NURBS Programming via Blossoming

Xianming Chen

March 23, 2005

Contents

1 The Affine Combination Notation 2
2 Knot Vectors, Knot Sequences, Distinct Value Knot Vectors, Knot Multiplicity Vectors 2
3 Interpolation in the Context of Piecewise Blossoms 2
4 Derivatives of NURBS Curves 4
5 Derivatives of NURBS Curves: A New Notation 5
   5.1 Derivatives of \(C^0\) NURBS Curves 6
   5.2 Derivatives of \(C^{-1}\) NURBS Curves 6
6 Direct Degree Elevation of NURBS 7
7 Multiplication of Two Scalar NURBS 8
1 The Affine Combination Notation

Most often we talk about a certain affine combination in the domain space, and then map the affine combination into the image space. For the sake of simple math formulation, we introduce the notation,

\[ \mathfrak{A}(t_1,t,t_2), \]

to mean that \( t_1, t, t_2 \) are 3 points in \( A^1 \) space, and \( t \) is an affine combination of \( t_1 \) and \( t_2 \), i.e.,

\[ \mathfrak{A}(t_1,t,t_2) = \frac{t_2-t}{t_2-t_1} t_1 + \frac{t-t_1}{t_2-t_1} t_2. \]

and the notation

\[ \mathfrak{A}_{t_1,t,t_2}(P_1,P_2) \]

to denote the image \( P \) of \( t \) under the considered affine mapping from \( A^1 \) to \( A^n \) for some \( n \) (usually \( n = 3 \)), where \( P_1 \) and \( P_2 \) are the images of \( t_1 \) and \( t_2 \), i.e.,

\[ \mathfrak{A}_{t_1,t,t_2}(P_1,P_2) = \frac{t_2-t}{t_2-t_1} P_1 + \frac{t-t_1}{t_2-t_1} P_2. \]

2 Knot Vectors, Knot Sequences, Distinct Value Knot Vectors, Knot Multiplicity Vectors

A NURBS is a piece-wise polynomials, blossoming to 'piece-wise' blossoms. Locally, a degree \( d \) polynomial blossoms to a \( d \)-variate symmetric multi-affine function, which has \( d+1 \) control points. The \( d+1 \) control points are defined as the blossom values from \( d+1 \) consecutive knots from a sequence of non-descending knot values of total length \( 2d \). Globally, all the adjacent polynomials (incl. degenerated ones) have \( C^{d-1} \) continuity, which translates into, that the \( d \) last control points of a blossom are the \( d \) first control points of the next blossom. The \( i \)-th polynomial \( x_i \), as well as the \( i \)-blossom \( b_i \), is defined by control points \( P_{i}^{j+d} \), which are the blossom values of \( b_i(t_i^{j+d-1}, \cdots, b_i(t_i^{j+2d-1}) \). By \( C^{d-1} \) continuity, control point \( P_j \) is actually a blossom value of as many as \( d+1 \) consecutive blossoms,

\[ P_j = b_i(t_j^{j+d-1}) = b_i(t_i^{j+d-1} + (j-i)), \quad j - i \geq 0 \wedge j - i \leq d, \quad j - d \leq i \geq j \]

Consequently, \( P_j \) is the \( (j - i) \)-th control point of \( i \)-th polynomial (blossom).

For the simplicity of mathematical formulation, we leave out the subscript in \( b_i \), in the above equation,

\[ P_j = b(t_j^{j+d-1}), \]

with the understanding that the \( b \) could be as many as \( d+1 \) different blossoms, which all evaluate to the same point.

A knot vector can also be expressed as a vector of ascending distinct knot values and a vector of multiplicity for each of the distinct value.

The knot vector representation is most common. However, the distinct knot value vector, together with the corresponding multiplicity vector, is more convenient for some NURBS topics, such as degree elevation (see Section 6) and NURBS multiplication (see Section 7).

3 Interpolation in the Context of Piecewise Blossoms

A NURBS is a piecewise polynomial curve or surface. Since a tensor surface is simply reduced to a congruence of NURBS curves with the same structure in each direction, we can just focus on NURBS curve case without loss of generality.

The most prominent new feature of a Bspline curve, compared to a Bézier curve, is that now there are many pieces of polynomial curves, joining together at the breaking points with the continuity of \( d - 1 \) (\( d \) is the degree).
Correspondingly, there are many pieces of blossoms, the adjacent two of which share all the control points except one \(^1\).

Now suppose the knot vector is changed by inserting a single valid knot value(by 'valid', we mean the inserted value is within the total defining domain of the NURBS) The inserted knot might be a new value, therefore creating a fake breaking point, or an old one, therefore introducing a new degenerated segment of zero domain interval.

Suppose further the newly inserted knot has index \(j\), after re-indexing the whole knot vector, we consider the new control points.

First, the first \(j - d + 1\) old control points, \(P_0^{j-d}\), become those of the new ones, i.e., \(\bar{P}_0^{j-d}\),

\[
\bar{P}_i = b(t_i^{j+(d-1)}) \\
= b(t_i^{j+(d-1)}) \\
= P_i, \text{ for } i = 0, \ldots, j - d.
\] (1)

Second, the \(d + 1\) old control points, \(P_j^{j-d}\), are affine combined into \(d\) new control points, i.e., \(\bar{P}_j^{j-d+1}\),

\[
\bar{P}_{j+k-(d-1)} = b(t_{j+k-(d-1)}) \\
= T_{j+k-d, j+k+1} \left( b(t_{j+k-1}), b(t_{j+k-d}) \right) \\
= \tilde{A}(P_{d+k-j}, P_{d-k+1}), \text{ for } k = 0, \ldots, d - 1.
\] (2)

Finally, the \(m - j\) old control points, \(P_{j+1}^{m-1}\), become of those of the new ones with indexing increased by one, i.e. \(P_{j+1}^{m-1}\),

\[
\bar{P}_{j+1+k} = b(t_{j+1+k}^{j+(d-1)}) \\
= b(t_{j+k}^{j+k+(d-1)}) \\
= P_{j+k}, \text{ for } k = 0, \ldots, m - j - 1.
\] (3)

Now suppose the knot vector is changed by inserting simultaneously some knot values (might be new values, therefore creating fake breaking points, or old values, therefore creating new segments of zero domain intervals).

Our task here is to find the blossom value at any new sequence of \(d\) consecutive knot values, say \(b_i(t_{i}, \ldots, t_{i+d-1})\). Suppose \(t_j\) is any knot value not in the original knot vector, we have the following interpolation,

\[
b_i(t_i, \ldots, t_{j-1}, t_j, t_{j+1}, \ldots, t_{i+d-1}) \\
= b_i(t_i, \ldots, t_{j-1}, \tilde{A}(t_{i-1}, t_j, t_{i+d}), t_{j+1}, \ldots, t_{i+d-1}) \\
= \tilde{A}_{t_{i-1}, t_i, t_{i+d}} \left( b_i(t_{i-1}, \ldots, t_{j-1}, t_{j+1}, \ldots, t_{i+d-1}), b_i(t_i, \ldots, t_{j-1}, t_{j+1}, \ldots, t_{i+d}) \right)
\] (4)

Since we are doing interpolation on \(t_j\), a new knot, the two knot sequences in the resulting blossoms in Eq. (4) can not have more new knots than the original one, and when we evaluate recursively, the number of new knots in these recursively generated knot sequences have to be decreased by one in some step if the interpolation is from some old knot(s). So the recursive evaluation will finally stop at the blossom values on knot sequences from the original knot vector. That is to say, \(b_i(t_i, \ldots, t_{i+d-1})\) is eventually interpolated from the original control points.

Also, for efficient implementation, all the intermediate blossom values are stored in a blossom table, and the blossom evaluation is actually carried out only if the table lookup is a miss. This way, with the help of a blossom table, all the new control points can be computed with minimal computation.

\(^1\)These statements are under the assumption that we take into account of any degenerated polynomial piece of zero domain interval. Otherwise, we should have stated that the continuity at a breaking point is degree less the multiplicity of the breaking knot value, and the adjacent two blossoms share all the control points except one if the continuity is maximal, and the number of shared control points decrease the same amount as the continuity decreases from the maximum.
Technically, however, there is one problem with the above described algorithm; to mathematically make sense, all the recursive interpolations have to be on those points which are the blossom values of one single blossom. The interpolation in Eq. (4) is based on \( t_j \) and consequently \( b_i \) is the the unique blossoming of the NURBS piece with its interval containing \( t_j \). The recursive interpolations followed are of course based on different new knot values, and the blossoms changed correspondingly. Nonetheless, we are able to justify the recursive interpolation scheme, by making the critical observation that any control point corresponding to a knot sequence belongs simultaneously to \( d + 1 \) (\( d \) is the degree) blossoms of \( d + 1 \) consecutive NURBS pieces (counting the degenerate one with zero interval). Therefore, we are on the safe side - even though the interpolation in each recursive step is with respect to some blossom, and the next step to some other blossoms, these blossoms share all the points that appear in the interpolation process.

## 4 Derivatives of NURBS Curves

Suppose a NURBS curve \( x(t) \) has

\[
\text{degree} = d,
\]

\[
\text{# segments (incl. degenerated segments of zero interval)} = s,
\]

defined by

\[
\text{control points} = P_0, \ldots, P_d, \ldots, P_{s+d-1},
\]

\[
\text{knot vector} = t_0, t_1, \ldots, t_d-1, \ldots, t_s+d-1, \ldots, t_s+2d-3, t_s+2d-2,
\]

defined on interval = \([t_{d-1}, t_{s+d-1}]\).

For \( i = 0, 1, \ldots, s-1 \), the \( i \)-th segment \( x_i \) has,

\[
\text{degree} = d,
\]

\[
\text{control points} = P_i, \ldots, P_{i+d},
\]

\[
\text{knot vector} = t_i, \ldots, t_{i+d-1}, \ldots, t_{i+d}, \ldots, t_{i+2d-1},
\]

defined on interval = \([t_{i+d-1}, t_{i+d}]\).

Suppose the \( i \)-th segment has blossom \( b_i \), from Eq. (??), we have,

\[
x'_i(t) = d \ast b_i(t^{d-1}, \overline{t}) = b_i(t^{d-1}),
\]

where

\[
b_i(u_0, \ldots, u_{d-2}) = d \ast b_i(u_0, \ldots, u_{d-2}, \overline{t}).
\]

(5)

With the knot vector \( t_{i+1}, \ldots, t_{i+d-1}, \ldots, t_{i+d}, \ldots, t_{i+2d-2} \) in mind, we define the symmetric (\( d - 1 \))-affine \( \tilde{b}_i \) by control points,

\[
\tilde{P}_i = \tilde{b}_i(t_{i+1}, t_{i+2}, \ldots, t_{i+d-1}), \ldots, \tilde{P}_{i+d} = \tilde{b}_i(t_{i+d-1}, t_{i+d}, \ldots, t_{i+2d-2})
\]

By Eq. (7),

\[
\tilde{P}_{i+j} = \frac{1}{t_{i+j+d} - t_{i+j}} (b_i(t_{i+j+1}, t_{i+j+2}, \ldots, t_{i+j+d}) - b_i(t_{i+j}, t_{i+j+2}, \ldots, t_{i+j+d-1}),
\]

\[
= d \ast \frac{1}{t_{i+j+d} - t_{i+j}} (P_{i+j+1} - P_{i+j}),
\]

(6)

where we have replaced \( \overline{t} \) with \( \frac{1}{t_{i+j+d} - t_{i+j}} (t_{i+j+d} - t_{i+j}). \)
It is obvious from the above equation that the new control points have the same formulation over all segments, and we have, for the derivative of $x(t)$,

\[
\text{degree} = d - 1, \quad \text{# segments (incl. degenerated segments of zero interval)} = s,
\]

defined by

- control points: $\tilde{P}_i = d \cdot \frac{1}{t_{i+d} - t_i} \cdot (P_{i+1} - P_i), i = 0, \ldots, s + d - 2$,
- knot vector: $t_1, \ldots, t_{d-1}, \ldots, t_{s+d-1}, \ldots, t_{s+2d-3}, t_{s+2d-3}$,

defined on interval: $[t_{d-1}, t_{s+d-1}]$.

### 5 Derivatives of NURBS Curves: A New Notation

Suppose a NURBS curve $x(t)$ has

\[
\text{degree} = d, \quad \text{# segments (incl. degenerated segments of zero interval)} = s,
\]

defined by

- control points: $P_i^d = P_i^d + (s-1)$,
- knot vector: $t_{0}^{2d-1} + (s-1)$,

defined on interval: $[t_{d-1}, t_{d} + (s-1)]$.

For $i = 0, 1, \ldots, s - 1$, the $i$-th segment $x_i$ has,

- degree = $d$,
- control points: $P_i^{i+d}$,
- knot vector: $t_i^{i+2d-1}$,

defined on interval: $[t_{i+d-1}, t_{i+d}]$.

Suppose the $i$-th segment has blossom $b_i$, from Eq. (7), we have,

\[
x'_i(t) = d \cdot b_i(t^{d-1}) = b_i(t^{d-1}),
\]

where

\[
\tilde{b}_i(u_0, \cdots, u_{d-2}) = d \cdot b_i(u_0, \cdots, u_{d-2}, \bar{1}).
\]

The symmetric $(d - 1)$- affine $\tilde{b}_i$ has control net $P_i^{i+d-1}$, based on knot vector $t_i^{i+2d-2}$,

\[
\tilde{P}_{i+j} = \tilde{b}_i(t_i^{i+d-1} + j)
= d \cdot b_i(t_i^{i+d-1} + j, t_i^{i+1} + j, \bar{1})
= \frac{d}{t_{i+d+j} - t_{i+j}} \cdot b_i(t_i^{i+d-1} + j, (t_{i+d+j} - t_{i+j}))
= \frac{d}{t_{i+d+j} - t_{i+j}} \cdot (b_i(t_i^{i+d+j} - t_{i+j}) - b_i(t_i^{i+d-1+j}))
= \frac{d}{t_{i+d+j} - t_{i+j}} \cdot (P_{i+1+j} - P_{i+j})
\text{for } j = 0, \cdots, d - 1.
\]
where we have replaced \( \tilde{r} \) with \( \frac{1}{t_{i+j+d}-t_{i+j}}(t_{i+j+d} - t_{i+j}) \).

It is obvious from the above equation that the new control points have the same formulation over all segments. If the derivative has at least \( C^0 \) continuity, then it has

\[
\text{degree} = d - 1, \\
\# \text{ segments (incl. degenerated segments of zero interval)} = s,
\]
defined by

\[
\begin{align*}
\text{control points } \tilde{P}_i^{(d-1) + (s-1)}: & \quad \tilde{P}_i = d * \frac{1}{t_i+d - t_i} * (P_{t_i+1} - P_t), \\ 
& \quad i = 0, \ldots, s + d - 2, \\
\text{knot vector: } & \quad t_1, \ldots, t_d-1, \ldots, t_s+d-1, \ldots, t_s+2d-3, \ldots, t_s+2d-3,
\end{align*}
\]
defined on interval: \([t_{d-1}, t_{s+d-1}]\),

\subsection{5.1 Derivatives of \( C^0 \) NURBS Curves}

It turns out the scheme above (Eq. (9) applies just as well to the derivatives of \( C^0 \) NURBS. The justification follows.

If the original NURBS has \( C^0 \) break point between the \( i \)-th and \((i + d)\)-th segments (we count degenerated segments of zero intervals), i.e., the corresponding knot multiplicity is \( d \) (\( d \) is the degree, and there are \( d - 1 \) degenerated segments here), then the derivative has \( C^{(-1)} \) there, with the same \( d \) multiple knots but in the context of now \( d - 1 \) degree. Consequently there are two control points, the last control point of segment \( i \): \( P_{i+(d-1)} \) and the first of segment \( i + d \): \( P_{i+d} \), one being the left limit of the derivative and the other the right limit. The left limit is computed by Eq. (8) from the \( i \)-th segment, while the right one by the same equation but from the \( i + d \)-th segment.

\[
\begin{align*}
\tilde{P}_{i+d-1} &= \frac{d}{t_{i+2d-1} - t_{i+d-1}} * (P_{i+d} - P_{i+d-1}) \\
\tilde{P}_{i+d} &= \frac{d}{t_{i+2d} - t_{i+d}} * (P_{i+d+1} - P_{i+d})
\end{align*}
\]

\subsection{5.2 Derivatives of \( C^{-1} \) NURBS Curves}

The derivatives of a \( C^{-1} \) NURBS is still \( C^{-1} \) at the appropriate breaking points.

If the original NURBS has \( C^{-1} \) break point between the \( i \)-th and \((i + d + 1)\)-th segments (once again remember that we count degenerated segments of zero intervals), i.e., the corresponding knot multiplicity is \( d + 1 \) (\( d \) is the degree, and there are \( d \) degenerated segments here), then the derivative still has \( C^{(-1)} \) there, with \( d \) multiple knots in the context of now \( d - 1 \) degree. Consequently there are two control points, the last control point of segment \( i \): \( P_{i+(d-1)} \) and the first of segment \( i + d \): \( P_{i+d} \), one being the left limit of the derivative and the other the right limit. The left limit is computed by Eq. (8) from the \( i \)-th segment, while the right one by the same equation but from the \( i + d + 1 \)-th segment.

\[
\begin{align*}
\tilde{P}_{i+d-1} &= \frac{d}{t_{i+2d-1} - t_{i+d-1}} * (P_{i+d} - P_{i+d-1}) \\
\tilde{P}_{i+d+1} &= \frac{d}{t_{i+2d+1} - t_{i+d+1}} * (P_{i+d+2} - P_{i+d+1})
\end{align*}
\]

It is obvious there is a labeling gap here. However, notice that in the process, one knot has to be deleted from the \( d + 1 \) copies of the knot, otherwise a \( C^{(-2)} \) results which makes no sense. This implies that the labeling of the new control point in the second equation of Eq. (11) should decrease by one, while the labeling on the right side of it keeps the same (BTW, this also implies the the segments of the derivative NURBS is one less than the original one instead of the same for the non-\( C^{-1} \) cases).

The conclusion is,
When using Eq. (9) to compute the new control points of the derivative, the labeling (indexing from the implementation perspective) has to be increased by one each time a $C_1$ occurs, which is simply indicated by zero interpolation interval, i.e. $t_{i+d} - t_i = 0$ in Eq. 9.

6 Direct Degree Elevation of NURBS

For the specific topic of this section, the multiplicity knot vector representation is used, and consequently any degenerated polynomial piece of zero domain interval is not counted as one segment.

This section gives an algorithm that develops NURBS degree elevation with any amount in a single step. Since degree elevation of a NURBS surface is basically done in each direction, in exactly the same as a NURBS curve, we only consider NURBS curves in the following.

Let $b(t_1, \cdots, t_d)$ be the blossom of some NURBS segment. Define another symmetric multi-affine mapping as,

$$
\bar{b}(t_1, \cdots, t_{d+n}) = \sum_{I \in I_{n+d}, \|I\| = d} b(t_I)
$$

(12)

where $I_n$ is the index set,

$I_n = \{1, 2, \cdots, n\},$

and $(t_I)$ stands for $(t_{k_1}, \cdots, t_{k_d})$, where $I = \{k_1, \cdots, k_d\}$.

From the specific construction of $\bar{b}$, obviously its diagonalization is a NURBS segment that degree raises the original one by the amount of $d$.

Now coming back to the actual NURBS to be degree raised by an amount of $d$, i.e. we should construct the above symmetric multi-affine mapping for each single segment, find the control points of each, and join all these separate pieces together. It turns out that we can just compute the new control points regardless of the knowledge which segments of the computed control points are defined for; the interpolation scheme in the algorithm below actually automatically makes this always correct.

Suppose the original knot vector has distinct ascending knot values,

$$
t_0, \cdots, t_s,
$$

and multiplicities,

$$
m_i <= d \text{ for } i = 0, \cdots, s,
$$

Suppose further this NURBS is to be degree raised by an amount of $n$, then the degree raised NURBS has the same distinct knot values, but with the multiplicities increased by $n$ for each knot value, i.e.,

$$
\tilde{m}_i = m_i + n \text{ for } i = 0, \cdots, s,
$$

And with the knowledge of the knot vector of the degree raised NURBS, and Eq. (12), the new control points can be computed from the corresponding points of the original NURBS blossom, which are usually not in the original control points and therefore need to be interpolated from them. The new control knot vector has each multiplicity raised by $d$, and the knot sequence in each term of the right side of Eq. (12) has total $d$ knots removed; hence, the knot sequence in each term of the right side of Eq. (12) is either a sequence exactly from the original knot vector, or a sequence of the original knot vector with some knots inserted - either case, it is well defined and the interpolation make sense.$^2$

With the proper iteration method on control mesh and knot sequence defined, Algorithm 1 implements a direct general dimensional NURBS degree elevation.

$^2$There is a problem if the original knot vector does not have an open end condition; there would be extrapolation at the end(s). So the following algorithm changes the end condition into open end in the degree raised direction.
Algorithm 1 Direct Degree Elevation of a General Dimensional NURBS

Input:
- $x$, the tensor NURBS of any dimension to degree raised in some direction.
- $i$, the direction along which the given NURBS is to be degree raised.
- $d$, the original degree in direction $i$ of $x$.
- $n$, the amount of degree elevation.

Output:
- $x$, the degree raised NURBS.

Begin
convert $x$ to open end condition in direction $i$.
raise the knot multiplicity of $x$ by $n$ for each knot value.

knot-sequence-iterate through the new knot vector and through the slices (across direction $i$) of control points as well,
extract consecutive $n + d$ knot values, $\text{Seq}$, in each iteration,
initialize the new control points of the whole current slice to the zero value in the appropriate affine space.

degree-reduce-iterate over $\text{Seq}$, extract a sequence of $d$ knot values, $\text{seq}$, in each iteration,
for each control point in the current slice,
add it the blossom value of $x$ at $\text{seq}$ by interpolation on the original control points.
(this is typically done with a blossom table).
divide each control point in the current slice by the number of total iterations above.

End

7 Multiplication of Two Scalar NURBS

In this section, we use the same knot vector representation as in Section 6, and do not count any zero domain interval segment.

Given two scalar NURBS, $f_1(t)$ and $f_2(t)$, of degree $d_1$ and $d_2$, we try to find their multiplication via blossoming.

First we make the two NURBS to be open end conditioned, and have the same distinct knot values (by knot insertion),
$$t_0, \ldots, t_i, t_{i+1}, \ldots t_s,$$
but with different multiplicities,
$$m_1_{0i}, m_1_{i}, m_1_{i+1}, \ldots, m_1_s, \text{ where } m_1_i <= d_1 \text{ for } i = 0, \ldots, s;$$
$$m_2_{0i}, m_2_{i}, m_2_{i+1}, \ldots, m_2_s, \text{ where } m_2_i <= d_2 \text{ for } i = 0, \ldots, s;$$

Since the multiplied scalar NURBS, $f(t) = f_1(t) * f_2(t)$, have the same continuity as the lower one of the given two NURBS at any knot breaking point, the knot vector of $f(t)$ has the same distinct knot values, and with multiplicities of
$$m_i = \{ \begin{array}{ll}
\text{if } & d_1 - m_1_i < d_2 - m_2_i, \text{ i.e } f_1 \text{ has lower continuity, } d_2 + m_1_i \\
\text{else} & d_1 + m_2_i \\
\end{array} \text{ for } i = 0, \ldots, s;$$

Now focusing on the $i$-th (non-degenerated) segment. $f_1(t)$ is defined on interval $[t_i, t_{i+1})$, with knot vector of
$$\ldots, t_i^{m_1_i}, t_{i+1}^{m_1_{i+1}}, \ldots,$$
while $f_2(t)$ is defined on the same interval $[t_i, t_{i+1})$, with knot vector of
$$\ldots, t_i^{m_2_i}, t_{i+1}^{m_2_{i+1}}, \ldots,$$
and \( f(t) \) is defined on the interval \([t_i, t_{i+1}]\), with knot vector of

\[
\cdots, t_i^{m_i}, t_{i+1}^{m_{i+1}}, \cdots,
\]

Let the blossoms of \( f_1 \) and \( f_2 \) be, \( b_1(u_0, \cdots, u_{d_1-1}) \) and \( b_1(u_{d_1}, \cdots, u_{d_1+d_2-1}) \). Define a symmetric multi-affine mapping as,

\[
b(u_I) = \frac{\sum_{I_1 \oplus I_2 = I, \|I_1\| = d_1, \|I_2\| = d_2} b_1(u_{I_1}) * b_2(u_{I_2})}{(d_1 + d_2)_{d_1}}, \tag{13}
\]

where \( I \) is the index set \( \{0, 1, \cdots, d_1 + d_2 - 1\} \), and \( \{I_1, I_2\} \) is any partition of \( I \), with fixed cardinality of \( d_1 \) and \( d_2 \) respectively; also, \( u_I \) (similarly for \( u_{I_1} \) and \( u_{I_2} \)) stands for \((u_0, u_1, \cdots, u_{d_1+d_2-1})\), i.e. all \( u \)-s with the subscripts in \( I \). Obviously the diagonalization of \( b \) is the multiplication of those of \( b_1 \) and \( b_2 \), i.e. the \( i \)-th segment of \( f_1 \) and \( f_2 \). That is \( b \) is the blossom of \( f(t) \).

To gain some intuition, let us work out an example first.

**Example 1** Suppose a degree 4 NURBS function \( f_1 \) has knot vector\(^3\),

\[
0, 1^2, 2, 3, 4^2, 5, 6^4, 7^2,
\]

and another degree 5 NURBS function \( f_2 \),

\[
0, 1, 2^3, 3, 4^2, 5^5, 6^2, 7.
\]

The multiplied degree 9 NURBS function \( f = f_1 f_2 \) has knot vector,

\[
0^6, 1^7, 2^7, 3^6, 4^7, 5^9, 6^9, 7^7.
\]

Let us focus on the segment of interval \([3, 4]\), with knot vector of

\[
2^3, 3^6, 4^7, 5^2.
\]

The blossom values (control points) of \( f \) can be interpolated from the those of \( f_1 \) and \( f_2 \), as

\[
\begin{align*}
&b(2^33^6) \leftarrow \begin{pmatrix}
    b_1(2^33^1) * b_2(3^3) \\
    b_1(2^22^2) * b_2(2^13^4) \\
    b_1(2^13^3) * b_2(2^23^3) \\
    b_1(2^03^4) * b_2(2^33^2)
\end{pmatrix} \\
&b(2^23^64^1) \leftarrow \begin{pmatrix}
    b_1(2^22^22^4) * b_2(0^63^44^1) \\
    b_1(2^23^34^1) * b_2(0^35^49) \\
    b_1(2^13^34^0) * b_2(2^13^34^1) \\
    b_1(2^12^34^1) * b_2(2^23^44^0) \\
    b_1(2^03^44^0) * b_2(2^23^44^1) \\
    b_1(2^03^44^1) * b_2(2^33^44^0)
\end{pmatrix} \\
&b(2^13^64^2) \leftarrow \begin{pmatrix}
    b_1(2^13^34^0) * b_2(2^03^34^2) \\
    b_1(2^13^34^1) * b_2(2^03^34^1) \\
    b_1(2^13^34^2) * b_2(2^03^34^0) \\
    b_1(2^03^44^0) * b_2(2^13^24^2) \\
    b_1(2^03^44^1) * b_2(2^13^34^1) \\
    b_1(2^03^44^2) * b_2(2^13^34^0)
\end{pmatrix}
\end{align*}
\]

\(^3\)The two NURBS in this example donot have open end conditions. Using the same method in this example, it is easy to see that the end control points of the multiplied NURBS have to be extrapolated, instead of interpolated as shown in this example for the inner control points. Therefore, this also demonstrate the necessity of making the two NURBS open end conditioned before doing any multiplication.
Algorithm 2 Component-Wise Multiplication of Two NURBS

Input:

\(x_1, x_2\), the two NURBS of the same dimension and domain intervals.

Output:

\(x\), the component-wise product of \(x_1\) and \(x_2\).

Begin

convert \(x_1, x_2\) to open ends,
create blossom tables \(\text{table}_1\) and \(\text{table}_2\) of \(x_1\) and \(x_2\), respectively,
compute the multiplicity knot vector, \(K_t\), of \(x = x_1 x_2\),
resize the control mesh of \(x\) according to \(K_t\) and the total degree,
insert all new distinct knot value that is in \(K_t\) but not in the knot vectors of \(\text{table}_1\) and \(\text{table}_2\), respectively.

iterate over \(K_t\) and the control mesh of \(x\); let \(t_{Seq}\) and \(\text{CtlPnt}\) the current tensor knot sequence and its control point,
\(\text{CtlPnt} \leftarrow 0\).

degree-reduce-iterate over \(t_{Seq}\); let \(t_{Seq_1}\) and \(t_{Seq_2}\) be the iterated pair, and \(\text{weight}\) the weight,
compute blossom value, \(\text{blsm}_1\), of \(x_1\) at \(t_{Seq_1}\), by table looking up at \(\text{table}_1\) (interpolation going on here).
compute blossom value, \(\text{blsm}_2\), of \(x_2\) at \(t_{Seq_2}\), by table looking up at \(\text{table}_2\) (interpolation going on here).
\(\text{CtlPnt} = \text{CtlPnt} + \text{weight} \times \text{blsm}_1 \times \text{blsm}_2\).

End

etc, where the left arrow \(\leftarrow\) means the left side is an affine combination of all rows at the right side (the weights are omitted here). All blossom terms in the right side, can in turn be interpolated from the original control points of \(f_1\) and \(f_2\).

Notice all the zero multiplicities in the knot sequences on the right side in the above example appear only at the two ends. Actually exactly as we did in Section 6, we can make a critical observation that any knot sequence \(\text{Seq}\) of the blossom values (either for \(f_1\) or for \(f_2\)) to be interpolated from is exactly from the original knot vector, or a sequence of the original knot vector with some knots inserted - either case, it is well defined and the interpolation makes sense. This again is true because each knot value has been raised multiplicity by either \(d_1\) or \(d_2\) depending on which one results in a larger total multiplicity, and henceforth any knot value not in the both ends of \(\text{Seq}\) cannot be zero multiplicity.

Given the similarity to the degree elevation problem, not surprising we have the following similar Algorithm 2.