

High Order B-Spline Networks and Its Applications to Learning Feedforward Control

Zhen Song, *Student Member, IEEE* and YangQuan Chen, *Senior Member, IEEE*
 Center for Self-Organizing and Intelligent Systems (CSOIS),
 Dept. of Electrical and Computer Engineering, 4160 Old Main Hill,
 Utah State University (USU), Logan, UT 84322-4160, USA.

Abstract—This paper presents a derivation of closed-form expression of an arbitrary high order uniform B-spline. This close-form expression is then applied to the stability analysis of a learning feedforward controller (LFFC) using B-spline network (BSN) in frequency domain. BSN as a promising tool discussed in the existing literature is mainly limited to the 2nd-order B-splines case. Detailed derivation of high order BSN is still missing from the literature. In this paper, we propose to use a special type of matrix called “B-matrix” to facilitate the analysis and computation of high-order BSN for use with the LFFC scheme. Unlike the common recursive definition, the technique we developed is in “piecewise closed form,” which is a nice feature especially suitable for the LFFC analysis and design.

Index Terms—Learning feedforward control (LFFC); iterative learning control (ILC); B-spline network (BSN); high-order B-spline; frequency domain analysis;

I. INTRODUCTION

B-splines with their orders higher than 2 are referred to as “high-order B-splines”. In this paper, we first presents a detailed derivation of closed-form expression of an arbitrary high order uniform B-spline. This close-form expression is then applied to the stability analysis of a learning feedforward controller (LFFC) using B-spline network (BSN) in frequency domain. This paper is an extension of [1] where a mechatronic application example can also be found.

A. Basics of B-splines

B-spline is a common piecewise polynomial fitting tool developed by Carl De Boor and M. G. Cox independently [2] in 1972. In most references, the De Boor form B-splines are described in the continuous form. For our application, we are interested in the discrete counterpart, which is shown in (1) by referring to Fig. 1.

$$\begin{cases} \mu_{1,0}(k) = \begin{cases} 1, & k \in [mi, mi + m - 1] \\ 0, & k \notin [mi, mi + m - 1]. \end{cases} \\ \mu_{p,i}(k) = \frac{k - mi}{m(p-1)}\mu_{p-1,i}(k) + \frac{m(i+p) - k}{m(p-1)}\mu_{p-1,i+1}(k), & p > 1 \end{cases} \quad (1)$$

Several remarks on (1) follow.

- The sampling period is h . Thus, we can transform between the discrete and continuous forms by the definition $t = kh$, where t is the variable of the continuous form. So, $\mu_{p,i}(k)$ is discrete, and $\mu_{p,i}(t)$ is continuous.
- $p, i, k \in \mathbb{Z}$, and $h \in \mathbb{R}$.
- p is the order of the B-Splines. i is the index of a particular B-spline. Note that $i \in [-N_1, N_2]$, where $N_1, N_2 \in \mathbb{Z}; N_1, N_2 \geq 0$. Normally, we are interested in the cases when $p \geq 2$, then $N_1, N_2 \in \mathbb{N}$. Then, we define $N := N_2 - N_1 + 1$ as the total number of B-Splines that present in the interval of interest.
- The support of a B-Spline is d , which is defined as $d := pm$. Thus, for a p th order B-Spline, it is composed of p polynomials, with m the uniform width of each interval.

Corresponding author: Dr. YangQuan Chen. Tel.: 1(435)797-0148, Fax: 1(435)797-3054, Email: yqchen@ece.usu.edu. <http://www.csois.usu.edu/people/yqchen/>

- We also define N_p as the total number of samplings. Thus, $k \in [0, N_p - 1]$.

Now, take k as the input, and $y^j(k)$ as the output, we can construct a BSN that similar to the definition of an artificial neural network (ANN).

$$y^j(k) = \sum_{i=-N_1}^{N_2} w_i^j \mu_{p,i}(k). \quad (2)$$

Here, $\mu_{p,i}(k)$ is the i -th B-Spline of p -th order as defined in (1). w_i^j is the i -th weight in the j -th iteration. Accordingly, $y^j(k)$ is the output that associated with the input k in the j -th iteration. This relationship can be seen clearly in Fig. 1. The center of a B-spline $\mu_{p,i}$ in the bottom subplot is aligned with its associated weight, w_i^j , in the top subplot. The solid line in the top subplot is $y^j(k)$, which is smooth, and its shape is controlled by w_i^j . By adjusting w_i^j in the iteration domain, it is possible that $y^j(k)$ fits our objective function better and better, as explained in the following.

B. Basics of LFFC and ILC

LFFC [3], [4], [5], [6], [7] is a variant of “Iterative Learning Control” (ILC) [8], [9]. The basic ideas are the same in the sense that LFFC and ILC are both considered to be a *value-added block* for enhancing the feedback control performance of some classes of systems by capitalizing on the repetitiveness of these systems’ operation. The general LFFC scheme is illustrated in Fig. 2, where **FBC** stands for “feedback controller” and y_d is the given desired output trajectory to be tracked. The scheme works as follows. After the $(j-1)$ th iteration, the feedforward control signal u_F^{j-1} and the feedback control signal u_C^{j-1} are to be store in the memory bank for constructing the feedforward control signal at the next repetition,

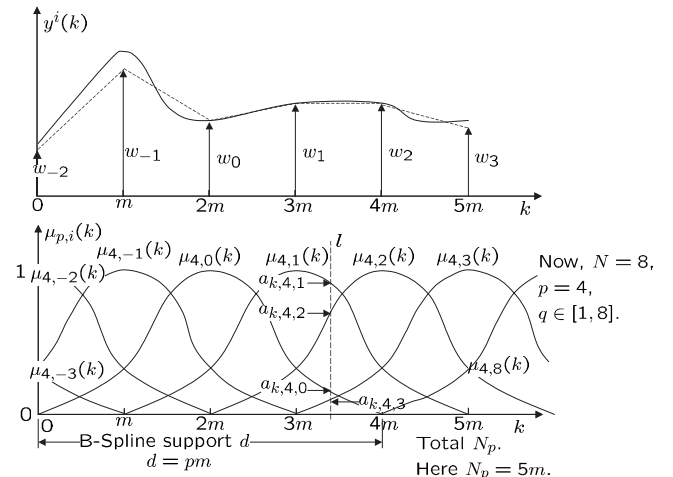


Fig. 1. An illustration of high-order (4-th order) B-splines and the related filtering process.

i.e., u_F^j . The stored feedback control signal u_C^{j-1} is filtered through $H(z, z^{-1})$ and multiplied by a learning gain γ . Note that the general filter $H(z, z^{-1})$ is the key LFFC element. The correct selection of this filter and its parameters is the key to a successful LFFC scheme [1].

Although a general neural network can be used in LFFC, so far only 2nd order BSNs have been considered. The features of BSN made it easier for stability analysis [6], [4], [7], [1], [10] and in [1], [10], two parameters tuning rules were proposed for LFFC with the 2nd order BSN. Also, the effects of dilation to the 2nd order B-spline were discussed in [1]. However, the performance of high-order B-spline is still not very clear. It is mainly due to the difficulty of analysis. Since 2nd order B-splines are actually triangles, it is straightforward to write down their closed-form polynomial representation, but it is difficult for high-order B-splines. In this paper, we will propose a new mathematic tool named B-matrix, and extend some results in [1] to the arbitrary high-order uniform B-spline system.

C. Problem Description

When we use BSN to filter the learning feedforward control signal $u_F^j(kh)$, we have (3).

$$u_F^j(k) = y^j(k) = \sum_{i=-N_1}^{N_2} w_i^j \mu_{p,i}(k), \quad (3)$$

In addition to the generic BSN properties that we have described, we assume this BSN has no dilation now. In [1], dilation was defined as the ‘‘basis functions do not overlap more than 50% each other.’’ It is valid for 2nd BSN. Now, we extend this definition to high-order BSN: A p th high-order BSN is dilated if the basis functions do not overlap exactly $(1-1/p) \times 100\%$ of each two consecutive B-splines. Accordingly, it is non-dilation if each two consecutive B-splines overlap exactly $(1-1/p) \times 100\%$.

Figure 1 is an example of non-dilation 4th order BSN. It is easy to observe that the delay between two consecutive B-splines is m , and the support d is equals to $4m$. On the right hand side there is a vertical line labeled l , which intersects with the B-splines $\mu_{4,0}$, $\mu_{4,1}$, $\mu_{4,2}$ and $\mu_{4,3}$ at the points $a_{k,4,0}$, $a_{k,4,1}$, $a_{k,4,2}$ and $a_{k,4,3}$ respectively. These intersections are the values of B-splines when the variable is k . In [1], a_k is defined as the value of $\mu_{2,i}(k)$ at

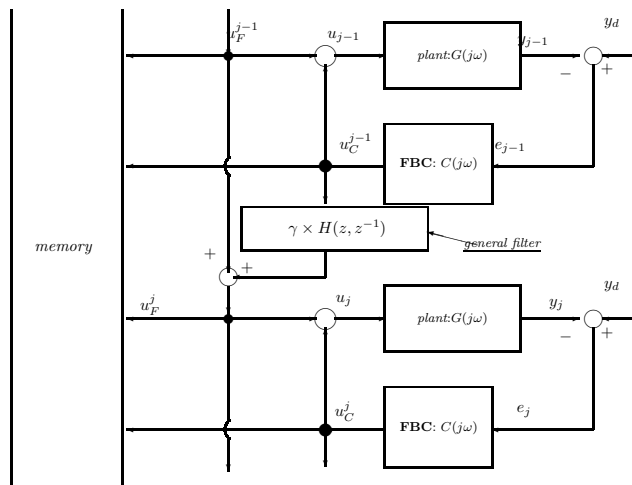


Fig. 2. Block diagram of LFFC using a general filter $H(z, z^{-1})$.

the point k . When we denote a_k as $a_{k,2,i}$, the value of $\mu_{2,i-1}(k)$ is $1 - a_k$. For high-order B-spline, the relationships among the consecutive B-splines are not that simple. In this paper, we propose the so-called ‘‘B-matrix’’ to simplify this procedure, and find out the closed-form analytical representation of B-splines.

The remaining part of the paper is organized as follows. Section II explains the proposed B-matrix idea, the construction of the B-matrix and the B-spline representation using B-matrix. Section III applies the B-matrix to the design procedure of LFFC originally described in [1]. Due to the page limit, after we briefly introduced the control law, we just present the different equations between 2nd order and high-order B-spline cases. Finally, Sec. IV concludes this paper with some proposed future research efforts.

II. ANALYSIS OF HIGH-ORDER UNIFORM B-SPLINE BY B-MATRIX

This section presents a new mathematical tool named B-matrix for the high-order B-spline analysis. The subsections II-A and II-B discuss B-matrix construction and its application on B-spline analysis, respectively. Subsection II-C explains the reason why B-matrix has this structure.

A. B-Matrices and Their Construction

First, let us see two examples: the 4-th order B-matrix B_4 in (4) and the 5-th order B-matrix B_5 in (5), which are associated with the 4-th and 5-th order B-splines, respectively.

$$B_4 = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 3 & 6 & 4 \\ 1 & 3 & 3 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (4)$$

$$B_5 = \begin{bmatrix} 1 & 2 & 4 & 8 & 16 \\ 1 & 3 & 6 & 12 & 8 \\ 1 & 3 & 7 & 6 & 4 \\ 1 & 3 & 3 & 3 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (5)$$

By careful observation, we can find the symmetric structure in (4) and (5) as marked in (6) and (7).

$$B_4 = \begin{bmatrix} \swarrow & \xrightarrow{\times 2} & 2 & \xrightarrow{\times 2} & 4 & \xrightarrow{\times 2} & 8 \\ & l_1 & \textcircled{2} & & l_2 & & \\ 1 & \xrightarrow{+2^1} & \swarrow & \xrightarrow{\times 2} & \swarrow & \textcircled{3} & 4 \\ & \textcircled{1} & & & & & \\ 1 & \xrightarrow{+2^1} & \swarrow & & \swarrow & & 2 \\ \swarrow & l_2 & & \textcircled{4} & & l_1 & \swarrow \\ \swarrow & & 1 & & 1 & & \swarrow \end{bmatrix} \quad (6)$$

$$B_5 = \begin{bmatrix} \swarrow & \xrightarrow{\times 2} & 2 & \xrightarrow{\times 2} & 4 & \xrightarrow{\times 2} & 8 & \xrightarrow{\times 2} & 16 \\ & l_1 & & \textcircled{2} & & l_2 & & & \\ 1 & \xrightarrow{+2^1} & \swarrow & \xrightarrow{\times 2} & 6 & \xrightarrow{\times 2} & 12 & & 8 \\ & \textcircled{1} & & & & & \textcircled{3} & & \\ 1 & \xrightarrow{+2^1} & 3 & \xrightarrow{+2^2} & \swarrow & & 6 & & 4 \\ & \textcircled{1} & & & & & & & \\ 1 & \xrightarrow{+2^1} & \swarrow & & 3 & & \swarrow & & 2 \\ \swarrow & l_2 & & \textcircled{4} & & l_1 & & & \swarrow \\ \swarrow & & 1 & & 1 & & 1 & & \swarrow \end{bmatrix} \quad (7)$$

The markings in (6) and (7) follow the following conventions:

- 1) The operations above arrows represent the transfer functions when along the direction of the arrow, e.g., $1 \xrightarrow{+2^1} 3$ represents $3 = 1 + 2^1$, and $4 \xrightarrow{\times 2} 8$ represents $8 = 4 \times 2$.
- 2) l_1 and l_2 are the two diagonals of the matrices. The slash or back slash on some of the entries indication the direction of the diagonals.
- 3) $\textcircled{1}, \textcircled{2}, \textcircled{3}$ and $\textcircled{4}$ represent 4 regions. They are segmented from each other by l_1 and l_2 .

Now, it is easy to conclude that regardless the order of a B-matrix is even, like \mathbf{B}_4 , or odd, like \mathbf{B}_5 , the same symmetric structure holds. So, we can construct a B-matrix of any order by the following rules:

- 1) Firstly, we define two diagonal lines for each matrix: l_1 and l_2 as the “upper left to lower right” diagonal and the “upper right to lower left” diagonal.
- 2) Then, as shown in equations 6 and 7, there are four regions defined as region ①, ②, ③, and ④.
- 3) At the very beginning, put a 1 at each entry of the first column.
- 4) In region ① the entry on right equals to the entry on left plus 2^n , where n is the column number of the left entry.
- 5) In region ② the entry on right equals to is the double of the left entry.
- 6) Region ③ and ④ are symmetric to region 1 and 2 with respect to l_2 .

The pseudo code for B-matrix generating is listed in Table I.

B. Compute the Closed-Form B-splines

Refer to Fig. 1 and equation 3. A p th order B-spline can be represented in a piecewise polynomial form with p segments. For the i th segment, we have written down its polynomial form from the i th column of the B-matrix by the following steps. Taking a 4th order B-spline as an example, Fig. 3 demonstrates how it was constructed from the lower-order B-splines. The plot on the top demonstrates the relationship. For a clear presentation, we separated every spline, and listed them below. For example, the $\mu_{4,0}$ was composed by two third-order B-splines: $\mu_{3,0}$ and $\mu_{3,1}$. $\mu_{3,0}$ was constructed by $\mu_{2,0}$ and $\mu_{2,1}$. In each segment, from seg 1 to seg 4, the 4th order B-spline can be represented by one polynomial. The polynomial of seg i can be computed from the i -th column of the B-matrix. To compute the polynomial associated with a specific segment, we can convert the entries of a corresponding column of the B-matrix into a binary form, thus generate a new matrix named *evaluation matrix*. For segment 0-1, we can write the first column

TABLE I
PSEUDO CODE FOR B-MATRIX GENERATION

```

Construct a  $n \times n$  matrix  $\mathbf{M}$ .
Every element of the first column of  $\mathbf{M}$  is 1.
For each row from 1 to  $n-1$ 
  For each column from 2 to  $n - row + 1$  (
    at the left of  $l_2$ )
    if column  $\leq$  row (at the left of  $l_1$ )
      We are in region ①.
       $\mathbf{M}(\text{row}, \text{column}) =$ 
         $\mathbf{M}(\text{row}, \text{column} - 1) + 2^{(\text{column}-1)}$ 
    else (at the right of  $l_1$ )
      We are in region ②.
       $\mathbf{M}(\text{row}, \text{column}) =$ 
         $\mathbf{M}(\text{row}, \text{column} - 1) \times 2$ 
    end if
  end for loop
end for loop

Now, fill the rest elements by the
symmetric property. For each row
from 2 to  $n$ 
  For each column from  $n + 1 - row$  to  $n$  (at
  the right of  $l_2$ )
     $\mathbf{M}(\text{row}, \text{column}) =$ 
       $\mathbf{M}(n + 1 - \text{column}, n + 1 - \text{row})$ 
  end for loop
end loop

```

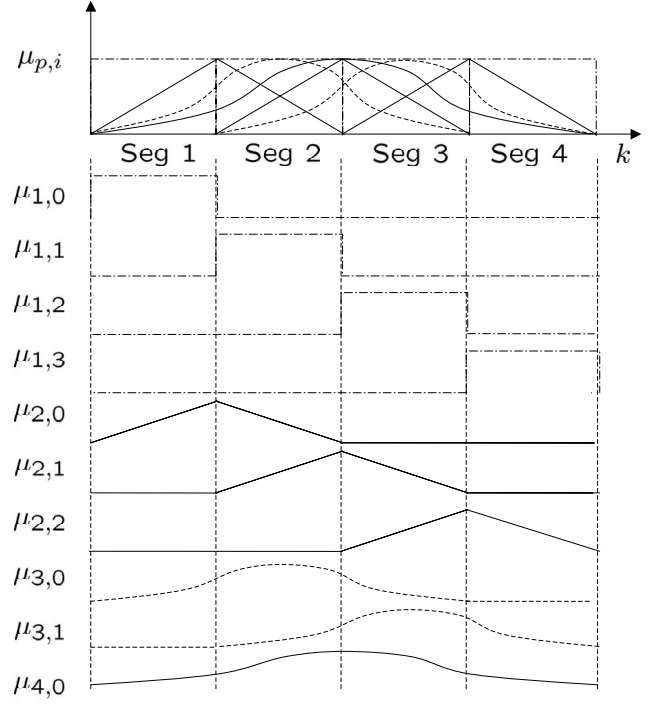


Fig. 3. Decomposing a high-order B-spline into low-order B-splines.

of \mathbf{B}_4 in the binary form, which is the first evaluation matrix \mathbf{E}_1 . We can then convert the entries of a corresponding column of the B-matrix into a binary form, thus generate a new matrix named *evaluation matrix*. For seg 1, we can write the first column of \mathbf{B}_4 in the binary form, which is the first evaluation matrix \mathbf{E}_1 .

- For each column of B-matrix, transfer it into binary form, thus generate what we called the *evaluation matrix*. For example, to compute the polynomial in seg 1, we can write the first column of \mathbf{B}_4 in the binary form, which is the first evaluation matrix \mathbf{E}_1 .

$$\mathbf{E}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- With a $n \times n$ evaluation matrix \mathbf{E}_i , we start from the right lower corner, $e_{n,n}$.
- With only two operations: Either going from an entry to the entry on the above, or going to the one on the left above, we can find one or many paths from $e_{n,n}$ to any element in the first row, $e_{1,\cdot}$, and this path passes only the entries that equal to 1.
- For each path, compute a path polynomial following the rules in (8).

$$\begin{cases} e_{i,j} \longrightarrow e_{i-1,j-1} & : \times (-b_{p+i-j}) \\ e_{i,j} \longrightarrow e_{i,j-1} & : \times b_{p-j} \end{cases} \quad (8)$$

where b_i is defined in (9)

$$b_i := \frac{k - mi}{m}. \quad (9)$$

Note that b_i is a function of k , thus b_i is a function instead of a constant.

- Add the b_i associate with all the possible paths together divided by $n!$.

Let us still use the 4-th order B-spline as the illustrative example. All the possible paths and the multiplication operations for each step are marked in equation 10.

$$\mathbf{E} = \begin{bmatrix} e_{1,1} & \swarrow -b_4 & e_{1,2} & \swarrow -b_3 & e_{1,3} & \swarrow -b_2 & e_{1,4} \\ e_{2,1} & \uparrow b_2 & e_{2,2} & \uparrow b_1 & e_{2,3} & \uparrow b_0 & e_{2,4} \\ e_{3,1} & & e_{3,2} & \swarrow -b_4 & e_{3,3} & \swarrow -b_3 & e_{3,4} \\ e_{4,1} & & e_{4,2} & & e_{4,3} & \swarrow -b_4 & e_{4,4} \end{bmatrix} \quad (10)$$

However, if any entry on a path is 0, this path is “invalid,” i.e., it is not a path at all. The path is valid only if all the entries on it are 1’s. In equation 11 we give four evaluation matrices with each one associated with one column of \mathbf{B}_4 . Here, we marked only operations on the valid paths.

Then, it is easy to see that the result of the evaluation matrix in the left most segment, *seg 1*, is $\frac{b_0^3}{3!}$. The polynomial of the second evaluation matrix is

$$\frac{-b_0 b_1 b_3}{3!} + \frac{-b_0^3}{3!} + \frac{-b_1^2 b_4}{3!}$$

Then, we can compute the explicit polynomial by replacing b_i with k .

C. Validity of B-Matrix

In the former subsection, we directly gave the structure of B-matrix without explanation. In this subsection, we will briefly explain why the structure looks like this.

The B-matrix was derived by observing the presentations of all the lower order splines. Let us use the 4-th B-spline as an example again.

In Fig. 4, “Seg 1” represents “Segment 1” etc. In the row of $\mu_{1,0}$, a line on right in the column of “Seg 1” indicates that the B-spline $\mu_{1,0}$ has non-zero values, where we say it “exists.” For the column without a line $\mu_{1,0}$ is always zero, e.g., Seg 2 is this case. We say $\mu_{1,0}$ does not exist in Seg 2.

Then, we can use matrices to represent the existence of very spline, as shown in Table II. In the table, the four areas with double vertical lines on both sides are the four segments in Fig. 4. If the item on the intersection of row $\mu_{j,\cdot}$ and column $\mu_{\cdot,i}$ is 1, then $\mu_{j,i}$ exists in the current segment, otherwise $\mu_{j,i}$ does not exist. It is easy to conclude that the matrices between the double vertical lines are the evaluation matrices and after converting them into the decimal, we can get the B-matrix.

With the help of the Table II, we can list the polynomial of each B-spline in every segment. This result leads to Table III. The first 10 rows of this table are filled with the recursive presentation format, i.e., the DeBoor-Cox form. The last row is the closed form that we need. Carefully comparing this table and its B-matrix (referring to Table II and the equation (4)), we can observe the structure as shown in equation 6. If the readers list the cases of other high-order B-splines, it is easy to see that this structure is not by accident. For example, the length of $\mu_{1,\cdot}$ is just one. Thus, after converting into binary form, we need to do one left shift in order to get the value of right segment from the left one. All the other B-splines have

	Seg 1	Seg 2	Seg 3	Seg 4
$\mu_{1,0}$				
$\mu_{1,1}$				
$\mu_{1,2}$				
$\mu_{1,3}$				
$\mu_{2,0}$				
$\mu_{2,1}$				
$\mu_{2,2}$				
$\mu_{3,0}$				
$\mu_{3,1}$				
$\mu_{4,0}$				

Fig. 4. Decomposition of a 4-th order B-Spline

the same relationship, except that the situation of the boundaries are different, where the overlapping relationships are different than what in between.

D. Computational Approach for Symbolic Closed-Form B-spline

The aim of this subsection is to find the symbolic closed-form (piecewise polynomial) representation of arbitrary high-order B-spline by Matlab programs.

The basic idea is the same as B-matrix, but we proposed an easy way of coding. Observing the evaluation matrix, we find out that the path always starts from $e_{p,p}$ to $e_{1,p-j+1}$, where p is the order of the B-spline and the j is the index of a certain segment. If we start from the entry $e_{p,p}$, we need exactly $p-1$ steps to either arrive $e_{1,p-j+1}$ or fail. This is due to the fact that no matter we move up or move upper-left, the row number will be reduced by 1 for each step. The only option we can choose are the time to turn left. Thus, we need to move $p-1$ steps, among them $j-1$ steps are going upper-left. So the total number of paths is C_{p-1}^{j-1} . Now, the question is: What are them exactly. Let us refer this “list of combination” problem: Choose N bits from M bits, how to list all the possible combinations? For example, if $f_2(C_3^2)$ is the binary value of selecting 2 out of 3, then it is a set of $\{011, 101, 110\}$, and its decimal form is $f_{10}(C_3^2) = \{3, 5, 6\}$. The question is: What is the algorithm to find $f(C_M^N)$? It is easy to prove:

$$f(C_M^N) = \{f(C_{M-1}^N), 2 \times f(C_{M-1}^{N-1}) + 1\}.$$

So we have the recursive algorithm as shown in Tab. IV

Then, transfer each element in L to binary, and check every bit of them with the following rules:

$$\begin{cases} \text{Initially} & row = p, col = p; \\ \text{If the next bit is 0,} & \times b_{p-col}, row = row - 1; \\ \text{If the next bit is 1,} & \times (-b_{p-col+row}), \\ & row = row - 1, col = col - 1. \end{cases}$$

For example, $f_{10}(C_3^2) = \{1, 2, 4\}$ is associated with *seg 2* of 4th order B-spline. It is binary form and the polynomial form (without being divided by $(p-1)!$) is

$$\begin{bmatrix}
0 & 0 & 0 & 1 & \vdots & 0 & 0 & 1 & 0 & \vdots & 0 & 1 & 0 & 0 & \vdots & 1 & 0 & 0 & 0 \\
& & \uparrow b_0 & \vdots & & \uparrow b_1 & \swarrow -b_2 & & & \vdots & \uparrow b_2 & \swarrow -b_3 & & & \vdots & \swarrow -b_4 & & & \\
0 & 0 & 0 & 1 & \vdots & 0 & 0 & 1 & 1 & \vdots & 0 & 1 & 1 & 0 & \vdots & 0 & 1 & 0 & 0 \\
& & \uparrow b_0 & \vdots & & \uparrow b_1 & \swarrow -b_3 & \uparrow b_0 & & \vdots & \swarrow -b_4 & \uparrow b_1 & \swarrow -b_3 & & \vdots & \swarrow -b_4 & & & \\
0 & 0 & 0 & 1 & \vdots & 0 & 0 & 1 & 1 & \vdots & 0 & 0 & 1 & 1 & \vdots & 0 & 0 & 1 & 0 \\
& & \uparrow b_0 & \vdots & & \swarrow -b_4 & \uparrow b_0 & & & \vdots & \swarrow -b_4 & \uparrow b_0 & & & \vdots & & \swarrow -b_4 & & \\
0 & 0 & 0 & 1 & \vdots & 0 & 0 & 0 & 1 & \vdots & 0 & 0 & 0 & 1 & \vdots & 0 & 0 & 0 & 1
\end{bmatrix} \quad (11)$$

TABLE II
THE EVALUATION MATRICES OF THE 4-TH ORDER B-MATRIX

	$\mu_{\cdot,3}$	$\mu_{\cdot,2}$	$\mu_{\cdot,1}$	$\mu_{\cdot,0}$	$\mu_{\cdot,3}$	$\mu_{\cdot,2}$	$\mu_{\cdot,1}$	$\mu_{\cdot,0}$	$\mu_{\cdot,3}$	$\mu_{\cdot,2}$	$\mu_{\cdot,1}$	$\mu_{\cdot,0}$	$\mu_{\cdot,3}$	$\mu_{\cdot,2}$	$\mu_{\cdot,1}$	$\mu_{\cdot,0}$
$\mu_{1,\cdot}$	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0
$\mu_{2,\cdot}$	0	0	0	1	0	0	1	1	0	1	1	0	0	1	0	0
$\mu_{3,\cdot}$	0	0	0	1	0	0	1	1	0	1	1	1	0	0	1	0
$\mu_{4,\cdot}$	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

TABLE III
COMPUTE A 4TH ORDER B-SPLINE

Spline	Seg 1	Seg 2	Seg 3	Seg 4
$\mu_{1,0}$	1	0	0	0
$\mu_{1,1}$	0	1	0	0
$\mu_{1,2}$	0	0	1	0
$\mu_{1,3}$	0	0	0	1
$\mu_{2,0}$	b_0	$-b_2$	0	0
$\mu_{2,1}$	0	b_1	$-b_3$	0
$\mu_{2,2}$	0	0	b_2	$-b_4$
$\mu_{3,0}$	$\mu_{2,0} \times \frac{b_0}{2}$	$\mu_{2,0} \times \frac{b_0}{2} + \mu_{2,1} \times \frac{-b_3}{2}$	$\mu_{2,1} \times \frac{-b_3}{2}$	0
$\mu_{3,1}$	0	$\mu_{2,1} \times \frac{b_1}{2}$	$\mu_{2,1} \times \frac{b_1}{2} + \mu_{2,2} \times \frac{-b_4}{2}$	$\mu_{2,2} \times \frac{-b_4}{2}$
$\mu_{4,0}$	$\mu_{3,0} \times \frac{b_0}{3}$	$\mu_{3,0} \times \frac{b_0}{3} + \mu_{3,1} \times \frac{-b_4}{3}$	$\mu_{3,0} \times \frac{b_0}{3} + \mu_{3,1} \times \frac{-b_4}{3}$	$\mu_{3,1} \times \frac{-b_4}{3}$
$\mu_{4,0}$	$\frac{b_0^3}{3!}$	$\frac{b_0}{3} \times \frac{-b_3}{2} \times b_1 +$ $\frac{b_0}{3} \times \frac{b_0}{2} \times (-b_2) +$ $\frac{-b_4}{3} \times \frac{b_1}{2} \times b_1$	$\frac{b_0}{3} \times \frac{-b_3}{2} \times (-b_3) +$ $\frac{-b_4}{3} \times \frac{b_1}{2} \times (-b_3) +$ $\frac{-b_4}{3} \times \frac{-b_4}{2} \times b_2$	$\frac{(-b_4)^3}{3!}$

TABLE IV
PSEUDO CODE FOR LIST OF COMBINATIONS

```

function L=comb(M,N,L)
if N==0
    L={L, 0};
end
if N==1
    L={2^0, 2^1, ..., 2^{M-1}};
end
if M==N
    L={L, 2^M - 1};
else
    L={L, 2 * comb(M-1, N-1, L) + 1, 2 * comb(M-1, N, L)}
end

```

$$\begin{cases}
100, & (-b_4)b_1b_1; \\
010, & b_0(-b_3)b_1; \\
001, & b_0b_0(-b_2).
\end{cases}$$

If we add these three polynomial together, this is the solution of 4th order B-spline at its 2nd segment.

III. LEARNING FEEDFORWARD CONTROL USING HIGH-ORDER B-SPLINE NETWORK

Now, we can plug the B-matrix into the LFFC derivation procedure of [1], and extend the 2nd order results to the high-order BSN.

A. The Control Law

The overall control signal at the j th repetitive operation is given by

$$u^j(k) = u_C^j(k) + u_F^j(k)$$

where $u_C^j(k)$ is the feedback control law, and u_F^j is the feedforward control law. We update the BSN by

$$\begin{aligned} w_i^j &= w_i^{j-1} + \Delta w_i^j \\ \Delta w_i^j &= \frac{\gamma \sum_{l=0}^{N_p-1} \mu_i(l) u_C^{j-1}(l)}{\sum_{l=0}^{N_p-1} \mu_i(l)} \end{aligned}$$

So,

$$u_F^j(k) = u_F^{j-1}(k) + \gamma \sum_{i=1}^N \mu_i(k) \frac{\sum_{l=0}^{N_p-1} \mu_i(l) u_C^{j-1}(l)}{\sum_{l=0}^{N_p-1} \mu_i(l)} \quad (12)$$

Therefore, in general, the LFFC updating law can be written in the following filter form:

$$u_F^j = u_F^{j-1} + \gamma H(z, z^{-1}) u_C^{j-1}.$$

B. Frequency domain analysis

The frequency stability analysis presented in [1] still holds for high-order BSN. So, without repeating the detailed derivations, here we simply extend some of the 2nd order BSN-based LFFC design into the high-order case. Following the convention in [1], we denote

$$a_{k,p,i} := \mu_{p,0}(L_{k,p,i}), \quad (13)$$

$$L_{k,p,i} = k - mi + \frac{mp}{2}, \quad (14)$$

$$A_k(z) = Z(\mu_{p,0}(k)). \quad (15)$$

Since $mi - \frac{mp}{2}$ is the center of the i th B-spline, $L_{k,p,i}$ is the distance from k to the center of i th B-spline. Then,

$$Z(a_{k,p,i}) = A_k(z) z^{-mi + \frac{mp}{2}}.$$

Thus, $a_{k,p,i}$ is the value (height) of the i th B-spline, with an order of p , at the k th sample time. $L_{k,p,i}$ is the distance between the k th sample and the center of the i th B-spline. For a p th order B-spline, its area in unit of h is S_p , i.e.,

$$S_p := \sum_{l=0}^{N_p-1} \mu_{p,i}(l). \quad (16)$$

Clearly, equation 16 is compatible with equation 14 in [1]. So $S_2 = m$. Observe that $\sum_{k=im+1}^{im+m} \mu_{p,i}(k) = \sum_{k=1}^m \mu_{p,0}(k) = S_p$. We can prove the following:

$$\begin{aligned} S_{P+1} &= \sum_{k=1}^{pm} \left[\frac{k}{mp} \mu_{p,0}(k) + \frac{m(i+p+1)-k}{mp} \mu_{p,1}(k) \right] \\ &= \frac{1}{mp} \sum_{k=1}^{pm-m} k \mu_{p,0}(k) + \frac{p+1}{p} \sum_{k=m+1}^{mp} \mu_{p,1}(k) \\ &\quad - \sum_{k=m+1}^{mp} \frac{k}{mp} \mu_{p,1}(k) \\ &= \frac{1}{mp} \sum_{k=1}^{pm-m} k \mu_{p,0}(k) + \frac{p+1}{p} S_p \\ &\quad - \sum_{k=1}^{pm-m} \frac{k}{mp} \mu_{p,1}(k+m) - \sum_{k=1}^{pm-m} \frac{1}{p} \mu_{p,1}(k+m) \\ &= \frac{p+1}{p} S_p - \frac{1}{p} \sum_{k=1}^{pm-m} \mu_{p,0}(k) \\ &= S_p \end{aligned}$$

Thus, $S_p \equiv m$. Equation (12) can then be rewritten as:

$$u_F^j(k) = u_F^{j-1}(k) + \frac{\gamma}{m} \left[\sum_{i=\lfloor \frac{k}{m} \rfloor}^{\lfloor \frac{k}{m} \rfloor + p - 1} \left(a_{k,q,i} \sum_{k=0}^{N_p-1} \mu_{p,i}(k) u_C^{j-1}(k) \right) \right]$$

Therefore, finally, we have

$$u_F^j(k) = u_F^{j-1}(k) + \gamma H_2(z, z^{-1}) \times \sum_{i=0}^{p-1} A_k(z) z^{-mi+mp/2} u_C^{j-1}(k).$$

Based on the above derivation, we can design d and γ by the similar method described in [1]. The key point here is that the parameterized z -transfer function of the high-order B-spline network filter can be obtained via the known form of $A_k(z) z^{-mi+mp/2} u_C^{j-1}(k)$ as shown in the above equation, which is a generalized form of the 2nd order BSN filtering process.

IV. CONCLUSION

In this paper we proposed an approach to compute and analyze high-order uniform B-spline by the so-called B-matrix method. Subsequently, we are able to extend the existing LFFC design method of the 2nd order BSN to the high-order BSN. The stability criterions of the 2nd order BSN still holds for the high-order BSN case. The future research efforts could be 1) Finding the optimum order of BSN; 2) Checking the dilation effect; 3) Optimal trade-off between the BSN order and the dilation level.

V. REFERENCES

- [1] Y. Chen, K. Moore, and V. Bahl, "Learning feedforward control using a dilated b-spline network: frequency domain analysis and design," *IEEE Transactions on Neural Networks*, vol. 15, no. 2, pp. 355–366, Mar 2004.
- [2] K. Joy, "The DeBoor-Cox calculation." [Online]. Available: <http://graphics.cs.ucdavis.edu/CAGDNotes/Deboor-Cox-Calculation/Deboor-Cox-Calculation.html>
- [3] J. G. Starrenburg, W. T. C. van Luene, W. Oelen, and J. V. Amerongen, "Learning feedforward controller for a mobile robot vehicle," *Control Engineering Practice*, vol. 14, no. 9, pp. 1221–1230, 1996.
- [4] T. de Veries, W. Velthuis, and J. van Amerongen, "Learning feedforward control of a flexible beam," in *Proc. of the IEEE International Symposium on Intelligent Control*, Dearborn, MI, USA, 1996, pp. 103–108.
- [5] G. Otten, T. J. A. de Vries, J. van Amerongen, A. M. Rankers, and E. W. Gaal, "Linear motor motion control using a learning feedforward controller," *IEEE/ASME Trans. Mechatronics*, vol. 2, no. 3, pp. 179–187, 1997.
- [6] W. Velthuis, T. de Veries, P. Schaak, and E. Gaal, "Stability analysis of learning feedforward control," *Automatica*, vol. 36, no. 12, pp. 1889–1895, Dec. 2000.
- [7] W. J. Velthuis, "Learning feedforward control: Theory, design and applications," Ph.D. dissertation, University of Twente, Dutch, Feb. 2000.
- [8] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *J. of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984.
- [9] K. L. Moore, *Iterative learning control for deterministic systems*. Advances in Industrial Control. Springer-Verlag, 1993.
- [10] Y. Chen and K. Moore, "Frequency domain adaptive learning feedforward control," in *Proc. of the IEEE International Symposium on Intelligent Control*, Banff, Alberta, Canada, July 2001. [Online]. Available: <http://csois.usu.edu/publications/pdf/pub098.pdf>