Available online at www.sciencedirect.com
ScienceDirect
Computer Aided Geometric Design 25 (2008) 837–849
COMPUTER AIDED GEOMETRIC DESIGN
www.elsevier.com/locate/cagd

A local fitting algorithm for converting planar curves to B-splines

Chongyang Deng, Xunnian Yang *

Department of Mathematics, Zhejiang University, Yuquan, Hangzhou 310027, People's Republic of China

Received 22 May 2006; received in revised form 25 October 2007; accepted 2 November 2007

Available online 17 November 2007

Abstract

In this paper we present a local fitting algorithm for converting smooth planar curves to B-splines. For a smooth planar curve a set of points together with their tangent vectors are first sampled from the curve such that the connected polygon approximates the curve with high accuracy and inflexions are detected by the sampled data efficiently. Then, a G^1 continuous Bézier spline curve is obtained by fitting the sampled data with shape preservation as well as within a prescribed accuracy. Finally, the Bézier spline is merged into a C^2 continuous B-spline curve by subdivision and control points adjustment. The merging is guaranteed to be within another error bound and with no more inflexions than the Bézier spline. In addition to shape preserving and error control, this conversion algorithm also benefits that the knots are selected automatically and adaptively according to local shape and error bound. A few experimental results are included to demonstrate the validity and efficiency of the algorithm.

Keywords: Curve conversion; B-spline curve; Cubic Bézier curve; Knot selection; Shape preserving

1. Introduction

In the field of computer aided geometric design (CAGD) and related applications, there are various ways to define and represent planar curves. For example, parametric, implicit and subdivision curves are basic forms for curve and surface modeling. Moreover, other types of curves, such as offset curves, intersection curves and so on, are frequently encountered in CAGD but cannot be represented in a simple form generally. Among all forms of curves and surfaces, B-spline is most widely used in many CAD/CAM systems. For the convenience of application, it is often necessary to convert other types of planar curves to B-splines.

Most often, implicit curves, subdivision curves or even a different type of parametric curves cannot be transformed to B-spline exactly. Then curve conversion from other forms to B-spline means the approximation of a given curve by a B-spline curve. Generally, the conversion should satisfy the following three requirements: (1) The approximating curve lies on the original curve within a given tolerance. (2) The control points of the B-spline are as few as possible. (3) The shape of the fitting B-spline is compatible with the original curve.

In the literature several methods have been proposed for curve conversion. Hoschek (1987), Hoschek and Wissel (1988) proposed conversion algorithms for spline curve by degree reduction and degree elevation. Patrikalakis (1989)

* Corresponding author.
E-mail addresses: ddd1742@163.com (C. Deng), yxn@zju.edu.cn (X. Yang).

0167-8396/$ – see front matter
doi:10.1016/j.cagd.2007.11.001

tried to approximate rational B-spline curves by integer B-spline curves. Tai et al. (2003) proposed to merge two B-spline curves into one curve with single interior knots, but without control of fitting error. To convert general types of planar curves to B-spline, Park (2004) adopted a two-step algorithm for curve conversion, where the original curve is first approximated by a polygon with an error bound and then the polygon is refitted by a B-spline curve with another error bound. However, the fitting B-spline curve is not guaranteed to be shape preserving. To find a shape preserving B-spline curve, additional constraints such as local convexity should be considered along with curve fitting. Using a reference curve, Jüttler (1997) generates linear sufficient conditions for the convexity of the approximant. However, this method has not taken error tolerance into account.

A general type of parametric curves can be approximated by polygons with proper parametric steps (Filip et al., 1986; Kosters, 1991). Besides parametric curves, implicit curves can also be numerically parameterized (Hartmann, 2000) or an algebraic curve can be plotted (Taubin, 1994) efficiently. To fit a point set or a polygonal curve by B-spline, one should set proper knots and solve control points for the B-spline curve (Hoschek and Lasser, 1993; Piegl and Tiller, 1997; Yang et al., 2004). The choice of knots has considerable effect on the shape of a B-spline curve (Farin, 2002), and knot placement is in fact a multivariate and multimodal nonlinear optimization problem (Yoshimoto and Harada, 2003). To obtain a B-spline curve with small number of control points, Eck and Hadenfeld (1995), Lyche and Mørken (1987, 1988) employed knot removal algorithms to reduce the number of control points. Alternatively, the knots of the B-spline curve can be chosen iteratively (Ma and Kruth, 1995; Park, 2004) or adaptively with the data (Saux and Daniel, 1999; Razdan, 1999; Li et al., 2005; Park and Lee, 2007).

In this paper we present a novel numerical algorithm for converting planar curves to B-splines. For a given smooth planar curve, the distance from an approximating polygon can be estimated explicitly(e.g. Park, 2004). If we sample points and tangent vectors from the original curve with enough density and accuracy, the deviation from the connecting polygon to the original curve is always much less than the tolerance for B-spline fitting. Then we here neglect the sampling error and pay our attention to fitting a polygon curve by a B-spline with shape preservation and error control. Instead of fitting a B-spline curve directly, we first fit the sampled data with a G^1 continuous Bézier spline curve. Each Bézier curve is obtained to fit as many as possible the sampled data without unwanted inflexion and in a given tolerance. Then we subdivide each of intermediate Bézier curves and perturb the control points further to make a C^2 Bézier spline curve. The subdivision parameters are computed adaptively and the distance from the perturbed Bézier curves to their original positions is bounded by another given tolerance. Moreover, the inflexion number is guaranteed to be no more than that of the original Bézier spline curve. Finally the C^2 continuous Bézier spline is merged into a B-spline curve.

The major contributions of this paper lie in two aspects: (a) we achieve shape preserving in curve conversion by fitting the sampled data with a spline of Bézier curves locally; (b) we propose a method to merge a sequence of Bézier curves which are G^1 continuous to a C^2 continuous B-spline under the constraints of error control and shape preserving.

The rest of this paper is organized as follows: In Section 2, data fitting by a G^1 cubic Bézier spline is discussed. How to merge a sequence of G^1 continuous cubic Bézier curves to a C^2 cubic B-spline curve is presented in Section 3. The entire converting algorithm is given in Section 4. We generalize the algorithm for higher order B-spline curves in Section 5. Examples are given in Section 6 and we conclude the paper in Section 7.

2. Cubic Bézier spline curve fitting

Let $\{\boldsymbol{A}_i\}$ $(i = 0, 1, \ldots, n)$ be a set of sampled points and $\{\boldsymbol{q}_i\}$ $(i = 0, 1, \ldots, n)$ be the unit tangent vectors at these points, we would search for a sequence of cubic Bézier curves to fit the data within a prescribed tolerance ε_1. The Bézier curves should be as few as possible and the inflexion number inferred by these Bézier curves is no more than that defined by the input data. We solve this problem by fitting the data locally and obtain the resulting Bézier curves sequently. If a fitting cubic Bézier curve lies within the given tolerance and has no undesired inflexion, it will be added to the sequence. Otherwise, we choose less data and fit another cubic Bézier curve. This procedure is repeated until a shape preserving Bézier curve within permitted tolerance is found.

Without loss of generality, we assume $\{\boldsymbol{A}_i\}$ $(i = 0, 1, \ldots, m)$ be the point set to be fitted by a cubic Bézier curve

$$\boldsymbol{P}(t) = \sum_{i=0}^{3} B_i^3(t)\boldsymbol{P}_i,$$

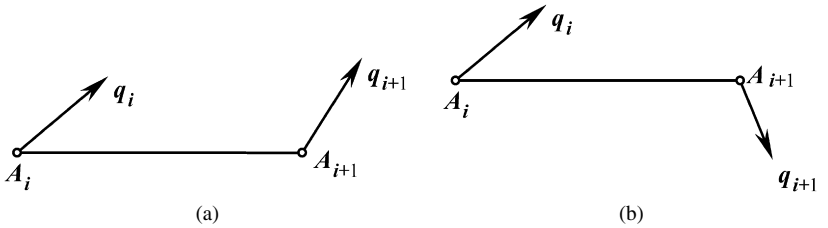


Fig. 1. Inflexion detection between points A_i and A_{i+1}.

where $B_i^3(t)$ $(i = 0, 1, 2, 3)$ are the Bernstein basis functions. To obtain a G^1 continuous Bézier spline curve in the end, two end points and tangents at the ends should be interpolated. Then we have $P_0 = A_0$, $P_3 = A_m$, $P_1 - P_0 = \alpha_1 q_0$, $P_3 - P_2 = \alpha_2 q_m$. The unknowns α_1 and α_2 are positive and can be obtained by solving a least square problem (Shao and Zhou, 1996).

2.1. Inflexion identification

To judge whether a new fitting Bézier segment is shape preserving or not, we should check the numbers of inflexions for the sampled data, the cubic Bézier segment, and the Bézier spline curve.

Let $\{A_i\}$ $(i = 0, 1, \ldots, m)$ be a sequence of sampled points and $\{q_i\}$ $(i = 0, 1, \ldots, m)$ be the unit tangent vectors at these points, the inflexion number defined by the point set can be computed by checking every pair of neighboring points and their tangents. Let $q = A_{i+1} - A_i$ and $h = (q_i \times q) \cdot (q \times q_{i+1})$, if $h < 0$ we consider that the original curve has one inflexion between A_i and A_{i+1} (Fig. 1(a)). If $h > 0$ there is no inflexion between these two points (Fig. 1(b)). If A_{i-1}, A_i, and A_{i+1} are collinear and q_i is parallel with this line too, we can ignore point A_i and check inflexion between A_{i-1} and A_{i+1} directly.

For a cubic Bézier curve, it may be convex, with one inflexion, two inflexions, a cusp, or a loop (Su and Liu, 1989). However, for most engineering applications, a cubic Bézier curve that has a cusp, or two inflexions, or a loop is usually forbidden (Li et al., 2004). In this paper we fit points by cubic Bézier curves with at most one inflexion and without cusp or loop. For the ease of implementation, the control polygon of a cubic Bézier curve should be convex or the turning direction of the control polygon shifts only once which yield a convex curve or a curve with one inflexion (Su and Liu, 1989). As a quadratic Bézier curve is definitely convex, we can construct a quadratic Bézier curve from the boundary data when the fitted cubic Bézier curve has unwanted inflexion and the two directed lines determined by $\{A_0, q_0\}$ and $\{A_m, -q_m\}$ intersect.

The inflexions for a Bézier spline curve are inflexions lying in each Bézier curve and inflexions implied by the joining points between adjacent Bézier curves. Assume the curvatures of two adjacent cubic Bézier curves $P_{j-1}(t)$ and $P_j(t)$ at their joint point are k_e and k_s, we deal the joint point as an inflexion when $k_e k_s < 0$. If $k_e k_s > 0$ the two Bézier curves are local convex at the joint point.

2.2. Inflexion control

To guarantee that the fitting B-spline is shape preserving in the end, the inflexion number of the cubic Bézier spline should not be larger than that defined by the corresponding points and tangent vectors.

Let $\{A_i^j\}$ $(i = 0, 1, \ldots, m)$ be a subset of the sampled data with I_j inflexions and $P_j(t)$ be the cubic Bézier curve fitting $\{A_i^j\}$ with K_j inflexions. Assume that $\{P_i(t)$ $(i = 0, \ldots, j-1)\}$ are obtained, whether $P_j(t)$ can be added to the cubic Bézier spline or not will be judged in the following way.

(1) $K_j > I_j$. This means that the inflexion number of $P_j(t)$ is larger than that of point set $\{A_i^j\}$ $(i = 0, 1, \ldots, m)$. Obviously, there exists unwanted inflexion within the curve $P_j(t)$ and the curve can not be accepted.

(2) $K_j \leqslant I_j$. At this moment, the inflexion number of the Bézier segment is not larger than that of the corresponding data. Whether the Bézier segment can be accepted or not is also determined by the convexity at the joint point between $P_{j-1}(t)$ and $P_j(t)$. If the two curves are convex at their joint point, $P_j(t)$ will be accepted. If the joint point is an inflexion, we check these two Bézier curves together. When the total number of inflexions within $P_{j-1}(t)$ and $P_j(t)$

and the inflexion at the joint is no larger than that of the sampled data, i.e., $K_{j-1} + K_j + 1 \leqslant I_{j-1} + I_j$, $\boldsymbol{P}_j(t)$ will be accepted. Otherwise, $\boldsymbol{P}_j(t)$ should not be accepted.

2.3. Bézier spline curve fitting algorithm

With the techniques of least square fitting of individual Bézier curve and inflexion control for cubic Bézier spline curve, we now present a fitting algorithm for Bézier spline curve which preserves shapes as well as a prescribed accuracy. For a given point set the Bézier curves can be constructed and checked sequently. If a fitting Bézier curve satisfies accuracy and inflexion requirements, it will be added to the sequence.

To fit a proper subset of given points by a Bézier curve with accuracy and inflexion requirements, there are always two ways to do that in practice. The first method is to test subsets of original point set by reducing point one by one, until a subset satisfying the mentioned requirements is reached. The second way for subset choosing is the bisection method (Park et al., 2000). The efficiency of these two methods depends on the number of points. One may choose the first method for point set with small point number and the second method for point set with large point number. The algorithm for G^1 continuous Bézier spline curve fitting is presented as follows.

Procedure 1. Data fitting by G^1 continuous cubic Bézier spline curve.
Input: A set of points and tangents $\{\boldsymbol{A}_i, \boldsymbol{q}_i\}$ $(i = 0, 1, \ldots, n)$, a tolerance ε_1.
Output: a G^1 continuous cubic Bézier spline curve $\boldsymbol{P}_j(t)$, $j = 0, 1, \ldots$.

Step 1: Fit a Bézier curve $\boldsymbol{P}(t)$ to first $m + 1$ points from the current data set.
Step 2:
  Step 2.1: If the fitting error for $\boldsymbol{P}(t)$ is larger than ε_1, decrease m and go to Step 1.
  Step 2.2: If $\boldsymbol{P}(t)$ has unwanted inflexion, go to Step 3.
  Step 2.3: Let $\boldsymbol{P}_j(t) = \boldsymbol{P}(t)$, go to Step 4.
Step 3:
  Step 3.1: If two directed lines determined by $\{\boldsymbol{A}_0, \boldsymbol{q}_0\}$ and $\{\boldsymbol{A}_m, -q_m\}$ intersect, construct a quadratic Bézier curve $\bar{\boldsymbol{P}}(t)$, else decrease m and go to Step 1.
  Step 3.2: If the fitting error for $\bar{\boldsymbol{P}}(t)$ is larger than ε_1, decrease m and go to Step 1; otherwise, elevate the degree of $\bar{\boldsymbol{P}}(t)$ to a cubic Bézier curve $\boldsymbol{P}(t)$ and let $\boldsymbol{P}_j(t) = \boldsymbol{P}(t)$.
Step 4: Let $j = j + 1$, delete the first m points and tangent vectors from the current data.
  If the rest data set has more than one point, go to Step 1.

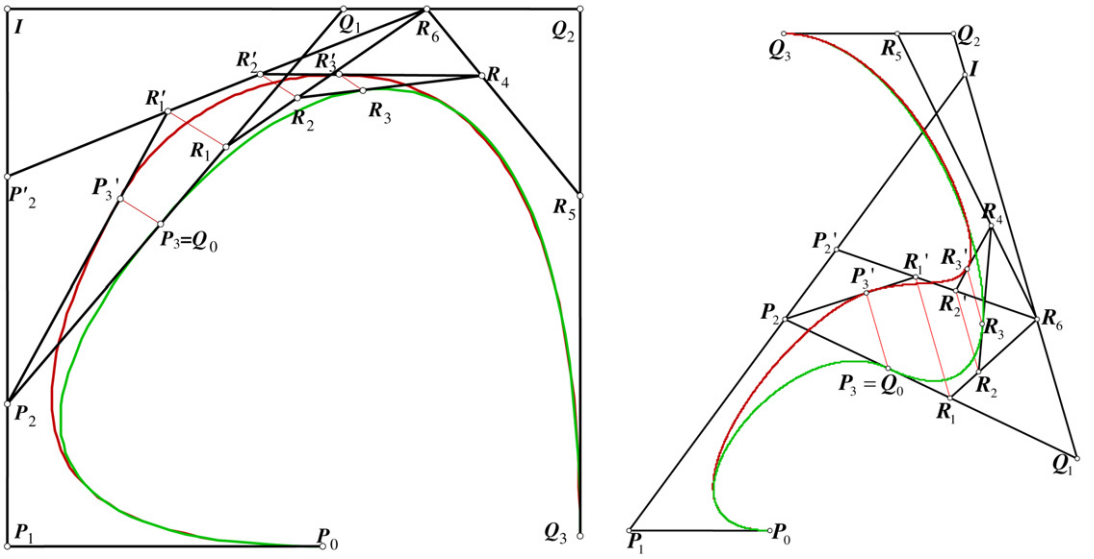
3. Merge to cubic B-spline curves

Though a G^1 continuous cubic Bézier spline curve can be represented as a cubic B-spline curve with multiple knots, B-spline with singular interior knots own higher order of continuity and are preferred for most practical applications. In this section we introduce a new method to merge G^1 continuous cubic Bézier curves to a C^2 continuous cubic B-spline curve. The merged curve is guaranteed to be within prescribed error bound ε_2 and no more inflexions would occur during the merging. We first show how to merge two G^1 continuous cubic Bézier curves to a C^2 continuous cubic B-spline curve. After that, we give formulae for merging a cubic B-spline curve and cubic Bézier curve which are G^1 continuous to a new cubic B-spline curve.

3.1. Merge two cubic Bézier curves

Let cubic Bézier curves $\boldsymbol{P}(t) = \sum_{i=0}^{3} B_i^3(t)\boldsymbol{P}_i$ $(0 \leqslant t \leqslant 1)$ and $\boldsymbol{Q}(t) = \sum_{i=0}^{3} B_i^3(t)\boldsymbol{Q}_i$ $(0 \leqslant t \leqslant 1)$ are G^1 continuous at point $\boldsymbol{P}_3 = \boldsymbol{Q}_0$, we first construct C^2 continuous Bézier curves with end derivatives $\boldsymbol{P}'(0)$, $\boldsymbol{P}''(0)$, $\boldsymbol{Q}'(1)$ and $\boldsymbol{Q}''(1)$ unchanged. This makes the merging a local algorithm and merging of a set of Bézier or B-spline curves can be implemented sequently. Then a B-spline curve will be constructed from the set of Bézier curves.

To allow enough degrees of freedom for achieving C^2 continuity, we subdivide $\boldsymbol{Q}(t)$ into two sub-Bézier curves $\boldsymbol{Q}_1(t)$, $\boldsymbol{Q}_2(t)$ with parameter $\lambda(0 < \lambda < 1)$. The control points for these two new Bézier curves are $\boldsymbol{P}_3$, $\boldsymbol{R}_1$, $\boldsymbol{R}_2$, $\boldsymbol{R}_3$ and $\boldsymbol{R}_3$, $\boldsymbol{R}_4$, $\boldsymbol{R}_5$, $\boldsymbol{Q}_3$, respectively (see Fig. 2). We have



(a) (b)

Fig. 2. Merge two G^1 continuous cubic Bézier curves to a cubic B-spline curve: (a) the joint point is local convex and (b) the joint point is dealt as an inflexion.

$$\boldsymbol{R}_1 = (1-\lambda)\,\boldsymbol{Q}_0 + \lambda\,\boldsymbol{Q}_1 \tag{1}$$

$$\boldsymbol{R}_2 = (1-\lambda)^2\,\boldsymbol{Q}_0 + 2(1-\lambda)\lambda\,\boldsymbol{Q}_1 + \lambda^2\,\boldsymbol{Q}_2 \tag{2}$$

$$\boldsymbol{R}_3 = (1-\lambda)^3\,\boldsymbol{Q}_0 + 3(1-\lambda)^2\lambda\,\boldsymbol{Q}_1 + 3(1-\lambda)\lambda^2\,\boldsymbol{Q}_2 + \lambda^3\,\boldsymbol{Q}_3 \tag{3}$$

$$\boldsymbol{R}_4 = (1-\lambda)^2\,\boldsymbol{Q}_1 + 2(1-\lambda)\lambda\,\boldsymbol{Q}_2 + \lambda^2\,\boldsymbol{Q}_3 \tag{4}$$

$$\boldsymbol{R}_5 = (1-\lambda)\,\boldsymbol{Q}_2 + \lambda\,\boldsymbol{Q}_3 \tag{5}$$

Let $u = \frac{\|\boldsymbol{P}_3\boldsymbol{Q}_1\|}{\|\boldsymbol{P}_2\boldsymbol{P}_3\|}$, due to the condition of G^1 continuity between $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$, we have

$$\boldsymbol{Q}_1 = (1+u)\boldsymbol{P}_3 - u\boldsymbol{P}_2 \tag{6}$$

To achieve C^2 continuity between $\boldsymbol{P}(t)$, $\boldsymbol{Q}_1(t)$ and $\boldsymbol{Q}_2(t)$, we perturb control points $\boldsymbol{P}_3, \boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}_3$ with $\boldsymbol{\delta}_0, \boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \boldsymbol{\delta}_3$, respectively. Assume that new positions for these points are $\boldsymbol{P}'_3, \boldsymbol{R}'_1, \boldsymbol{R}'_2, \boldsymbol{R}'_3$, we obtain three new Bézier curves: $\boldsymbol{P}'(t)$ (with control points $\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}'_3$), $\boldsymbol{Q}'_1(t)$ (with control points $\boldsymbol{P}'_3, \boldsymbol{R}'_1, \boldsymbol{R}'_2, \boldsymbol{R}'_3$) and $\boldsymbol{Q}'_2(t)$ (with control points $\boldsymbol{R}'_3, \boldsymbol{R}_4, \boldsymbol{R}_5, \boldsymbol{Q}_3$). The C^2 continuity condition requires that:

$$\boldsymbol{R}'_1 = (1+\lambda u)\boldsymbol{P}'_3 - \lambda u\boldsymbol{P}_2 \tag{7}$$

$$\boldsymbol{R}'_2 = (1+\lambda u)^2\boldsymbol{P}'_3 - 2(1+\lambda u)\lambda u\boldsymbol{P}_2 + (\lambda u)^2\boldsymbol{P}_1 \tag{8}$$

$$\boldsymbol{R}_4 = \frac{1}{\lambda}\boldsymbol{R}'_3 + \left(1-\frac{1}{\lambda}\right)\boldsymbol{R}'_2 \tag{9}$$

$$\boldsymbol{R}_5 = \left(\frac{1}{\lambda}\right)^2\boldsymbol{R}'_3 + 2\frac{1}{\lambda}\left(1-\frac{1}{\lambda}\right)\boldsymbol{R}'_2 + \left(1-\frac{1}{\lambda}\right)^2\boldsymbol{R}'_1 \tag{10}$$

Solving Eqs. (7)–(10) we have:

$$\boldsymbol{\delta}_0 = \frac{\lambda[u^2(2\boldsymbol{P}_2 - \boldsymbol{P}_1 - \boldsymbol{P}_3) + (\boldsymbol{Q}_2 + \boldsymbol{Q}_0 - 2\boldsymbol{Q}_1)]}{(1+u)(1+\lambda u)} \tag{11}$$

$$\boldsymbol{\delta}_1 = \frac{\lambda[u^2(2\boldsymbol{P}_2 - \boldsymbol{P}_1 - \boldsymbol{P}_3) + (\boldsymbol{Q}_2 + \boldsymbol{Q}_0 - 2\boldsymbol{Q}_1)]}{(1+u)} \tag{12}$$

$$\boldsymbol{\delta}_2 = \frac{(1-\lambda)\lambda[u^2(2\boldsymbol{P}_2 - \boldsymbol{P}_1 - \boldsymbol{P}_3) + (\boldsymbol{Q}_2 + \boldsymbol{Q}_0 - 2\boldsymbol{Q}_1)]}{(1+u)} \tag{13}$$

$$\boldsymbol{\delta}_3 = \frac{(1-\lambda)^2\lambda[u^2(2\boldsymbol{P}_2 - \boldsymbol{P}_1 - \boldsymbol{P}_3) + (\boldsymbol{Q}_2 + \boldsymbol{Q}_0 - 2\boldsymbol{Q}_1)]}{(1+u)} \tag{14}$$

By the knot removing algorithm (Piegl and Tiller, 1997) we can rewrite $\boldsymbol{P}'(t)$, $\boldsymbol{Q}'_1(t)$, $\boldsymbol{Q}'_2(t)$ into a cubic B-spline curve

$$\tilde{\boldsymbol{C}}(t) = \sum_{i=0}^{5} \tilde{\boldsymbol{C}}_i N_{i,4}(t) \tag{15}$$

where the knot vector is $\{0, 0, 0, 0, 1, 1+\lambda u, 1+u, 1+u, 1+u, 1+u\}$. The control points $\tilde{C}_i$ $(i = 0, 1, \ldots, 5)$ are $\boldsymbol{P}_0$, $\boldsymbol{P}_1$, $\boldsymbol{P}'_2$, $\boldsymbol{R}_6$, $\boldsymbol{R}_5$, $\boldsymbol{Q}_3$ as in Fig. 2, where $\boldsymbol{P}'_2 = (1+\lambda u)\boldsymbol{P}_2 - \lambda u \boldsymbol{P}_1$, $\boldsymbol{R}_6 = (1-\lambda)\boldsymbol{Q}_1 + \lambda \boldsymbol{Q}_2$.

3.2. Error and inflexion control

In this subsection we show how to control the fitting error and inflexion number by choosing a proper subdivision parameter for Bézier curves merging.

From Eqs. (11)–(14) it is easily derived that $\|\boldsymbol{\delta}_1\|$ is the maximum value among $\|\boldsymbol{\delta}_0\|$, $\|\boldsymbol{\delta}_1\|$, $\|\boldsymbol{\delta}_2\|$, $\|\boldsymbol{\delta}_3\|$. By Eq. (12), $\|\boldsymbol{\delta}_1\| \leqslant \varepsilon_2$ is equal to:

$$\lambda \leqslant \lambda_1 = \frac{(1+u)\varepsilon_2}{\|u^2(2\boldsymbol{P}_2 - \boldsymbol{P}_1 - \boldsymbol{P}_3) + (\boldsymbol{Q}_2 + \boldsymbol{Q}_0 - 2\boldsymbol{Q}_1)\|} \tag{16}$$

When we choose λ satisfying Eq. (16), the distance between $\tilde{\boldsymbol{C}}(t)$ and $\boldsymbol{P}(t)$ or $\boldsymbol{Q}(t)$ is guaranteed to be less than ε_2.

To control the inflexion number for the B-spline curve merged by $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$, we should compute another bound for the subdivision parameter λ such that the control polygon of $\tilde{\boldsymbol{C}}(t)$ owns the same turning number as the total turning number of control polygons of $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$. We compute the bound according to the case whether the original two joining curves are convex or inflection at the joint point.

(1) $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$ are convex at $\boldsymbol{P}_3 = \boldsymbol{Q}_0$. In this case the polygon $\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3\boldsymbol{Q}_1\boldsymbol{Q}_2$ is convex. If the two directed lines $\boldsymbol{P}_1\boldsymbol{P}_2$ and $\boldsymbol{Q}_2\boldsymbol{Q}_1$ intersect, namely at $\boldsymbol{I}$ (see Fig. 2(a)). For $0 < \lambda \leqslant 1$ and $\lambda \leqslant \lambda_2 = \frac{\|\boldsymbol{I}\boldsymbol{P}_1\| - \|\boldsymbol{P}_1\boldsymbol{P}_2\|}{u\|\boldsymbol{P}_1\boldsymbol{P}_2\|}$, and because $\boldsymbol{P}'_2 = (1+\lambda u)\boldsymbol{P}_2 - \lambda u \boldsymbol{P}_1$, $\boldsymbol{P}'_2$ lies between $\boldsymbol{P}_2$ and $\boldsymbol{I}$. If the two directed lines $\boldsymbol{P}_1\boldsymbol{P}_2$ and $\boldsymbol{Q}_2\boldsymbol{Q}_1$ do not intersect, we set $\lambda_2 = 1$. Then the polygon $\boldsymbol{P}_0\boldsymbol{P}_1\boldsymbol{P}'_2\boldsymbol{R}_6\boldsymbol{R}_5\boldsymbol{Q}_3$ owns the same turning number as the polygon $\boldsymbol{P}_0\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3\boldsymbol{Q}_1\boldsymbol{Q}_2\boldsymbol{Q}_3$. Due to the variation diminishing property of B-spline, the inflexion number of $\tilde{\boldsymbol{C}}(t)$ is equal to the sum of inflexion numbers of $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$.

(2) $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$ are inflectional at $\boldsymbol{P}_3 = \boldsymbol{Q}_0$. In this case the turning direction of the polygon $\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3\boldsymbol{Q}_1\boldsymbol{Q}_2$ changes one time (see Fig. 2(b)). Let $\boldsymbol{I}$ be the intersection point of directed lines $\boldsymbol{P}_1\boldsymbol{P}_2$ and $\boldsymbol{Q}_1\boldsymbol{Q}_2$, then $\boldsymbol{R}_6$ lies between $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$ when $0 < \lambda \leqslant 1$ and $\boldsymbol{R}_6$ lies between $\boldsymbol{Q}_1, \boldsymbol{I}$ when $\lambda \leqslant \lambda_2 = \frac{\|\boldsymbol{I}\boldsymbol{Q}_1\|}{\|\boldsymbol{Q}_1\boldsymbol{Q}_2\|}$. If directed lines $\boldsymbol{P}_1\boldsymbol{P}_2$ and $\boldsymbol{Q}_1\boldsymbol{Q}_2$ do not intersect, we also set $\lambda_2 = 1$. For $\lambda \leqslant \min(1, \lambda_2)$, the polygon $\boldsymbol{P}_0\boldsymbol{P}_1\boldsymbol{P}'_2\boldsymbol{R}_6\boldsymbol{R}_5\boldsymbol{Q}_3$ has the equal turning number as the polygon $\boldsymbol{P}_0\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3\boldsymbol{Q}_1\boldsymbol{Q}_2\boldsymbol{Q}_3$. Then the inflexion number of $\tilde{\boldsymbol{C}}(t)$ is equal to that of the Bézier spline composing of $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$.

To achieve fitting accuracy and inflexion control simultaneously, we choose $0 < \lambda \leqslant \lambda_m$, where $\lambda_m = \min(\lambda_1, \lambda_2, 1)$. If $\lambda_m = 1$ and $\lambda = \lambda_m$, $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$ can even be merged with no subdivision of $\boldsymbol{Q}(t)$. In our experiments we choose $\lambda = 0.5$ when $0.5 < \lambda_m \leqslant 1$ for uniform subdivision purpose.

Remark. Though the subdivision parameter λ can be computed explicitly with the bounds λ_1 and λ_2 evaluated above, one of the Bézier segments obtained by subdivision of $Q(t)$ is tiny when λ is a small value. In fact, $\|\delta_1\| \leqslant \varepsilon_2$ is a conservative estimation for curve deviation and then λ_1 derived by Eq. (16) is also a conservative bound for the choice of λ. We find that λ_1 is round about 0.05 in most practical cases. To make as uniform as possible Bézier segments, one may increase the tolerance ε_2 or subdivide the curve $Q(t)$ to more sub-segments. We propose here a numerical optimization procedure to pick λ according to the criteria that the curve $Q(t)$ is subdivided into two sub-curves and the fitting error is computed directly from the original data to the perturbed fitting curves. Let $\lambda_{\min} = \min\{\lambda_1, \lambda_2\}$, $\lambda_{\max} = \min\{1, \lambda_2\}$, we adopt the bisection method to determine λ:

Procedure 2. Numerical optimization of λ.
Input: cubic Bézier curves $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$, a set of points $\boldsymbol{A}_i$ $(i = 0, 1, \ldots, i_n)$ fitted by $\boldsymbol{P}(t)$, $\boldsymbol{Q}(t)$, tolerance $\varepsilon, \varepsilon_1$.
Output: the subdivision parameter λ for $\boldsymbol{Q}(t)$.

Step 1: Set $\lambda = 0.5(\lambda_{\max} + \lambda_{\min})$.
Step 2: Merge $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$ to a cubic B-spline curve $\tilde{\boldsymbol{P}}(t)$ by subdividing $\boldsymbol{Q}(t)$ with parameter λ.
Step 3: If the distance from each point $\boldsymbol{A}_i$ to the perturbed curve $\boldsymbol{P}'(t)$ or $\boldsymbol{Q}'_1(t)$ is less than ε or to curve $\boldsymbol{Q}'_2(t)$ is less than ε_1, set $\lambda_{\min} = \lambda$; else set $\lambda_{\max} = \lambda$.
Step 4: If $(\lambda_{\max} - \lambda_{\min}) < 0.001$, set $\lambda = \lambda_{\min}$ and output it; else go to Step 1.

3.3. Merge a cubic B-spline and a cubic Bézier curve

To merge a cubic B-spline curve and a cubic Bézier curve which are G^1 continuous at the joint point, we first convert the last segment of the B-spline curve into a Bézier curve by knot insertion. Then the two cubic Bézier curves are merged to a B-spline curve using the method introduced in Sections 3.1 and 3.2. Because the first and second order derivatives of the derived Bézier curve at its start point do not change, the two B-spline curves can be exactly merged to a new B-spline curve by knot removing algorithm.

Let cubic B-spline curve $\boldsymbol{P}(t) = \sum_{i=0}^{n} \boldsymbol{P}_i N_{i,4}(t)$ (knot vector $T = \{0, 0, 0, 0, t_4, \ldots, t_{n-1}, t_n, 1, 1, 1, 1\}$) and cubic Bézier curve $\boldsymbol{Q}(t) = \sum_{i=0}^{3} B_i^3(t)\boldsymbol{Q}_i$ $(0 \leqslant t \leqslant 1)$ are G^1 continuous at $\boldsymbol{Q}_0 = \boldsymbol{P}_n$. By inserting t_n two times we convert $\boldsymbol{P}(t)(t \in [t_n, 1])$ to a cubic Bézier curve:

$$\hat{\boldsymbol{P}}(s) = B_0^3(s)\boldsymbol{P}''_{n-2} + B_1^3(s)\boldsymbol{P}'_{n-1} + B_2^3(s)\boldsymbol{P}_{n-1} + B_3^3(s)\boldsymbol{P}_n$$

where

$$\begin{aligned}
\boldsymbol{P}''_{n-2} &= \frac{1-t_n}{1-t_{n-1}}\boldsymbol{P}'_{n-2} + \frac{t_n - t_{n-1}}{1-t_{n-1}}\boldsymbol{P}'_{n-1},\\
\boldsymbol{P}'_{n-1} &= \frac{1-t_n}{1-t_{n-1}}\boldsymbol{P}_{n-2} + \frac{t_n - t_{n-1}}{1-t_{n-1}}\boldsymbol{P}_{n-1},\\
\boldsymbol{P}'_{n-2} &= \frac{1-t_n}{1-t_{n-2}}\boldsymbol{P}_{n-3} + \frac{t_n - t_{n-2}}{1-t_{n-2}}\boldsymbol{P}_{n-2}.
\end{aligned}$$

After the merging of $\hat{\boldsymbol{P}}(s)$ and $\boldsymbol{Q}(t)$, we merge $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$ to a new cubic B-spline curve

$$\tilde{\boldsymbol{P}}(t) = \sum_{i=0}^{n+2} \tilde{\boldsymbol{P}}_i N_{i,4}(t).$$

The control points of $\tilde{\boldsymbol{P}}(t)$ are as follows: $\tilde{\boldsymbol{P}}_i = \boldsymbol{P}_i$ $(i = 0, 1, 2, \ldots, n-2)$, $\tilde{\boldsymbol{P}}_{n-1} = (1+\lambda u)\boldsymbol{P}_{n-1} - \lambda u \boldsymbol{P}'_{n-1}$, $\tilde{\boldsymbol{P}}_n = (1-\lambda)\boldsymbol{Q}_1 + \lambda \boldsymbol{Q}_2$, $\tilde{\boldsymbol{P}}_{n+1} = (1-\lambda)\boldsymbol{Q}_2 + \lambda \boldsymbol{Q}_3$, $\tilde{\boldsymbol{P}}_{n+2} = \boldsymbol{Q}_3$. The parameters λ and u are defined and computed as in Section 3.1 and Section 3.2. Let $a = 1 + u(1-t_n)$, the knot vector for the final B-spline curve is $T = \{0, 0, 0, 0, t_4/a, \ldots, t_{n-1}/a, t_n/a, 1/a, (1+\lambda u(1-t_n))/a, 1, 1, 1, 1\}$.

4. Convert planar curves to B-splines

With the techniques of Bézier curve fitting and B-spline curve merging introduced above, we can convert any type of smooth planar curves to B-splines. The conversion is consisting of two main parts, sampling points and tangents on the original curve and fitting the sampled data with prescribed tolerance and known inflexion number.

4.1. Sampling points and tangents

Sampling points and tangent vectors on a given smooth curve has been studied extensively in the literature, and various methods have been proposed with wide applications. In this paper we focus that the connected polygon approximates the original curve with high accuracy and inflexions on the sampled curve can be detected explicitly. So we just mention some existing methods here for our purpose.

To convert a parametric curve within prescribed accuracy the parameter steps should be evaluated (Filip et al., 1986). Park (2004) has also proposed a detailed algorithm for approximating a smooth curve by a polygon within a given tolerance. For the purpose of high accuracy approximation and inflexion checking points as well as sample tangents should be sampled (Yang, 2002). For a shape preserving subdivision curve, see for example (Yang, 2006), the sampled points can be simply picked as on the subdivided polygon after certain subdivision steps. The tangent vector $\boldsymbol{q}_i$ at vertex $\boldsymbol{A}_i$ can be set as the bisector of the turning angle at vertex $\boldsymbol{A}_i$. In a same way as sampling data on parametric curves, the error between the sampled polygon and the limit curve can be estimated explicitly. As to implicit or algebraic curve, the method introduced by Hartmann (2000) will be adopted. We note that the sampling can be with any high density and accuracy, and we ignore the sampling error for the following fitting steps.

4.2. The conversion algorithm

As discussed above, the conversion algorithm consists three main steps: sampling data from the given curve, fitting the sampled data with a G^1 continuous cubic Bézier spline curve, and merging the Bézier spline into a B-spline curve. The whole conversion algorithm is as follows.

Procedure 3. Converting a given curve to a cubic B-spline.
Input: a given curve $\boldsymbol{S}(t)$, tolerance ε.
Output: a cubic B-spline curve $\tilde{\boldsymbol{P}}(t)$.

Step 1: Sample points and tangents $\{\boldsymbol{A}_i, \boldsymbol{q}_i\}$ $(i = 0, 1, \ldots, n)$ from $\boldsymbol{S}(t)$.
Step 2: Set tolerances $\varepsilon_1, \varepsilon_2$ which satisfy $\varepsilon = \varepsilon_1 + \varepsilon_2$ and $\varepsilon_1 > \varepsilon_2$.
Step 3: Fit cubic Bézier curves $\{\boldsymbol{P}_j(t)\}$ within ε_1 to $\{\boldsymbol{A}_i\}$ by Procedure 1.
Step 4: Rewrite $\boldsymbol{P}_0(t)$ as a cubic B-spline curve $\boldsymbol{P}(t)$, merge $\boldsymbol{P}(t)$ and $\boldsymbol{P}_j(t)$ $(j = 1, 2, \ldots, k)$ within ε_2 in turn, and return the final cubic B-spline curve $\tilde{\boldsymbol{P}}(t)$.

Although the merging process can be implemented for any bound $\varepsilon_2 > 0$, we point out that the magnitude of ε_2 affects the quality of the resulting curve. In our experiments, we set $\varepsilon_1 = \frac{3}{4}\varepsilon, \varepsilon_2 = \frac{1}{4}\varepsilon$ to reduce the number of control points and ensure the quality of the resulting B-spline curve.

5. Extend the algorithm to arbitrary degree B-spline curves

In a similar way as converting planar curves to cubic B-splines, a smooth plane curve can also be converted to k $(k > 3)$-degree B-spline curve with C^2 or higher order continuity at interior knots. There are two main steps for the conversion algorithm: one is fitting the sampled data by a G^1 continuous k-degree Bézier spline curve, the other is merging the G^1 continuous Bézier spline curve to a C^{k-1} continuous k-degree B-spline curve.

To fit a k-degree Bézier curve with fixed boundary point and tangent vectors to a sequence of points, one can fit the data by the least squares fitting method. Inflexion checking for the k-degree Bézier curve is more complicated than for the cubic Bézier curve. But based on the variation diminishing property of Bézier curves, one can control the inflexion number of the fitting Bézier spline curve by its control polygon explicitly.

To merge two C^m $(0 < m < k - 1)$ continuous k-degree Bézier curves $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t)$ to a C^{m+1} continuous k-degree B-spline curve $\tilde{\boldsymbol{C}}(t)$, one should subdivide $\boldsymbol{Q}(t)$ into two sub-Bézier curves $\boldsymbol{Q}_1(t)$, $\boldsymbol{Q}_2(t)$ with a parameter λ $(0 < \lambda < 1)$. Similar to Section 3.1, $k + 1$ control points are perturbed such that $\boldsymbol{P}'(t)$ and $\boldsymbol{Q}'_1(t)$ are C^{m+1} continuous and $\boldsymbol{Q}'_1(t)$ and $\boldsymbol{Q}'_2(t)$ are C^{k-1} continuous at their joint points, respectively. From the continuity conditions $k + 1$ equations are established and new control points are obtained as the solutions to the equations. Repeat this process $k - 2$ times, we can merge two G^1 continuous k-degree Bézier curves to a C^{k-1} continuous k-degree B-spline curve with $k - 2$ interior knots.

As C^2 continuity works for most practical cases, we present here the result of conversion and merging method for $k(k > 3)$-degree B-spline curve with C^2 continuity. We omit here the details and present the main result for merging a k-degree B-spline curve and k-degree Bézier curve which are G^1 continuous into a new k-degree B-spline with C^2 continuity.

Assume that B-spline curve $\boldsymbol{P}(t) = \sum_{i=0}^{n} \boldsymbol{P}_i N_{i,k+1}(t)$ (knots vector $T = \{\underbrace{0, \ldots, 0}_{k+1}, t_{k+1}, \ldots, t_{n-1}, t_n, \underbrace{1, \ldots, 1}_{k+1}\}$) and Bézier curve $\boldsymbol{Q}(t) = \sum_{i=0}^{k} B_i^k(t)\boldsymbol{Q}_i (0 \leqslant t \leqslant 1)$ are G^1 continuous at $\boldsymbol{Q}_0 = \boldsymbol{P}_n$, they can be merged to a C^2 continuous k-degree B-spline curve $\tilde{\boldsymbol{P}}(t) = \sum_{i=0}^{n+k-1} \tilde{\boldsymbol{P}}_i N_{i,k+1}(t)$. The control points of $\tilde{\boldsymbol{P}}(t)$ are as follows: $\tilde{\boldsymbol{P}}_i = \boldsymbol{P}_i$ $(i = 0, 1, \ldots, n-2)$, $\tilde{\boldsymbol{P}}_{n-1} = (1+\lambda u)\boldsymbol{P}_{n-1} - \lambda u \boldsymbol{P}'_{n-1}$, $\tilde{\boldsymbol{P}}_{n+j} = (1-\lambda)\boldsymbol{Q}_{j+1} + \lambda \boldsymbol{Q}_{j+2}$ $(j = 0, 1, \ldots, k-2)$, $\tilde{\boldsymbol{P}}_{n+k-1} = \boldsymbol{Q}_k$. The parameters λ, u are defined as those in Section 3 and $\boldsymbol{P}'_{n-1} = \frac{1-t_n}{1-t_{n-1}}\boldsymbol{P}_{n-2} + \frac{t_n - t_{n-1}}{1-t_{n-1}}\boldsymbol{P}_{n-1}$. Let $a = 1 + u(1-t_n)$, the knot vector for the final B-spline curve is

$$T = \{\underbrace{0, \ldots, 0}_{k+1}, t_{k+1}/a, \ldots, t_{n-1}/a, t_n/a, \underbrace{1/a, \ldots, 1/a}_{k-2}, (1+\lambda u(1-t_n))/a, \underbrace{1, \ldots, 1}_{k+1}\}.$$

6. Experimental results

We have tested the local fitting algorithm for many examples and we show two examples here to illustrate the efficiency of the algorithm.

We also compare our approach with traditional least square fitting technique. When fitting a point set in a least squares sense, the parameter corresponding to every point and the knot vector for the B-spline curve are computed by the method introduced in Piegl and Tiller (1997). The number of control points starts with 4 and increases by 1 iteratively until the error bound is satisfied.

In Example 1, we convert a functional curve $\boldsymbol{C}_1 : f(t) = t(2-t) + 0.2\sin(12t)$ $(0 \leqslant t \leqslant 1)$ to a cubic B-spline curve under various tolerances. Table 1 shows the numbers of control points and numbers of inflexions by different methods. The inflexion number of the B-spline curve by local fitting algorithm is equal to that of the original curve, but the B-spline curve by least square fitting may have more inflexions. Note that when tolerance $\varepsilon = 1 \times 10^{-3}$, the resulting cubic B-spline curve has 18 control points, but with tolerance $\varepsilon = 2 \times 10^{-3}$, the number is 20. This implies that local fitting algorithm can generate B-spline curves with fewer but not the least number of control points in general.

Figs. 3(a)–(c) are the graphical results of example 1 with $\varepsilon = 5 \times 10^{-3}$. The dotted blue line is the curve to be converted, and red real lines are cubic B-spline curves.[1] The points corresponding to knots are plotted as red "∘"s along the fitting curves. The knot vector of the B-spline curve shown in Fig. 3(a) (by local fitting algorithm) is {0, 0, 0, 0, 0.07682, 0.08072, 0.15223, 0.15295, 0.22342, 0.22419, 0.31391, 0.31591, 0.43699, 0.43903, 0.96924, 0.97157, 1, 1, 1, 1}. After numerical optimization the knot vector becomes {0, 0, 0, 0, 0.07657, 0.15171, 0.16495, 0.22265, 0.22527, 0.31281, 0.43547, 0.48072, 0.96631, 1, 1, 1, 1}. The knot vector of the B-spline curve shown in Fig. 3(c) (by least square fitting) is {0, 0, 0, 0, 0.12590, 0.21919, 0.26489, 0.29944, 0.33965, 0.37237, 0.42851, 0.52195, 0.62681, 0.70701, 0.74693, 0.79172, 0.85729, 0.91294, 0.94689, 1, 1, 1, 1}. To understand the quality of the resulting curves more clearly, we plot the curvature of the original curve and the resulting B-spline curves in Fig. 4.

In the second example, we convert a subdivision curve $\boldsymbol{C}_2$ as Fig. 7(a) in Yang (2006). The main results are shown in Table 2. Notice that the top part of curve $\boldsymbol{C}_2$ is flat, and the B-spline curves by least square technique have many undesired inflexions and unwanted undulations in that part. As a comparison, the approximating B-spline curves by local fitting algorithm keep the original shape successfully.

Table 1
Control points (CP) and inflexions for the B-spline curves approximating the functional curve $\boldsymbol{C}_1$

Tolerance	Local fitting			Least square	
	CP without optim.	CP with optim.	Inflexion	Control points	Inflexion
1×10^{-2}	12	12	3	15	3
5×10^{-3}	16	13	3	19	8
2×10^{-3}	20	17	3	25	7
1×10^{-3}	18	15	3	28	3

[1] For colors see the web version of this article.

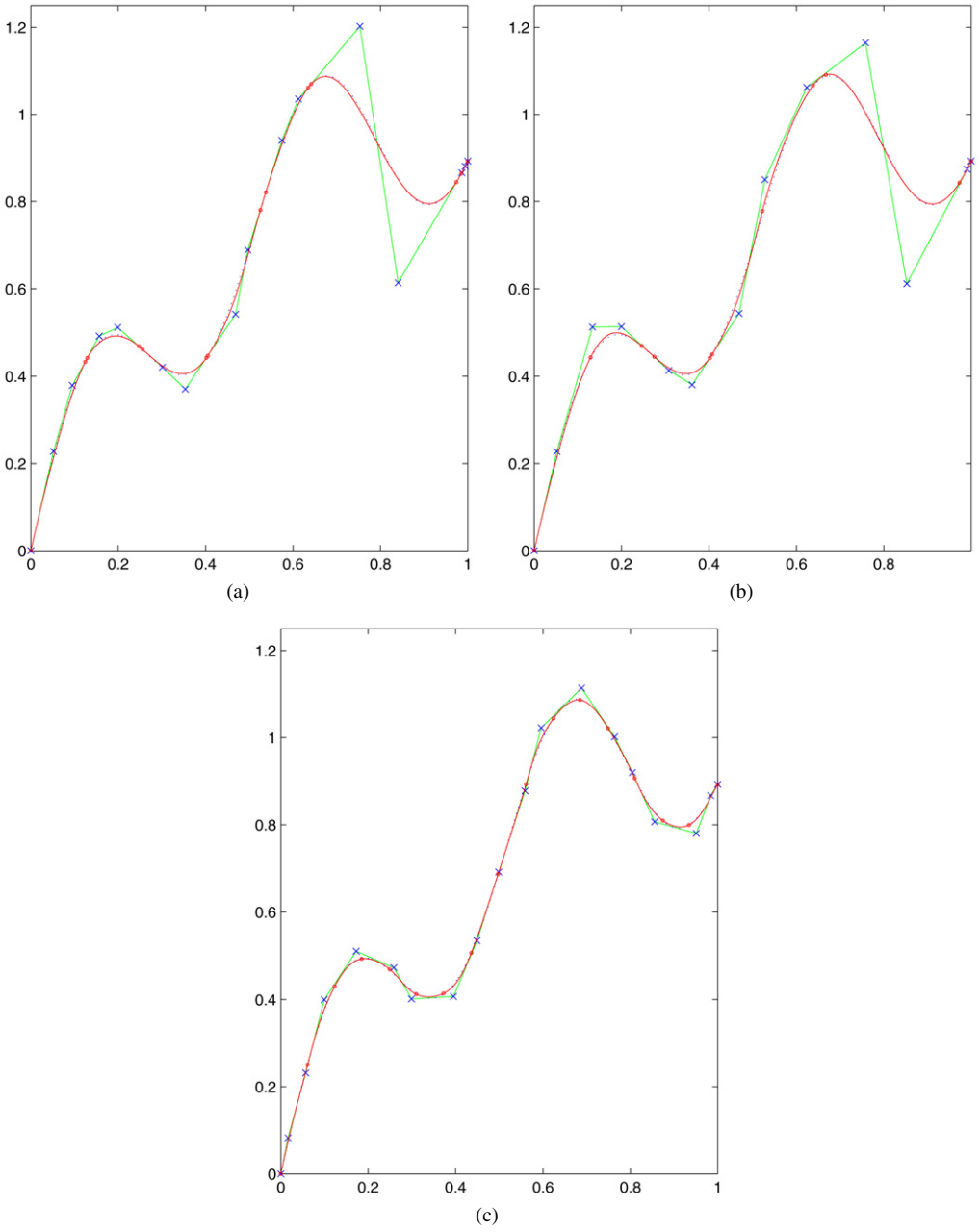


Fig. 3. Converting curve $\boldsymbol{C}_1$ to cubic B-spline curves: (a) by local fitting without numerical optimization of λ; (b) by local fitting with numerical optimization of λ; (c) by least square fitting.

Table 2
Control points and inflexions for the B-spline curves approximating the subdivision curve $\boldsymbol{C}_2$

Tolerance	Local fitting			Least square	
	CP without optim.	CP with optim.	Inflexion	Control points	Inflexion
4×10^{-2}	16	15	2	21	10
2×10^{-2}	18	17	2	27	8
1×10^{-2}	26	25	2	33	10
5×10^{-3}	30	28	2	39	6

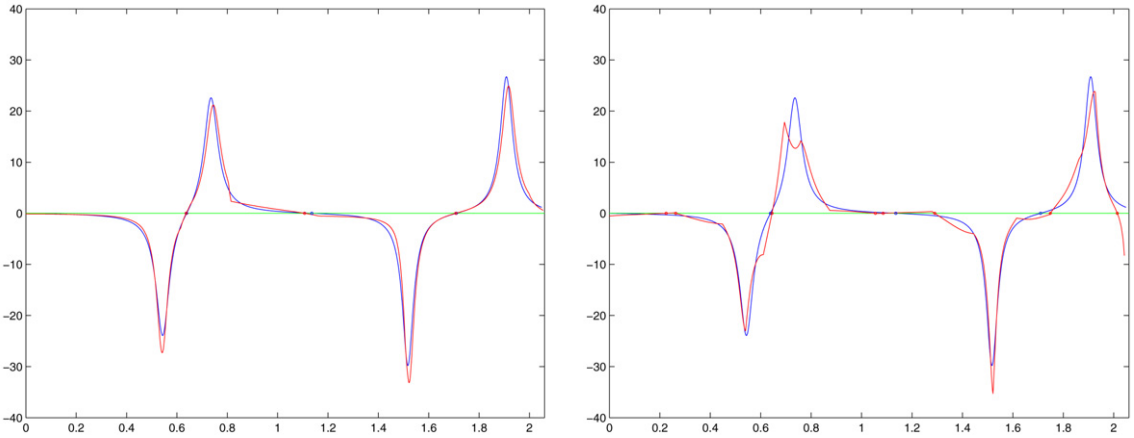


Fig. 4. Curvature plots along arc length for the original curve (blue) and the approximating B-spline curves (red): (a) by local fitting with optimized knots; (b) by least square fitting.

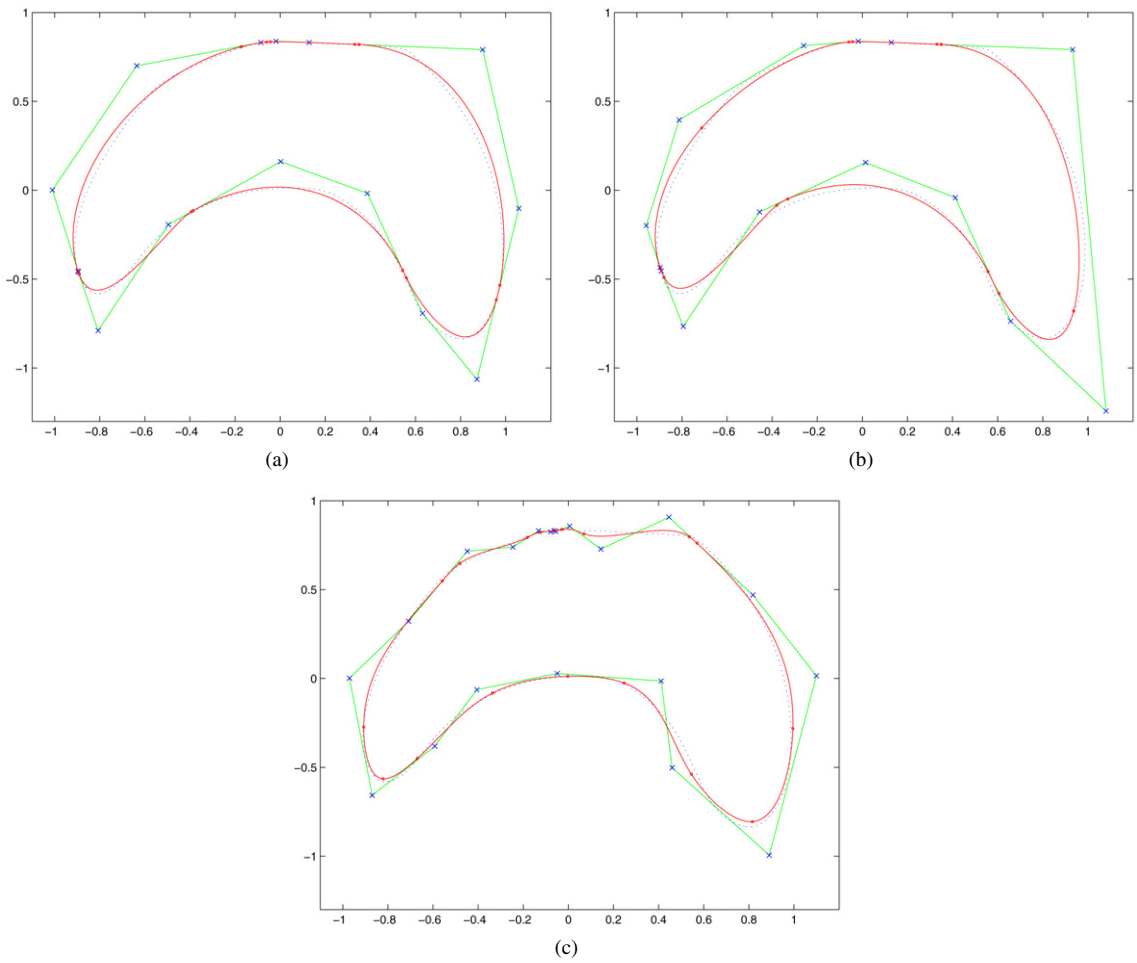


Fig. 5. Approximating a subdivision curve by B-spline: (a) by local fitting without numerical optimization of λ; (b) by local fitting with numerical optimization of λ, (c) by least square fitting method.

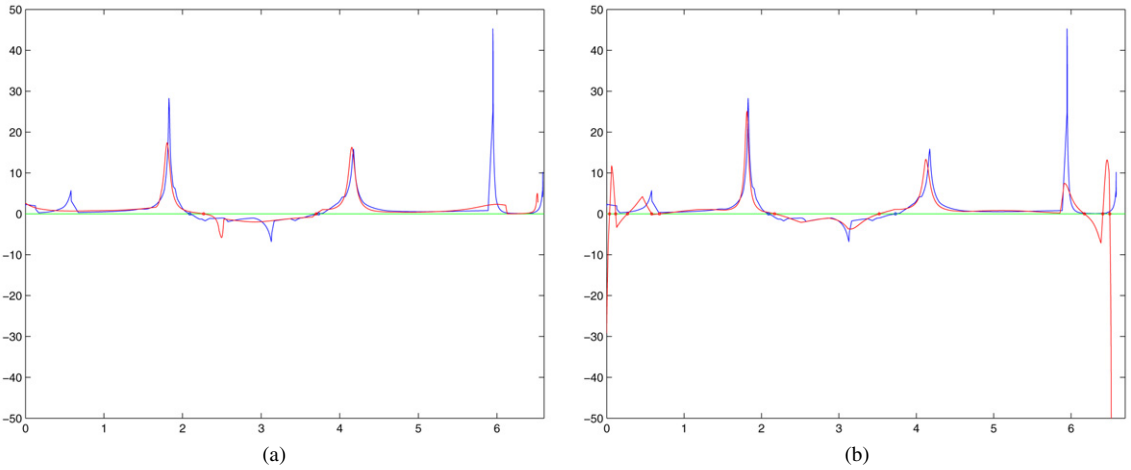


Fig. 6. Curvature plots for the subdivision curve (blue) and the approximating B-spline curves (red): (a) by local fitting with optimized knots; (b) by least square fitting.

Figs. 5(a)–(c) are the conversion results by our local fitting algorithm and the least square fitting algorithm, all with $\varepsilon = 4 \times 10^{-2}$. As the curve $\boldsymbol{C}_2$ is close, the first cubic Bézier segment is regarded as the last segment and it is merged twice. In Fig. 5(a) the knot vector of the B-spline curve by local fitting algorithm is {0, 0, 0, 0, 0.00162, 0.05761, 0.05906, 0.27076, 0.27215, 0.37887, 0.37981, 0.48370, 0.48510, 0.52202, 0.52491, 0.52781, 0.53657, 1, 1, 1, 1}. After numerical optimization the knot vector becomes {0, 0, 0, 0, 0.00352, 0.05762, 0.066000, 0.27076, 0.28943, 0.37888, 0.48370, 0.48485, 0.52203, 0.52781, 0.76391, 1, 1, 1, 1}. In Fig. 5(c) the knot vector of the B-spline curve by least square fitting is {0, 0, 0, 0, 0.00913, 0.019500, 0.06954, 0.08825, 0.22571, 0.27691, 0.31079, 0.38591, 0.43895, 0.47583, 0.56763, 0.63352, 0.72538, 0.90128, 0.90880, 0.97839, 0.99508, 1, 1, 1, 1}. The curvature plots for these curves are illustrated in Fig. 6.

From the examples presented above we can see that the main advantage of the proposed algorithm is that the fitting B-spline curve keeps the shape of the original curve very well. Moreover, the knots of the final B-spline curve are determined efficiently and the control points are always less than those obtained by least square fitting too. When we merge two G^1 curves to a B-spline with conservative subdivision parameter λ, it may arouse tiny Bézier segments and uneven knot intervals for the final B-spline curve. With simple optimization of the subdivision parameter, the regularity of knot intervals can be improved much and the curvature plots also show the quality of the final B-spline curves.

7. Conclusions and discussions

This paper has presented a local fitting algorithm for converting planar curves to B-splines. An original curve or the original point set is first approximated by a G^1 Bézier spline curve and then the Bézier curves are merged into a B-spline curve via subdivision and control points adjustment. The fitting accuracy and shape control are considered along with the fitting and merging processes. Another property of the local algorithm is that the control points and knots of the final B-spline curve are created automatically and adaptively. Though we deal with planar curve conversion in this paper, we believe that the method can be generalized for spacial curve conversion by combining the techniques of shape control for Bézier and B-spline curves in 3D space.

Acknowledgements

The authors owe thanks to referees for their detailed and helpful comments and suggestions which have helped to improve the paper greatly. This work is supported by NSFC (60673032, 60333010) and 973 program of China (2002CB312101).

References

Eck, M., Hadenfeld, J., 1995. Knot removal for B-spline curves. Computer Aided Geometric Design 12, 259–282.

Farin, G., 2002. Curves and Surfaces for CAGD, fifth ed. Morgan Kaufmann, San Francisco.

Filip, D., Magedson, R., Markot, R., 1986. Surface algorithms using bounds on derivatives. Computer Aided Geometric Design 3 (4), 295–311.

Hartmann, E., 2000. Numerical parametrization of curves and surfaces. Computer Aided Geometric Design 17 (3), 251–266.

Hoschek, J., 1987. Approximate conversion of spline curves. Computer Aided Geometric Design 4 (1–2), 59–66.

Hoschek, J., Lasser, D., 1993. Fundamentals of Computer Aided Geometric Design. A.K. Peters, London.

Hoschek, J., Wissel, N., 1988. Optimal approximate conversion of spline curves and spline approximation of offset curves. Computer-Aided Design 20 (8), 475–483.

Jüttler, B., 1997. Shape preserving least-squares approximation by polynomial parametric spline curves. Computer Aided Geometric Design 14 (8), 731–747.

Kosters, M., 1991. Curvature-dependent parametrization of curves and surfaces. Computer-Aided Design 23 (8), 569–578.

Li, W., Xu, S., Zheng, J., Zhao, G., 2004. Target curvature driven fairing algorithm for planar cubic B-spline curves. Computer Aided Geometric Design 21, 499–513.

Li, W., Xu, S., Zhao, G., Goh, L.P., 2005. Adaptive knot placement in B-spline curve approximation. Computer-Aided Design 37 (8), 791–797.

Lyche, T., Mørken, K., 1987. Knot removal for parametric B-spline curves and surfaces. Computer Aided Geometric Design 4, 217–230.

Lyche, T., Mørken, K., 1988. A data-reduction strategy for splines with applications to the approximation of functions and data. IMA J. Numer. Anal. 8, 185–208.

Ma, W.Y., Kruth, J.P., 1995. Parametrization of randomly measured points for least squares fitting of B-spline curves and surface. Computer-Aided Design 27 (9), 663–675.

Park, H., 2004. An error-bounded approximate method for representing planar curves in B-splines. Computer Aided Geometric Design 21 (5), 479–497.

Park, H., Kim, K., Lee, S.C., 2000. A method for approximate NURBS curve compatibility based on multiple curves refitting. Computer-Aided Design 32 (4), 237–252.

Park, H., Lee, J.H., 2007. B-spline curve fitting based on adaptive curve refinement using dominant points. Computer-Aided Design 39 (6), 439–451.

Patrikalakis, N.M., 1989. Approximate conversion of rational splines. Computer Aided Geometric Design 6 (2), 155–165.

Piegl, L., Tiller, W., 1997. The NURBS Book, second ed. Springer-Verlag, Berlin.

Razdan, A., 1999. Knot Placement for Piecewise Polynomial Approximation. Arizona State University, Tempe, AZ: http://3dk.asu.edu/archives/publication/publication.html.

Saux, E., Daniel, M., 1999. Data reduction of polygonal curves using B-splines. Computer-Aided Design 31 (8), 507–515.

Shao, L.J., Zhou, H., 1996. Curve fitting with Bézier cubics. Graphical Models and Image Processing 58 (3), 223–232.

Su, B., Liu, D., 1989. Computational Geometry-Curve and Surface Modeling. Academic Press, Boston.

Tai, C.L., Hu, S.M., Huang, Q.X., 2003. Approximate merging of B-spline curves via knot adjustment and constrained optimization. Computer-Aided Design 35 (10), 893–899.

Taubin, G., 1994. Rasterizing algebraic curves and surfaces. IEEE Computer Graphics and Applications 14 (2), 14–23.

Yoshimoto, F., Harada, T., 2003. Yoshimoto Y. Data fitting with a spline using a real-coded genetic algorithm. Computer-Aided Design 35 (8), 751–760.

Yang, H., Wang, W., Sun, J., 2004. Control point adjustment for B-spline curve approximation. Computer-Aided Design 36 (7), 639–652.

Yang, X., 2002. Efficient circular arc interpolation based on active tolerance control. Computer-Aided Design 34 (13), 1037–1046.

Yang, X., 2006. Normal based subdivision scheme for curve design. Computer Aided Geometric Design 23 (3), 243–260.