MERGING A PAIR OF INTERVAL BÉZIER CURVES

Yang Xiaofeng and Chen Falai
Department of Mathematics
University of Science and Technology of China
Hefei, Anhui 230026, People's Republic of China

ABSTRACT

This paper deals with the problem of merging a pair of interval Bézier curves, i.e., bounding two adjacent interval Bézier curves by a single interval Bézier curve. Two different constrained optimization methods are presented to solve the problem. Examples are provided to compare the two different approaches.

1 INTRODUCTION

In Computer Aided Design and Geometric Modeling, there are considerable interests in approximating curves and surfaces with other forms of curves and surfaces. This problem arises whenever the geometric information in a CAD system is transferred to another system, or the amount of CAD data need to be reduced.

One of the problems in data reduction is the merging problem, that is, merging as many as possible curve segments to one curve segment. The merging problem was recently considered by Hu et al [9], and an algorithm is devised to approximately merge a pair of Bézier curves. However, the algorithm mainly concerns how good the approximation is, and it doesn't deal with the numerical errors (or gaps). Such gaps, though extremely small in size, could cause geometric modeling system's failure, generate useless analysis results, and create defects in finished products. Models with gaps often need tremendous rework at the receiving end of data exchange.

To overcome the above difficulty, Sederberg et al. introduced interval representation forms of curves and surfaces that can embody a complete description of coefficient errors [2]. Recently, Hu et al.[3],[4],[5],[6] turned to interval forms of geometric objects and rounded interval arithmetic to deal with the problems of valid and consistent representations of geometric objects and robust geometric operations such as curve/curve and surface/surface intersections. Their results indicate that using interval arithmetic will substantially increase the numerical stability in geometric computations, and thus enhance the robustness of current CAD/CAM systems.

In previous works, we dealt with the problem of degree reduction of interval Bézier curves and B-spline curves [7],[8]. In this paper, we will consider the problem of merging a pair of interval Bézier curves, i.e., bounding two adjacent interval Bézier curves with a single interval Bézier curve. In the next section, some fundamental concepts about interval arithmetic and interval Bézier curves are reviewed. Then in Section 3, two different constrained optimization algorithms are presented to merge a pair of interval Bézier polynomials. In Section 4, the methods are applied in merging a pair of interval Bézier curves. Finally in Section 5, we provide some examples to demonstrate the algorithms and make some comparisons between the two different approaches.

2 INTERVAL BEZIER CURVES

An interval \([a,b]\) is the set of real numbers:

\[ [a,b] = \{ x \mid a \leq x \leq b \} \]

where the following interval arithmetic operations are defined:
An interval polynomial in Bernstein form is a polynomial whose coefficients are intervals:

\[ [a,b] + [c,d] = [a + c, b + d], \]

\[ [a,b] - [c,d] = [a - c, b - d], \]

\[ [a,b] \times [c,d] = \min(ac, ad, bc, bd), \max(ac, ad, bc, bd)], \]

\[ [a,b]/[c,d] = [a/b, 1/c], \quad 0 \not\in [c,d] \]

The width of an interval polynomial can be defined as

\[ w([p](t)) = \left\| \sum_{k=0}^{n} (b_k - a_k) B^n_k(t) \right\| \]

where \( \| \cdot \| \) is the standard norm, such as \( \| \cdot \|_1 \) or \( \| \cdot \|_2 \).

An interval Bézier curve is a Bézier curve whose control points are vector-valued intervals (i.e. rectangular region is the plane):

\[ [p](t) = \sum_{k=0}^{n} [a_k, b_k] B^n_k(t), \quad 0 \leq t \leq 1 \]

where \( [a_k, b_k] = [c_k, d_k] \times [e_k, f_k], k = 0, 1, \ldots, n. \)

The interval Bézier curve (4) can also be written in the vector form

\[ [p](t) = ([x](t)), [y](t)) \]

where \([x](t)\) and \([y](t)\) are interval polynomials:

\[ [x](t) = \sum_{k=0}^{n} [a_k, b_k] B^n_k(t) \]

\[ [y](t) = \sum_{k=0}^{n} [c_k, d_k] B^n_k(t) \]

The width of interval Bézier curve \([p](t)\) is defined to be the maximum of the widths of \([x](t)\) and \([y](t)\).

3 CONstrained OPTIMIZATION METHOD FOR MERGING

The merging problem of a pair of interval Bézier curves can be stated as follows:

**Problem 1:** Given two adjacent interval Bézier curves

\[ [p](u) = \sum_{i=0}^{n} [p_i] B^n_i(u), \quad 0 \leq u \leq 1 \]

\[ [q](v) = \sum_{i=0}^{n} [q_i] B^n_i(v), \quad 0 \leq v \leq 1 \]

find a single degree \( m \) interval Bézier curve

\[ [r](t) = \sum_{k=0}^{m} [r_k] B^m_k(t), \quad 0 \leq t \leq 1 \]

such that

\[ [p](u) \cup [q](v) \subseteq [r](t) \]

and the width of \([r](t)\) is as small as possible.

To solve the above problem, we can first solve the following

**Problem 2:** Given two adjacent Bernstein polynomials of degree \( n \):

\[ f(u) = \sum_{i=0}^{n} f_i B^n_i(u), \quad g(v) = \sum_{i=0}^{n} g_i B^n_i(v) \]

find a Bernstein polynomial of degree \( m(m \geq n) \)

\[ F(t) = \sum_{i=0}^{m} F_i B^m_i(t) \]

such that

\[ F(t) \geq (\leq) \overline{F}(t), \quad 0 \leq t \geq 1 \]

and

\[ ||F-\overline{F}|| \]

is minimized, where

\[ \overline{F}(t) = \left\{ \begin{array}{ll} \sum_{i=0}^{n} f_i B^n_i(t), & 0 \leq t \leq \lambda \\ \sum_{i=0}^{n} g_i B^n_i(t), & 1-\lambda \leq t \leq 1 \end{array} \right. \]

and \( \lambda \in (0,1) \) is some constant.
subdivide \( F(t) \) at parameter \( t = \lambda \) and let
\[
f(u) = \sum_{i=0}^{m} f_i B_{i}^m(u), \quad g(v) = \sum_{i=0}^{m} g_i B_{i}^m(v)
\]
be the two halves of polynomial \( F(t) \) after subdivision. We also assume \( n = m \), since otherwise we can degree elevate polynomials \( \bar{f}(u) \) and \( \bar{g}(v) \) to degree \( m \). Then a sufficient condition for (12) is:
\[
f_i \geq \bar{f}_i, \quad g_i \geq \bar{g}_i, \quad i = 0,1,...,m
\]
The basic idea to solve problem \( 2 \) is to modify the given pair of polynomials such that they precisely form a single polynomial, and the polynomial is an upper (or a lower) bound of the given polynomials. We will describe the details of the algorithm for find an upper bound polynomial in the following subsections.

(3.1) The condition of precise merging for Bernstein polynomials

Given two polynomials \( f(u) \) and \( g(v) \) of degree \( m \), we say \( f(u) \) and \( g(v) \) are merged precisely if there exists a degree \( m \) Bernstein Polynomial \( F(t) \) such that \( F(t) = \bar{F}(t) \), where \( \bar{F}(t) \) is defined by (14).

**Lemma 1** [9] Two Bernstein polynomials \( f(u) \), \( g(v) \) of degree \( m \) can be merged precisely if and only if there exists some constant \( \mu > 0 \), such that
\[
\Delta f_{m+i} = \mu \Delta g_{0}, \quad i = 0,1,...,m
\]
where \( \Delta \) is the forward difference operator defined by \( \Delta f_i = f_{i+1} - f_i \).

(3.2) Constrained optimization method (I)

In this subsection, we use \( \| \cdot \|_1 \) to define the width of interval polynomials, that is,
\[
\| f(t) \| = \int_0^1 |f(t)| dt
\]
We solve Problem \( 2 \) as follows. We perturb the coefficients \( f_i \) and \( g_i \) of polynomials \( f(u) \) and \( g(v) \) by some non-negative values \( \epsilon_i \) and \( \delta_i \) respectively, \( i = 0,1,...,m \). The polynomials after perturbations are
\[
\begin{align*}
\bar{f}(u) &= \sum_{i=0}^{m} f_i B_{i}^m(u) + \sum_{i=0}^{m} \epsilon_i B_{i}^m(u) \\
\bar{g}(v) &= \sum_{i=0}^{m} g_i B_{i}^m(v) + \sum_{i=0}^{m} \delta_i B_{i}^m(v)
\end{align*}
\]
If \( f(u) \) and \( g(v) \) merge precisely, then the polynomial \( F(t) \) merged by \( f(u) \) and \( g(v) \) is exactly the upper bound polynomial. Notice that
\[
\| F(t) - \bar{F}(t) \|_1 = \sum_{i=0}^{m} (\epsilon_i + \delta_i).
\]

**Problem 2** can be converted to solving the following linear programming problem:
\[
\begin{align*}
\text{Min} & \sum_{i=0}^{m} (\epsilon_i + \delta_i) \\
\text{s.t.} & \Delta (\bar{f}_{m+i} + \epsilon_{m-i}) = \mu \Delta (\bar{g}_0 + \delta_0), \quad i = 0,1,...,m
\end{align*}
\]
From the constrained condition
\[
\Delta (\bar{f}_{m+i} + \epsilon_{m-i}) = \mu \Delta (\bar{g}_0 + \delta_0), \quad i = 0,1,...,m
\]
we get
\[
\begin{align*}
\delta_0 &= \phi_0(\epsilon_m) \\
\delta_i &= \phi_i(\epsilon_{m-i}, \epsilon_m) \\
&\vdots \\
\delta_m &= \phi_m(\epsilon_0, \epsilon_1, ..., \epsilon_m)
\end{align*}
\]
where, \( \phi_i, \quad i = 0,1,...,m \) are linear functions of \( \epsilon_{m-i}, ..., \epsilon_m \). Thus the linear programming problem (20) is equivalent to
\[
\begin{align*}
\text{Min} & \sum_{i=0}^{m} \phi(\epsilon_0, ..., \epsilon_{m-i}, \epsilon_m) \\
\text{s.t.} & \phi_j(\epsilon_{m-j}, ..., \epsilon_m) \geq 0, \quad j = 0,1,...,m
\end{align*}
\]
\[ \varphi(e_0, \ldots, e_{m-1}, e_m) = \sum_{i=0}^{m} \phi_i(e_{m-i}, \ldots, e_m) + \sum_{i=0}^{m} e_i \quad (23) \]

After solving the above linear programming problem, the upper bound polynomial \( F(t) \) can be easily obtained.

(3.3) Constrained optimization method (II)

In this subsection, we use

\[
\sum_{i=0}^{m} e_i = f_i \quad (24)
\]

to define the norm of a Bernstein polynomial.

To solve Problem 2, again we perturb the coefficients of \( \tilde{f}(u) \) and \( \tilde{g}(v) \) by some \( \varepsilon_i \) and \( \delta_i \), \( i = 0,1,\ldots,m \). However, we do not require the perturbations are non-negative, so we will solve the following linear programming problem

\[
\begin{aligned}
\text{Min} & \quad \sum_{i=0}^{m} (e_i^2 + \delta_i^2) \\
\text{s.t.} & \quad \Delta(m_i, m_{i-1}) = \mu' \Delta(\bar{g}_0 + \delta_0) \\
& \quad i = 0,1,\ldots,m
\end{aligned}
\quad (25)
\]

The above linear programming problem can be solved using Lagrange multipliers. The solution is as follows [9]:

\[
\begin{aligned}
\varepsilon_0 &= 0 \\
\varepsilon_j &= -\frac{1}{2} \sum_{i=m-j}^{m} (-1)^{m-j} \binom{m-j}{i} \lambda_i, \ j = 1,2,\ldots,m \\
\delta_j &= \frac{1}{2} \sum_{i=j}^{m} (-1)^{m-j} \binom{m-j}{i} \mu' \mu_i, \ j = 0,1,\ldots,m-1 \\
\delta_m &= 0
\end{aligned}
\quad (26)
\]

and \( \lambda_i (i = 0,1,\ldots,m) \) can be computed by:

\[
\frac{1}{2} \sum_{i=0}^{m} (1 + (-\mu'^{(i)}) (l + i) \lambda_i = \Delta(m_i, m_{i-1}) - \mu' \Delta(\bar{g}_0), \quad (27)
\]
i = 0,1,\ldots,m-1, and

\[
\frac{1}{2} \sum_{i=0}^{m} (1 + (-\mu'^{(i)}) (l + i) \mu' \mu_i^m \Delta^m \Delta^m \bar{g}_0 = \Delta^m \bar{f}_0 - \mu^m \Delta^m \Delta^m \bar{g}_0
\quad (28)
\]

After solving the above problem, we get an approximate merge \( F(t) \) of \( \tilde{f}(u) \) and \( \tilde{g}(v) \). To further make \( F(t) \) an upper bound polynomial, we add \( F(t) \) by a positive constant which is the absolute value of the smallest negative value among \( \varepsilon_i \) and \( \delta_i \), \( i = 0,1,\ldots,m \).

3 MERGING A PAIR OF INTERVAL BÉZIER CURVES

After having solved the problem of merging a pair of Bernstein polynomials, the problem of merging a pair of interval Bézier curves can be solved easily. The basic idea is based on the following

**Proposition** Given two adjacent interval Bézier curves \([P](u)\) and \([Q](v)\) of degree \(n\):

\([P](u) = ([x^p][u],[y^p][u]), \quad [Q](v) = ([x^q][v],[y^q][v])\)

Suppose \([x^p](t)\) and \([y^p](t)\) are interval polynomials merged by \([x^p](u)\), \([y^p](u)\) and \([x^q](v)\), \([y^q](v)\) respectively. Then

\([R](t) = ([x^r](t),[y^r](t))\)

are the interval Bézier curve which bound \([P](u)\) and \([Q](v)\).

The parameter \(\mu\) in the linear programming problem (20) and (25) can be defined as

\[
\mu = \frac{1}{n} \sum_{j=0}^{n} \frac{||P_m||^{||Q_m||}}{||||Q_m||^{||P_m||}}
\quad (29)
\]

where \(\bar{P}_j\) and \(\bar{Q}_j\) are the centers of the interval control points \([P](u)\) and \([Q](v)\) respectively.
5 EXAMPLES AND COMPARISONS

In this section, we will provide two examples to demonstrate the algorithms presented in the last section. We will also make some comparisons between the two different approaches.

Example 1 Given a pair of cubic interval Bézier curves $[P](u)$ and $[Q](v)$ whose control points are:

$[P_0] = [-3.6,-3.4] \times [2.0,2.2]$, $[P_1] = [-3.2,-3] \times [4.5,4.7]$ $[P_2] = [-2.2,-2] \times [5.6,5.7]$, $[P_3] = [-2.2,-0.1] \times [4.3,4.4]$ $[Q_0] = [-0.2,-0.1] \times [4.3,4.4]$, $[Q_1] = [2.1] \times [4.9,5.0]$ $[Q_2] = [2.9,3] \times [4.5,4.6]$, $[Q_3] = [3.9,4,1] \times [2.3,2.5]$ respectively. We use a degree $m = 4,5,6,7$ interval Bézier curve $[R](t)$ to merge $[P](u)$ and $[Q](v)$ by the two optimization methods respectively. The widths of $[R](t)$ are listed in Table 1.

![Table 1](image)

Table 1. Comparison of the widths of the merged curves by the two methods (1)

<table>
<thead>
<tr>
<th>Degree</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method (I)</td>
<td>1.25</td>
<td>0.91</td>
<td>0.80</td>
<td>0.81</td>
</tr>
<tr>
<td>Method (II)</td>
<td>1.21</td>
<td>1.03</td>
<td>0.93</td>
<td>0.89</td>
</tr>
</tbody>
</table>

Fig. 1 depicts the merged curve of degree five and the original curves.

Example 2 Let $[P](u)$ and $[Q](v)$ be a pair of interval Bézier curves with control points

$[P_0] = [-3.6,-3.4] \times [-0.5,-0.3]$, $[P_1] = [-3.2,-3] \times [-3,-2.8]$ $[P_2] = [-2.2,-2] \times [-2.6,-2.3]$, $[P_3] = [-2.2,-0.1] \times [-0.3,-0.1]$ $[Q_0] = [0.2,0.4] \times [-0.5,-0.4]$, $[Q_1] = [2.2,1] \times [2.9,3.0]$ $[Q_2] = [2.9,3] \times [3.1,3.3]$, $[Q_3] = [3.9,4.1] \times [-0.4,-0.3]$ respectively. Table 2 lists the widths of the merged interval Bézier curves.

![Table 2](image)

Table 2. Comparison of the widths of the merged curves by the two methods (2)

<table>
<thead>
<tr>
<th>Degree</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method (I)</td>
<td>1.84</td>
<td>1.01</td>
<td>0.76</td>
</tr>
<tr>
<td>Method (II)</td>
<td>1.71</td>
<td>1.09</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Fig. 2 shows the degree six merged curves by the two different methods.

![Fig. 1(a)](image) The degree five merged curve by the first method

![Fig. 1(b)](image) The degree five merged curve by the second method

![Fig. 2(a)](image) The merged curve of degree six by the first method

![Fig. 2(b)](image) The merged curve of degree six by the second method
From tens of examples we have tested so far, we conclude that the first method generally produces tighter bound than the second method. However, since there exist an analytic solution to the second method, the second method takes less computational costs than the first method.

REFERENCES