Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Aided Geometric Design 22 (2005) 183–197
COMPUTER AIDED GEOMETRIC DESIGN
www.elsevier.com/locate/cagd

Fast degree elevation and knot insertion for B-spline curves

Qi-Xing Huang [a], Shi-Min Hu [a,*], Ralph R. Martin [b]

[a] *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China*
[b] *School of Computer Science, Cardiff University, Cardiff, CF24 3AA, United Kingdom*

Received 30 April 2004; received in revised form 15 October 2004; accepted 10 November 2004

Available online 30 November 2004

Abstract

We give a new, simple algorithm for simultaneous degree elevation and knot insertion for B-spline curves. The method is based on the simple approach of computing derivatives using the control points, resampling the knot vector, and then computing the new control points from the derivatives. We compare our approach with previous algorithms and illustrate it with examples.

Keywords: B-splines; Degree elevation; Knot insertion

1. Introduction

Several methods have been given for degree elevation of B-spline curves (Cohen et al., 1985; Liu and Wayne, 1997; Prautzsch, 1984; Prautzsch and Piper, 1991; Pigel and Tiller, 1994), the fastest of which is Prautzsch and Piper's algorithm (Prautzsch and Piper, 1991). Unfortunately, their algorithm suffers from being complicated and hard to implement. On the other hand, Piegl and Tiller's (1994) algorithm is more straightforward, and easier to understand. It splits the B-spline curve into Bézier curve pieces, raises the degree of each piece, and then recombines the degree-elevated Bézier curves to produce the new B-spline curve. Liu's algorithm (Liu and Wayne, 1997) has the benefits of being both simple to implement and fast. It takes the approach of computing the new control points using a series of knot insertions followed by a series of knot deletions.

* Corresponding author.
E-mail address: shimin@tsinghua.edu.cn (S.-M. Hu).

0167-8396/$ – see front matter
doi:10.1016/j.cagd.2004.11.001

We give here a new fast, simple method for degree elevation of B-spline curves, which can also simultaneously insert knots. Our approach is based on the well known fact that a polynomial curve is uniquely determined by its value at a given point together with the values of all its derivatives at that point. This observation can be generalised to B-splines which are piecewise polynomials, and hence B-spline curves: each curve segment over the knot interval $[t_i, t_{i+1}]$ is determined by the value of the curve and its derivatives at the knot t_i. We have already used this observation to devise a knot adjustment algorithm for B-splines (Tai et al., 2003).

The derivatives of a B-spline can be computed efficiently using a variety of approaches (Ferrari et al., 1994; Wang et al., 1996). It has been shown that using the inverse approach leads to a knot refinement algorithm (Sankar et al., 1994) which is significantly faster than the usual Oslo algorithm (Cohen et al., 1980).

In a similar vein to (Sankar et al., 1994), this paper gives a new degree elevation algorithm based on representing a B-spline using derivatives, and converting it back to the usual piecewise polynomial form. Our new algorithm has the following advantages:

- our algorithm is more efficient than existing algorithms,
- our algorithm is simple to implement, with a readily understood conceptual basis,
- our algorithm can handle unclamped B-spline curves, whereas previous algorithms only work in the clamped case,
- our algorithm can be generalized to perform knot insertion.

Section 2 outlines various B-spline formulae we need later. Section 3 describes our new algorithms. Section 4 provides examples and a comparison with other approaches. We close the paper with some conclusions and a discussion.

2. B-spline formulae

Here we summarize various relevant B-spline formulae. A B-spline curve of order k is defined by a linear combination of B-spline basis functions as:

$$P(t) = \sum_{i=0}^{n} P_i N_{i,k}(t), \quad t_{k-1} \leqslant t \leqslant t_{n+1}, \tag{1}$$

where the P_i are control points, and $N_{i,k}(t)$, $i = 0, \ldots, n$, are B-spline basis functions defined recursively on the knot vector $T = [t_0, \ldots, t_{n+k}]$ as (Piegl and Tiller, 1997):

$$N_{i,1}(t) = \begin{cases} 1 & \text{if } t_i \leqslant t < t_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t). \tag{3}$$

(By convention, if $0/0$ appears in the formula, we replace it by 0.)

We could explicitly require that $t_{i+k} > t_i$: if $t_{i+k} = t_i$, this leads to $N_{i,k}(t) = 0$, resulting in a B-spline curve which splits into two separate B-spline curves. However, there is no need to impose this condition,

and Eq. (1) is still valid with this choice of knot vector. This fact is important because the $(k-p)$th derivative of a B-spline curve may not satisfy the condition that $t_{i+p} > t_i$, but it is still convenient to treat it as a single B-spline curve.

As some knots with consecutive subscripts may be equal, for the sake of convenience, we rewrite the knot vector in another form as follows:

$$T = [t_0, \ldots, t_{k-2}, u_0, \underbrace{u_1, \ldots, u_1}_{z_1}, \underbrace{u_2, \ldots, u_2}_{z_2}, \ldots, \underbrace{u_{S-1}, \ldots, u_{S-1}}_{z_{S-1}}, u_S, t_{n+2}, \ldots, t_{n+k}], \tag{4}$$

where $t_0 \leqslant \cdots \leqslant t_{k-2} \leqslant u_0$, $u_S \leqslant t_{n+2} \leqslant \cdots \leqslant t_{n+k}$, and $\{u_i\}_{i=0,\ldots,S}$ is a strictly increasing sequence, with $\{z_i\}_{i=1,\ldots,S-1}$ being a positive integer sequence giving the multiplicities of each of the knots: $1 \leqslant z_i \leqslant k$; $i = 1, 2, \ldots, S-1$. The multiplicity of each u_i is z_i.

Let $P^{(l)}(t)$ denote the lth derivative of $P(t)$. Then

$$P^{(l)}(t) = \sum_{i=0}^{n-l} P_i^l N_{i+l,k-l}(t), \tag{5}$$

where $N_{i+l,k-l}(t)$ are the B-spline basis functions defined over the knot vector given by Eq. (4), and the P_i^l are defined recursively by:

$$P_i^l = \begin{cases} P_i & \text{if } l = 0, \\ \frac{k-l}{t_{i+k}-t_{i+l}}(P_{i+1}^{l-1} - P_i^{l-1}) & \text{if } l > 0 \text{ and } t_{i+k} > t_{i+l}, \\ 0 & \text{if } l > 0 \text{ and } t_{i+k} = t_{i+l}. \end{cases} \tag{6}$$

Alternatively, we can compute P_{i+1}^{l-1} from P_i^{l-1} and P_i^l by a rearrangement of Eq. (6):

$$P_{i+1}^{l-1} = P_i^{l-1} + \frac{t_{i+k} - t_{i+l}}{k-l} P_i^l. \tag{7}$$

We call P_i^j the *derivative coefficients* of the B-spline $P(t)$. When a B-spline curve has only simple knots, Wang (Wang et al., 1996) gives the following formula to compute the $(k-1)$th derivatives at the knots using the P_i^j as follows:

$$P^{(k-1)}(u_i) = P_i^{k-1}. \tag{8}$$

For curves having multiple knots, we now give a similar formula:

Theorem 1.

$$P^{(j)}(u_i) = P_{\beta_i}^j, \quad k - z_i \leqslant j \leqslant k-1,\ 1 \leqslant i \leqslant S-1, \tag{9}$$

where $\beta_i = \sum_{l=1}^{i} z_l$.

Proof.

$$P^{(j)}(u_i) = \sum_{i=0}^{n-j} P_i^j N_{i+j,k-j}(u_i) = \sum_{i=0}^{n-j} P_i^j N_{i+j,k-j}(t_{\beta_i+k-1}) = \sum_{i=\beta_i}^{\beta_i+k-j-1} P_i^j N_{i+j,k-j}(t_{\beta_i+k-1})$$

$$= P_{\beta_i}^j N_{\beta_i+j,k-j}(t_{\beta_i+k-1}) \tag{10}$$

$$= P_{\beta_i}^j. \tag{11}$$

Eq. (10) follows from $u_i = t_{\beta_i+k-1} = t_{i+j}$ and Eq. (11) follows from

$$N_{\beta_i+j,k-j}(t_{\beta_i+k-1}) = \sum_{i=\beta_i}^{\beta_i+k-j-1} N_{i+j,k-j}(t_{\beta_i+k-1}) = 1. \qquad \square$$

Knot vectors of B-splines can be classified as *clamped* or *unclamped* (Piegl and Tiller, 1997). The knot vector of a clamped B-spline curve satisfies $t_0 = t_1 = \cdots = t_{k-2} = u_0$ and $u_S = t_{n+2} = \cdots = t_{n+k}$. We may also say that $P(t)$ is *left-clamped* if $t_0 = t_1 = \cdots = t_{k-1}$. It is well known that a left-clamped B-spline curve $P(t)$ satisfies

$$P^{(j)}(u_0) = P_0^j, \quad 0 \leqslant j \leqslant k-1. \tag{12}$$

Existing degree elevation algorithms (Liu and Wayne, 1997; Prautzsch and Piper, 1991; Pigel and Tiller, 1994) can only handle clamped B-spline curves, so existing approaches to raising the degree of an unclamped B-spline curve first clamp its knot vector by means of a clamping algorithm (Piegl and Tiller, 1997). As an alternative, we use a knot adjustment algorithm (Tai et al., 2003) for this purpose; it can easily be combined with our new degree elevation algorithm to obtain greater overall efficiency.

3. Degree elevation

3.1. Degree elevation of a clamped B-spline curve

Since a B-spline curve is a piecewise polynomial curve, it is possible to raise its degree from k to $k+m$, where m is an integer greater than or equal to 1. Thus, there must exist control points Q_i and a new knot vector $\overline{T} = [\bar{t}_0, \ldots, \bar{t}_{\bar{n}+k+m}]$ such that

$$P(t) = Q(t) = \sum_{i=0}^{\bar{n}} \overline{N}_{i,k+m}(t) Q_i, \tag{13}$$

where $\bar{n}$ is the number of control points of $Q(t)$, and $\overline{N}_{i,k+m}(t), i = 0, \ldots, \bar{n}$, are the B-spline basis functions of order $k+m$ defined on the knot vector $\overline{T}$.

The curves $P(t)$ and $Q(t)$ have the same geometry and parameterization. The computation of $\bar{n}$, Q_i, and $\overline{T}$ is referred to as *raising the degree* of the curve (Pigel and Tiller, 1994).

The knot vector $\overline{T}$ and $\bar{n}$ can be computed as follows. Assume that T takes the form given in Eq. (4). Since degree elevation preserves continuity, $Q(t)$ has continuity of order C^{k-z_i} at u_i, and the new knot vector must take the form

$$T = [\underbrace{u_0, \ldots, u_0}_{k+m}, \underbrace{u_1, \ldots, u_1}_{z_1+m}, \underbrace{u_2, \ldots, u_2}_{z_2+m}, \ldots, \underbrace{u_{S-1}, \ldots, u_{S-1}}_{z_{S-1}+m}, \underbrace{u_S, \ldots, u_S}_{k+m}], \tag{14}$$

so that $\bar{n} = n + S \times m$.

We now consider how to find the Q_i.

3.1.1. The new degree elevation algorithm

Theorem 2. *The derivative coefficients of $P(t)$ and $Q(t)$ are related as follows*

$$Q_0^j = P_0^j, \quad 0 \leqslant j \leqslant k-1, \tag{15}$$

$$Q_{\beta_p+pm}^t = P_{\beta_p}^i, \quad 1 \leqslant p \leqslant S-1,\ k - z_p \leqslant i \leqslant k-1, \tag{16}$$

$$Q_{\beta_p+pm+j}^{k-1} = Q_{\beta_p+pm}^{k-1}, \quad 1 \leqslant p \leqslant S-1,\ 1 \leqslant j \leqslant m. \tag{17}$$

Proof. Theorem 1 gives that,

$$P^{(i)}(u_0) = P_0^i, \quad 0 \leqslant i \leqslant k-1,$$
$$Q^{(i)}(u_0) = Q_0^i, \quad 0 \leqslant i \leqslant k-1,$$
$$P^{(i)}(u_p) = P_{\beta_p}^i,\ Q^{(i)}(u_p) = Q_{\beta_p}^i, \quad 1 \leqslant p \leqslant S-1,\ k - z_p \leqslant i \leqslant k-1.$$

As $P(t)$ and $Q(t)$ have the same geometry and parametrization, so do their derivatives, which proves Eqs. (15) and (16).

Consider one segment of the knot vector, $t \in [u_p, u_{p+1})$. It is well known that at most $k+m$ of the B-spline basis functions $\overline{N}_{i,k+m}(t)$ are nonzero in this knot segment; more precisely, $\overline{N}_{i+k,m}(t)$ is nonzero on $[u_p, u_{p+1})$ when $\beta_p + (p-1)m - k + 1 \leqslant i \leqslant \beta_p + pm - k$. Consider the kth derivatives of $P(t)$ and $Q(t)$. As the degree of $P(t)$ is $k-1$, its kth derivative equals zero, and thus so does the kth derivative of $Q(t)$, so that:

$$0 = P^{(k)}(t) = Q^{(k)}(t) = \sum_{i=0}^{n+S\cdot m} Q_i^k \bar{n}_{i+k,m}(t) = \sum_{i=\beta_p+(p-1)m-k+1}^{\beta_p+pm-k} Q_i^k \bar{n}_{i+k,m}(t). \tag{18}$$

Eq. (18) allows us to deduce that $Q_i^k = 0$, $\beta_p + (p-1)m - k + 1 \leqslant i \leqslant \beta_p + pm - k$, and as a result we can deduce Eq. (17):

$$Q_{i+1}^{k-1} = Q_i^{k-1} + \frac{\bar{t}_{i+k+m} - \bar{t}_{i+k}}{(k+m)-k} Q_i^k. \qquad \square$$

Remark. It is obvious that Theorem 2 holds as long as $P(t)$ and $Q(t)$ are just left-clamped; it does not matter if the curve is right-unclamped. To do degree elevation for an unclamped curve, we can turn it into a left-clamped curve using the knot adjustment algorithm mentioned earlier.

For a given l in Eqs. (6) and (7), we can see that division by a common factor of $(k-l)$ is needed for all i, so from a software engineering point of view, it simplifies matters if we define instead

$$\widetilde{P}_i^j = P_i^j \Big/ \prod_{l=1}^{j} (k-l),$$

$$\widetilde{Q}_i^j = Q_i^j \Big/ \prod_{l=1}^{j} (k+m-l),$$

which lets us rewrite Eqs. (6) and (7) in simpler form:

$$\widetilde{P}_i^j = \frac{\widetilde{P}_{i+1}^{j-1} - \widetilde{P}_i^{j-1}}{t_{i+k} - t_{i+l}}, \tag{19}$$

$$\widetilde{P}_{i+1}^{j-1} = \widetilde{P}_{i}^{j-1} + (t_{i+k} - t_{i+l})\,\widetilde{P}_{i}^{j}. \tag{20}$$

Eqs. (15)–(17) now become

$$\widetilde{Q}_{0}^{j} = \prod_{l=1}^{j}\left(\frac{k-l}{k+m-l}\right)\widetilde{P}_{0}^{j}, \quad 0 \leqslant j \leqslant k-1, \tag{21}$$

$$\widetilde{Q}_{\beta_p+pm}^{j} = \prod_{l=1}^{j}\left(\frac{k-l}{k+m-l}\right)\widetilde{P}_{\beta_p}^{j}, \quad 1 \leqslant p \leqslant S-1,\ k-z_p \leqslant j \leqslant k-1, \tag{22}$$

$$\widetilde{Q}_{\beta_p+pm+i}^{k-1} = \widetilde{Q}_{\beta_p+pm}^{k-1}, \quad 1 \leqslant p \leqslant S-1,\ 1 \leqslant i \leqslant m. \tag{23}$$

This leads to greater efficiency. For example, in the case where the knots of $P(t)$ are not repeated, then $\prod_{l=1}^{j}(\frac{k-l}{k+m-l})$ can be computed a priori, so while Eqs. (15) and (16) add a further n multiplications, Eqs. (19) and (20) save a total of $n(k-1)$ multiplications and $mn(k-1)$ divisions respectively.

Based on the equations developed above, we now give a procedural method for degree elevation of a clamped B-spline curve as follows:

Algorithm 1. Raise a clamped B-spline curve from degree k to degree $k+m$.

- Use Eq. (19) to compute $\widetilde{P}_{0}^{j}$, $0 \leqslant j \leqslant k-1$ and $\widetilde{P}_{\beta_p}^{i}$, $1 \leqslant p \leqslant S-1$, $k-z_p \leqslant i \leqslant k-1$.
- Use Eq. (14) to compute $\widetilde{T}$ and set $\tilde{n}$ to $n + S \times m$.
- Use Eqs. (21)–(23) to get $\widetilde{Q}_{0}^{j}$, $0 \leqslant j \leqslant k-1$, $\widetilde{Q}_{\beta_p+pm}^{i}$, $\widetilde{Q}_{\beta_p+pm+j}^{k-1}$.
- Use Eq. (20) to compute new control points $\widetilde{Q}_{i}^{0}$.

3.1.2. Example

We now give a numerical example showing how our algorithm works. Fig. 1 shows a clamped B-spline curve. The original curve $P(t)$ is an order 4 B-spline curve with knot vector {0, 0, 0, 0, 0.5, 0.5, 1, 1, 1, 1} and control points {(260, 100), (100, 260), (260, 420), (420, 420), (580, 260), (420, 100)}. The curve $Q(t)$ after degree elevation is an order 5 B-spline with knots {0, 0, 0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1, 1, 1} and with control points {(260, 100), (140, 220), (180, 340), (280, 420), (400, 420), (500, 340), (540, 220), (420, 100)}. The differential coefficients of the two curves are presented below together with the relations from Theorem 2:

$$\widetilde{Q}_{0}^{0} = \widetilde{P}_{0}^{0}, \qquad \widetilde{Q}_{1}^{0} = \frac{3}{4}\widetilde{P}_{1}^{0}, \qquad \widetilde{Q}_{2}^{0} = \frac{1}{2}\widetilde{P}_{2}^{0}, \qquad \widetilde{Q}_{3}^{0} = \frac{1}{4}\widetilde{P}_{3}^{0},$$

$$\widetilde{Q}_{2}^{3} = \frac{1}{2}\widetilde{P}_{2}^{2}, \qquad \widetilde{Q}_{3}^{3} = \frac{1}{4}\widetilde{P}_{2}^{3}, \qquad \widetilde{Q}_{3}^{1} = \widetilde{Q}_{3}^{0}, \qquad \widetilde{Q}_{3}^{4} = \widetilde{Q}_{3}^{3}.$$

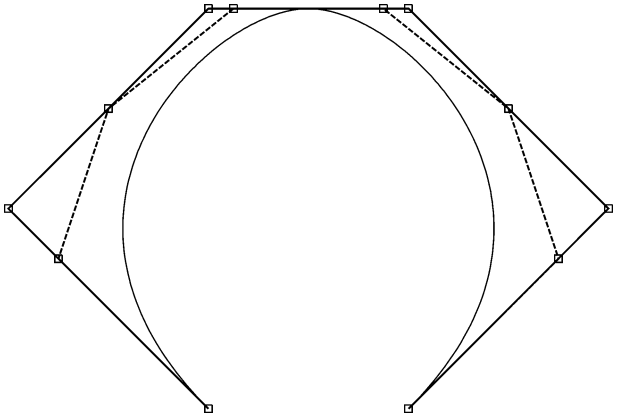

Fig. 1. Degree elevation from order 4 to order 5.

Differential coefficients of $P(t)$:

$\widetilde{P}_0^0(260, 100)$ $\widetilde{P}_1^0(100, 260)$ $\widetilde{P}_2^0(260, 420)$

$\widetilde{P}_0^1(-320, 320)$ $\widetilde{P}_1^1(320, 320)$ $\widetilde{P}_2^1(160, 0)$

$\widetilde{P}_0^2(1280, 0)$ $\widetilde{P}_1^2(-320, -640)$ $\widetilde{P}_2^2(320, -640)$

$\widetilde{P}_0^3(-3200, -1280)$ $\widetilde{P}_1^3(320, -640)$ $\widetilde{P}_2^3(-3200, 1280)$

$\widetilde{P}_3^0(420, 420)$ $\widetilde{P}_4^0(580, 260)$ $\widetilde{P}_5^0(420, 100)$

$\widetilde{P}_3^1(320, -320)$ $\widetilde{P}_4^1(-320, -320)$

$\widetilde{P}_3^2(-1280, 0)$.

Differential coefficients of $Q(t)$:

$\widetilde{Q}_0^0(260, 100)$ $\widetilde{Q}_1^0(140, 220)$ $\widetilde{Q}_2^0(180, 340)$ $\widetilde{Q}_3^0(280, 420)$

$\widetilde{Q}_0^1(-240, 240)$ $\widetilde{Q}_1^1(80, 240)$ $\widetilde{Q}_2^1(200, 160)$ $\widetilde{Q}_3^1(120, 0)$

$\widetilde{Q}_0^2(640, 0)$ $\widetilde{Q}_1^2(240, -160)$ $\widetilde{Q}_2^2(-160, -320)$ $\widetilde{Q}_3^2(160, -320)$

$\widetilde{Q}_0^3(-800, -320)$ $\widetilde{Q}_1^3(-800, -320)$ $\widetilde{Q}_2^3(160, -320)$ $\widetilde{Q}_3^3(-800, 320)$

$\widetilde{Q}_0^4(0, 0)$ $\widetilde{Q}_1^4(0, 0)$ $\widetilde{Q}_2^4(0, 0)$ $\widetilde{Q}_3^4(0, 0)$

$\widetilde{Q}_4^0(400, 420)$ $\widetilde{Q}_5^0(500, 340)$ $\widetilde{Q}_6^0(540, 220)$ $\widetilde{Q}_7^0(420, 100)$

$\widetilde{Q}_4^1(200, -160)$ $\widetilde{Q}_5^1(80, -240)$ $\widetilde{Q}_6^1(-240, -240)$

$\widetilde{Q}_4^2(-240, -160)$ $\widetilde{Q}_5^2(-640, 0)$

$\widetilde{Q}_4^3(-800, 320)$.

3.2. Degree elevation of an unclamped B-spline curve

The above method can readily be generalised to the case of an unclamped B-spline curve. This is done by considering the appropriate knot adjustment and degree raising results in turn, and combining them.

3.2.1. Problem statement

Let $P(t) = \sum_{i=0}^{n} N_{i,k}(t) P_i$ be a B-spline curve defined on the knot vector

$$T = [t_0, \ldots, t_{k-2}, u_0, \underbrace{u_1, \ldots, u_1}_{z_1}, \ldots, \underbrace{u_{S-1}, \ldots, u_{S-1}}_{z_{S-1}}, u_S, t_{n+2}, \ldots, t_{n+k}].$$

We wish to raise the degree of this B-spline from k to $k+m$, and compute an appropriate new knot vector. We denote the final curve by $Q(t)$, with knot vector of the form:

$$\overline{T} = [\bar{t}_0, \ldots, \bar{t}_{k+m-2}, u_0, \underbrace{u_1, \ldots, u_1}_{z_1+m}, \ldots, \underbrace{u_{S-1}, \ldots, u_{S-1}}_{z_{S-1}}, u_S, \bar{t}_{n+Sm+2}, \ldots, \bar{t}_{n+(S+1)m+k}].$$

As before $\{\bar{t}_0 \leqslant \bar{t}_1 \leqslant \cdots \leqslant \bar{t}_{k+m-2} \leqslant u_0\}$ and $\{u_S \leqslant \bar{t}_{n+Sm+2} \leqslant \ldots \leqslant \bar{t}_{n+(S+1)m+k}\}$ are input values which may be chosen by the user.

3.2.2. Degree elevation algorithm

First, we change the knot vector of $P(t)$ to its left-clamped form

$$T_1 = [\underbrace{u_0, \ldots, u_0}_{k}, \underbrace{u_1, \ldots, u_1}_{z_1}, \ldots, \underbrace{u_{S-1}, \ldots, u_{S-1}}_{z_{S-1}}, u_S, t_{n+1}, \ldots, t_{n+k}].$$

The new curve $R(t)$ defined on T_1 has control points $R_i, i = 0, \ldots, n$. By considering the derivative coefficients of $P(t)$ and $R(t)$ arising in the knot adjustment algorithm, we have

$$\widetilde{R}_0^{k-1} = \widetilde{P}_0^{k-1} \widetilde{R}_{k-2-l}^{l} = \widetilde{P}_{k-2-l}^{l}, \quad 0 \leqslant l \leqslant k-2. \tag{24}$$

$$\widetilde{R}_{\beta_p}^{i} = \widetilde{P}_{\beta_p}^{i}, \quad 1 \leqslant p \leqslant S-1,\ k - z_p \leqslant i \leqslant k-1, \tag{25}$$

$$\widetilde{R}_i^{l} = \widetilde{R}_{i+1}^{l} - (t_{i+k} - t_{k-1}) \widetilde{R}_i^{l+1}, \quad i = 0, \ldots, k-3-l,\ l = k-3, \ldots, 0. \tag{26}$$

We now use the degree elevation algorithm for a clamped B-spline curve to raise the degree of $R(t)$ as desired. Let the result be $U(t)$ with control points U_i and knot vector

$$T_2 = [\underbrace{u_0, \ldots, u_0}_{k+m}, \underbrace{u_1, \ldots, u_1}_{z_1+m}, \ldots, \underbrace{u_{S-1}, \ldots, u_{S-1}}_{z_{S-1}}, u_S, \tilde{t}_{n+Sm+2}, \ldots, \tilde{t}_{n+(S+1)m+k}].$$

The relations between the $\widetilde{R}_i^j$ and the $\widetilde{U}_i^j$ are given by Eqs. (21)–(23) where we replace P by R and Q by U.

Finally, we change the knot vector T_2 to $\widetilde{T}$ and find $Q(t)$. By considering the derivative coefficients of $U(t)$ and $Q(t)$, and as $Q_i^l = U_i^l = 0, l \geqslant k$ from the knot adjustment algorithm, we have

$$\widetilde{Q}_{\beta_p+pm}^{j} = \widetilde{U}_{\beta_p+pm}^{j}, \quad 1 \leqslant p \leqslant S-1,\ k - z_p \leqslant j \leqslant k-1, \tag{27}$$

$$\widetilde{Q}_{\beta_p+pm+i}^{k-1} = \widetilde{Q}_{\beta_p+pm}^{k-1} = \widetilde{U}_{\beta_p+pm+i}^{k-1} = \widetilde{U}_{\beta_p+pm}^{k-1}, \quad 1 \leqslant p \leqslant S-1,\ 1 \leqslant i \leqslant m, \tag{28}$$

$$\widetilde{Q}_{m+k-1-j}^{j} = \widetilde{U}_{m+k-1-j}^{j}, \quad 1 \leqslant j \leqslant k-1, \qquad \widetilde{Q}_i^{k-1} = \widetilde{U}_i^{k-1}, \quad 1 \leqslant l \leqslant m, \tag{29}$$

$$\widetilde{Q}_i^j = \widetilde{Q}_{i+1}^j - (\tilde{t}_{i+k+m} - \tilde{t}_{i+j+1})\widetilde{Q}_i^{j+1}, \quad i = 0, \ldots, k+m-2-j,\ j = 0, \ldots, k-2. \tag{30}$$

These may readily be cast into algorithmic form as before.

3.3. Combining knot insertion and degree elevation

3.3.1. Problem statement

In this section, we only consider the case of a clamped B-spline curve. Let $P(t) = \sum_{i=0}^{n} P_i N_{i,k}(t)$ be a B-spline curve defined over the knot vector

$$T = [\underbrace{u_0, \ldots, u_0}_{k}, \underbrace{u_1, \ldots, u_1}_{z_1}, \underbrace{u_2, \ldots, u_2}_{z_2}, \ldots, \underbrace{u_{S-1}, \ldots, u_{S-1}}_{z_{S-1}}, \underbrace{u_S, \ldots, u_S}_{k}]$$

as before. We now wish to raise its degree from k to $k+m$, and also to insert a set of new knots s_i each with multiplicity y_i:

$$[\underbrace{s_0, \ldots, s_0}_{y_0}, \underbrace{s_1, \ldots, s_1}_{y_1}, \ldots, \underbrace{s_l, \ldots, s_l}_{y_l}]. \tag{31}$$

We denote the final curve by $Q(t)$. We may express the final knot vector as:

$$\overline{T} = [\underbrace{\bar{u}_0, \ldots, \bar{u}_0}_{k+m}, \underbrace{\bar{u}_1, \ldots, \bar{u}_1}_{\bar{z}_1}, \ldots, \underbrace{\bar{u}_{l[1]}, \ldots, u_{l[1]}}_{\bar{z}_{l[1]}}, \ldots, \underbrace{\bar{u}_{l[S]-1}, \ldots, \bar{u}_{l[S]-1}}_{z_{l[S]-1}}, \underbrace{\bar{u}_{l[S]}, \ldots, \bar{u}_{l[S]}}_{k+m}], \tag{32}$$

where $\bar{u}_{l[i]} = u_i,\ 0 \leqslant i \leqslant S$, are the knots of original curve and the knots

$$[\underbrace{\bar{u}_1, \ldots, \bar{u}_1}_{\bar{z}_1}, \ldots, \underbrace{\bar{u}_{l[1]-1}, \ldots, \bar{u}_{l[1]-1}}_{\bar{z}_{l[1]-1}}, \underbrace{\bar{u}_{l[1]}, \ldots, \bar{u}_{l[1]}}_{\bar{z}_{l[1]}-z_1-m}, \ldots, \underbrace{\bar{u}_{l[S]-1}, \ldots, \bar{u}_{l[S]-1}}_{\bar{z}_{l[S]-1}}] \tag{33}$$

are the ones inserted. Thus, the number of new knots is $\bar{n} = n + Sm + \sum_{i=0}^{l} y_i$, where $\sum_{i=0}^{l} y_i$ is the number of knots being inserted. Here Eq. (33) gives another way of expressing Eq. (31).

3.3.2. Combining knot insertion and degree elevation

Existing degree elevation and knot insertion algorithms use different approaches which are hard to combine. However, our degree elevation algorithm and the knot insertion algorithm proposed by (Sankar et al., 1994) use the same idea, i.e., computing derivatives from control points, resampling the knot vector, and computing new control points from derivatives. It is thus possible to combine the two algorithms. The resulting algorithm is more efficient than performing degree elevation and knot insertion separately. The following theorem describes the relations between derivatives of the curve before and after degree elevation and knot insertion.

Theorem 3.

$$\widetilde{Q}_0^j = \widetilde{P}_0^j, \quad 0 \leqslant j \leqslant k-1, \tag{34}$$

$$\widetilde{Q}_{\tilde{\beta}_{l[i]}}^j = \widetilde{P}_{\beta_i}^j, \quad k - z_i \leqslant j \leqslant k-1,\ 1 \leqslant i \leqslant S-1, \tag{35}$$

$$\widetilde{Q}_{\tilde{\beta}_{l[i]+h}+j}^{k-1} = \widetilde{P}_{\beta_i}^{k-1}, \quad \begin{array}{l} 1 \leqslant i \leqslant S-1, \\ 0 \leqslant h \leqslant l[i+1] - l[i] - 1, \\ 0 \leqslant j \leqslant \min(\tilde{z}_{l[i]+h} - 1, m), \end{array} \tag{36}$$

$$\widetilde{Q}^{j}_{\tilde{\beta}_h} = \widetilde{Q}^{j}_{\tilde{\beta}_h+(k-m-1-\tilde{z}_i-j)}, \quad \begin{cases} k+m-\tilde{z}_h \leqslant j \leqslant k-2 & \text{if } h \neq l[i], \\ k+m-\tilde{z}_h \leqslant j \leqslant k-z_i-1 & \text{if } h = l[i]. \end{cases} \tag{37}$$

Proof. The ideas of the proof follow those of Theorem 2 and we omit them for brevity. □

We may now give an algorithm for simultaneously raising the degree and inserting new knots, which follows by analogy with Algorithm 1.

Algorithm 2. Simultaneous degree elevation and knot insertion for a clamped B-spline curve.
The order is being raised from k to $k+m$.

- Use Eq. (19) to compute $\widetilde{P}^{j}_{0}$, $0 \leqslant j \leqslant k-1$, and $\widetilde{P}^{i}_{\beta_p}$, $1 \leqslant p \leqslant S-1$, $k-z_p \leqslant i \leqslant k-1$.
- Set $\widetilde{T}$ by Eq. (32) and set $\tilde{n} = n + Sm + \sum_{i=0}^{l} y_i$ as above.
- Use Theorem 3 to get $\widetilde{Q}^{j}_{0}$, $0 \leqslant j \leqslant k-1$, $\widetilde{Q}^{j}_{\tilde{\beta}_{l[i]}}$, $k-z_i \leqslant j \leqslant k-1$, $1 \leqslant i \leqslant S-1$, and

$$\widetilde{Q}^{k-1}_{\tilde{\beta}_{l[i]+h}+j}, \quad \begin{array}{l} 1 \leqslant i \leqslant S-1, \\ 0 \leqslant h \leqslant l[i+1]-l[i]-1, \\ 0 \leqslant j \leqslant \min(\tilde{z}_{l[i]+h}-1, m). \end{array}$$

- Use Eqs. (20) and (37) to compute the new control points $\widetilde{Q}^{0}_{i}$.

Our algorithm is very efficient. For a 2D B-spline curve with unique knots, we need only $3(k-1)$ additions and $2(k-1)$ multiplications per inserted knot, while Böhm's knot insertion algorithm (Goldman and Lyche, 1993) takes $3(k+m-1)$ additions and $2(k+m-1)$ multiplications.

4. Comparison and discussion

We now give the results of comparing our new algorithm with existing methods.

4.1. Degree elevation

Here we only consider the case of a clamped B-spline curve, as the algorithms proposed by Piegl (Pigel and Tiller, 1994) and Prautzsch (Prautzsch and Piper, 1991) cover this case.

Firstly, we counted the number of operations needed by various algorithms. Prautzsch's algorithm is faster than Piegl's for raising the order by one, but Piegl's algorithm is more efficient when raising the order by an arbitrary degree m. Thus we compared our algorithm with Prautzsch's in the former case and with Piegl's in the latter. The following tables give the number of operations required, assuming all knots are of multiplicity one; we consider knots with higher multiplicity later.

Tables 1 and 2 clearly show that for arbitrary n, k, and m, our algorithm takes less arithmetic operations.

Secondly, we experimentally tested and compared our algorithm to the algorithms considered above. The following tests were performed:

- The growth rate as a function of m using different starting degrees ($k = 2, 3, \ldots$).

Table 1
Raising the degree by one

Operations	Prautzsch's algorithm	Our algorithm
$+,-$	$(n-k+1)(10k-8)+k$	$6(n-k+1)(k-1)+2k(k-1)$
$\times$	$(n-k+1)(5k-2)+2k$	$(n-k+1)(2k-1)+k(k-1)/2$
$/$	$(n-k+1)(2k-2)$	$(n-k+1)(k-1)+k(k-1)/2$

Table 2
Raising the degree by m

Operations	Pigel's algorithm	Our algorithm
$+,-$	$((2m+1)(k-1)+2k*k)(n-k+1)$	$2(m+2)(n-k+1)(k-1)+2k(k-1)$
$\times$	$((m+1)(k-1)+3k(k-1)/2)(n-k+1)$	$(n-k+1)(km+(k-m))+k(k-1)/2$
$/$	$k^2(n-k+1)/2$	$(n-k+1)(k-1)+k(k-1)/2$

Table 3
Times (in seconds) taken to elevate the degree starting at order 3

Elevation	Pigel's algorithm	Prautzsch's algorithm	Our algorithm
3 to 4	0.9432	0.7916	0.5338
3 to 5	0.9432	1.8412	0.689
3 to 6	1.5162	3.1578	0.8352
3 to 7	1.8116	4.7107	0.9734
3 to 8	2.108	6.5777	1.1446
3 to 9	2.3834	8.6342	1.3

- The growth rate as a function of the starting order k using different elevations of order (k goes to $k+1$, k goes to $k+2,\ldots$).

In an attempt to be as fair as possible to previous methods, we used the previous authors' own code given in (Prautzsch and Piper, 1991) and (Pigel and Tiller, 1994). The hardware used was an Intel Pentium IV, 1.4 GHZ computer. We ran each program 10000 times, and counted the total time taken in seconds. Each test curve was a 2D B-spline curve with 20 randomly chosen control points and randomly distributed knots.

These tests all led to similar conclusions; representative results for order 3 curves are shown in Table 3 and Fig. 2.

These tests showed that:

- Our algorithm is best in terms of absolute time taken.
- Prautzsch and Piper's algorithm has the worst growth characteristics; our algorithm has a slightly slower growth rate than Piegl and Tiller's algorithm.

These examples used B-spline curves with interior knots of multiplicity one. We also investigated timings in the order 4 case when the interior knots were all double or triple knots, again using a B-spline

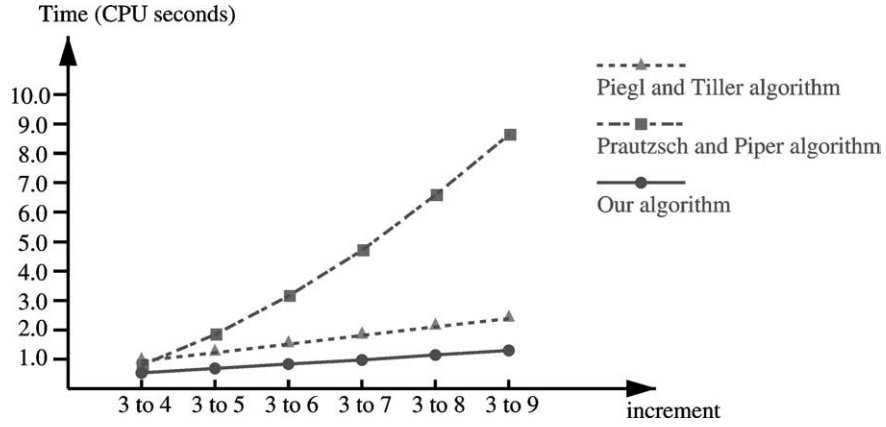

Fig. 2. Times taken to elevate the degree starting at order 3.

Table 4
Times (in seconds) taken to raise the order by 3 starting at various orders

Elevation	Pigel's algorithm	Prautzsch's algorithm	Our algorithm
2 to 4	0.8344	1.3525	0.4135
3 to 5	1.2218	1.8412	0.689
4 to 6	1.701	2.3516	0.9441
5 to 7	2.3283	2.9312	1.1355
6 to 8	2.8171	3.1965	1.3819
7 to 9	3.381	3.5331	1.564
8 to 10	4.057	3.8652	1.7143

with 20 control points. Similar results were again obtained, again showing that our algorithm to be best both in absolute time taken, and in growth rate, when the splines have multiple knots.

Next, we studied the performance of the various algorithms with respect to different starting orders, and differing amounts of degree elevation. Again, as a representative sample, we illustrate the results obtained in the order k to order $k+3$ case in Table 4 and Fig. 3.

The results follow a similar pattern to the previous tests, and our algorithm is the clear winner.

The results of our practical experiments validated our theoretical comparisons in terms of the number of operations used. Our new algorithm is clearly more efficient for degree elevation than either of the existing algorithms used as benchmarks.

4.2. Degree elevation and knot insertion

We also experimentally tested and compared our combined degree elevation and knot insertion algorithm to the use of a separate degree elevation algorithm followed by a knot insertion algorithm. For the same reasons noted above, we used Prautzsch's degree elevation algorithm when raising the degree by one, and Piegl's algorithm when raising the degree by more than one. We used Böhm's (1980) knot insertion algorithm, being the fastest. Other test conditions were as described in the previous Section.

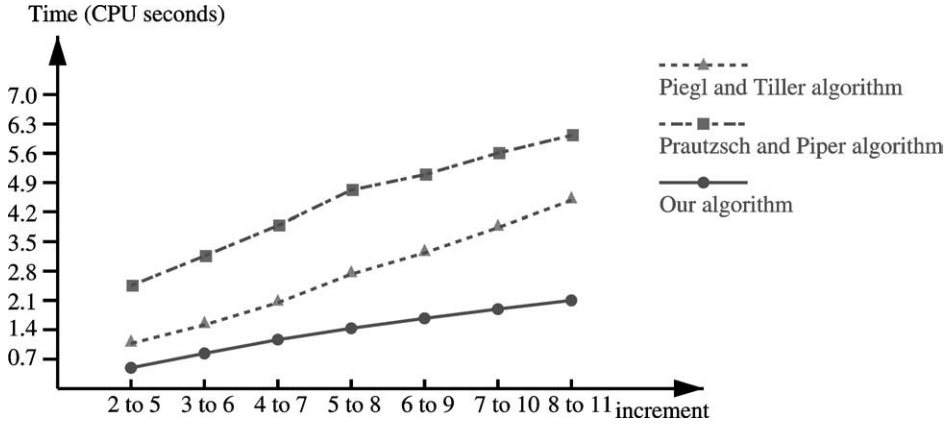

Fig. 3. Times taken to raise the order by 3 starting at various orders.

Table 5
Times (in seconds) taken to raise the order by 1 and insert varying numbers of knots

Number of knots inserted	Separate algorithms	Our algorithm
10	0.874	0.5894
30	1.042	0.7106
50	1.203	0.8118
70	1.375	0.9231
90	1.542	1.045
110	1.709	1.153

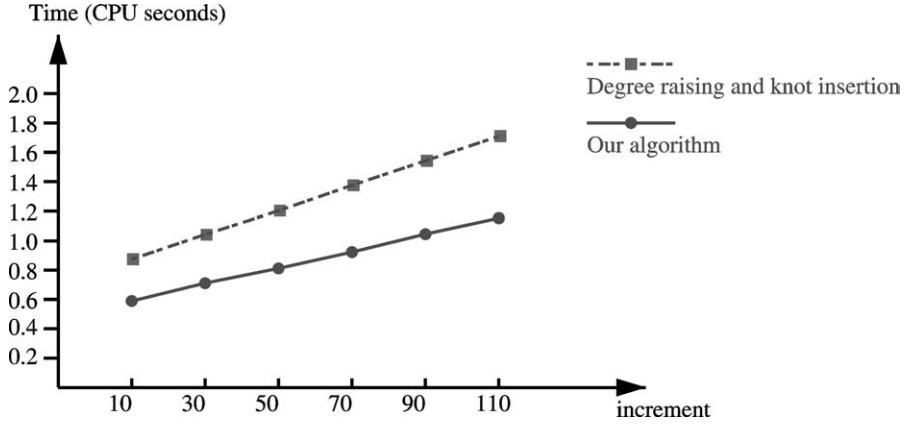

Fig. 4. Times (in seconds) taken to raise the order by 1 and insert varying numbers of knots.

Two tests were carried out. In the first, we always started with an order 3 curve, and raised its order to 4, while at the same time inserting some number of knots, with the number being inserted increasing from 10 to 110. Times taken are shown in Table 5, and graphed in Fig. 4.

In the second test, we started with an order 4 curve, and raised its order by varying amounts to order 5, 6, ..., 10, while at the same time always inserting 100 knots. In this case, the relative advantage of our method increased with the amount of degree elevation.

Overall, our combined algorithm is faster than using separate algorithms, as expected.

4.3. Handling rounding errors

Real arithmetic on a computer is not performed to perfect accuracy. Normally, the resulting errors (*rounding errors*) can be ignored because the noise they represent is extremely small compared to the signal. However, in certain ill-conditioned cases, the errors can be significant.

Usually, normal floating point arithmetic works well for our algorithms, but the results can be unstable when the knots are distributed in an extremely non-uniform manner, e.g., if $(u_{i+1} - u_i)/(u_i - u_{i-1}) < 10^{-7}$ for some i.

In this case, a modification to our new method may be used. We insert u_{i+1} and u_i, $k - z_{i+1}$ and $k - z_i$ times respectively, using Böhm's approach (Böhm, 1980), so that the curve is segmented into three separate B-spline curves. Degree elevation can then be performed for each curve using our algorithm, and then Eck's method (Eck and Hadenfeld, 1995) can be used to remove the knots u_i and u_{i+1}, k_{z_i} and $k_{z_{i+1}}$ times respectively, to obtain the overall B-spline curve after degree elevation. Stability is achieved by dividing the B-spline curve into segments such the that knots are relatively uniformly distributed within each segment.

Note that this process only takes $O(k^2)$ time for each knot interval $[u_i, u_{i+1}]$, which has to be processed in this way. If we do this procedure for *every* knot, we obtain Piegl and Tiller's algorithm.

5. Conclusion

In this paper, we have given an efficient algorithm to elevate the degree of a B-spline curve, and we have also shown how the process can be combined with a complementary algorithm working on similar principles for knot insertion, to give an efficient algorithm which can do degree elevation and knot insertion simultaneously. These methods are computationally superior to existing approaches. The new method presented in this paper can clearly also be extended to the case of degree reduction.

Acknowledgement

The work was supported by the Natural Science Foundation of China (Project Number 60225016, 60273012, 60321002) and the National Basic Research Project of China (Project Number 2002CB312101).

References

Böhm, W., 1980. Inserting new knots into B-spline curves. Computer Aided Design 12 (4), 199–201.

Cohen, E., Lyche, T., Riesenfeld, R.F., 1980. Discrete B-spline subdivision techniques in computer aided geometric design and computer graphics. Computer Graphics and Image Processing 14 (2), 87–111.

Cohen, E., Lyche, T., Schumaker, L., 1985. Algorithms for degree raising of splines. ACM Trans. Graph. 4 (3), 171–181.
Eck, M., Hadenfeld, J., 1995. Knot removal for B-spline curves. Computer Aided Geometric Design 12 (3), 259–282.
Goldman, R.N., Lyche, T., 1993. Knot Insertion Deletion Algorithms for B-Spline Curves and Surfaces. Society for Industrial and Applied Mathematics.
Ferrari, L.A., Sankar, P.V., Silbermann, M.J., 1994. Efficient algorithms for the implementations of general B-splines. Computer Vision, Graphics and Image Processing 56 (1), 102–105.
Liu, W., Wayne, 1997. A simple, efficient degree raising algorithm for B-spline curves. Computer Aided Geometric Design 14 (7), 693–698.
Prautzsch, H., 1984. Degree elevation of B-spline curves. Computer Aided Geometric Design 18 (12), 193–198.
Prautzsch, H., Piper, B., 1991. A fast algorithm to raise the degree of B-spline curves. Computer Aided Geometric Design 8 (4), 253–266.
Pigel, L., Tiller, W., 1994. Software-engineering approach to degree elevation of B-spline curves. Computer-Aided Design 26 (1), 17–28.
Piegl, L., Tiller, W., 1997. The NURBS Book, second ed. Springer-Verlag.
Sankar, P.V., Silbermann, M.J., Ferrari, L.A., 1994. Curve and surface generation and refinement based on a high speed derivative algorithm. Computer Vision, Graphics and Image Processing 56 (1), 94–101.
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.
Wang, S.Y., Ferrari, L., Silbermann, M.J., 1996. High speed computation of spline functions and applications. Internat. J. Imag. Syst. Technol. 7, 1–15.