Engineering with Computers (2004) 20: 22–30
DOI 10.1007/s00366-004-0275-5

ORIGINAL ARTICLE

Jun Qu · Radha Sarma

The continuous non-linear approximation of procedurally defined curves using integral B-splines

Received: 9 December 2002 / Accepted: 31 December 2003 / Published online: 5 March 2004

Abstract This paper outlines an algorithm for the continuous non-linear approximation of procedurally defined curves. Unlike conventional approximation methods using the discrete L_2 form metric with sampling points, this algorithm uses the continuous L_2 form metric based on minimizing the integral of the least square error metric between the original and approximate curves. Expressions for the optimality criteria are derived based on exact B-spline integration. Although numerical integration may be necessary for some complicated curves, the use of numerical integration is minimized by *a priori* explicit evaluations. Plane or space curves with high curvatures and/or discontinuities can also be handled by means of an adaptive knot placement strategy. It has been found that the proposed scheme is more efficient and accurate compared to currently existing interpolation and approximation methods.

Keywords Approximation · B-spline · CAD · Continuous · Interpolation · Reparametrization

Introduction

The equivalence of two parametric curves is an issue in many practical computer-aided design and manufacturing applications, e.g., surface-surface intersection, NC tool path generation, parametric surface trimming and robotic trajectory planning. Curves may be termed as parametrically equivalent or geometrically equivalent [1]. For example, curves $u(t)$ and $v(s)$ are parametrically equivalent if $s = t$ implies that $u(t) = v(s)$ in the common domain of s and t. On the other hand $u(t)$ and $v(s)$ are geometrically equivalent if they occupy the same locus of points but may be parameterised differently. In many applications, it is important that the parametric equivalence of curves is maintained after various conversions e.g., knot removal, degree elevation, degree reduction, or reparametrization.

Reparametrization is the process of altering the parametric speed along a curve or surface by the specification of a linear or non-linear function. In concept, an exact reparametrization is desired. A solution to the problem of exact non-linear reparametrization is outlined in The NURBS Book [2]. It involves a repeated application of the chain rule of differentiation to exactly compute the control points of the resulting curve. However, exact reparameterized functions may be very hard to obtain or too complex for practical use. An example will demonstrate the need for approximate algorithms such as those described in this paper. Consider a parametric surface $\boldsymbol{S}(u,v)$ in $\boldsymbol{R}^3$ of degree $n_0 \times m_0$. Also consider a curve that is represented in the parametric domain of the surface by parametric functions $u(t)$ and v(t), each of degree n_1 and m_1 respectively. In many applications the representation of the curve $\boldsymbol{C}(t)$ in $\boldsymbol{R}^3$ (which lies on the surface $\boldsymbol{S}(u,v)$) is needed for further processing. The equation of $\boldsymbol{C}(t)$ is shown in Eq. 1, where: B_i^n are the B-spline basis functions of degree n and the $\boldsymbol{Q}_{i,j}$ are the control points of the surface (a similar notation is adopted for curves). The degree of $\boldsymbol{C}(t)$ is $n_0 \cdot n_1 + m_0 \cdot m_1$, which can turn out to be a large number. For example, if $n_0 = n_1 = m_0 = m_1 = 3$ (a reasonable assumption), the resulting degree of $\boldsymbol{C}(t)$ will be 18 (too large for practical purposes). Thus, there exists the need for an algorithm for approximate reparametrization that closely maintains parametric equivalence, while ensuring that the degree of the resulting curve is within practical limits.

J. Qu (✉)
Metals and Ceramics Division,
Oak Ridge National Laboratory,
P.O. Box 2008, MS 6063 Oak Ridge,
TN 37831-6063, USA
E-mail: qujn@ornl.gov
Tel.: +1-865-5744560
Fax: +1-865-5746918

R. Sarma
Department of Mechanical Engineering,
University of Michigan, Ann Arbor, MI 48109, USA

$$\boldsymbol{C}(t) = \sum_{i=0}^{N_0} \sum_{j=0}^{M_0} B_i^{n_0}(u(t)) B_j^{m_0}(v(t)) \boldsymbol{Q}_{i,j} = \boldsymbol{S}(u(t), v(t)) \quad (1)$$

$$u(t) = \sum_{k=0}^{N_1} B_k^{n_1}(t) U_k \quad v(t) = \sum_{k=0}^{M_1} B_k^{m_1}(t) V_k$$

Interpolation and approximation methods have been used to approximate various types of procedurally defined curves based on sampled points. Piecewise cubic curves are quite commonly used to solve the Hermite interpolation problem of passing a curve through a set of points with specified end derivatives [3, 4, 5, 6]. Integral B-spline curve interpolation has also been used [7, 8]. In approximating curves, the discrete L_2 form metric, i.e., the sum of squared deviations between the sampled points and the approximated curve, are minimized to estimate the unknowns of the approximated curve [9, 10, 11 ,12, 13]. Usually, the unknowns are the control points of the approximated curve. Most often, this results in linear equations [9, 10, 11]. However, depending on how the problem is posed, non-linear systems could also result [12, 13].

This paper presents an alternate algorithm, with continuous approximation, using the continuous L_2 form metric by minimizing the integral of the least square error metrics between the original and approximate curves. This proposed scheme may be used to approximate any curves whose positions and derivatives can be found procedurally. The next two sections outline the core and high-level procedures of this non-linear approximation algorithm. Remarks and results are presented and followed by conclusions. In the context of expressions, **bold** faced letters are used to indicate vectors/matrices and ordinary letters to represent scalars.

The core procedure

Consider an ideal curve $\boldsymbol{C}(t)$ whose positions and derivatives may be obtained procedurally, i.e., the final B-spline representation of $\boldsymbol{C}(t)$ is not necessary to be available (or may be difficult to obtain). The goal of this section is to develop the core procedure for finding a B-spline curve $\tilde{\boldsymbol{C}}(t)$ that approximates the ideal curve $\boldsymbol{C}(t)$. A typical expression of $\tilde{\boldsymbol{C}}(t)$ is shown in Eq. 2:

$$\tilde{\boldsymbol{C}}(t) = \sum_{k=0}^{N} B_k^n(t) \boldsymbol{P}_k \quad (2)$$

$$T = [t_0, t_1, ..., t_{N+n+1}], t \in [a, b] = [t_n, t_{N+1}]$$

The assumptions made in developing the core procedure outlined in this section, are listed below.

- The curve $\tilde{\boldsymbol{C}}(t)$ is an integral B-spline curve.
- The degree (n) and the number of control points $(N+1)$ of $\tilde{\boldsymbol{C}}(t)$ are known.
- Parametric correspondence exists between the curves $\boldsymbol{C}(t)$ and $\tilde{\boldsymbol{C}}(t)$.
- T is a pre-determined knot sequence.
- The control points $\boldsymbol{P}_k$ are unknown and need to be determined such that a given metric is minimized.

Let the lth derivative of $\tilde{\boldsymbol{C}}(t)$ be represented as $\tilde{\boldsymbol{C}}^{(l)}(t)$, as shown in Eq. 3, where $\tilde{\boldsymbol{C}}^{(0)}(t) = \boldsymbol{C}(t)$. Note that the resulting control points $\boldsymbol{P}_k{}^{(l)}$ can be determined by the original control points $\boldsymbol{P}_k$ [2].

$$\tilde{\boldsymbol{C}}^{(l)}(t) = \sum_{k=0}^{N-l} B_k^{n-l}(t) \boldsymbol{P}_k^{(l)} \quad (3)$$

$$T^{(l)} = [t_l, t_{l+1}, ..., t_{N+n+1-l}], t \in [a, b] = [t_n, t_{N+1}]$$

The metric that determines the closeness of the ideal curve from the approximated curve is chosen such that the deviations in positions as well as derivatives are represented. The metric, shown in Eq. 4, represents the sum of square deviations of positions and derivatives between the ideal and approximated curves. A similar metric was used in curve fitting by Fang and Gossard [14]. The number of derivatives d $(0 \le d \le n)$ may be chosen depending on the desired closeness, in a least square sense, of positions and derivatives along the curve. The α_l is an arbitrary constant that scales various components of the metric.

$$E = \sum_{l=0}^{d} \alpha_l \int_a^b \left| \tilde{\boldsymbol{C}}^{(l)}(t) - \boldsymbol{C}^{(l)}(t) \right|^2 dt \quad (4)$$

The curve (in this case, its control points) can be determined by minimizing the metric in Eq. 4:

$$\frac{\partial E}{\partial \boldsymbol{P}_k} = 0 = \sum_{l=0}^{d} \alpha_l \int_a^b [\tilde{\boldsymbol{C}}^{(l)}(t) - \boldsymbol{C}^{(l)}(t)] \cdot r_{k,n}^{(l)}(t) dt \quad (5)$$

where:

$$r_{k,n}^{(l)}(t) = \frac{\partial \tilde{\boldsymbol{C}}^{(l)}(t)}{\partial \boldsymbol{P}_k} = \sum_{i=0}^{N-l} B_i^{n-l}(t) Q_{i,k,n}^{(l)} \quad (5a)$$

$$Q_{i,k,n}^{(0)} = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases} \quad (5b)$$

$$Q_{i,k,n}^{(l)} = \begin{cases} (n-l+1) \cdot \frac{Q_{i+1,k,n}^{(l-1)} - Q_{i,k,n}^{(l-1)}}{t_{i+n+1} - t_{i+l}}, & k-l \leqslant i \leqslant k \\ 0, & \textit{otherwise} \end{cases} \quad (5c)$$

By taking all the terms containing the unknowns (control points) to the left hand side, Eq. 5 can be expressed as:

$$\sum_{l=0}^{d} \alpha_l \int_a^b \tilde{\boldsymbol{C}}^{(l)}(t) r_{k,n}^{(l)}(t) dt = \sum_{l=0}^{d} \alpha_l \int_a^b \boldsymbol{C}^{(l)}(t) r_{k,n}^{(l)}(t) dt \quad (6)$$

Equation 6 yields $N+1$ equations in $N+1$ (vector) unknowns, $\boldsymbol{P}_k$, as represented by:

$$\frac{a_k \cdot \bar{P} = b_k}{0 \leqslant k \leqslant N} \tag{7}$$

where $\boldsymbol{a}_k$ are 1×(N + 1) row vectors, $\boldsymbol{b}_k$ are 1×1 vectors, and $\bar{\boldsymbol{P}}$ are the composite vector of control points that results from assembling $\boldsymbol{P}_k$.

For the left hand side of Eq. 6, the summation will be temporarily ignored and only one term will be considered. The main idea is to express the term shown in Eq. 8 as a product of matrices and the vector of unknowns $\bar{\boldsymbol{P}}$. Thus $\boldsymbol{H}_k^l$ are row vectors of size 1×(N-l + 1) and $\boldsymbol{G}_q$ are matrices of size (N-m)×(N-m + 1). They are defined in Eqs. 9 and 10, respectively.

$$\begin{aligned}\int_a^b \tilde{\boldsymbol{C}}^{(l)}(t) r_{k,n}^{(l)}(t)dt &= \sum_{i=0}^{N-1}\left(\int_a^b B_i^{n-l}(t)\cdot r_{k,n}^{(l)}(t)dt\right)P_i^{(l)} \\ &= \boldsymbol{H}_k^{(l)}\cdot\left(\prod_{m=l-1}^{0}\boldsymbol{G}_m\right)\cdot\bar{\boldsymbol{P}} = \boldsymbol{a}_k^{(l)}\cdot\bar{\boldsymbol{P}}\end{aligned} \tag{8}$$

where:

$$\begin{aligned}&\boldsymbol{G}_m = \left[g_{m,i,j}\right]_{(N-m)\times(N-m+1)} \\ &g_{m,i,j} = \begin{cases}\frac{m-n}{t_{i+n+1}-t_{i+m+1}}, & i=j \\ \frac{n-m}{t_{i+n+1}-t_{i+m+1}}, & i=j-1 \\ 0, & otherwise\end{cases}\end{aligned} \tag{8a}$$

$$\begin{aligned}&H_k^{(l)} = \left[h_{i,k}^{(l)}\right]_{1x(N-l+1)} \\ &h_{i,k}^{(l)} = \begin{cases}\int_a^b r_{i,n-l}^{(0)}(t)\cdot r_{k,n}^{(l)}(t)dt, & k-l-n\leqslant i\leqslant k+n \\ 0, & otherwise\end{cases}\end{aligned} \tag{8b}$$

The calculation of $h_{i,k}^{(l)}$ is involved in the integrals of products of B-spline functions, whose derivations are shown in the Appendix. Finally, $\boldsymbol{a}_k$ can be obtained as follows:

$$\boldsymbol{a}_k = \sum_{l=0}^{d}\alpha_l\boldsymbol{a}_k^{(l)} = \sum_{l=0}^{d}\left(\alpha_l\cdot\boldsymbol{H}_k^{(l)}\cdot\prod_{m=l-1}^{0}\boldsymbol{G}_m\right) \tag{9}$$

The $\boldsymbol{b}_k$ can be evaluated in two cases as shown in Eqs. 10 and 11 when: $\frac{n}{2} < l \leqslant n$ $\boldsymbol{b}_k^{(l)}$ can be calculated exactly by Eq. 10b (see the Appendix for integrals of products of B-spline functions); and when $0 < l \leqslant \frac{n}{2}$, numerical integration may be necessary.

$$\begin{aligned}\boldsymbol{b}_k^{(l)} &= \int_a^b \boldsymbol{C}^{(l)}(t) r_{k,n}^{(l)}(t)dt \\ &= \begin{cases}\boldsymbol{I}_k^{(l)} + (-1)^l\int_a^b \boldsymbol{C}(t)\cdot r_{k,n}^{(2l)}(t)dt, & 0<l\leqslant\frac{n}{2} \\ \boldsymbol{J}_k^{(l)}, & \frac{n}{2}<l\leqslant n\end{cases}\end{aligned} \tag{10}$$

where:

$$\begin{aligned}\boldsymbol{I}_k^{(l)} = \sum_{j=0}^{l-1}(-1)^j\Big[&\boldsymbol{C}^{(l-j-1)}(b)\cdot r_{k,n}^{(l+j)}(b) \\ &- \boldsymbol{C}^{(l-j-1)}(a)\cdot r_{k,n}^{(l+j)}(a)\Big]\end{aligned} \tag{10a}$$

$$\begin{aligned}\boldsymbol{J}_k^{(l)} = &\sum_{j=0}^{n-l-1}(-1)^j\Big[\boldsymbol{C}^{(l-j-1)}(b)\cdot r_{k,n}^{(l+j)}(b) \\ &- \boldsymbol{C}^{(l-j-1)}(a)\cdot r_{k,n}^{(l+j)}(a)\Big] \\ &+ (-1)^{n-}\sum_{j=0}^{N-n}\Big[\boldsymbol{C}^{(2l-n-1)}(t_{j+n+1}) \\ &- \boldsymbol{C}^{(2l-n-1)}(t_{j+n})\Big]\cdot Q_{j,k,n}^{(n)}\end{aligned} \tag{10b}$$

Thus:

$$\boldsymbol{b}_k = \sum_{l=\frac{n}{2}+1}^{d}\alpha_l\boldsymbol{J}_k^{(l)} + \sum_{l=0}^{\min(d,\frac{n}{2})}\alpha_l\boldsymbol{I}_k^{(l)} + \int_{t_k}^{t_{k+n+1}}\boldsymbol{C}(t)\cdot f_k(t)dt \tag{11}$$

where:

$$f_k(t) = \sum_{l=0}^{\min(d,\frac{n}{2})}(-1)^l\alpha_l\cdot r_{k,n}^{(2l)}(t) = \begin{cases}\neq 0, & t_k\leqslant t\leqslant t_{k+n+1} \\ = 0, & otherwise\end{cases} \tag{11a}$$

Note that if $\boldsymbol{C}(t)$ could be represented directly as a B-spline function, the $\boldsymbol{b}_k$ would be evaluated exactly without numerical integration involved. However, when $\boldsymbol{C}(t)$ is a procedurally defined function or difficult to represent in a final B-spline form, numerical integration has to be applied.

Once $\boldsymbol{a}_k$ and $\boldsymbol{b}_k$ are evaluated, a linear system of equations obtained as shown in Eq. 12, which can be solved to obtain the control points $\bar{\boldsymbol{P}}$ of $\tilde{C}(t)$. Each $\boldsymbol{a}_k$ is a 1×(N + 1) row vector, and A turn out to be a (N + 1)×(N + 1) matrix.

$$\boldsymbol{A}\cdot\boldsymbol{P} = \boldsymbol{B}$$

$$\begin{bmatrix}[\boldsymbol{a}_0]_{1x(N+1)} \\ [\boldsymbol{a}_1]_{1x(N+1)} \\ \vdots \\ [\boldsymbol{a}_N]_{1x(N+1)}\end{bmatrix}_{(N+1)\times(N+1)}\cdot\begin{bmatrix}\boldsymbol{P}_0 \\ \boldsymbol{P}_1 \\ \vdots \\ \boldsymbol{P}_N\end{bmatrix}_{(N+1)\times 1} = \begin{bmatrix}\boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_N\end{bmatrix}_{(N+1)\times 1} \tag{12}$$

A high-level procedure

The core procedure described in Sec. 2 assumes that the degree (n), the number of control points $(N+1)$, and the knot vector are known *a priori*. While it is realistic to assume a certain degree for the resulting curve, the number of control points and the knot vector need to be determined based on user-specified error bounds on the positions and derivatives. This error specified is generally different from the least square error $E^{(l)}$, since most

users are more interested in an estimate of the set theoretic distance measure: the Hausdorff distance $e^{(l)}$ between the ideal and approximated curves [15]. Thus, a high-level iterative procedure for finding the number of control points and the knot vector based on repeatedly using the core procedure is outlined.

1. The degree n of the approximated curve is fixed (usually degree 3 or 4 curves are preferred). Initially, the number of control points is set to the smallest possible number, i.e., $n+1$ for a B-spline curve. The knot vector is initialised to a uniform sequence in the parametric domain of interest as $T=[t_0,t_1,\ldots t_{2n+1}]$.
2. The core procedure is used to find the optimal control points of the approximated curve.
3. The errors between the ideal and approximate curves are evaluated numerically based on Eq. 13. The discussion of the error evaluation can be found in [16].

$$e^{(l)} = \sup \left| \boldsymbol{C}^{(l)}(t) - \tilde{\boldsymbol{C}}^{(l)}(t) \right| \leqslant e^{(l)}_{Specified} \tag{13}$$

4. If all the errors calculated in Eq. 12 are less than the values of the user specified errors, the iteration stops. Otherwise, an adaptive knot spacing strategy is used to enhance the knot vector (see the section The knot insertion strategy) and steps 2–4 are repeated.

Remarks

The knot insertion strategy

The algorithm for continuous non-linear approximation, while avoiding point sampling, does not get rid of the problem of determining an appropriate knot placement strategy. The knot vector needs to be enhanced for each high-level iteration undertaken. In general, placing additional knots in regions of high curvature (or regions where higher derivatives have large magnitudes) is known to improve the quality of the approximation [8, 17]. In this algorithm, a similar strategy is adapted that additional knots are inserted at selected parameters, where high local maximum errors occur. If r knots are inserted, the number of control points increases from $N+1$ to $N+r+1$. Thus, the knot insertion strategy automatically determines the unknown number of control points for the next iteration. Internal multiple knots are restricted to a multiplicity of $\leq n$ and end knots (if dealing with end-point interpolating curves) to a multiplicity of $\leq n+1$. More details of the adaptive knot insertion strategy can be found in [16].

Non-singularity

In order to make certain that Eq. 12 can be uniquely solved every single time, we must guarantee the coefficient matrix $\boldsymbol{A}$ is not singular or invertible. In the context of interpolation and approximation, rules of thumb ensure that the resulting linear system of equations can be solved [18]. For this continuous approximation method, it can be proved that the coefficient matrix $\boldsymbol{A}$ is always non-singular, as discussed below.

In the context of interpolation, let the t_i represent knots in the knot vector and τ_i represent parameters corresponding to the sampled points. DeBoor states that a non-singular system of linear equations will result if and only if $B^n_i(\tau) \neq 0$, i.e., if and only if $t_i < \tau_i < t_i + n + 1$ (see [18]). For conventional approximation, $S+1$ $(S>N)$ discrete points must be sampled, and the resulting linear system of equations will be non-singular if and only if for some $0 \leq j_0 \leq \ldots \leq j_N \leq S$ the condition $t_i < \tau_j < t_i + n + 1$ is satisfied (see [18]). In other words, at least $N+1$ of the parameter values (τ_{j_i}) corresponding to the sampled points must lie in the corresponding knot intervals $[t_i, t_i+n+1]$, in order to guarantee a non-singular system of linear equations.

For the proposed continuous approximation in this paper, since integration is used instead of summation, all points on the ideal curve are "sampled". Obviously, there must exist a set of points τ_{j_i} that satisfy the condition $t_i < \tau_{j_i} < t_{i+n+1}$. Thus by [18], matrix $\boldsymbol{A}$ will never be singular.

Special points and discontinuities

This continuous non-linear approximation can handle special points (e.g., cusps) or discontinuities of the ideal curve or the reparametrizing function without any separate efforts or increases in computational complexity. Assuming that there is a special point or discontinuity at the parameter $\tilde{t}$, the knot insertion strategy exploits the fact that the error $e^{(l)}(\tilde{t}) > e^{(l)}_{Specified}$ allowing the placement of multiple knots at $\tilde{t}$. Through numerical experiments it was found that the resulting multiplicity of knots reasonably agrees with the degree of discontinuity in the curve. For example, for a C^0 discontinuous point, the corresponding knot will be of multiplicity n, and for a C^1 discontinuous point the multiplicity of the knot will be $n-1$, etc.

To accommodate multiple knots, the algorithm for integrating products of B-spline functions was modified in the following manner. Wherever there is a repeated knot, a knot refinement is performed to increase the multiplicity to $n+1$. In other words, the curve is divided into several individual B-spline curves, each having no multiple interior knots (this is done only for the purpose of integration; the curve itself is not modified). The integration is then performed using equations in the Appendix.

This approximation method can also be applied to curves with parametric discontinuities or piecewise curves, as discussed in [16].

Table 1 The original plane curve in example 1

Curve expression	Degree	No. of control points
$\boldsymbol{C}(t) = \boldsymbol{S}_{xy}(u(t), v(t)) = \sum_{i=0}^{N_0} \sum_{j=0}^{M_0} B_i^{n_0}(u(t)) B_j^{m_0}(v(t)) \boldsymbol{Q}_{i,j}$ $u(t) = \sum_{k=0}^{N_1} B_k^{n_1}(t) U_k \quad v(t) = \sum_{k=0}^{M_1} B_k^{m_1}(t) V_k$	$n_0 = 3$ $m_0 = 3$ $n_1 = 3$ $m_1 = 3$	$N_0 = 6$ $M_0 = 6$ $N_1 = 6$ $M_1 = 6$

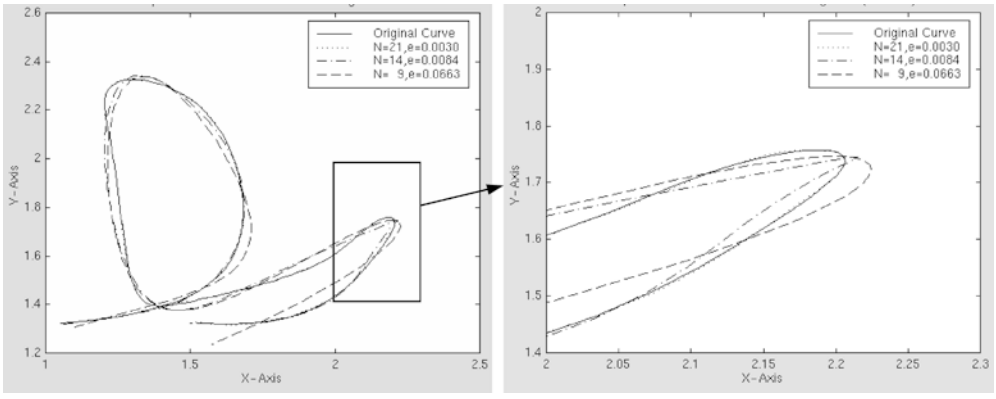

Fig. 1 The original curve and continuous approximate curves at different iterations in example 1

Examples and discussions

Three examples are presented and results are compared between the proposed continuous approximation and conventional interpolation and discrete approximation in this section. Details of the control points and knot vectors are omitted for brevity. All the examples used $d=0$, i.e. only positional errors ($e^{(0)}$) have been specified by the user (see the section The selection of weights in the error metric for a discussion when d is non-zero). The metrics for comparing the three different approximation methods are: the number of control points, the number of high-level iterations, and the average curvature deviation. The average curvature deviation can be calculated by Eq. 14, where the curvatures on the original curve and the approximated curve were sampled at $S+1$ parameter values.

$$\kappa_{dev} = \frac{\sum_{i=0}^{S} \left| \kappa_{orig}(t_i) - \kappa_{approx}(t_i) \right|}{S+1} \tag{14}$$

Examples and results

Example 1: the plane curve

A two-dimensional curve $\boldsymbol{C}(t)$ is defined in Table 1 and shown in Fig. 1. The desired degree n of $\tilde{\boldsymbol{C}}(t)$ was set to 3 and the specified error limit was:

$$e^{(0)}_{Specified} = 0.005$$

Table 2 presents the curves generated by the proposed continuous approximation, interpolation and discrete approximation methods, respectively. The continuous approximate curves at different iterations in Fig. 1 clearly show the approximation approach.

Example 2: The space curve

The original curve $\boldsymbol{C}(t)$ was a three-dimensional curve, as shown in Fig. 2. Table 3 gives the expression with the information of degree and number of control points. The desired curve $\tilde{\boldsymbol{C}}(t)$ has degree $n = 3$ and the specified error limit was:

Table 2 Approximate plane curves generated by different methods in example 1

Method	Degree	No. of control points	No. of iterations	No. of sampled points	Ave. curvature deviation κ_{dev}
Continuous approximation	3	22	5	N/A	0.633
Discrete approximation	3	23	5	71	0.972
Interpolation	3	32	6	32	0.867

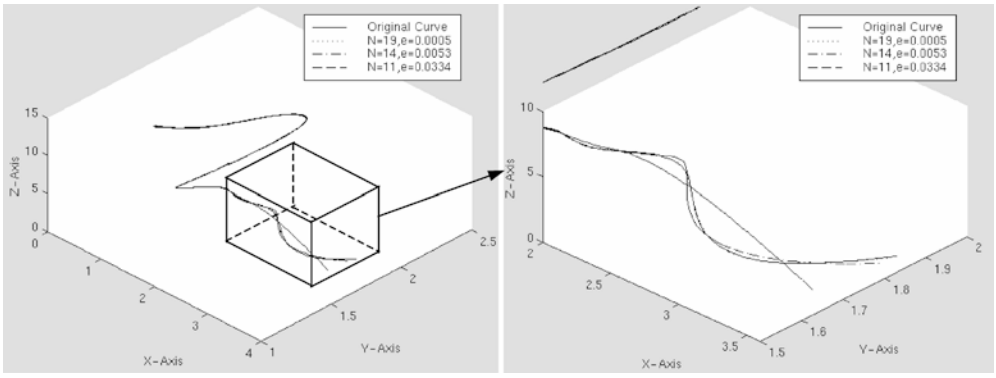

Fig. 2 The original curve and continuous approximate curves at different iterations in example 2

Table 3 The original space curve in example II

Curve expression	Degree	No. of control points
$\boldsymbol{C}(t) = \boldsymbol{S}_{xyz}(u(t), v(t)) = \sum_{i=0}^{N_0} \sum_{j=0}^{M_0} B_i^{n_0}(u(t)) B_j^{m_0}(v(t)) \boldsymbol{Q}_{i,j}$ $u(t) = \sum_{k=0}^{N_1} B_k^{n_1}(t) U_k \quad v(t) = \sum_{k=0}^{M_1} B_k^{m_1}(t) V_k$	$n_0 = 3$ $m_0 = 3$ $n_1 = 3$ $m_1 = 3$	$N_0 = 6$ $M_0 = 6$ $N_1 = 6$ $M_1 = 6$

Table 4 Approximate space curves generated by different methods in example 2

Method	Degree	No. of control points	No. of iterations	No. of sampled points	Ave. curvature deviation κ_{dev}
Continuous approximation	3	20	5	N/A	0.089
Discrete approximation	3	26	5	73	0.134
Interpolation	3	30	6	30	0.296

Table 5 Original piecewise curve in example 3

Curve expression	Degree	No. of control points	No. of discontinuities
$\boldsymbol{C}(t) = \boldsymbol{C}(u(t)) = \begin{cases} \sum_{i=0}^{N_0} B_i^{n_0}(u) \boldsymbol{P}_i, & 0 \leqslant u \leqslant 0.7 \\ \sum_{i=0}^{M_0} B_i^{m_0}(u) \boldsymbol{Q}_i, & 0.7 < u \leqslant 1 \end{cases}$	$n_0 = 1$ $m_0 = 4$	$N_0 = 5$ $M_0 = 12$	6
$u(t) = \sum_{k=0}^{N_1} B_k^{n_1}(t) U_k$	$n_1 = 3$	$N_1 = 6$	N/A

$$e^{(0)}_{Specified} = 0.001$$

Table 4 lists the curves generated by the three approximation methods. The continuous approximate curves at different iterations are shown in Fig. 2.

Example 3: The piecewise curve with multiple tangent discontinuities

The original curve $\boldsymbol{C}(t)$ is a piecewise curve with multiple tangent discontinuities, as defined in Table 5 and shown in Fig. 3. The desired degree n of $\tilde{\boldsymbol{C}}(t)$ was set to 3 and the specified error limit was:

$$e^{(0)}_{Specified} = 0.005$$

The approximate curves generated by the proposed and conventional methods are presented in Table 6. The curves produced by the continuous approximation algorithm at different iterations are shown in Fig. 3.

Continuous approximation versus interpolation and discrete approximation

The interpolation and discrete approximation used in the abovementioned examples are also comprised of a core procedure and high-level iterative procedure. The implementation of the interpolation basically followed the algorithm described in [8], except for an improvement on the sampling strategy. Points were added at the locations of local maximum errors (adaptive point sampling strategy) instead of at mid-spans. This enhanced the interpolation algorithm with a faster convergence of the high-level iteration procedure. The

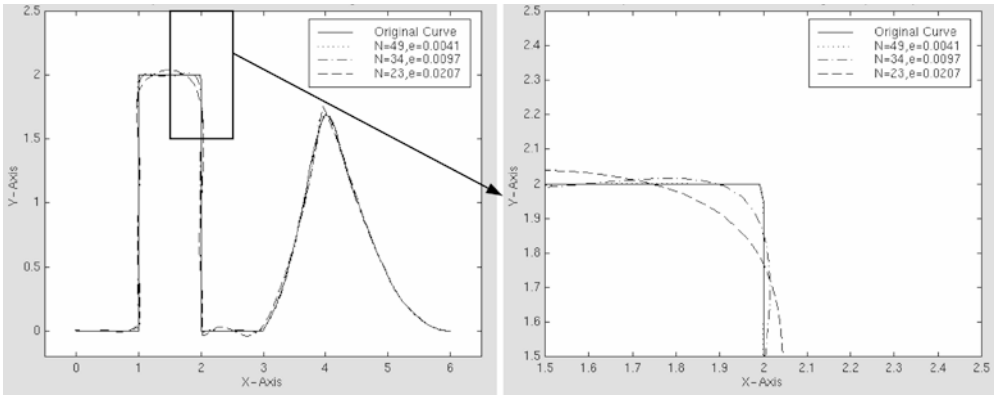

Fig. 3 The original curve and continuous approximate curves at different iterations in example 3

Table 6 Approximate curves generated by different methods in example 3

Method	Degree	No. of control points	No. of discontinuities detected	No. of iterations	No. of sampled points	Ave. curvature deviation κ_{dev}
Continuous approximation	3	50	5	7	N/A	0.472
Discrete approximation	3	61	4	9	108	0.634
Interpolation	3	71	0	10	71	0.752

discrete approximation used in this study was based on the algorithm presented in [17]. Again, the adaptive point sampling strategy was introduced to improve the performance.

Based on the results presented in Tables 2, 4 and 6, the proposed continuous approximation has shown several advantages over the other two conventional methods:

1. more concise expression (less control points),
2. faster convergence of high-level procedure (less iterations),
3. better capability of detecting and accommodating special points and discontinuities,
4. lower average curvature deviation.

These make the continuous approximation a very competitive alternative to previously existing discrete approximation methods. While numerical integration may be involved, this new method has minimized the opportunities of performing numerical integration by giving most evaluations explicitly.

The selection of weights in the error metric

The parameter α_1 in the error metric (Eq. 4) acts as weights for balancing the errors of the function and its derivatives. When the error bounds of the function and its derivatives are independently specified, the weights can be adjusted to ensure that the error bounds are being satisfied without an excessive increase in the number of control points. That is another advantage over the interpolation method, which can only control the error of the function.

The heuristics for specifying α_1 depend on the range of magnitudes of positions and derivatives and on the desired accuracy of positions and derivatives. The range of magnitude of the position or derivatives is defined in Eq. 15, where t_i and t_j are distinct parameter values. Then the recommended values of αl are shown in Eq. 16.

$$range(\boldsymbol{C}^{(l)}(t)) = \sup \left| \boldsymbol{C}^{(l)}(t_i) - \boldsymbol{C}^{(l)}(t_j) \right| \tag{15}$$

$$\alpha_l = \frac{\tilde{\alpha}_l}{\left[range(\boldsymbol{C}^{(l)}(t)) \right]^2} \tag{16}$$

Without weight control, the errors on the derivatives of the function are usually orders of magnitude higher [19], e.g., for a given error of the function $(\propto N^{-4})$, the errors of the first, second and third derivatives could be: $\propto N^{-3}$, $\propto N^{-2}$, and $\propto N^{-1}$, respectively. Here $N+1$ is the number of control points. Thus, it is reasonable to set $\tilde{\alpha}_l = 10^l$ to give heavier weights to the derivatives, and eventually reduce the error magnitudes of the derivatives.

Example 1 was used to demonstrate the effect of weights on the approximate curve as shown in Table 7. Apparently, the weights adjustment reduced the errors of derivatives effectively.

Conclusions

A non-linear approximation algorithm, using the continuous L_2 norm metric, has been developed. It is a tractable alternative to the currently available interpolation and approximation algorithms using the discrete L_2 norm metric. Based on several numerical examples, this method can be described as being robust and able to approximate complex curves. The curves generated by

Table 7 Effect of weights on the approximate curve

No. of derivatives	Specified error bounds	$\tilde{\alpha}_l$	No. of control points	Errors
d = 0	$e^{(0)}_{Specified} = 0.01$	$\tilde{\alpha}_0 = 1$	14	$e^0 = 0.008$ $e^1 = 0.090$ $e^2 = 0.585$
d = 1	$e^{(0)}_{Specified} = 0.05$	$\tilde{\alpha}_0 = 1$	13	$e^0 = 0.026$
	$e^{(1)}_{Specified} = 0.05$	$\tilde{\alpha}_1 = 10$		$e^1 = 0.044$ $e^2 = 0.449$
d = 2	$e^{(0)}_{Specified} = 0.1$	$\tilde{\alpha}_0 = 1$	10	$e^0 = 0.026$
	$e^{(1)}_{Specified} = 0.1$	$\tilde{\alpha}_1 = 10$		$e^1 = 0.020$
	$e^{(2)}_{Specified} = 0.1$	$\tilde{\alpha}_2 = 100$		$e^2 = 0.076$

the proposed method have shown higher efficiency and precision, with less control points, better discontinuity handling and lower curvature deviation, than those of the conventional interpolation and approximation. Although the numerical integration cannot be avoided for some procedurally defined curves, the possibility of performing numerical integration has been minimized by explicitly expressing most evaluations in this continuous approximation algorithm.

Appendix: Integrals of products of B-spline functions

The derivatives and integrals of B-spline functions can be found in [2, 20], and are shown in this appendix for completeness. Consider a B-spline function defined over a non-decreasing knot sequence as shown in Eq. 17:

$$\begin{aligned} u(t) &= \sum_{k=0}^{N} B_k^n(t) U_k \\ T &= [t_0, t_1, ..., t_{N+n+1}] \end{aligned} \tag{17}$$

The general recursive scheme for evaluating the lth derivative ($l > 0$) or integral ($l < 0$) for a B-spline function is as follows:

$$u^{(l)}(t) = \sum_{k=0}^{N-l} B_k^{n-l}(t) U_k^{(l)} \tag{18}$$

$$U_k^{(l)} = \begin{cases} \frac{n-l+1}{t_{k+n+1} - t_{k+i}} \left(U_{k+1}^{(l-1)} - U_k^{(l-1)} \right), & l > 0 \\ \sum_{j=0}^{k-1} \frac{t_{j+n+1} - t_{j+l+1}}{n-l} \cdot U_j^{(l+1)}, & l < 0 \end{cases} \tag{18a}$$

$$T^{(l)} = \begin{cases} [t_l, t_{l+1}, \ldots, t_{N+n-l+1}], & l > 0 \\ [t_l, t_{l+1}, \ldots, t_0, \ldots, t_{N+n+1}, \ldots, t_{N+n-l+1}], & l < 0 \end{cases} \tag{18b}$$

Note that when $l < 0$, the new knots t_l, t_{l+1},...,t_{-1} and t_{N+n+2},...,$t_{N+n-l+1}$ can be chosen arbitrarily outside of the original knot vector T, as long as it ensures that the new knot vector $T^{(l)}$ has a non-decreasing sequence.

The integrals of products of B-spline functions may be derived by repeated applications of the chain rule of integration [20]. Consider evaluating the following integral:

$$\int_a^b v(t)u(t)dt = v(t) \int u(t)dt \Big|_a^b - \int_a^b \left[\frac{d}{dt} v(t) \cdot \int u(t)dt \right] dt \tag{19}$$

where $v(t)$ and $u(t)$ are B-spline functions as shown below:

$$\begin{aligned} u(t) &= \sum_{k=0}^{N} B_k^n(t) U_k \qquad & v(t) &= \sum_{k=0}^{M} B_k^m(t) V_k \\ T_u &= [t_{u,0}, t_{u,1}, ..., t_{u,N+n+1}] & T_v &= [t_{v,0}, t_{v,1}, ..., t_{v,M+m+1}] \end{aligned} \tag{20}$$

The two B-spline functions $v(t)$and $u(t)$ need not have same knot vectors, while the range of the definite integral satisfies $[a, b] \subseteq [t_{v,m}, t_{v,M+1}] \cap [t_{u,n}, t_{u,N+1}]$. Using Eq. 18 and the notations for derivatives and integrals introduced above, Eq. 19 becomes:

$$\begin{aligned} \int_a^b v(t)u(t)dt = \sum_{i=0}^{m-1} (-1)^i \Big[& v^{(i)}(b) \cdot u^{(-i-1)}(b) \\ & - v^{(i)}(a) \cdot u^{(-i-1)}(a) \Big] \\ & + (-1)^m \int_a^b v^{(m)}(t) \cdot u^{(-m)}(t)dt \end{aligned} \tag{21}$$

Note that $v(m)(t)$ is a constant because its degree is zero. Thus Eq. 21 can be simplified to:

$$\begin{aligned} &\int_a^b v(t)u(t)dt \\ &= \sum_{i=0}^{m} (-1)^i \left[v^{(i)}(b) \cdot u^{(-i-1)}(b) - v^{(i)}(a) \cdot u^{(-i-1)}(a) \right] \end{aligned} \tag{22}$$

Acknowledgements Partial support from the National Science Foundation, grant DMI-9713818, and from Iowa State University startup funds is gratefully acknowledged.

References

1. Alt L (1992) Rational linear reparametrization of NURBS and the blossoming principle. Comp Aid Geom Des 9:313–319
2. Piegl L, Tiller W (1997) The NURBS book. Springer, Berlin Heidelberg New York
3. DeBoor C, Hollig K, Sabin M (1987) High accuracy geometric Hermite interpolation. Comp Aid Geom Des 4:269–278
4. Sakai M, Usmani RA (1990) On orders of approximation of plane curves by parametric cubic splines. BIT 30:735–741
5. Sederberg TW, Nishita T (1991) Geometric Hermite approximation of surface patch intersection curves. Comp Aid Geom Des 8:97–114
6. Meek DS, Walton DJ (1997) Geometric Hermite interpolation with Tschirnhausen cubics. J Comp Appl Math 81:299–309
7. Patrikalakis NM (1989) Approximate conversion of rational splines. Comp Aid Geom Des 6:155–165
8. Wolter FE, Tuhoy ST (1992) Approximation of high-degree procedural curves. Eng Comp 8:61–80
9. Patrikalakis NM, Bardis L (1991) Localization of rational B-spline surfaces. Eng Comp 7:237–252
10. Piegl LA, Tiller W (2000) Least-squares B-spline curve approximation with arbitrary end derivatives. Eng Comp 16(2):109–116
11. Borges CF, Pastva T (2002) Total least squares fitting of Bezier and B-spline curves to ordered data. Comp Aid Geom Des 19(4):275–289
12. Hoschek J (1988) Intrinsic parametrization for approximation. Comp Aid Geom Des 5:27–31
13. Hoschek J, Wissel N (1988) Optimal approximate conversion of spline curves and spline approximation of offset curves. Comp Aid Des 20:475–483
14. Fang L, Gossard DC (1995) Multidimensional curve fitting to unorganized data points by nonlinear minimization. Comp Aid Des 27:48–58
15. Fritsch FN, Nielson GM (1989) On the problem of determining the distance between parametric curves. In: Proceedings of the SIAM Conference on Curve and Surface Design, Tempe, AZ, 6–10 November 1989–
16. Qu J (1999) Approximate non-linear reparametrization of procedural curves using integral B-splines. Masters Thesis, Iowa State University
17. Patrikalakis NM (1989) Approximate conversion of rational splines. Comp Aid Geom Des 6:155–165
18. DeBoor C (1978) A practical guide to splines. Springer, Berlin Heidelberg New York
19. Watson GA (1980) Approximation theory and numerical methods. Wiley, New York
20. Vermeulen AH, Bartels RH, Heppler GR (1992) Integrating products of B-splines. SIAM J Sci Stat Comp 13:1025–1038