Efficient Degree Elevation and Knot Insertion for B-spline Curves using Derivatives

Qi-Xing Huang[1], Shi-Min Hu[1] and Ralph R Martin[2]

[1]Tsinghua University, Beijing, huangqx@cg.tsinghua.edu.cn
[2]Tsinghua University, Beijing, shimin@tsinghua.edu.cn
[3]Cardiff University, Ralph.Martin@cs.cardiff.ac.uk

ABSTRACT

This paper presents a new algorithm for raising the degree of a B-spline curve which can also insert new knots at the same time. The new algorithm is faster than existing algorithms, and is much easier to understand and to implement. The new control points are computed using the following three simple steps: computing derivatives from control points, resampling the knot vector, and computing new control points from derivatives. Comparisons with previous methods and examples are given.

Keywords: B-splines, elevation, knot insertion

1. INTRODUCTION

Several algorithms have been published for raising the degree of B-spline curves [3,7-10]. The fastest of these is the algorithm by Prautzsch and Piper [9]; a simpler and easier-to-understand algorithm is the one by Piegl and Tiller [10]. The latter converts the B-spline curve to Bezier form, raises the degree of each Bezier curve, and then rejoins the Bezier curves to give the new B-spline curve. Liu[7] gives another degree elevation algorithm, which has the benefits of being fast and simple. It simply computes the new control points via a series of knot insertions followed by a series of knot deletions.

As is well-known, polynomial curve is uniquely determined by its value and the values of its derivatives at given point. Because B-spline curves are piecewise polynomial curves, they share similar property, i.e., each curve segment over knot interval $[t_i, t_{i+1}]$ is determined by the value and derivatives of the curve at the knot t_i. We have previously used this property for knot adjustment of B-splines [14].

Various papers [5,13,15] give rapid algorithms for the computation of the derivatives of an arbitrary order B-Spline, and demonstrate that knot refinement algorithms based on the above observation are an order of magnitude faster than the well-known Oslo algorithm [2].

In this paper, we develop new efficient degree elevation algorithm based on derivatives. While existing algorithms only work for clamped B-spline curves, our new algorithm also handles the case of unclamped B-spline curves. Comparisons between our new algorithm and existing ones are made, showing that our algorithm is more efficient, and is also easy to understand and implement. In Section 2, various B-spline formulae are stated. Section 3 describes our new algorithms. Comparisons and examples are given in Section 4. A conclusion Section closes the paper.

2. B-SPLINE FORMULAE

Here we summarize various relevant B-spline formulae and define our notation. Parametric B-spline curve of order k is defined by linear combination of B-spline basis functions as follows:

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

where the P_i are control points forming a control polygon, and $N_{i,k}(t), i = 0,...,n$, are the B-spline basis functions of order k defined on the knot vector $T = [t_0,...,t_{k-1},t_k,...,t_n,t_{n+1},...,t_{n+k}]$.

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 Equation (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:

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

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

Let $P^{(l)}(t)$ denote the l^{th} derivative of $P(t)$. Then $P^{(l)}(t)=\sum_{i=0}^{n-l}P_i^l N_{i+l,k-l}(t)$ where $N_{i+l,k-l}(t)$ are the B-spline basis functions defined over the knot vector given by Equation (2), and the P_i^l are defined recursively by:

$$P_i^l=\begin{cases}P_i & \text{if } l=0\\ \dfrac{k-l}{t_{i+k}-t_{i+l}}(P_{i+1}^{l-1}-P_i^{l-1}) & \text{if } l>0 \text{ and } t_{i+k}>t_{i+l}\\ 0 & \text{if } l>0 \text{ and } t_{i+k}=t_{i+l}\end{cases} \quad (3)$$

Alternatively, we can compute P_{i+1}^{l-1} from P_i^{l-1} and P_i^l by a rearrangement of Equation (3):

$$P_{i+1}^{l-1}=P_i^{l-1}+\frac{t_{i+k}-t_{i+l}}{k-l}P_i^l \quad (4)$$

We call P_i^j the *derivative coefficients* of the B-spline $P(t)$. When a B-spline curve has only simple knots, Wang [15] gives the following formula to compute the $(k-1)^{th}$ derivatives at the knots using the P_i^j as follows:

$$P^{(k-1)}(u_i) = P_i^{k-1} \quad (5)$$

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

Theorem 1

$$P^{(j)}(u_i) = P_{\beta_i}^{j} \quad {\scriptstyle k-z_i\le j\le k-1 \atop \scriptstyle 1\le i\le S-1}, \text{ where } \beta_i=\sum_{l=1}^{i}z_l \quad (6)$$

PROOF

$$P^{(j)}(u_i)=\sum_{i=0}^{n-j}P_i^j N_{i+j,k-j}(u_i)=\sum_{i=0}^{n-j}P_i^j N_{i+j,k-j}(t_{\beta_i+k-1}) \quad (7)$$

$$=\sum_{i=\beta_i}^{\beta_i+k-j-1}P_i^j N_{i+j,k-j}(t_{\beta_i+k-1})=P_{\beta_i}^j N_{\beta_i+j,k-j}(t_{\beta_i+k-1}) \quad (8)$$

$$=P_{\beta_i}^j \quad (9)$$

Equation (8) follows from $u_i=t_{\beta_i+k-1}=t_{i+j}$ and Equation (9) follows from

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

Knot vectors of B-splines can be classified as *clamped* and *unclamped* [6,12]. 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

$$P^{(j)}(u_0) = P_0^j, 0\le j\le k-1 \quad (10)$$

3. DEGREE ELEVATION

3.1 Degree Elevation of a B-spline curve

We now consider degree elevation of a clamped 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 $\overline{T}=[\overline{t_0},\ldots,\overline{t}_{n+k+m}]$ such that

$$P(t) = Q(t) = \sum_{i=0}^{\pi} \overline{N}_{i,k+m}(t) Q_i \qquad (11)$$

where $\overline{n}$ is the number of control points of $Q(t)$, and $\overline{N}_{i,k+m}(t), i = 0,\ldots,n$, 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 $\overline{n}$, Q_i, and $\overline{T}$ is referred to as raising the degree of the curve [10].

The knot vector $\overline{T}$ and $\overline{n}$ can be computed as follows. Assume that T takes the form given in equation (2). 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

$$T = [\underbrace{u_0,\ldots,u_0}_{k+m}, \underbrace{u_1,\ldots,u_1}_{z_1+m}, \ldots, \underbrace{u_{S-1},\ldots,u_{S-1}}_{z_{S-1}+m}, \underbrace{u_S,\ldots,u_S}_{k+m}], \quad (12)$$

so that $\overline{n} = n + S \times m$. We now consider how to find the Q_i.

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

$$Q_j = P_j, 0 \le j \le k-1. \quad (13)$$

$$Q^t_{\beta_p+pm} = P^i_{\beta_p}, \ {}^{1\le p\le S}_{k-z_p\le i\le k-1} \quad (14)$$

$$Q^{k-1}_{\beta_p+pm+j} = P^{k\text{-}1}_{\beta_p+pm}, \ {}^{1\le p\le S}_{1\le j\le m} \quad (15)$$

PROOF. Theorem 1 gives that,

$$P^{(i)}(u_0) = P^i_0, \quad 0 \le i \le k-1$$
$$Q^{(i)}(u_0) = Q^i_0, \quad 0 \le i \le k-1$$
$$P^{(i)}(u_p) = P^i_{\beta_i}, Q^{(i)}(u_p) = Q^i_{\beta_i}, {}^{1\le p\le S}_{k-z_p\le i\le k-1}$$

As $P(t)$ and $Q(t)$ have the same geometry and parametrization, so do their derivatives, which proves Equations (13) and (14).

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 segment; more precisely, $\overline{N}_{i,k+m}(t)$ is nonzero on $[u_p, u_{p+1})$ when $\beta_p + (p-1)m - k + 1 \le i \le \beta_p + pm - k$. Consider the k^{th} derivatives of $P(t)$ and $Q(t)$. As the degree of $P(t)$ is $k-1$, its k^{th} derivative equals zero, and thus so is the k^{th} derivative of $Q(t)$. Thus

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

The above equation allows us to deduce that $Q_i^k = 0, \beta_p + (p-1)m - k + 1 \le i \le \beta_p + pm - k$, and as a result we can deduce Equation (15) from

$$Q_{i+1}^{k-1} = Q_i^{k-1} + \frac{t_{i+k+m} - t_{i+k}}{(k+m)-k} Q_i^k.$$

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 our earlier knot adjustment algorithm.

For a given l in Equations (3) and (4), 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

$$\tilde{P}_i^j = P_i^j / \prod_{l=1}^{j}(k-l), \quad \tilde{Q}_i^j = Q_i^j / \prod_{l=1}^{j}(k+m-l),$$

which lets us rewrite Equations (3) and (4) in simpler form:

$$\tilde{P}_i^j = \frac{\tilde{P}_{i+1}^{j-1} - \tilde{P}_i^{j-1}}{t_{i+k} - t_{i+l}}, \qquad (16)$$

$$\tilde{P}_{i+1}^{j-1} = \tilde{P}_i^{j-1} + (t_{i+k} - t_{i+l}) \cdot \tilde{P}_i^j. \quad (17)$$

Equations (13–15) now become

$$Q_0^j = \prod_{l=1}^{j} \left(\frac{k-l}{k+m-l}\right) \cdot P_0^j, 0 \le j \le k-1, \quad (18)$$

$$Q_{\beta_p+pm}^j = \prod_{l=1}^{j} \left(\frac{k-l}{k+m-l}\right) \cdot P_{\beta_p}^j, \ {}^{1\le p\le S}_{k-z_p\le i\le k-1} \quad (19)$$

$$Q_{\beta_p+pm+j}^{k-1} = Q_{\beta_p+pm}^{k-1}, \ {}^{1\le p\le S}_{1\le j\le m} \quad (20)$$

This leads to greater efficiency. For example, in the case where the knots of $P(t)$ are not repeated, then $\prod_{l=1}^{j} \left(\frac{k-l}{k+m-l}\right)$ can be computed a priori, so while Equations (13) and (14) add a further *n* multiplications, Equations (16) and (17) 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 Equation (16) to compute* $P_0^j, 0 \le j \le k-1$ *and* $P_{\beta_p}^j, \ {}^{1\le p\le S}_{k-z_p\le i\le k-1}$
- *Use Equation (12) to compute* $\tilde{T}$ *and set* $\tilde{n}$ *to* $n + S \times m$
- *Use Equations (18–20) to get* $Q_0^j, 0 \le j \le k-1$ *and* $Q_{\beta_p+pm}^j$, $Q_{\beta_p+pm+j}^{k-1}$
- *Use Equation (17) to compute new control points* Q_i^0,

Remarks

Existing degree elevation algorithms [7,9,10] can only handle clamped B-spline curves. The degree of an unclamped B-spline curve is raised by first clamping its knot vector using a suitable algorithm [12]. As an alternative, we may use a knot adjustment algorithm for this purpose [14]; it can easily be combined with our new degree elevation algorithm to obtain greater overall efficiency.

3.2 Combining Knot Insertion and Degree Elevation

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

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

in a similar way to before. We now wish to raise its degree from k to $k+m$, and also to insert a set of new knots

$$T = [\underbrace{s_0,\ldots,s_0}_{y_0}, \underbrace{s_1,\ldots,s_1}_{y_1}, \ldots, \underbrace{s_l,\ldots,s_l}_{y_{l-1}}]$$

where each y_i gives the multiplicity of knot s_i.

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

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

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

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

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. Equation (23) is another form of to Equation (21).

Existing degree elevation and knot insertion algorithms use different methods which are hard to combine. However, our degree elevation algorithm and the knot insertion algorithm proposed by [13] can be combined as they use the same idea, i.e. computing derivatives from control points, resampling the knot vector, and computing new control points from derivatives. In this section, we do so, and show that 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

$$\mathbf{Q}_0^j = \mathbf{P}_0^j,\ 0 \le j \le k-1 \qquad (24)$$

$$\mathbf{Q}_{\beta_{l[i]}}^j = \mathbf{P}_i^j \ , \ \begin{matrix} k-z_i \le j \le k-1 \\ 1 \le i \le S-1 \end{matrix} \qquad (25)$$

$$Q_{\beta_{l[i]+h}+j}^{k-1} = P_{\beta_i}^{k-1} \ , \ \begin{matrix} 1 \le i \le S-1, 0 \le h \le l[i+1]-l[i]-1 \\ 0 \le j \le \min(\tilde{z}_{l[i]+h}-1, m) \end{matrix} \qquad (26)$$

$$\mathbf{Q}_{\beta_h}^j = \mathbf{Q}_{\beta_h+(k-m-1-\tilde{z}_{i-j})}^j \ , \ \begin{matrix} k+m-\tilde{z}_h \le j \le k-2, h \ne l[i] \\ k+m-\tilde{z}_h \le j \le k-z_i-1, h = l[i] \end{matrix} . \qquad (27)$$

PROOF

Equation (24) follows because

$$Q^{(j)}(u_0) = P^{(j)}(u_0), 0 \le j \le k-1.$$

Equation (25) follows because

$$Q^{(j)}(\tilde{u}_{l[i]}) \ = \ Q^{(j)}(u_i), \begin{matrix} k-z_i \le j \le k-1 \\ 1 \le i \le S-1 \end{matrix} .$$

Equation (26) follows because $Q^{(k)}(t) = 0$. Equation (27) follows because

$$Q^{(j)}(\tilde{u}_h-) \ = \ Q^{(j)}(\tilde{u}_h+) \ \begin{matrix} k+m-\tilde{z}_h \le j \le k-2, h \ne l[i] \\ k+m-\tilde{z}_h \le j \le k-z_i-1, h = l[i] \end{matrix}$$

The detailed proof follows that of Theorem (2) and we omit it 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 raised from k to k + m.*

- *Use Equation (16) to compute* $\mathbf{P}_0^j,\ 0 \le j \le k-1$ *and* $\mathbf{P}_{\beta_p}^i \ , \ \begin{matrix} k-z_i \le i \le k-1 \\ 1 \le p \le S-1 \end{matrix}$.
- *Set* $\mathbf{T}$ *by Equation (22) and set* $\tilde{n} = n + Sm + \sum_{i=1}^{l} y_i$ *as above.*
- *Use Theorem 3 to get* $\mathbf{Q}_0^j, 0 \le j \le k-1$, $\mathbf{Q}_{\beta_{l[i]}}^j \ , \ \begin{matrix} k-z_i \le j \le k-1 \\ 1 \le i \le S-1 \end{matrix}$ *and* $Q_{\beta_{l[i]+h}+j}^{k-1} \ , \ \begin{matrix} 1 \le i \le S-1, 0 \le h \le l[i+1]-l[i]-1 \\ 0 \le j \le \min(\tilde{z}_{l[i]+h}-1, m) \end{matrix}$
- *Use Equations (17) and (27) to compute the new control points* $\mathbf{Q}_0^i$

Our algorithm is very efficient. For a 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 [4] takes *3(k + m - 1)* additions and *2(k + m - 1)* multiplications.

4. COMPARISON

4.1Degree elevation only

Here we only consider clamped B-spline curves, as the algorithms proposed by Piegl [10] and Prautzsch [9] cover this case. We measured the time taken by the algorithm as we varied:

- the amount of degree elevation *m*, using different starting degrees (*k* =2, 3, . . .).
- the starting order *k*, using different elevations of order (*k* goes to k + 1, *k* goes to *k + 2*, . . .).

To be fair to previous methods, we used the authors' own code given in [9] and [10]. We used an Intel Pentium IV, 1.4GHz computer. We ran each program 10000 times, and measured the total time taken. B-spline test curves with 20 randomly chosen control points and randomly distributed knots were used.

We first made tests starting with order 3 and order 4 curves, and raised them to varying new orders up to order 9. Timings are graphed in Figures 1 and 2.

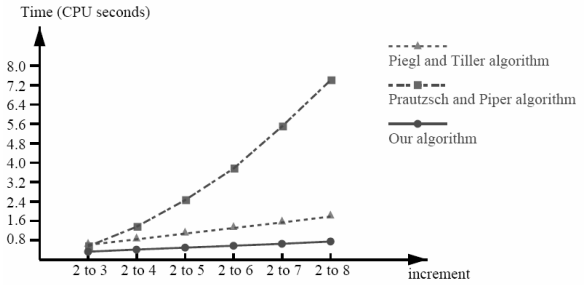

Fig.1. Times Taken to Elevate the Degree Starting at Order 3

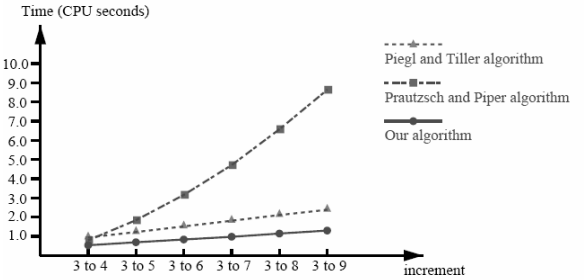

Fig.2. Times Taken to Elevate the Degree Starting at Order 4

The previous examples used B-spline curves with interior knots of multiplicity one. We also investigated the timings in the order 4 case when the interior knots were all multiple knots, again using a B-spline with 20 control points.

Figure 3 shows results for order 4 curves with interior

knots of multiplicity three. These results show that our algorithm is again the best, both in absolute time, 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. Figure 4 shows the timing results for elevation from order k to order $k + 1$ elevation using starting orders between 2 and 8.

Finally, the order k to order $k + 2$ case was tested, and the results are shown in Figure 5

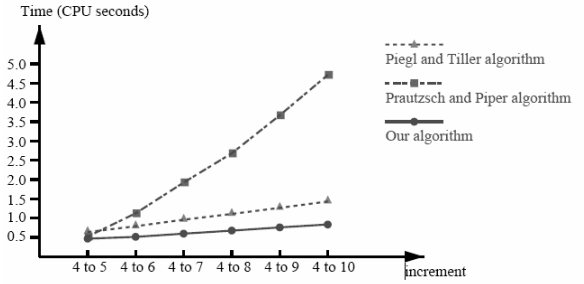

Fig.3. Times Taken with Interior Knots of Multiplicity 3

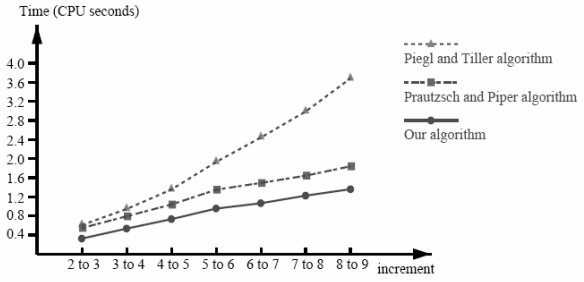

Fig.4. Times Taken to Raise the Order by 1 Starting at Various Orders

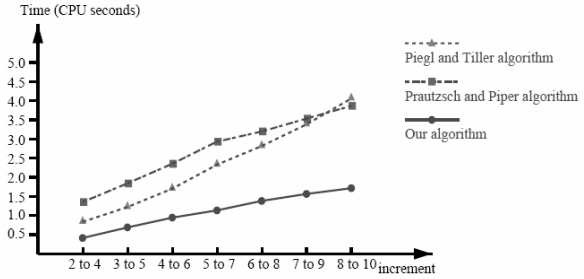

Fig.5. Times Taken to Raise the Order by 3 Starting at Various Orders

We can see from the above that our algorithm is the best in terms of absolute time taken. Furthermore, Prautzsch and Piper's algorithm usually has the worst growth rate; our algorithm usually has a slightly slower growth rate than Piegl and Tiller's algorithm. Our algorithm is the clear winner. The conclusions drawn by our practical experiments validate 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 separate degree elevation and knot insertion algorithms. For the separate algorithms we used the fastest available: for degree elevation, we used Prautzsch and Piper's algorithm when raising the degree by one, and Piegl and Tiller's algorithm when raising the degree by more than one; for knot insertion, we used Böhm's [1] algorithm. The test conditions were the same as those described in the previous Section.

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 a varying number of between 10 and 110 knots. The timings are graphed in Figure 6.

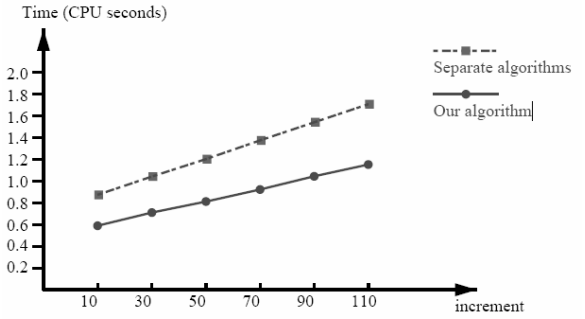

Fig.6 Times (in Seconds) Taken to Raise the Order by 1 and Insert Varying Numbers of Knots

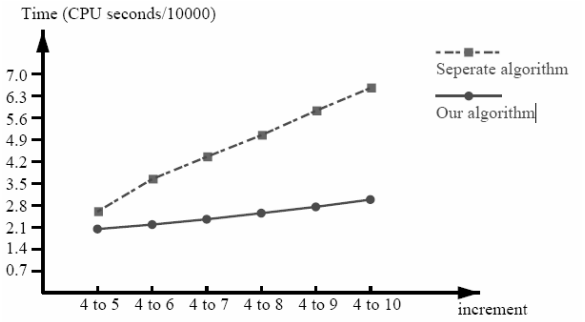

Fig.7. Times Taken to Raise the Order by Varying Amounts and Insert 100 Knots

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. The timings are graphed in Figure 7.

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

5. CONCLUSIONS

We have given a new algorithm to elevate the degree of a B-spline curve based on derivatives. It can also be combined with an algorithm using 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 also be extended to the case of degree reduction.

6. ACKNOWLEDGEMENTS

This work was supported by the Natural Science Foundation of China (Project Number 60225016, 60273012), the Specialized Research Fund for the Doctoral Program of Higher Education (Project Number 20020003051) and the National Basic Research Project of China (Project Number 2002CB312100).

7. REFERENCES

[1] W. Beohm, Inserting new knots into B-spline curves, *Computer Aided Design 12 (4), 1980. pp 199--201*

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

[3] E. Cohen, T. Lyche, L. Schumaker., Algorithms for degree raising of splines, ACM Transactions on Graphics 4 (3) , 1985, pp 171-181.

[4] R. N. Goldman, T. Lyche., *Knot insertion and deletion algorithms for B-spline curves and surfaces*, Society for Industrial and Applied Mathematics, 1993.

[5] L. A. Ferrari, P. V. Sankar, M. J. Silbermann., *Efficient algorithms for the implementations of general B-splines*, Computer Vision, Graphics and Image Processing 56 (1) , 1994. pp 102-105

[6] S.-M. Hu, C.-L. Tai and S.-H. Zhang., An Extension algorithm for B-spline curves by curve unclamping, Computer Aided Design, 2002, Vol. 34, No. 5, pp 415-419.

[7] W. Liu, Wayne., A simple, efficient degree raising algorithm for B-spline curves, Computer Aided Geometric Design 14 (7) 693--698, 1997.

[8] H. Prautzsch, Degree elevation of B-spline curves, Computer Aided Geometric Design 18 (12), 1984. pp 193--198

[9] H. Prautzsch, B. Piper., A fast algorithm to raise the degree of B-Spline curves, Computer Aided Geometric Design 8 (4), 1991. pp 253--266

[10] L. Pigel, W. Tiller., Software-engineering approach to degree elevation of B-spline curves, Computer Aided Design 26 (1) , 1994. pp 17--28

[11] L. Piegl, W. Tiller., Algorithm for approximate NURBS skinning, Computer-Aided Design 28 (9), 1995. pp 699--706

[12] L. Piegl, W. Tiller., The NURBS Book, Springer-Verlag, 2nd Edition, 1997.

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

[14] S. Y. Wang, L. Ferrari, M. J. Silbermann., High speed computation of spline functions and applications, International Journal of Imaging Systems and Technology, 1996. pp 71-75