Iterative process for $G^2$-multi degree reduction of Bézier curves

Abedallah Rababah a, b, Stephen Mann a,*

a David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
b Jordan University of Science and Technology, Jordan

ARTICLE INFO

Keywords:
Bézier curves
Multi-degree reduction
Iterative methods
Geometric continuity
$G^2$-continuity

ABSTRACT

In this paper, the issue of multi-degree reduction of Bézier curves with $C^1$ and $G^2$-continuity at the end points of the curve is considered. An iterative method, which is the first of this type, is derived. It is shown that this algorithm converges and can be applied iteratively to get the required accuracy. Some examples and figures are given to demonstrate the efficiency of this method.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

The issue of degree reduction is concerned with the solution of the following problem: for a given Bézier curve $P_n(t)$ of degree $n$ with Bézier points $\{p_i\}_{i=0}^n$, find an approximative Bézier curve $R_m(t)$ of lower degree $m$, where $m < n$, with the set of Bézier points $\{r_i\}_{i=0}^m$, so that $P_n$ and $R_m$ satisfy boundary conditions at the end points, and the error between $P_n$ and $R_m$ is minimum.

The issue of degree reduction of Bézier curves and surfaces is of particular importance in the field of Computer Aided Geometric Design (CAGD). Degree reduction decreases the required memory for storage and is important for dealing with data comparison, compression, and transfer. It began as a simple operation as an inverse process of degree elevation.

Degree reduction of the Bézier curves has been conducted according to different norms, mostly the $L^2$- and $l^1$-norms, for both unconstrained and constrained conditions. In general, unconstrained degree reduction gives lower error than the constrained one. However, Bézier curves are often a part of a piecewise curve, so constrained degree reduction is preferred.

Hoschek [7] used non-linear optimization techniques to do degree reduction, which he called approximate conversions of spline curves. Watkins and Worsley [21] developed a detailed error analysis of degree reduction based on minimax approximation technique. Lachance [11] improved the common way of accomplishing degree reduction at that time, which was via black box methods. He used Chebyshev economization with the $l^1$-norm for the degree reduction. Furthermore, he improved these results to the constrained case [12]. Eck [8] made the work of Lachance [11] more transparent by giving formulas that express the geometric relationship between the control points. Moreover, constrained Chebyshev polynomials were used therein to obtain best constrained approximations. These results have been improved by Eck [9] by minimizing the constrained $L^2$-norm between the two curves by performing the degree reduction within a prescribed error tolerance. Bogacki et al. [2] presented a component-wise best uniform degree reduction of a given Bézier curve. Brunnett et al. [3] studied the geometry of optimal degree reduction of Bézier curves. Lutterkort et al. [16] showed that the best $L^2$-approximation is equivalent to the best Euclidean approximation of the vector of Bézier coefficients. Lee et al. [13] studied the relationship between Legendre and Bernstein bases and used these transformations to present a multiple degree reductions of Bézier curves with respect to the $L^2$-norm. The results in [16] were generalized to the constrained case by Ahn et al. [1]. Rababah et al. [17] used the bases transformations between Chebyshev and Bernstein for multi-degree reduction of Bézier curves. They used the
bases transformations between Bernstein and Jacobi bases in [18]. Woźniak and Lewanowicz [22] used the dual constrained Bernstein basis associated with the Jacobi scalar product for constrained multi-degree reduction of Bézier curves.

Lately, the concept of geometric continuity has been combined with degree reduction. Lu and Wang [15] presented an optimal multi-degree reduction of Bézier curves with $G^2$-continuity. However, to find the solution, they had to solve a nonlinear equation of degree 8 using the conjugate gradient method to solve the minimization problem.

Lu et al. [14] developed a progressive iterative degree reduction method. Their method is not optimal nor $G^k$-continuous. At each iteration, they update the location of the control points of the degree reduced curve by computing a difference between the degree reduced curve and the original curve at the Greville abscissa; see Delgado [4]. This differs from our method, as at each iteration we update the location of the control points using least squares.

The additional conditions and requirements on the degree reduction made the solution of the problem more complicated.

In this paper, we construct an iterative algorithm to find the $G^2$ multi-degree reduction of Bézier curves. To avoid the non-linearity problem, we require $C^1$ continuity in addition to $G^2$ continuity at the boundaries. At each iteration, we solve two least squares equations; the first adjusts the Bézier control points, while the second adjusts the continuity parameters. To measure the error, the $L_2$-norm is used. This algorithm converges, and it did well in all the examples we tried.

The outline of this paper is as follows. The Bernstein polynomials are given in Section 2. The Bézier curves together with some examples to illustrate our method in Section 6 and some conclusions in Section 7.

2. Bernstein polynomials

The Bernstein polynomials of degree $n$ are defined by

$$B^n_i(t) = \binom{n}{i} (1 - t)^{n-i} t^i, \quad i = 0, 1, \ldots, n.$$  

For the derivation of our iterative method, the following properties of the Bernstein polynomials will be needed. The value of the integral of $B^n_i(t)$ for all $i = 0, \ldots, n$ is given by

$$\int_0^1 B^n_i(t) dt = \frac{1}{n+1}.$$  

The product of two Bernstein polynomials is also a Bernstein polynomial and is given by

$$B^n_i(t) B^n_j(t) = \binom{m+n}{i+j} B^n_{i+j}(t).$$  

These products will be integrated, and it is convenient to define the following $(m + 1) \times (n + 1)$-matrix $G_{m,n}$. The elements of $G_{m,n}$ are given by

$$g_{ij} = \int_0^1 B^n_i(t) B^n_j(t) dt = \binom{m}{i} \binom{n}{j} \binom{m+n}{i+j}, \quad i = 0, \ldots, m, \quad j = 0, \ldots, n.$$  

The matrix $G_{m,n}$ is the Gram matrix, which is real, symmetric, and positive definite; see [19].

3. Bézier curves

In this section, Bézier curves are defined and some of their properties are stated; for more details, see Farin [5], Hoschek-Lasser [6], and Farouki-Rajan [10]. At the end of the section, the concept of geometric continuity is reviewed.

A Bézier curve $P_n(t)$ of degree $n$ with Bézier control points $(p_i)_{i=0}^n$ is given by

$$P_n(t) = \sum_{i=0}^n p_i B^n_i(t), \quad 0 \leq t \leq 1.$$  

It is convenient to write the Bézier curve in vector form by defining the vector of Bézier points $P^t = [p_0, p_1, \ldots, p_n]$ and the vector of Bernstein polynomials $B^n = [B^n_0(t), B^n_1(t), \ldots, B^n_n(t)]$; therefore, $P_n(t) = B^n P^t$.

The operator $\Delta$ that acts on the Bézier points is defined in standard form as follows: $\Delta^0 p_i = p_i$, $\Delta p_i = p_{i+1} - p_i$, $\Delta^k p_i = \Delta^{k-1}(\Delta p_i)$, $k \geq 1$, $i = 0, 1, \ldots, n - k$. This operator is used to express the $k$th derivatives of the Bézier curve in (2) at the end points $t = 0, 1$ as follows:
\[
\frac{d^k}{dt^k} P_n(0) = \frac{n!}{(n-k)!} \Delta^k p_0, \quad \frac{d^k}{dt^k} P_n(1) = \frac{n!}{(n-k)!} \Delta^k p_{n-k}, \quad k = 0, \ldots, n.
\]

For Bézier curves, degree reduction is used to approximate \( P_n(t) \) by the Bézier curve
\[
R_m(t) = \sum_{i=0}^{m} r_i B^n_i(t) =: B_m R, \quad 0 \leq t \leq 1,
\]
of degree \( m \), \( m < n \). Our goal in this paper is to degree reduce the curve \( P_n(t) \) by \( R_m(t) \), where the two curves are \( C^2 \)-continuous at the end points.

Two Bézier curves \( P_n \) and \( R_m \) are \( C^k \)-continuous at \( t = 0, 1 \) if there exist a strictly increasing parameterization \( s(t) : [0,1] \rightarrow [0,1] \) with \( s(0) = 0, s(1) = 1 \), and
\[
R_m^{(i)}(t) = P_n^{(i)}(s(t)), \quad t = 0, 1, \quad i = 0, 1, \ldots, k.
\]
In general, continuity of order \( k \) implies geometric continuity of order \( k \); however, the converse, in general, is not necessarily true.

4. \( C^2 \)-degree reduction

According to the conditions in (5), to get \( C^2 \)-continuity between \( P_n(t) \) and \( R_m(t) \) at \( t = 0, 1 \) the following conditions have to be fulfilled:
\[
R_m(t) = P_n(s(t)),
\]
\[
R_m'(t) = s'(t) P_n'(s(t)), \quad s'(t) > 0, \quad t = 0, 1,
\]
\[
R_m''(t) = (s'(t))^2 p_n''(s(t)) + s'(t) P_n'(s(t)).
\]
These conditions are simplified by substituting \( s'(t) = \delta^2_1 \), \( s''(t) = \delta_1 \), \( i = 0, 1 \) to get the following system of equations:
\[
R_m(i) = P_n(i),
\]
\[
R_m'(i) = \delta^2_1 P_n'(i),
\]
\[
R_m''(i) = \delta^3_1 P_n''(i) + \delta_1 P_n'(i).
\]

Unfortunately, this system of equations becomes non-linear in \( \delta_1, i = 0, 1 \). Instead, we consider the following approach. We simplify this system by considering \( C^1 \)-continuity, i.e., taking \( \delta_1 = 1, i = 0, 1 \); this approach simplifies the system and avoids the instability, complexity, and difficulties in computing the degree reduced curve. We call this method \( C^1/G^2 \)-degree reduction.

Substituting these in Eq. (7) and applying (3) yields
\[
r_0 = p_0, \quad r_m = p_n,
\]
\[
r_1 = p_0 + \frac{n}{m} \Delta p_0, \quad r_{m-1} = p_n - \frac{n}{m} \Delta p_{n-1},
\]
\[
r_2 = 2r_1 - r_0 + \frac{n(n-1)}{m(m-1)} \Delta^2 p_0 + \frac{n}{m(m-1)} \Delta p_0 s_0,
\]
\[
r_{m-2} = 2r_{m-1} - r_m + \frac{n(n-1)}{m(m-1)} \Delta^2 p_{n-2} + \frac{n}{m(m-1)} \Delta p_{n-1} s_1.
\]
So far, the points \( r_0, r_1, r_2, r_{m-2}, r_{m-1}, \) and \( r_m \) are determined by the boundary conditions; accordingly, it is proper to decompose the elements of \( R \) into two parts: the part of the continuity conditions at the boundaries \( R^c = (r_0, r_1, r_2, r_{m-2}, r_{m-1}, r_m) \) and the interior part \( R^f = (r_3, \ldots, r_{m-3}) \). Similarly, \( B_m \) is decomposed alike into \( B_m^c \) and \( B_m^f \). The \( L^2 \)-norm is used to measure the distance between \( R_m \) and \( P_n \); consequently, the error term becomes
\[
\varepsilon = \int_0^1 \| B_m^c P - B_m^c R \|^2 dt = \int_0^1 \| B_m^f P - B_m^f R^f - B_m^c R^c \|^2 dt.
\]
The minimum occurs when the partial derivatives vanish. Differentiating \( \varepsilon \) with respect to \( R^f \) and equating to zero gives the following system:
\[
G_{m,m}^f R^f = C_{m,n}^f P - C_{m,m}^f R^c,
\]
where
\[ G'_{m,n} := G_{m,n}(3, \ldots, m - 3; 3, \ldots, m - 3), \]
\[ G''_{m,n} := G_{m,n}(3, \ldots, m - 3; 0, 1, \ldots, n), \]
\[ G'''_{m,n} := G_{m,n}(3, \ldots, m - 3; 0, 1, 2, m - 2, m - 1, m). \]

and \[ G_{m,n}(\ldots) \] is the sub-matrix of \( G_{m,n} \) formed by the indicated rows and columns.

The matrix \( G'_{m,n} \) is real, symmetric, and positive definite; therefore, Eq. (13) can be solved explicitly for \( R' \) as follows
\[ R' = (G'_{m,n})^{-1}(G_{m,n}P - G_{m,m}R^e). \]

This solution for \( R' \) involves the two parameters \( s_0 \) and \( s_1 \).

Thereafter, the values of \( R' \) and \( R^e \) are substituted into the objective function (12); thus the objective function contains \( s_0 \) and \( s_1 \) as unknown parameters linearly. Differentiating the objective function (12) with respect to the parameters \( s_0 \) and \( s_1 \), respectively, gives the following two equations:
\[ \left( G_{m,n,3}P - G''_{m,m,3}R^e - r_2 \begin{pmatrix} m \\ 2 \\ m \\ 2 \\ m \\ 2 \\ m \end{pmatrix} - r_{m-2} \begin{pmatrix} m \\ 2 \\ m \\ 2 \\ m \end{pmatrix} \right) \cdot \Delta p_0 = 0, \]
\[ \left( G_{m,n,m-1}P - G'''_{m,m,m-1}R^e - r_2 \begin{pmatrix} m \\ 2 \\ m \\ 2 \\ m \\ 2 \\ m \end{pmatrix} - r_{m-2} \begin{pmatrix} m \\ 2 \\ m \\ 2 \\ m \end{pmatrix} \right) \cdot \Delta p_{n-1} = 0, \]

where \( G_{m,n,i} \) is the \( i \)th row of \( G_{m,n} \), \( G'_{m,m,i} \) is the \( i \)th row of \( G_{m,m} \) after extracting the 3rd and \( (m - 1) \)st columns, and \( R^e \) is that part of \( R \) after extracting the 3rd and \( (m - 1) \)st elements.

In this stage of the solution, the minimum is guaranteed by the least squares method. Moreover, having better solutions of \( R \) will lead to having less deviation from the curve \( P_x(t) \). Solving this linear system for \( s_0 \) and \( s_1 \) and substituting these values of \( s_0 \) and \( s_1 \) into \( r_2, r_{m-2} \) and into \( R^e \) produces a \( G^2 \)-degree reduced curve.

5. Iterative method

This approach suggests an iterative method to find the \( G^2 \)-degree reduction. We proceed as follows. We begin with the values of \( R \) from a \( G^2 \)-degree reduced curve, i.e., choosing \( s_0 = s_1 = 1 \); after that, Eqs. (15) and (16) are used to find the values of \( s_0 \) and \( s_1 \). These values are substituted in the equations for \( r_2, r_{m-2} \) together with the Eq. (14) of \( R' \). This gives the Bézier points \( R \) of the \( G^2 \)-degree reduced Bézier curve \( R_m \). These new points are used in the new stage and this process is repeated until a satisfactory solution is reached. We summarize the method in the following algorithm:

---

The \( C^1/G^2 \)-degree reduction Algorithm:

**Given:** a Bézier curve of degree \( n \) as in (2).

**Find:** a \( G^2 \)-degree reduced curve of degree \( m \) as in (4).

**Initial Values:** (A \( C^2 \)-degree reduced curve of degree \( m \).) Set \( s_0 = s_1 = 1 \) and find \( R \) from Eqs. (8), (9), (10), (11), and (14).

**STEP I:** Solve the two Eqs. (15) and (16) for \( s_0 \) and \( s_1 \).

**STEP II:** Update \( R \) from (10), (11), and (14).

**STEP III:** Repeat STEPS I and II until a satisfactory solution is reached.

---

The complexity of the \( C^2 \) method of Ahn et al. [1] is dominated by the cost of inverting \( G'_{m,n} \); since this matrix is fixed (for any given \( m \)), the inverse of \( G_{m,n} \) could be precomputed, reducing the cost to the cost of the matrix multiplies in (14). The interior loop of our method performs the same matrix multiplies, so our method is more expensive than the \( C^2 \) method by a factor of the number of iterations; see also Rababah [20].

The \( G^2 \) method of Lu-Wang [15] also evaluates (14). Further, it requires using a non-linear technique to minimize a degree 8 polynomial with four unknowns. Thus, both our method and the \( G^2 \) method require evaluating (14), with the difference in the cost of the two methods being that \( G^2 \) method has the additional cost of minimizing the non-linear equation, while our method only has the additional cost of solving a \( 2 \times 2 \) system of linear equations.

**Remarks:**

(1) There are some similarities between our steps I and II and two steps used in [15], in that they also solve for the geometric continuity parameters in one step and for the control points in a second step. However, Lu-Wang’s step I is non-linear and they do not discuss repeating these steps, resorting instead to subdivision of the curve to improve the approximation (note that subdivision would also work using our method).
(2) With regards to the solvability of the linear system in (15) and (16), the matrix of coefficients has the following form

\[
\begin{pmatrix}
\frac{\Delta p_0 \Delta p_0}{m^2} & \frac{\Delta p_0 \Delta p_{n-1}}{m} & \frac{\Delta p_0 \Delta p_{n-1}}{4} \\
\frac{\Delta p_{n-1} \Delta p_{n-1}}{m^2} & \frac{\Delta p_{n-1} \Delta p_{n-1}}{m} & \frac{\Delta p_{n-1} \Delta p_{n-1}}{4} \\
\frac{\Delta p_0 \Delta p_{n-1}}{m^2} & \frac{\Delta p_0 \Delta p_{n-1}}{m} & \frac{\Delta p_0 \Delta p_{n-1}}{4}
\end{pmatrix}
= \begin{pmatrix}
\frac{\|\Delta p_0\|^2}{m^2} & \frac{\|\Delta p_0\| |\Delta p_{n-1}| |\cos \theta|}{m} & \frac{\|\Delta p_0\| |\Delta p_{n-1}|}{4} \\
\frac{\|\Delta p_{n-1}\| |\Delta p_{n-1}| |\cos \theta|}{m^2} & \frac{\|\Delta p_{n-1}\| |\Delta p_{n-1}|}{m} & \frac{\|\Delta p_{n-1}\|}{4} \\
\frac{\|\Delta p_0\| |\Delta p_{n-1}|}{m^2} & \frac{\|\Delta p_0\| |\Delta p_{n-1}|}{m} & \frac{\|\Delta p_0\|}{4}
\end{pmatrix},
\]

where \(\theta \in [0, \pi]\) is the angle between \(\Delta p_0\) and \(\Delta p_{n-1}\). Usually, \(\|\Delta p_0\|\) and \(\|\Delta p_{n-1}\|\) are close in value; consequently, this matrix is diagonally dominant for all \(m \geq 2\). If \(\|\Delta p_0\|\) and \(\|\Delta p_{n-1}\|\) are not close in value, then either \(\|\Delta p_0\|^2\) or \(\|\Delta p_{n-1}\|^2\) is greater in value than \(\|\Delta p_0\| \|\Delta p_{n-1}\| \cos \theta\). Again in this case one of the diagonal elements can be used as a pivoting element.

Another factor is the fact that \(\frac{2m}{4} \ll \frac{2m}{m}\), for large values of \(m\), which gives additional weight to the diagonal of the matrix of coefficients.

(3) The fact that the matrix of coefficients \(G_{n,m}\) is non-singular guarantees the consistency of the linear system (13).

(4) The matrix \(G_{n,m}\) satisfies the following conditions: All of its elements are strictly positive, and the sum of each row and each column is less than \(\frac{1}{m+n}\), see [19]. Therefore, \(\|G_{n,m}\|_2 < 1\); consequently, since the spectral radius \(\rho(G_{n,m}) \leq \|G_{n,m}\|_2\), we get \(\rho(G_{n,m}) < 1\), which is the required condition for the convergence of the iterative method based on (13).

6. Examples

We give several examples to show how well our method works. Since our method is iterative and starts from the \(C^2\) method, we give a simple example showing how much improvement our method has after a single iteration.

We then compare our method to the \(C^2\) method of Lu-Wang; while we know that the error of the Lu-Wang method will never be larger than the error of our method, the example indicates that the improvement in the Lu-Wang method compared to ours will be small relative to the reduction in error from the \(C^2\) method to our method.

We then give some additional examples showing how the error decreases as we increase the number of iterations in our method.

As a first test of our \(C^1/G^2\) method, we degree reduce an “S” curve, shown in Fig. 1. In this figure, we see that our iterative \(C^1/G^2\) method gives a significantly better approximation after a single iteration than the \(C^2\) method (by “iteration”, we mean one application of Step I of our algorithm followed by one application of Step II). In this example, our method does not improve significantly after more than one iteration. In general (and as seen in the next two examples), multiple iterations are needed to converge, where the number of iterations will depend on the complexity of the curve being approximated as well as the degree of the approximation.

As a second example, we use a dataset of Lu and Wang [15], which they used to compare their non-linear \(C^2\) method to the \(C^2\) method in [1]; there a degree 10 curve was reduced to degree 6. Fig. 2 compares our \(C^1/G^2\) method (long dashed curve) to the \(C^2\) method (short dashed curve) in [1] and to the \(C^2\) method (dotted curve) in [15].1 From this figure, we see that both our \(C^1/G^2\) method and the \(C^2\) method in [15] produce curves that are better approximations to the original; the method in [15] produces in some places a slightly better approximation than our method.

From this example, we see that the first iteration of the \(C^1/G^2\) method gives a noticeable improvement over the \(C^2\) method; each successive iteration gives a smaller improvement. In this example, our method converged to pixel resolution after 30 iterations. Table 1 gives the \(L_2\) and \(l_\infty\) errors for the \(C^2\) approximation, the \(G^2\) approximation, and for our iterative method.

1 Our curve plot looks different than the one that appears in [15] because we used a uniform scale of the axes while they used a non-uniform scale.
for 1, 2, 5, 10, 20, and 30 iterations; a plot of these errors appears in Fig. 3. Further, in our experience, the number of iterations until convergence increases as the degree \( m \) increases.

As a third example, we use the “heart” curve in Fig. 4, where the solid curve is the original degree 13 curve; the long dashed curve is the \( C^2 \) degree 8 approximation; and the short dashed curve is our \( C^1/G^2 \) degree 8 approximation. In this example, we see that the extra degrees of freedom in our \( C^1/G^2 \) scheme allow us to reproduce the inner loop in this heart that the \( C^2 \) scheme was unable to reproduce.

### Table 1

<table>
<thead>
<tr>
<th></th>
<th>( C^2 )</th>
<th>( C^1/G^2 ), iterations</th>
<th>( G^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_2 )</td>
<td>0.03665</td>
<td>0.03299</td>
<td>0.02490</td>
</tr>
<tr>
<td>( L_\infty )</td>
<td>0.06644</td>
<td>0.05572</td>
<td>0.05041</td>
</tr>
</tbody>
</table>

**Fig. 2.** Reduction from degree 10 to degree 6 with \( C^2 \), \( C^1/G^2 \), and \( G^2 \) continuity. Left: 1 iteration for \( C^1/G^2 \); right: 30 iterations for \( C^1/G^2 \).

**Fig. 3.** \( L_2 \) error (bottom curve) and \( L_\infty \) error (top curve) versus iterations for approximations to Lu-Wang curve. Zero iterations correspond to the \( C^2 \) method. The \( L_2 \) (black circles) and \( L_\infty \) (white circles) are also marked for the \( C^2 \) and fully \( G^2 \) methods.

for 1, 2, 5, 10, 20, and 30 iterations; a plot of these errors appears in Fig. 3. Further, in our experience, the number of iterations until convergence increases as the degree \( m \) increases.

As a third example, we use the “heart” curve in Fig. 4, where the solid curve is the original degree 13 curve; the long dashed curve is the \( C^2 \) degree 8 approximation; and the short dashed curve is our \( C^1/G^2 \) degree 8 approximation. In this example, we see that the extra degrees of freedom in our \( C^1/G^2 \) scheme allow us to reproduce the inner loop in this heart that the \( C^2 \) scheme was unable to reproduce.
7. Conclusions

An iterative method has been introduced to find the multi-degree reduction of Bézier curves. This method is $C^1$- and $C^2$-continuous at the end points of the curve. Our $C^1/C^2$ method can be easily applied to find approximative degree reduced Bézier curves of lower degree and small error.

The examples show that our $C^1/C^2$ method is significantly better than the $C^2$ method of Ahn et al. [1]; their $C^2$ method failed to reproduce the inner loop in one example, while our $C^1/C^2$ method reproduced this loop correctly. Further note that our iterative $C^1/C^2$ algorithm is only slightly more complex than Ahn et al.'s $C^2$ algorithm: each iteration of our algorithm solves a $2 \times 2$ system of equations and then solves a least squares problem similar to that of Ahn et al.

Further, the examples and figures show that our $C^1/C^2$ algorithm produces results nearly as good as the non-linear $G^2$ method of Lu-Wang [15]. While the Lu-Wang method affords in some places a slightly better approximation than our method, they have to use the conjugate gradient method to solve systems of degree 8 equations, while our method only requires solving linear equations.

Our method can be extended from the unweighted $L_2$-norm to the weighted $L_2$-norm

$$\int_0^1 (1 - t)^{2\beta} \| P_n(t) - R_m(t) \|^2 dt, \quad \alpha, \beta > -1,$$

as was done in the unconstrained case by Rababah et al. [18]. It is known that using $\alpha, \beta$ close to $-1$ will improve the approximation near the boundaries, while large $\alpha, \beta$ will improve the approximation at the middle of the curve. Since our constrained method already has good approximation near the boundaries, using non-negative $\alpha, \beta$ should improve the overall approximation. However, use of this norm will complicate our approach, and requires that the integrals be approximated with a numerical method, which likely rules out the weighted norm for industrial applications.

Acknowledgements

The authors thank the referees for their valuable comments. This work was funded in part by the National Research Council of Canada.

References