DEGREE ELEVATION OF NURBS CURVES BY WEIGHTED BLOSSOM

BYUNG-GOOK LEE AND YUNBEOM PARK

Abstract. An algorithmic approach to degree elevation of NURBS curves is presented. The new algorithms are based on the weighted blossoming process and its matrix representation. The elevation method is introduced that consists of the following steps: (a) decompose the NURBS curve into piecewise rational Bézier curves, (b) degree elevate each rational Bézier piece, and (c) compose the piecewise rational Bézier curves into NURBS curve.

1. Introduction

NURBS(Non-uniform rational B-splines) are parametric objects at the form of a fraction, with polynomial B-splines in nominator and denominator. The nominator and denominator are of the same degree, and defined on the same knot vector. The denominator is always a B-spline in $\mathbb{R}^1$ and we will restrict it futher by saying that all its vertices are positive. It is convenient to introduce rational B-splines by a projective mapping from a space of polynomial B-spline of higher dimension.

Let $\mathbb{R}^3_+ = \{(x, y, w) \in \mathbb{R}^3 : w > 0\}$. We construct $\mathbb{P}^2$, the projective space of dimension 2, by identifying all points in $\mathbb{R}^3$ on the same line through the origin. This is also called a space of homogeneous coordinates. For each element $\mathbb{P}^2$ with $w \neq 0$ we may choose a representative al the form $(x, y, 1)$. We will not be concerned with the other elements of $\mathbb{P}^2$, the points of infinity.

The connection between $\mathbb{R}^3_+$ and $\mathbb{P}^2$ is described by the mapping

$$\psi : \mathbb{R}^3_+ \to \mathbb{P}^2 \text{ where } \psi(x, y, w) = (\frac{x}{w}, \frac{y}{w}, 1).$$

Further, we have the natural projection

$$\pi : \mathbb{P}^2 \to \mathbb{R}^2 \text{ where } \pi(x, y, 1) = (x, y),$$

and the composition of these gives the mapping

$$\Psi = \pi \circ \psi : \mathbb{R}^3_+ \to \mathbb{R}^2 \text{ where } \Psi(x, y, w) = (\frac{x}{w}, \frac{y}{w}).$$

1991 *Mathematics Subject Classification.* 65D17, 68U05.

Key words and phrases. NURBS, Weighted Blossom, Knot Insertion, Subdivision, Degree Elevation.

This work was supported by Korea Research Foundation Grant(KRF-99-015-DP0037).

The mapping is clearly not one-to-one. We observe that the inverse image of a point is a line and the inverse image of a curve is the cone of the curve.

Let $B(u) = \sum_{i=0}^{n} B_i N_{i,p}(u)$ be a p-th degree B-spline curve, where the $\{B_i\}$ are the control points in $\mathbb{R}^3$, and the $\{N_{i,p}(u)\}$ are the p-th degree B-spline basis functions defined on the non-uniform knot vector

$$U = \{\underbrace{a, a, \ldots, a}_{p+1}, u_{p+1}, u_{p+2}, \ldots, u_{m-p-1}, \underbrace{b, b, \ldots, b}_{p+1}\}.$$

If all B_i are in $\mathbb{R}^3_+$, then clearly $B(u)$ is in $\mathbb{R}^3_+$, and the mapping $\Psi(B(u))$ is well defined. We may write the B_i's at the form $(x_i w_i, y_i w_i, w_i)$. We define a p-th degree two dimensional rational B-spline curve as the image under Ψ of a polynomial B-spline curve in $\mathbb{R}^3_+$:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}, \quad a \le u \le b$$

where the $P_i = \Psi(B_i) = (x_i, y_i)$.

2. Definition of the Weighted Blossom

Let $G(u)$ be a polynomial curve of degree p or less. The weighted blossom $B_G(u_1, u_2, \ldots, u_p : w)$ of the polynomial $G(u)$ is the unique multivariate polynomial with the following properties:

1. *multiaffine* : for all indices i and all real number c

$$\begin{aligned} &B_G(u_1, \ldots, u_{i-1}, cu + (1-c)v, u_{i+1}, \ldots, u_p : cw_1 + (1-c)w_2) \\ &= \frac{cw_1}{cw_1 + (1-c)w_2} B_G(u_1, \ldots, u_{i-1}, u, u_{i+1}, \ldots, u_p : w_1) \\ &+ \frac{(1-c)w_2}{cw_1 + (1-c)w_2} B_G(u_1, \ldots, u_{i-1}, v, u_{i+1}, \ldots, u_p : w_2) \end{aligned}$$

2. *symmetry* : for any permutation π of $\{1, 2, \ldots, p\}$

$$B_G(u_1, u_2, \ldots, u_p : w) = B_G(u_{\pi(1)}, u_{\pi(2)}, \ldots, u_{\pi(p)} : w)$$

3. *diagonal* : B_G reduces to G when evaluated on its diagonal : i.e.

$$G(u) = B_G(u, u, \ldots, u : w)$$

4. *dual functional* : any control vertex

$$P_i = B_G(u_{i+1}, u_{i+2}, \ldots, u_{i+p} : w) \text{ with weight } w$$

Let us consider first the de Casteljau algorithm to subdivide rational Bézier curves. By the dual functional property the control vertices P_i for the rational Bézier representation of the curve are given in terms of the blossom by $P_i = B_G(0, \ldots, 0, 1, \ldots, 1 : w_i)$ where 0 appears as an argument $n - i$ times and 1 appears i times.

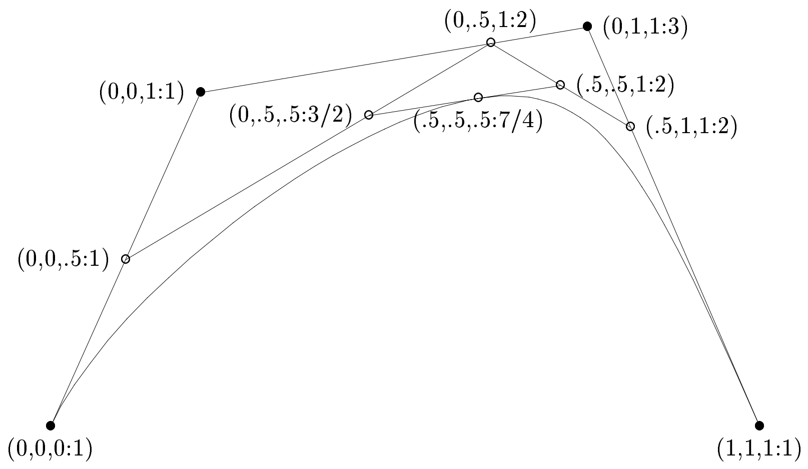

FIGURE 1. Subdivision of the cubic rational Bézier curve with $a = 0.5$, ($w_0 = 1$, $w_1 = 1$, $w_2 = 3$, $w_3 = 1$).

For the sake of simplicity, consider the cubic case. We begin with the control vertices $B_G(0,0,0 : w_0)$, $B_G(0,0,1 : w_1)$, $B_G(0,1,1 : w_2)$, and $B_G(1,1,1 : w_3)$. From these we wish to get new control vertices representing the curve over segments$[0,a]$ and $[a,1]$, respectively, as rational Bézier curves. The new control vertices are $B_G(0,0,0)$, $B_G(0,0,a)$, $B_G(0,a,a)$, $B_G(a,a,a)$, $B_G(a,a,1)$, $B_G(a,1,1)$, and $B_G(1,1,1)$. The question is now how to find all the new control vertices from the old control vertices. The multiaffine property provide us to derive new blossom values from old ones. For example, consider finding the value $B_G(0,0,a : w)$. Note

$$B_G(0,0,a : w) = B_G(0,0,(1-a)0 + a1 : (1-a)w_0 + aw_1)$$

$$= \frac{(1-a)w_0}{(1-a)w_0 + aw_1} B_G(0,0,0 : w_0) + \frac{aw_1}{(1-a)w_0 + aw_1} B_G(0,0,1 : w_1)$$

expressing one of the new control vertices in terms of two old control vertices.

In the figure 1, to conserve space the points are labelled using only the blossom arguments. The parameter $a(=.5)$ subdivides the interval $[0,1]$ into two subintervals; similarly, the point $b^n(.5)$ subdivideds the curve segment $b^n(0)$, $b^n(1)$ into two subsegments. Hence, we can conclude directly that : the rational Bézier curve segment $b^n(0)$, $b^n(.5)$ has the control polygon B_G(0,0,0), B_G(0,0,.5), B_G(0,.5,.5), B_G(.5,.5,.5), with weights 1, 1, 3/2, 7/4, and the rational Bézier curve segment $b^n(.5)$, $b^n(1)$ has the control polygon $B_G(.5,.5,.5)$, $B_G(.5,.5,1)$, $B_G(.5,1,1)$, $B_G(1,1,1)$, with weights 7/4, 2, 2, 1.

As a second example, consider the de Boor algorithm for evaluating a NURBS curve at value $a \in [u_i, u_{i+1})$. The input to the algorithm, expressed in terms of blossom, is the control vertices $B_G(u_{i-2}, u_{i-1}, u_i :$

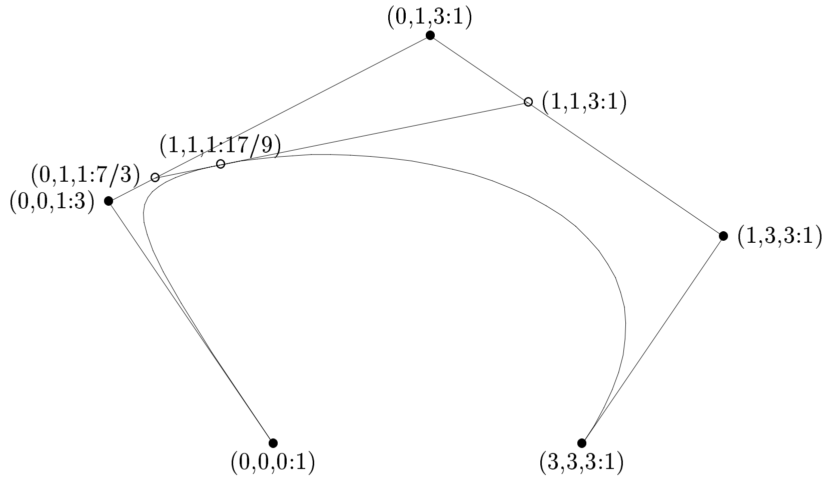

FIGURE 2. Applying the de Boor algorithm to a cubic NURBS curve.

$w_1)$, $B_G(u_{i-1}, u_i, u_{i+1} : w_2)$, $B_G(u_i, u_{i+1}, u_{i+2} : w_3)$, and $B_G(u_{i+1}, u_{i+2}, u_{i+3} : w_4)$ governing the segment of the curve over $[u_i, u_{i+1})$. The output will be the point $B_G(a, a, a : w)$ on the curve. We can get from the original control vertices to the point on the curve by a set of affine combinations as shown in the figure 2. For example, consider finding $B_G(u_{i-1}, u_i, a : w)$ from $B_G(u_{i-2}, u_{i-1}, u_i : w_1)$, and $B_G(u_{i-1}, u_i, u_{i+1} : w_2)$.

$$B_G(u_{i-1}, u_i, a : w) = B_G(u_{i-1}, u_i, cu_{i-2} + (1-c)u_{i+1} : cw_1 + (1-c)w_2)$$

$$= \frac{cw_1}{cw_1 + (1-c)w_2} B_G(u_{i-2}, u_{i-1}, u_i : w_1) + \frac{(1-c)w_2}{cw_1 + (1-c)w_2} B_G(u_{i-1}, u_i, u_{i+1} : w_2)$$

$$\text{where } c = \frac{u_{i+1} - a}{u_{i+1} - u_{i-2}}.$$

The curve in the example is over the knot vector $(0, 0, 0, 0, 1, 3, 3, 3, 3)$ with weights 1,3,1,1,1, and is evaluated at $a = 1$.

As a third example, consider the knot insertion algorithm. It is very convenient in curve design when it is necessary to apply fine shape control. Suppose that we insert a new knot 2 in the knot vector in the figure 3. The curve is over the knot vector $(0, 0, 0, 0, 1, 3, 3, 3, 3)$ with weights 1,3,1,1,1, and is evaluated at $a = 2$. From these we can get new control vertices $B_G(0, 0, 0 : v_0)$, $B_G(0, 0, 1 : v_1)$, $B_G(0, 1, 2 : v_2)$, $B_G(1, 2, 3 : v_3)$, $B_G(2, 3, 3 : v_4)$, and $B_G(3, 3, 3 : v_5)$ over the knot vector (0,0,0,0,1,2,3,3,3,3)

$$\begin{aligned}
B_G(0,0,0 : v_0) &= B_G(0,0,0 : 1) &&= \tfrac{1}{v_0} w_0 B_G(0,0,0 : 1)\\
B_G(0,0,1 : v_1) &= B_G(0,0,1 : 3) &&= \tfrac{1}{v_1} w_1 B_G(0,0,1 : 3)\\
B_G(0,1,2 : v_2) &= B_G(0,1,\tfrac{1}{3}0 + \tfrac{2}{3}3 : \tfrac{1}{3}3 + \tfrac{2}{3}1) &&= \tfrac{1}{v_2}(\tfrac{1}{3} w_1 B_G(0,0,1 : 3) + \tfrac{2}{3} w_2 B_G(0,1,3 : 1))\\
B_G(1,2,3 : v_3) &= B_G(1,\tfrac{1}{3}0 + \tfrac{2}{3}3, 3 : \tfrac{1}{3}1 + \tfrac{2}{3}1) &&= \tfrac{1}{v_3}(\tfrac{1}{3} w_2 B_G(0,1,3 : 1) + \tfrac{2}{3} w_3 B_G(1,3,3 : 1))\\
B_G(2,3,3 : v_4) &= B_G(\tfrac{1}{2}1 + \tfrac{1}{2}3, 3, 3 : \tfrac{1}{2}1 + \tfrac{1}{2}1) &&= \tfrac{1}{v_4}(\tfrac{1}{2} w_3 B_G(1,3,3 : 1) + \tfrac{1}{2} w_4 B_G(3,3,3 : 1))\\
B_G(3,3,3 : v_5) &= B_G(3,3,3 : 1) &&= \tfrac{1}{v_5} w_4 B_G(3,3,3 : 1)
\end{aligned}$$

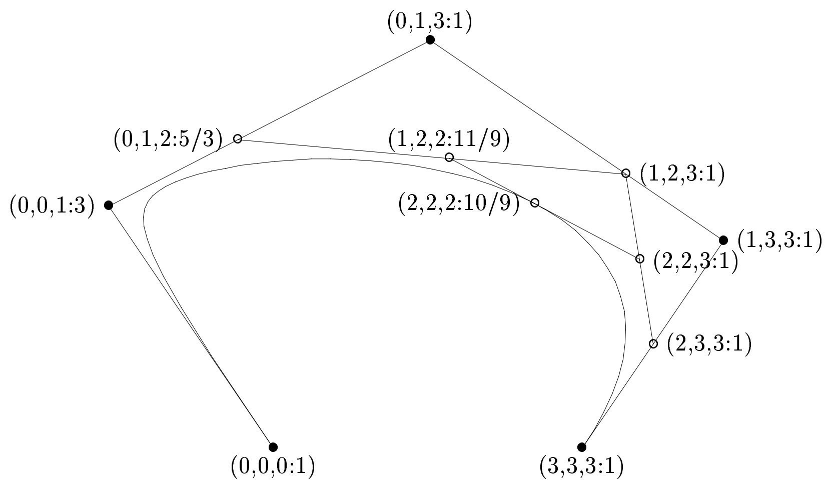

FIGURE 3. Applying the de Boor algorithm to a cubic NURBS curve.

The knot insertion algorithm in the figure 3 is expressed in a matrix form as

$$V = X^w W,\ Q = X^p P$$

where

$$V = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{pmatrix}, X^w = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1/3 & 2/3 & 0 & 0 \\ 0 & 0 & 1/3 & 2/3 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, W = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix},$$

$Q = (B_G(0,0,0), B_G(0,0,1), B_G(0,1,2), B_G(1,2,3), B_G(2,3,3), B_G(3,3,3))^t$,
$P = (B_G(0,0,0), B_G(0,0,1), B_G(0,1,3), B_G(1,3,3), B_G(3,3,3))^t$,
$X^p = diag(1/V) X^w diag(W)$,

$$diag(1/V) = \begin{pmatrix} 1/v_0 & & & \\ & 1/v_1 & & \\ & & \ddots & \\ & & & 1/v_5 \end{pmatrix}, \text{and } diag(W) = \begin{pmatrix} w_0 & & & \\ & w_1 & & \\ & & \ddots & \\ & & & w_4 \end{pmatrix}.$$

Curve decomposition is normally done via knot insertion. Consider the cubic NURBS curve in the figure 2. Here, the rational Bézier segments are extracted by inserting knot 1 two times. From these we can get first rational Bézier piece $B_G(0,0,0)$, $B_G(0,0,1)$, $B_G(0,1,1)$, $B_G(1,1,1)$ with weights $1, 3, 7/3, 17/9$, and second rational Bézier piece $B_G(1,1,1)$, $B_G(1,1,3)$, $B_G(1,3,3)$, $B_G(3,3,3)$ with weights $17/9, 1, 1, 1$.

Now, we consider the knot removable algorithm. It is the reverse process of inserting a knot. While knot insertion is a precise procedure, i.e. the knot-inserted curve is precisely the same as the original one, knot removal, in general, procedures an approximation of the original curve. Clearly, after a knot inserted, it can be removed precisely. Consider the curve shown in the figure 4. The knot $u = 1$ is removable two times since the curve is C^2-continuous at $u = 1$.

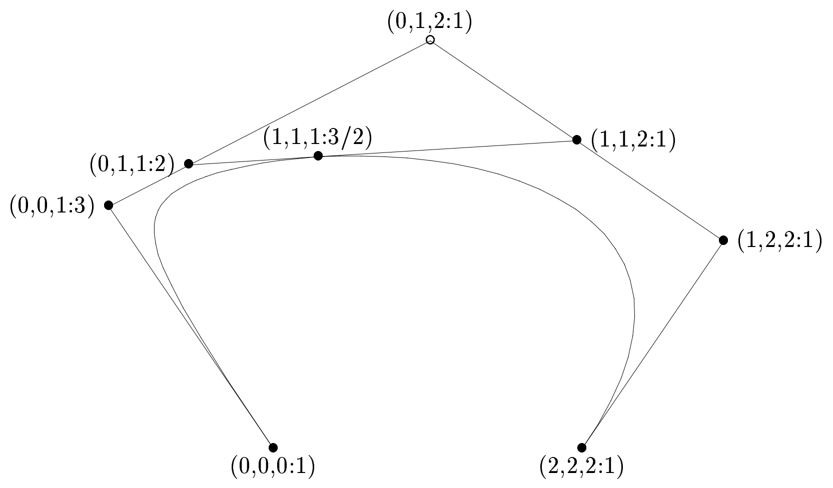

FIGURE 4. A cubic NURBS curve over the knot vector $(0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2)$ with weights 1,3,2,3/2,1,1,1, C^2-continous at $u = 1$

The knot removable algorithm in the figure 4 is expressed in a linear equation as

$$V = X^w W, \; Q = X^p P$$

where

$$V = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{pmatrix}, X^w = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 1/4 & 1/2 & 1/4 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, W = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix},$$

$Q = (B_G(0,0,0), B_G(0,0,1), B_G(0,1,1), B_G(1,1,1), B_G(1,1,2), B_G(1,2,2), B_G(2,2,2))^t$,
$P = (B_G(0,0,0), B_G(0,0,1), B_G(0,1,2), B_G(1,2,2), B_G(2,2,2))^t$, and
$X^p = diag(1/V)X^w diag(W)$.

Then, because X^w has full column rank and $(X^w)^t X^w$ is non-singular, $G^w = [(X^w)^t X^w]^{-1}(X^w)^t$ is a left inverse of the matrix X^w. Therefore the consistent equations $V = X^w W$ and $Q = X^p P$ have the unique solution

$$W = G^w V \text{ and } P = diag(1/W)G^w diag(V)Q.$$

3. Degree Elevation

Since a NURBS curve is a piecewise polynomial curve, it must be possible to evaulate its degree from p to $p+r$. That is, there must exist control points $\tilde{P}$, a knot vector $\tilde{U}$, and a weight vector $\tilde{W}$ such that

$$C(u) = \tilde{C}(u) = \frac{\sum_{i=0}^{\tilde{n}} N_{i,p+r}(u)\tilde{w}_i\tilde{P}_i}{\sum_{i=0}^{\tilde{n}} N_{i,p+r}(u)\tilde{w}_i}$$

The curve $C(u)$ and $\tilde{C}(u)$ are the same geometrically and parametrically. The computing of $\tilde{n}$,$\tilde{P}$, $\tilde{U}$, and $\tilde{W}$ is referred to as degree elevation. The knot vector $\tilde{U}$ and $\tilde{n}$ can easily be computed as follows. Assume that U has the form

$$U = \{\underbrace{a, a, \ldots, a}_{p+1}, \underbrace{u_1, \ldots, u_1}_{m_1}, \ldots, \underbrace{u_s, \ldots, u_s}_{m_s}, \underbrace{b, b, \ldots, b}_{p+1}\}$$

where the end knots a and b are repeated with multiplicity $p+1$ and the interior knots u_i are repeated with multiplicity m_i. Since the curve $C(u)$ is $C^{p-m_i}-$continuous at the knot of multiplicity m_i, $\tilde{C}$ must have the same continuity. Consequently, the new vector must take the form

$$\tilde{U} = \{\underbrace{a, a, \ldots, a}_{p+r+1}, \underbrace{u_1, \ldots, u_1}_{m_1+r}, \ldots, \underbrace{u_s, \ldots, u_s}_{m_s+r}, \underbrace{b, b \ldots, b}_{p+r+1}\}$$

which gives $\tilde{n} = n + (s+1)r$.

We provide a procedural method that can be summarized as follows:

(1) Decompose the NURBS curve into piecewise rational Bézier curves.
(2) Degree elevate each rational Bézier piece.
(3) Make the NURBS curve from the piecewise rational Bézier segment.

Curve Decomposition : Curve decomposition is normally done via knot insertion is very convenient in curve design when is necessary to apply fine sharp control. In the case of a NURBS curve a new knot can be inserted to increase the number of curve defining vectors and the number of curve segments. Consider the cubic NURBS curve in the figure 5. The knot 1 is inserted two times bring the total multiplicity to 3.

The Decomposition algorithm in the figure 5 is expressed in a matrix form as

$$V = M_d^w W, \ Q = M_d^p P$$

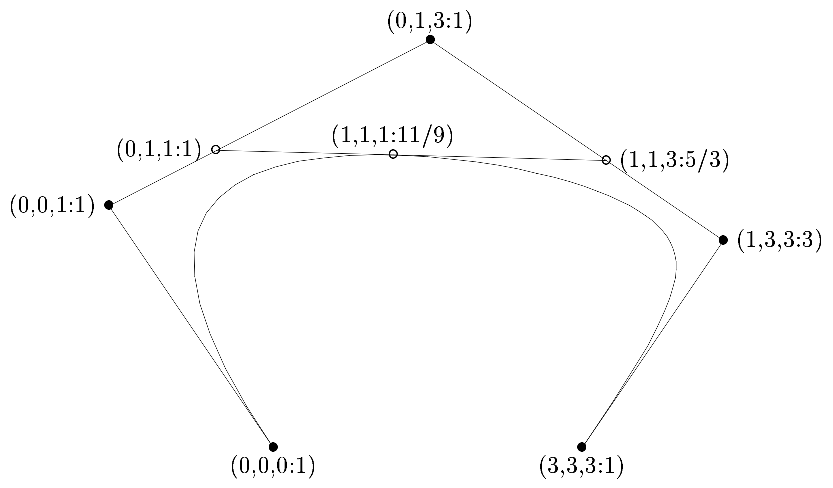

FIGURE 5. a cubic NURBS curve over the knot vector (0,0,0,0,1,3,3,3,3) with weights 1,1,1,3,1 to be degree elevated. Decomposed the NURBS curve into piecewise rational Bézier curves.

where

$$V = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{pmatrix}, M_d^w = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 2/3 & 1/3 & 0 & 0 \\ 0 & 4/9 & 4/9 & 1/9 & 0 \\ 0 & 0 & 2/3 & 1/3 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, W = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix},$$

$Q = (B_G(0,0,0), B_G(0,0,1), B_G(0,1,1), B_G(1,1,1), B_G(1,1,3), B_G(1,3,3), B_G(3,3,3))^t$,
$P = (B_G(0,0,0), B_G(0,0,1), B_G(0,1,3), B_G(1,3,3), B_G(3,3,3))^t$, and
$M_d^p = diag(1/V) M_d^w diag(W)$.

Next we give detailed pseudocode to compute the decomposition matrix M_d^w. It uses a local array V of size s to store the ith knot interior values and another one M of size s to store the multiplicity of the ith interior knots.

```
**Make_DecompositionMatrix**($U$, $m$, $p$)
// Input: Knots vector $U = (a, a, \ldots, a, u_{p+1}, \ldots, u_{m-p-1}, b, b, \ldots, b)$,
//        number of knots $m+1$ and degree $p$
// Output: $(ps+p+1) \times (n+1)$ matrix $M_d^w$

(* In case of Bezier curve *)
**if** $(m = 2p+1)$ **exit**;

(* initialize some variables *)
$s = 1$; $t = 1$; $l = p$; $n = m - p - 1$;
```

```
(* initialize $M_d^w$ matrix *)
**for** $(i = 0$ to $n$ by $1)$
   **for** $(j = 0$ to $n$ by $1)$
      **if** $(i = j)$ $M_d^w[i][j] = 1$; **else** $M_d^w[i][j] = 0$;

(* compute knot multiplicity *)
$V[s] = U[p+1]$; $M[s] = 1$;
**for** $(i = p+2$ to $n$ by $1)$
   **if** $(V[s] = U[i])$ $M[s] = M[s] + 1$;
   **else** $s = s + 1$; $V[s] = U[i]$; $M[s] = 1$;
   **endif**
**endfor**

(* make $M_d^w$ matrix *)
**for** $(i = 1$ to $s$ by $1)$
   $l = l + M[i]$;
   **for** $(j = M[i]$ to $p - 1$ by $1)$
      **KnotsInsertion** $(U, l, V[i], n, n+t, p)$;
      **for** $(k = m + t$ to $l + 1$ by $-1)$ $U[k] = U[k-1]$;
      $U[l] = V[i]$; $l = l + 1$; $t = t + 1$;
   **endfor**
**endfor**


**KnotsInsertion**$(U, l, v, n, k, p)$
// Input: Knots vector $U$, new knot $v(u_l \le v < u_{l+1})$,
//        number of control points $n+1$ and degree $p$

**for** $(i = l - p + 1$ to $l - 1$ by $1)$
   $\alpha = (v - U[i])/(U[i+p] - U[i])$;
   **for** $(j = 0$ to $n$ by $1)$
      $T[i][j] = (1-\alpha)M_d^w[i-1][j] + \alpha M_d^w[i][j]$;
   **endfor**
**endfor**
**for** $(i = k$ to $l$ by $-1)$
   **for** $(j = 0$ to $n$ by $1)$ $M_d^w[i][j] = M_d^w[i-1][j]$;
**for** $(i = l - p + 1$ to $l - 1$ by $1)$
   **for** $(j = 0$ to $n$ by $1)$ $M_d^w[i][j] = T[i][j]$;
```

Degree Elevation of rational Bézier Curves: The degree elevation of rational Bézier curves is well understood and well documented. As a first step, to raising the degree of the rational Bézier curve by one. We can show that weights of new control points are obtained from the old weights by piecewise linear interpolation at the parameter values $j/(n+1)$. We may repeat this process and obtain a sequence of control points. After r degree elevation, we have a linear system

$$U = T^w_{n,r} V, \ \ R = T^p_{n,r} Q,$$

where the $(n+r+1) \times (n+1)$ matrix $T^w_{n,r} = \{t_{i,j}\}$ has elements

$$t_{i+j,i} = \frac{\binom{n}{i}\binom{r}{j}}{\binom{n+r}{i+j}}, \quad \begin{cases} i = 0, 1, \ldots, n \\ j = 0, 1, \ldots, r. \end{cases}$$

and

$$T^p_{n,r} = diag(1/U) T^w_{n,r} diag(V).$$

Rational Bézier curves are known to be a special polynomial type of NURBS curve with the knot vector given by n knots at 0 and n knots at 1. By the dual functional property the control vertices Q_i for the rational Bézier representation of the curve are given in terms of blossom by $Q_i = B_G(0{,}0{,}\ldots{,}0{,}1{,}1{,}\ldots{,}1)$ where 0 appears as an argument $n-i$ times and 1 appears i times. For the sake of simplicity, consider the cubic case. We begin with the control vertices from the blossom representations $B_G(0,0,0)$, $B_G(0,0,1)$, $B_G(0,1,1)$, and $B_G(1,1,1)$. From these we wish to raise the degree of the rational Bézier curve by two. We can represent the new rational Bézier curve as having a knot vector with $n+2$ knots at 0 and another $n+2$ at 1. as shown in the figure 6.

$$\begin{array}{ll}
R_0 = B_G(0,0,0,0,0 : 1) & = \frac{1}{u_0} v_0 B_G(0,0,0) \\
R_1 = B_G(0,0,0,0,1 : \frac{2}{5} \times 1 + \frac{3}{5} \times 1) & = \frac{1}{u_1}(\frac{2}{5} v_0 B_G(0,0,0) + \frac{3}{5} v_1 B_G(0,0,1)) \\
R_2 = B_G(0,0,0,1,1 : \frac{1}{10} \times 1 + \frac{3}{5} \times 1 + \frac{3}{10} \times 1) & = \frac{1}{u_2}(\frac{1}{10} v_0 B_G(0,0,0) + \frac{3}{5} v_1 B_G(0,0,1) + \frac{3}{10} v_2 B_G(0,1,1)) \\
R_3 = B_G(0,0,1,1,1 : \frac{3}{10} \times 1 + \frac{3}{5} \times 1 + \frac{1}{10} \times \frac{11}{9}) & = \frac{1}{u_3}(\frac{3}{10} v_1 B_G(0,0,1) + \frac{3}{5} v_2 B_G(0,1,1) + \frac{1}{10} v_3 B_G(1,1,1)) \\
R_4 = B_G(0,1,1,1,1 : \frac{3}{5} \times 1 + \frac{2}{5} \times \frac{11}{9}) & = \frac{1}{u_4}(\frac{3}{5} v_2 B_G(0,1,1) + \frac{2}{5} v_3 B_G(1,1,1)) \\
R_5 = B_G(1,1,1,1,1 : \frac{11}{9}) & = \frac{1}{u_5} v_3 B_G(1,1,1)
\end{array}$$

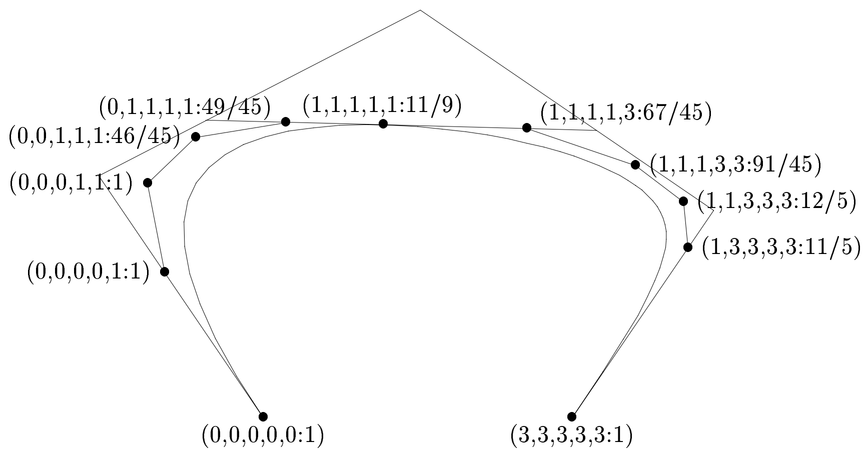

FIGURE 6. Degree elevated each rational Bézier piece.

The degree elevation algorithm in the figure 6 is expressed in a matrix form as

$$U = M_e^w V,\ R = M_e^p Q$$

where

$$U = \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \end{pmatrix}, M_e^w = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2/5 & 3/5 & 0 & 0 & 0 & 0 & 0 \\ 1/10 & 3/5 & 3/10 & 0 & 0 & 0 & 0 \\ 0 & 3/10 & 3/5 & 1/10 & 0 & 0 & 0 \\ 0 & 0 & 3/5 & 2/5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2/5 & 3/5 & 0 & 0 \\ 0 & 0 & 0 & 1/10 & 3/5 & 3/10 & 0 \\ 0 & 0 & 0 & 0 & 3/10 & 3/5 & 1/10 \\ 0 & 0 & 0 & 0 & 0 & 3/5 & 2/5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, V = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{pmatrix},$$

$R = (B_G(0,0,0,0,0), B_G(0,0,0,0,1), B_G(0,0,0,1,1), B_G(0,0,1,1,1), B_G(0,1,1,1,1),$
$B_G(1,1,1,1,1), B_G(1,1,1,1,3), B_G(1,1,1,3,3), B_G(1,1,3,3,3), B_G(1,3,3,3,3), B_G(3,3,3,3,3))^t,$
$Q = (B_G(0,0,0), B_G(0,0,1), B_G(0,1,1), B_G(1,1,1), B_G(1,1,3), B_G(1,3,3), B_G(3,3,3))^t,$
and
$M_e^p = diag(1/U) M_e^w diag(V).$

The following algorithm compute degree elevation M_e^w matrix.

```
**Make_ElevationMatrix**($p$, $r$, $s$)
// Input: Degree $p$ and elevation degree $r$
// Output: $(ps + p + rs + r + 1) \times (ps + p + 1)$ matrix $M_e^w$

$div = 1$;
**for** ($i = 1$ to $r$ by 1) $div = div \times (p + i)$;

$M[0] = 1$;
**for** ($i = 0$ to $p$ by 1)
   **for** ($j = 1$ to $r$ by 1) $M[j] = M[j] + M[j-1]$;
   **for** ($j = 0$ to $r$ by 1)
      $left = 1$;
      **for** ($k = 1$ to $j$ by 1) $left = left \times (r - k + 1)$;
      $right = 1$;
      **for** ($k = 1$ to $r - j$ by 1) $right = right \times (p - i + k)$;
      $T[i + j][i] = left \times M[j] \times right/div$;
   **endfor**
**endfor**

$M_e^w[0][0] = T[0][0]$;
**for** ($i = 0$ to $s$ by 1)
   **for** ($j = 1$ to $p + r$ by 1)
      **for** ($k = 0$ to $p$ by 1) $M_e^w[j + (p + r)i][(k + p)i] = T[j][k]$;
```

Curve Composition: Curve composition is the inverse of decomposition. In the figure 7, it can be expressed in a linear equation as

$$U = M_D^w \tilde{W}, \;\; R = M_D^p \tilde{P}$$

where

$$U = \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \end{pmatrix}, M_D^w = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4/9 & 4/9 & 1/9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2/3 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tilde{W} = \begin{pmatrix} \tilde{w}_0 \\ \tilde{w}_1 \\ \tilde{w}_2 \\ \tilde{w}_3 \\ \tilde{w}_4 \\ \tilde{w}_5 \\ \tilde{w}_6 \\ \tilde{w}_7 \\ \tilde{w}_8 \end{pmatrix},$$

$R = (B_G(0,0,0,0,0), B_G(0,0,0,0,1), B_G(0,0,0,1,1), B_G(0,0,1,1,1), B_G(0,1,1,1,1),$
$B_G(1,1,1,1,1), B_G(1,1,1,1,3), B_G(1,1,1,3,3), B_G(1,1,3,3,3), B_G(1,3,3,3,3), B_G(3,3,3,3,3))^t,$
$\tilde{P} = (B_G(0,0,0,0,0), B_G(0,0,0,0,1), B_G(0,0,0,1,1), B_G(0,0,1,1,1), B_G(0,1,1,1,3),$
$B_G(1,1,1,3,3), B_G(1,1,3,3,3), B_G(1,3,3,3,3), B_G(3,3,3,3,3))^t$, and
$M_D^p = diag(1/U) M_D^w diag(\tilde{W})$.

Then, because M_D^w has full column rank and $(M_D^w)^t M_D^w$ is non-singular, $M_c^w = [(M_D^w)^t M_D^w]^{-1} (M_D^w)^t$ is a left inverse of the matrix M_D^w. Therefore the consistent equations $U = M_D^w \tilde{W}$ and $R = M_D^p \tilde{P}$ have a unique solution ,

$$\tilde{W} = M_c^w U \text{ and } \tilde{P} = M_c^p R$$

where $M_c^p = diag(1/\tilde{W}) M_c^w diag(U)$.

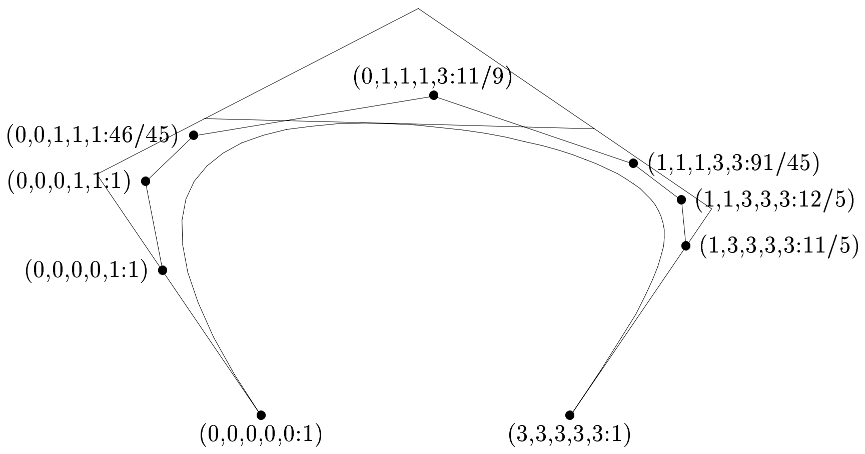

FIGURE 7. Composed the NURBS curve over the knot vector (0, 0, 0, 0, 0, 0, 1, 1, 1, 3, 3, 3, 3, 3, 3) from the piecewise rational Bézier segment.

We will write down the algorithm for the degree elevation of a NURBS curve in a shorthand notation.

1. Make_DecompositionMatrix(knots, m, p) : M_d^w
   knots= $\{\underbrace{a, a, \ldots, a}_{p+1}, \underbrace{u_1, \ldots, u_1}_{m_1}, \ldots, \underbrace{u_s, \ldots, u_s}_{m_s}, \underbrace{b, b, \ldots, b}_{p+1}\}$
2. Make_ElevationMatrix(p, r, s) : M_e^w
3. Make_DecompositionMatrix(knots, m+(s+2)r, p+r) : M_D^w
   knots= $\{\underbrace{a, a, \ldots, a}_{p+r+1}, \underbrace{u_1, \ldots, u_1}_{m_1+r}, \ldots, \underbrace{u_s, \ldots, u_s}_{m_s+r}, \underbrace{b, b, \ldots, b}_{p+r+1}\}$

$$M_c^w = [(M_D^w)^t M_D^w]^{-1}(M_D^w)^t$$

Then,

$$\tilde{W} = M^w W \text{ and } \tilde{P} = M^p P$$

where

$$M^w = M_c^w M_e^w M_d^w \text{ and } M^p = diag(1/\tilde{W}) M^w diag(W).$$

References

[1] Barry, P. J. (1993), An Introduction to Blossoming, in: Goldman, R. N. and Lyche, T. eds., *Knot Insertion and Deletion Algorithms for B–spline Curves and Surface*, SIAM, 1-10.

[2] Cohen, E. Lyche, T. and Schumaker, L. L. (1985), Algorithms for degree-rasing of splines, ACM Trans. Graph. 4, 171-181.

[3] Emery, J. D. (1986), The definition and computation of a metric on plane curves, Comput. Aided Des. 18, 25-28.

[4] Farin, G. (1993), *Curves and surfaces for computer aided geometric design : A practical guide*, Academic Press, San Diego.

[5] Lee, B.G. and Park, Y. (1997), The Distance for Bézier Curves and Degree reduction, Bull. Australian Math. Soc. 56, 507-515.

[6] Lee, B.G. and Park, Y. (1998), Degree Elevation and Reduction of B-spline Curves, *submitted to Comput. Aided Geom. Des.*

[7] Liu, W. (1997), A simple, efficient degree rasing algorithm for B–spline curves, Comput. Aided Geom. Des. 14, 693-698.

[8] Piegl, L. and Tiller, W. (1994), Software-engineering approach to degree elevation of B–spline curves, Comput. Aided Des. 26, 17-28.

[9] Piegl, L. and Tiller, W. (1995), Algorithm for degree reduction of B–spline curves, Comput. Aided Des. 27, 101-110.

[10] Prautzsch, H. (1984), Degree elevation of B–spline curves, Comput. Aided Geom. Des. 2, 193-198.

[11] Prautzsch, H. and Piper, B. (1991), A fast algorithm to raise the degree of B–spline curves, Comput. Aided Geom. Des. 8, 253-266.

[12] Searle, S. R. (1971), *Linear Models*, John Wiley & Sons, New York.

School of Internet Engineering, Dongseo University, Pusan, 617–716, Republic of Korea

Department of Mathematics Education, Seowon University, Chongju, 361–742, Republic of Korea