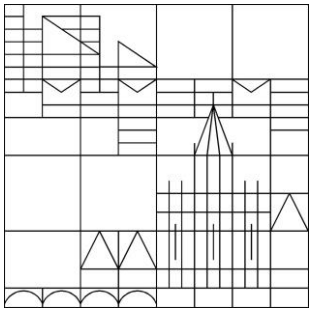

Universität Konstanz

Matrix Methods for the Simplicial Bernstein Representation and for the Evaluation of Multivariate Polynomials

Jihad Titi
Jürgen Garloff

Konstanzer Schriften in Mathematik

Nr. 363, Juli 2017

ISSN 1430-3558

Konstanzer Online-Publikations-System (KOPS)
URL: http://nbn-resolving.de/urn:nbn:de:bsz:352-0-416554

Matrix Methods for the Simplicial Bernstein Representation and for the Evaluation of Multivariate Polynomials

Jihad Titi[a] and Jürgen Garloff[a,b]

[a] *Department of Mathematics and Statistics, University of Konstanz, D-78464 Konstanz, Germany*

[b] *University of Applied Sciences / HTWG Konstanz, Institute for Applied Research, Brauneggerstr. 55, D-78405 Konstanz, Germany*

Abstract

In this paper, multivariate polynomials in the Bernstein basis over a simplex (simplicial Bernstein representation) are considered. Two matrix methods for the computation of the polynomial coefficients with respect to the Bernstein basis, the so-called Bernstein coefficients, are presented. Also matrix methods for the calculation of the Bernstein coefficients over subsimplices generated by subdivision of the standard simplex are proposed and compared with the use of the de Casteljau algorithm. The evaluation of a multivariate polynomial in the power and in the Bernstein basis is considered as well. All the methods solely use matrix operations such as multiplication, transposition, and reshaping; some of them rely also on the bidiagonal factorization of the lower triangular Pascal matrix or the factorization of this matrix by a Toeplitz matrix. The latter one enables the use of the Fast Fourier Transform hereby reducing the amount of arithmetic operations.

Keywords: Bernstein coefficient, simplicial Bernstein representation, range enclosure, simplicial subdivision, polynomial evaluation.

1. Introduction

Solving global optimization problems is of paramount importance in many real-life and scientific problems; polynomial global optimization problems form a significant part of them. In this paper, we consider the case that

the underlying region is a simplex. Applications of polynomial global optimization over a simplex include portfolio optimization, population dynamics, genetics, finding maximum stable sets in graphs, and lower bounds for the crossing number of certain classes of graphs, see [13]. A commonly used approach for solving global minimization problems is the branch and bound method. This is summarized as splitting of the search region into smaller parts and using suitable tests to discard subregions that cannot contain any global minimizer. The latter ones require the ability to compute tight bounds for the range of the objective function and the functions describing the constraints over the considered search region. In the case of polynomial optimization problems, one can make use of the expansion of a polynomial into Bernstein polynomials, see [7], [16], [18], [19], [21], [22]. Then the minimum and maximum of the coefficients of this expansion, the so-called Bernstein coefficients, provide bounds for the range of the polynomial over the search region. If the search region is a simplex, applications of this range enclosing property of the Bernstein coefficients includes reachability analysis for differential equations [5] and Lyapunow stability analysis [25], [26]. In some cases, not all Bernstein coefficients have to be computed in order to find their minimum and maximum due to monotonicity properties of the simplicial Bernstein coefficients of multivariate monomials [28]. The advantages and disadvantages of the use of the simplicial Bernstein basis compared to the use of the *tensorial* Bernstein basis, e.g. [7], where the underlying region is a box of $\mathbb{R}^n$, are discussed in [23]. For a comprehensive survey on the properties of the Bernstein bases see [6].

Using the same matrix representation of the array of the Bernstein coefficients as in [22] (in the case of the tensorial Bernstein basis), we present in this paper two methods for the calculation of the Bernstein coefficients over the standard simplex. Our methods use as [22] solely matrix operations such as multiplication, transposition, and reshaping but rely mainly on the use of the lower triangular Pascal matrix. The first method is based on the fact that this matrix can be represented as a product of bidiagonal matrices [27], [30]. The second method relies on a factorization of the Pascal matrix by using a Toeplitz matrix. This enables the application of the Fast Fourier Transform (FFT), see [27], [3, Chapter 13], hereby reducing the amount of arithmetic operations.

We also consider the subdivision of the standard simplex into two sub-

simplices. According to the different factorizations of the Pascal matrix, we obtain two methods for the computation of the Bernstein coefficients on the two subsimplices. We show that in nearly all cases these methods are superior to the use of the de Casteljau algorithm [15], [16], [19]. For the computation of the Bernstein coefficients on squares and triangles generated by subdivision of the standard simplex in the bivariate case see [10], [11].

The organization of our paper is as follows: In the next section, we introduce the notation and some matrices and factorizations which are used throughout the paper. In Section 3, we briefly recall the most important properties of the Bernstein expansion over a simplex. In Section 4, we derive the two proposed matrix methods for the computation of the Bernstein coefficients over the standard simplex. We present matrix methods for the calculation of the Bernstein coefficients on subsimplices generated by subdivision in Section 5 and for the evaluation of a polynomial in power as well as in the Bernstein representation in Section 6.

2. Notation

In this section, we introduce the notation that we are using throughout this paper. Let $n \in \mathbb{N}$ (set of the nonnegative integers) be the number of variables. A multi-index $(i_1, \ldots, i_n) \in \mathbb{N}^n$ is abbreviated by i. In particular, we write 0 for $(0, \ldots, 0)$. We put $|i| := i_1 + \cdots + i_n$. Comparison between multi-indices and arithmetic operations using them are understood entry-wise. For $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, its *monomials* are defined as $x^i := \prod_{s=1}^n x_s^{i_s}$. For $d = (d_1, \ldots, d_n) \in \mathbb{N}^n$ such that $i \leq d$, we use the compact notations $\sum_{i=0}^d := \sum_{i_1=0}^{d_1} \cdots \sum_{i_n=0}^{d_n}$ and $\binom{d}{i} := \prod_{s=1}^n \binom{d_s}{i_s}$.

Let $\mathbb{R}^{r,t}$ be the set of r-by-t real matrices. For $M \in \mathbb{R}^{r,r}$, the *antidiagonal* of M is formed by its entries in position $(r + 1 - v, v), v = 1, \ldots, r$. We introduce the following matrices[1] of $\mathbb{R}^{k+1,k+1}$. The lower triangular Toeplitz matrix T_k is defined as

$$(T_k)_{ij} := \begin{cases} \frac{1}{(i-j)!}, & j \leq i, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

[1] We will use small Latin letters to denote not only multiindices but also row and column indices of matrix entries.

P_k is the *lower triangular Pascal matrix*, i.e.,

$$(P_k)_{ij} := \begin{cases} \binom{i-1}{j-1}, & j \le i, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The matrices K_μ, $\mu = 1, \ldots, k$, are given by

$$(K_\mu)_{ij} := \begin{cases} 1, & \text{if } i = j, \\ 1, & \text{if } i = j+1, \quad k-\mu+1 \le j \le k, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

We will make use of the following factorization, e.g., [27, Lemma 2.4],

$$P_k = \prod_{\mu=1}^{k} K_\mu. \tag{4}$$

The matrices $D_k(t)$, $D'_k(t)$, G_k, and G'_k are the diagonal matrices $D_k(t) := \operatorname{diag}(1, t, t^2, \ldots, t^k)$, $D'_k(t) := \operatorname{diag}(t^k, t^{k-1}, \ldots, t, 1)$, $G_k := \operatorname{diag}(1, 1, 2!, \ldots, k!)$, and $G'_k := \operatorname{diag}(k!, (k-1)!, \ldots, 2!, 1, 1)$. The matrix P_k can also be factorized as

$$P_k = G_k T_k G_k^{-1}, \tag{5}$$

see, e.g., [27, Theorem 2.1]. We will suppress the lower or upper indices of these matrices when their order will be clear from the context.

3. Bernstein form over the standard simplex

In this section, we present fundamental properties of the Bernstein expansion over the standard simplex that are employed throughout the paper.

Let $v_0, \ldots, v_n$ be $n+1$ points of $\mathbb{R}^n$. The ordered list $V = [v_0, \ldots, v_n]$ is called *simplex of the vertices* $v_0, \ldots, v_n$. The *realization* $|V|$ of the simplex V is the set of $\mathbb{R}^n$ defined as the convex hull of the points $v_0, \ldots, v_n$. The *diameter* of V is the length of a largest edge of $|V|$.

Throughout the paper we will assume that the points $v_0, \ldots, v_n$ are affinely independent in which case the simplex V is non-degenerate. We denote by $e_1, \ldots, e_n$ the canonical basis of $\mathbb{R}^n$ and by e_0 the zero vector in $\mathbb{R}^n$. We

will often consider the *standard simplex* $\Delta := [e_0, e_1, \ldots, e_n]$. This is no restriction since any non-degenerate simplex V in $\mathbb{R}^n$ can be mapped affinely upon Δ, see, e.g., [15], [20, Section 3]. Recall that any vector $x \in \mathbb{R}^n$ can be written as an affine combination of the vertices $v_0, \ldots, v_n$ with weights $\lambda_0(x), \ldots, \lambda_n(x)$ called *barycentric coordinates*. If $x = (x_1, \ldots, x_n) \in \Delta$, then $\lambda = (\lambda_0(x), \ldots, \lambda_n(x)) = (1 - \sum_{s=1}^{n} x_s, x_1, \ldots, x_n)$. For $i = (i_0, i_1, \ldots, i_n) \in \mathbb{N}^{n+1}$, we put $\hat{i} := (i_1, \ldots, i_n)$ and extend the notation of the monomials introduced in Section 2 to λ^i and define $|i| := i_0 + i_1 + \ldots + i_n$. If $|i| = k$, $k \in \mathbb{N}$, we further use the notation $\binom{k}{i} := \frac{k!}{i_0! \ldots i_n!}$. Let $l \in \mathbb{N}^n$ and p be an n-variate polynomial with the power representation

$$p(x) = \sum_{\hat{i}=0}^{l} a_{\hat{i}} x^{\hat{i}}. \tag{6}$$

We call

$$l' := \max\left\{ |\hat{i}| \mid \hat{i} = 0, \ldots, l \text{ with } a_{\hat{i}} \neq 0 \right\}.$$

the *total degree of* p.

The *Bernstein polynomials of degree* k *over* Δ are the polynomials $(B_i^{(k)})_{|i|=k}$, defined as

$$B_i^{(k)} := \binom{k}{i} \lambda^i. \tag{7}$$

The Bernstein polynomials of degree k over Δ take nonnegative values on Δ and sum up to 1: $1 = \sum_{|i|=k} B_i^{(k)}$. Since the Bernstein polynomials of degree k form a basis of the vector space $\mathbb{R}_k[X]$ of the polynomials of total degree at most k, see, e.g., [14, Proposition 1.6], [15], [20], we expand p for $l' \leq k$ over Δ as

$$p(x) = \sum_{|i|=k} b_i^{(k)} B_i^{(k)}, \tag{8}$$

where $b_i^{(k)}$ are called the *Bernstein coefficients of* p *of degree* k *with respect to* Δ. We arrange the Bernstein coefficients in a multidimensional array $B(\Delta) = (b_{\hat{i}}^{(k)})_{|i|=k}$, the so-called *Bernstein patch*. Using the running index

$m \in \mathbb{N}^{n+1}$ with $|m| = k$, these coefficients are represented in a compact way as

$$b_i^{(k)} = \sum_{\hat{m} \leq \hat{i}} \frac{\binom{\hat{i}}{\hat{m}}}{\binom{k}{m}} a_{\hat{m}} \tag{9}$$

with the convention that

$$a_{\hat{m}} := 0 \text{ for } \hat{m}_s > l_s \text{ for at least one } s \in \{1, \ldots, n\}. \tag{10}$$

We call the representation (8) the *Bernstein representation of* p (more precisely, the *simplicial Bernstein representation* to distinguish it from the similar representation with respect to the tensorial Bernstein basis).

The Bernstein coefficients provide lower and upper bounds for the range of p over Δ [7], [14]

$$\min_{|i|=k} b_i^{(k)} \leq p(x) \leq \max_{|i|=k} b_i^{(k)}, \text{ for all } x \in \Delta. \tag{11}$$

Equality holds in the left or right inequality of (11) if and only if the minimum or the maximum, respectively, is attained at a vertex of $B(\Delta)$, i.e., if $\hat{i} = ke_{j_0}$, $j_0 \in \{0, \ldots, n\}$, see [29, Proposition 2].

Another property of the Bernstein coefficients which follows immediately from the representation (9) is their *linearity*: Let $p = \alpha p_1 + \beta p_2$, and $\alpha, \beta \in \mathbb{R}$, where the total degree of p_1 and p_2 is less than or equal to k. Then

$$b_i^{(k)}(p) \;=\; \alpha b_i^{(k)}(p_1) + \beta b_i^{(k)}(p_2), \text{ for all } |i| = k, \tag{12}$$

where $b_i^{(k)}(p_1)$ and $b_i^{(k)}(p_2)$ are the coefficients of the Bernstein expansions of p_1 and p_2 of the degree k with respect to Δ, respectively. In the sequel, we suppress the upper index if the degree of the Bernstein expansion is clear from the context.

We can improve the enclosure (11) for the range of p over Δ by elevating the degree k of the Bernstein expansion or subdividing Δ, see [15]. Subdivision is more efficient than degree elevation since iteratively applied subdivision generates a sequence of enclosures which converges quadratically in the Hausdorff metric to the range of p over Δ, in contrast to linear convergence when degree elevation is applied, see [15], [29]. Therefore, we choose

subdivision, put $k = l'$, and suppress the upper index; the methods investigated in this paper are easily extended to $l' < k$. The two sub-simplices generated by subdivision of Δ with respect to a point $Y = (Y_1, \ldots, Y_n)$ lying on the edge between e_f and e_g, $f, g \in \{0, 1, \ldots, n\}$, are defined as follows:

$$\Delta^{[w]} := [e_0, e_1, \ldots, e_{w-1}, Y, e_{w+1}, \ldots, e_n],\ w = f, g. \tag{13}$$

The Bernstein coefficients over $\Delta^{[w]}$, $w = f, g$, denoted by $(b_i(\Delta^{[w]}))_{|i|=k}$, can be computed from the Bernstein coefficients $b_i(\Delta)$ by using the de Casteljau algorithm, see, e.g., [15, p. 14], [16], [19]. This algorithm is summarized as follows:
Let ϵ_f and ϵ_g denote the f-th and g-th, respectively, unit vectors of $\mathbb{R}^{n+1}$ and let $i^{(w)} := (i_0, \ldots, i_{w-1}, 0, i_{w+1}, \ldots, i_n)$. Then

1) Put $b_i^{[0]} := b_i(\Delta)$, $|i| = k$;

2) for $\nu = 1, \ldots, k$:

3) for $|i| = k - \nu$

$$b_i^{[\nu]} := \lambda_f(Y) b_{i+\epsilon_f}^{[\nu-1]} + (1 - \lambda_f(Y)) b_{i+\epsilon_g}^{[\nu-1]}; \tag{14}$$

4) $b_i(\Delta^{[w]}) := b_{i^{(w)}}^{[i_w]}$, $|i| = k$, $w = f, g$.

4. Matrix methods for the computation of the Bernstein coefficients over the standard simplex

In this section, we propose two efficient matrix methods for the computation of the Bernstein coefficients of a given multivariate polynomial over Δ. These methods solely involve matrix operations such as multiplication, transposition, and reshaping. They are faster than the computation according to (9). However, they require an amount of formalism. Therefore, we present here only the bivariate case and delegate details of the general multivariate case to the Appendix 1. Let the polynomial p be given by (6).

4.1. Outline of the proposed matrix methods

a) Bivariate case

Equation (9) can be rewritten as

$$b_i = \sum_{\hat{m} \le \hat{i}} \binom{\hat{i}}{\hat{m}} a'_{\hat{m}}, \tag{15}$$

where $a'_{\hat{m}} = \frac{a_{\hat{m}}}{\binom{k}{m}}$. Keeping our convention (10), we embed the coefficients of p of total degree $l' \le k$ in a matrix A as follows:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,k} \\ a_{1,0} & a_{1,1} & \dots & a_{1,k} \\ \vdots & \vdots & \dots & \vdots \\ a_{k,0} & a_{k,1} & \dots & a_{k,k} \end{bmatrix}, \tag{16}$$

where $a_{j_1,j_2} := NaN$ (not a number) for $j_1 + j_2 > l'$.

We define the $(k+1)$-by-$(k+1)$ matrix $C(\Delta) = (\gamma_{j_1,j_2})$ by

$$\gamma_{j_1,j_2} := \begin{cases} a'_{j_1-1,j_2-1} = \frac{a_{j_1-1,j_2-1}(k-j_1-j_2+2)!(j_1-1)!(j_2-1)!}{k!}, & j_1 + j_2 \le k+2, \\ a_{j_1-1,j_2-1}, & \text{otherwise.} \end{cases} \tag{17}$$

Then the Bernstein coefficients (9) of the polynomial p over Δ are obtained as the entries above and on the antidiagonal of the matrix

$$(P_k(P_k C(\Delta))^T)^T, \tag{18}$$

where P_k is the lower triangular Pascal matrix of order $k+1$, see (2).

b) Multivariate case

We extend the procedure introduced in a) for the n-variate polynomial p of total degree $l' \le k$ given in (6). The coefficients of p are arranged in an $(k+1) \times k^*$ matrix A, where $k^* := (k+1)^{n-1}$. The correspondence between the coefficients $a_{i_1,\dots,i_n}$ of p and the entry of A in row ι and column κ is as follows:

$$\begin{aligned} \iota &= i_1 + 1, \\ \kappa &= i_2 + 1 + \sum_{s=3}^{n} i_s (k+1)^{s-2}. \end{aligned}$$

The matrix A can be represented explicitly as

$$\begin{bmatrix} a_{0,0,0,\ldots,0} & a_{0,1,0,\ldots,0} \cdots & a_{0,k,0,\ldots,0} & a_{0,0,1,\ldots,0} \cdots & a_{0,k,1,\ldots,0} \cdots & a_{0,0,k,\ldots,0} \cdots & a_{0,k,k,\ldots,0} \cdots \\ a_{1,0,0,\ldots,0} & a_{1,1,0,\ldots,0} \cdots & a_{1,k,0,\ldots,0} & a_{1,0,1,\ldots,0} \cdots & a_{1,k,1,\ldots,0} \cdots & a_{1,0,k,\ldots,0} \cdots & a_{1,k,k,\ldots,0} \cdots \\ \vdots & \vdots \quad \ldots & \vdots & \vdots \quad \ldots & \vdots \quad \ldots & \vdots \quad \ldots & \vdots \quad \ldots \\ a_{k,0,0,\ldots,0} & a_{k,1,0,\ldots,0} \cdots & a_{k,k,0,\ldots,0} & a_{k,0,1,\ldots,0} \cdots & a_{k,k,1,\ldots,0} \cdots & a_{k,0,k,\ldots,0} \cdots & a_{k,k,k,\ldots,0} \cdots \end{bmatrix}$$

$$\begin{bmatrix} \cdots & a_{0,0,k,\ldots,k} & a_{0,1,k,\ldots,k} & \cdots & a_{0,k,k,\ldots,k} \\ \cdots & a_{1,0,k,\ldots,k} & a_{1,1,k,\ldots,k} & \cdots & a_{1,k,k,\ldots,k} \\ \cdots & \vdots & \vdots & \cdots & \vdots \\ \cdots & a_{k,0,k,\ldots,k} & a_{k,1,k,\ldots,k} & \cdots & a_{k,k,k,\ldots,k} \end{bmatrix}, \tag{19}$$

where $a_j := NaN$ for $|j| > l'$.
For a graphical illustration in the trivariate case, see Figures 1 and 2 in [22]. The matrix $C(\Delta)$ is obtained from A by dividing a_i by $\binom{k}{i}$ for $|i| \leq l'$ and replacing all entries that correspond to a_i with $|i| > k$ by NaN. In the sequel, the superscript c denotes the *cyclic ordering* of the sequence of the indices, i.e., the order of the indices of the entries of the array under consideration is changed cyclically. This means that the index in the first position is replaced by the index in the second one, the index in the second position by the one in the third, ... , the index in the n-th position by the one in the first position, so that after n such steps the sequence of the indices is again in its initial order, see Figure 3 in [22] as an illustration in the trivariate case. Note that in the bivariate case the cyclic ordering is just the usual matrix transposition. The Bernstein coefficients (9) of the polynomial p over Δ can be obtained from the non-NaN entries of the matrix

$$(\underbrace{P_k(\cdots(P_k(P_k}_{n \text{ times}} C(\Delta))^c)^c \cdots)^c)^c. \tag{20}$$

We put $C_0(\Delta) := C(\Delta)$ and define for $t = 1, \ldots, n$

$$C_t := (P_k C_{t-1})^c. \tag{21}$$

We obtain the Bernstein patch $B(\Delta)$ from the non-NaN entries of the matrix C_n.

For larger n, the matrix in (20) may contain a huge number of NaN entries. In order to avoid operations with such entries we reduce the computation of (20) to the calculation of products of the Pascal matrices P_{k-z} and matrices $B_{t-1}^{(z)}$, see (51) in the Appendix 1, where also a matrix method for the computation of these products based on the factorization (4) is given.

Method 1: In this method we sequentially perform the multiplication in (51) by using the factorization (4), i.e., we firstly multiply K_{k-z} and $B_{t-1}^{(z)}$, multiply the resulting matrix by K_{k-z-1}, and so on. The number of arithmetic operations that are required by this method is $n\binom{n+k}{n+1}$, see Appendix 1.

Method 2: The factorial $j_s!$ appearing in $\binom{k}{j}$ and on the diagonal of G_k^{-1} cancel out, $s = 1, \ldots, n$. Therefore, if we multiply $a_{i_1,\ldots,i_n}$ by $\frac{i_0!}{k!}$ and name the resulting matrix $B_{t-1}^{\prime(z)}$ then

$$D_{t-1}^{(z)} = GTB_{t-1}^{\prime(z)},\ t = 1, \ldots, n, \tag{22}$$

yields the matrix in (51).

Method 2 employs factorization (5) and relies on the fact that the computation of a matrix-vector multiplication with the matrix T can be carried out by using the FFT [27, Theorem 1.6], see [3, Chapter 13]. At a first glance, it seems that by the use of method 2 the number of operations required for the computation of $B(\Delta)$ could significantly be reduced. However, to apply the FFT the order of the matrices being involved has possibly to be increased. Specifically, the multiplication of a vector of length t by the matrices $K_\mu, \mu = 1, \ldots, n$, requires $\frac{t(t+1)}{2}$ arithmetic operations, whereas the application of the FFT needs $5m \log_2 m + t$ operations [3, Section 3.1], where m is the smallest natural number greater than or equal to $2t-2$ such that $m = 2^\sigma$ for some σ [3, Chapter 13]. Thus, method 2 is superior to method 1 when $\frac{t(t+1)}{2} > 5\sigma 2^\sigma + t$. This inequality is fulfilled only for $t \in [216, 257]$ and for all $t \geq 321$. Since the number of the rows of the matrices $B_{t-1}^{(z)}$ is decreasing from $k+1$ to 1 when z is increased, see Appendix 1, we propose a mixed method in which method 1 or method 2 is chosen depending on which method is preferable for the actual number of rows.
In Table 1 we present the results for method 1 on the test problems listed in the Appendix 2. According to the last paragraph, method 1 outperforms

method 2 in all these cases. The timings in Table 1 are obtained on a laptop with Intel(R) Core(TM) i5-5200U CPU@ 3.30 GHz 2.20 GHz, 16.00 GB RAM. The computations are done in Visual Studio 2010.

Table 1: Time (in ms) required for computing the Bernstein patch over Δ by using method 1

Test case	n	k	method 1
Booth	2	2	0.11
Ler1	2	2	0.11
Himmelblau	2	4	0.31
Rosenbrock	2	4	0.31
Ler2	2	4	0.31
Ler3	2	4	0.31
Camel 2	2	6	0.52
Ler4	2	6	0.52
Ler5	2	8	0.89
Trid 3	3	2	0.24
Schwefel	3	4	0.79
L. V. 4	4	3	1.31
Cap 4	4	4	4.93
Wrig 5	5	2	1.5
Cyc 5	5	4	13.31
Reim 5	5	6	62.81
Mag 6	6	2	4.79
But 6	6	3	22.23

4.2. Bernstein coefficients on faces

In the tensorial case, the Bernstein coefficients on an r-dimensional face of an n-dimensional box, $r = 1, \ldots, n-1$, are identical with the coefficients lying on the respective face of the Bernstein patch, see [8, Lemma 2]. In the simplicial case, we have a parallel result. Let p be the n-variate polynomial given by (6). Then the Bernstein coefficients of p over an r-dimensional face of Δ, where $1 \leq r \leq n-1$, are the same as the Bernstein coefficients that are located at the corresponding r-dimensional face of the Bernstein patch of p over Δ. To show this it suffices to consider the case $r = n-1$. Let p^*

be the polynomial of $n-1$ variables given by

$$p(x_1,\ldots,x_{h-1},0,x_{h+1},\ldots,x_n)=\sum_{\substack{\hat{i}=0,\\ i_h=0}}^{l} a_{\hat{i}}x^{\hat{i}}.$$

Then the Bernstein coefficients of p^* on $|[e_0,e_1,\ldots,e_{h-1},e_{h+1},\ldots,e_n]|$ are

$$\sum_{\substack{\hat{m}\leq\hat{i},\\ m_h=0}} \frac{\binom{\hat{i}}{\hat{m}}}{\binom{k}{m}} a_{\hat{m}} \tag{23}$$

which coincide with the Bernstein coefficients $b_{i_0,\ldots,i_{h-1},0,i_{h+1},\ldots,i_n}$ of p over Δ. Also, if $\sum_{s=1}^{n} x_s = 1$, the only monomial terms in the Bernstein representation which are not identical zero are those containing the term $(1-\sum_{s=1}^{n-1} x_s)$ with a zero power, i.e., for which $|\hat{i}| = k$ holds. Therefore, we obtain

$$p(x_1,\ldots,x_{n-1},1-\sum_{s=1}^{n-1}x_s)=\sum_{|\hat{i}|=k} b_{\hat{i}}\binom{k}{\hat{i}}x_1^{i_1}\cdots x_{n-1}^{i_n-1}(1-\sum_{s=1}^{n-1}x_s)^{k-i_1-\ldots-i_{n-1}}.$$

By the uniqueness of the Bernstein representation over the face $\sum_{s=1}^{n} x_s = 1$, it follows that the Bernstein coefficients of p over this face are just the Bernstein coefficients lying in the Bernstein patch of p over Δ on the face given by $|\hat{i}| = k$.

Now let $V = [v_0,\ldots,v_n]$ be an arbitrary non-degenerate simplex of $\mathbb{R}^n$. Consider the affine transformation which maps v_s to e_s, $s = 0,\ldots,n$. It maps V onto Δ and any face of V onto the respective face of Δ. Since the Bernstein coefficients are invariant under an affine transformation, see [14, p. 24], [15, p. 19], the above result on the Bernstein coefficients on a face of Δ extends to any non-degenerate simplex of $\mathbb{R}^n$.

5. Simplicial subdivision

Let p be the n-variate polynomial given in (6) with Bernstein coefficients b_i over the standard simplex Δ arranged in the matrix $\mathcal{B}(\Delta)$ as in (19). In this section, we consider the subdivision of Δ into two subsimplices with respect to a point Y, which lies on an edge between any two vertices of Δ. Without loss of generality we assume that Y lies on the edge between e_0 and

e_1. This implies that $\lambda_2(Y) = \cdots = \lambda_n(Y) = 0$. In the case that Y lies between another pair of vertices of Δ we firstly arrange the Bernstein matrix $\mathcal{B}(\Delta)$ in such away that this pair plays the role of (e_0, e_1). Subdivision of Δ with respect to Y produces two subsimplices $\Delta^{[0]}$ and $\Delta^{[1]}$ defined as

$$\begin{aligned} \Delta^{[0]} &:= [Y, e_1, \ldots, e_n], \\ \Delta^{[1]} &:= [e_0, Y, \ldots, e_n], \end{aligned} \tag{24}$$

such that $\Delta = \Delta^{[0]} \cup \Delta^{[1]}$. We arrange their Bernstein coefficients $b_{\hat{i}}(\Delta^{[w]}),\ w = 0, 1$, likewise in matrices $\mathcal{B}(\Delta^{[0]})$ and $\mathcal{B}(\Delta^{[1]})$ of the form (19). For simplicity, we mean by the Bernstein coefficients of p over $\Delta^{[w]}$ the coefficients $b_{\hat{i}}(\Delta^{[w]})$.

We perform the de Casteljau algorithm, see Section 3, with the matrix $\mathcal{B}(\Delta)$ instead of the Bernstein patch $B(\Delta)$ and get then $k+1$ matrices $\mathcal{B}^{[\nu]}(\Delta)$. In the ν-th step, $\nu \geq 1$, $\mathcal{B}^{[\nu]}(\Delta)$ is obtained as follows: The first ν rows are identical with the first ν rows in the $(\nu-1)$-th step and the remaining rows are obtained as a convex combination of two consecutive rows. In matrix language, these matrices are obtained from the following matrices, see Example 5.1 below for an illustration. Let the columns of $F_z(\Delta) \in \mathbb{R}^{k+1, \psi_{n-1}(z)}$ be the columns of $\mathcal{B}(\Delta)$ such that each column has exactly z NaN entries and let $\psi_{n-1}(z)$ be the number of such columns, $z = 0, \ldots, k$. Then we form the matrices $F_z^{(1)}(\Delta) \in \mathbb{R}^{k-z+1, \psi_{n-1}(z)}$ from $F_z(\Delta)$ by deleting the last rows which contain exactly z NaN entries (if any), $z = 0, \ldots, k$. Since Y is a point on the edge between e_0 and e_1, $\lambda_0(Y)$ and $\lambda_1(Y)$ are related by $\lambda_0(Y) = 1 - \lambda_1(Y)$. In the sequel, we write λ_1 for $\lambda_1(Y)$. For $\mu = 1, \ldots, k-z,\ z = 0, \ldots, k$, we define the matrices L^{μ}_{k-z} and $R^{\mu}_{k-z} \in \mathbb{R}^{k-z+1, k-z+1}$ as follows (δ denotes the Kronecker delta)

$$(L^{\mu}_{k-z})_{ij} := \left\{ \begin{array}{ll} \delta_{i,j}, & \text{if } i = 1, \ldots, k-z-\mu+1, \\ 1-\lambda_1, & \text{if } j = i-1, \\ \lambda_1, & \text{if } j = i, \\ 0, & \text{otherwise,} \end{array} \right\} \text{for } i = k-z-\mu+2, \ldots, k-z+1, \tag{25}$$

and

$$(R^{\mu}_{k-z})_{ij} := \left\{ \begin{array}{ll} 1-\lambda_1, & \text{if } j = i, \\ \lambda_1, & \text{if } j = i+1, \\ 0, & \text{otherwise,} \\ \delta_{i,j}, & \text{if } i = \mu+1, \ldots, k-z+1. \end{array} \right\} \text{for } i = 1, \ldots, \mu, \tag{26}$$

Let $S_{k-z} \in \mathbb{R}^{k-z+1,k-z+1}$ be the permutation matrix defined by

$$(S_{k-z})_{ij} := \begin{cases} 1, & \text{if } i = k-z-j+2, \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

The matrices $F_z^{(2)}(\Delta) \in \mathbb{R}^{k-z+1,\psi_{n-1}(z)}$ are formed from $F_z^{(1)}(\Delta)$ as follows:

$$F_z^{(2)}(\Delta) := \prod_{\mu=1}^{k-z} L_{k-z}^{\mu} F_z^{(1)}(\Delta), \tag{28}$$

where $z = 0, \ldots, k$.

The matrices $F_z^{(3)}(\Delta) \in \mathbb{R}^{k+1,\psi_{n-1}(z)}$ are obtained from $F_z^{(2)}(\Delta)$, see (28), by appending at their bottom row z rows containing only NaN entries. Then we replace the columns in $\mathcal{B}(\Delta)$ by the corresponding resulting columns from $F_z^{(3)}(\Delta)$. It is not hard to see that we get in this way $\mathcal{B}(\Delta^{[1]})$ and the Bernstein coefficients over $\Delta^{[1]}$ are the non-NaN entries of $\mathcal{B}(\Delta^{[1]})$.

We define $L_{k-z}^{\dagger} \in \mathbb{R}^{k-z+1,k-z+1}$ by

$$(L_{k-z}^{\dagger})_{ij} := \begin{cases} \binom{i-1}{j-1} \lambda_1^{j-1} (1-\lambda_1)^{(i-1)-(j-1)}, & i \geq j, \\ 0, & \text{otherwise.} \end{cases} \tag{29}$$

Then, the factorization

$$L_{k-z}^{\dagger} = \prod_{\mu=1}^{k-z} L_{k-z}^{\mu}, \ z = 0, \ldots, k, \tag{30}$$

holds [1, Proposition 3]. It is easy to see that

$$L_{k-z}^{\dagger} = D_{k-z}(1-\lambda_1) P_{k-z} D_{k-z} \left(\frac{\lambda_1}{1-\lambda_1} \right). \tag{31}$$

Then by using (31), (28) becomes

$$F_z^{(2)}(\Delta) = D_{k-z}(1-\lambda_1) P_{k-z} D_{k-z} \left(\frac{\lambda_1}{1-\lambda_1} \right) F_z^{(1)}(\Delta). \tag{32}$$

After substituting (5) in (32), we have

$$F_z^{(2)}(\Delta) = D_{k-z}^{(2)} T_{k-z} D_{k-z}^{(1)} F_z^{(1)}(\Delta), \tag{33}$$

where $D^{(2)}_{k-z} := D_{k-z}(1-\lambda_1)G_{k-z}$ and $D^{(1)}_{k-z} := G^{-1}_{k-z}D_{k-z}\left(\frac{\lambda_1}{1-\lambda_1}\right)$.

To get the Bernstein coefficients matrix over $\Delta^{[0]}$ we first replace $F^{(1)}_z(\Delta)$ by $F'^{(1)}_z(\Delta)$ defined by

$$F'^{(1)}_z(\Delta) := S_{k-z}F^{(1)}_z(\Delta), \tag{34}$$

i.e., we permute the rows of $F^{(1)}_z(\Delta)$. We replace in L^{μ}_{k-z} λ_1 by $1-\lambda_1$ to obtain L'^{μ}_{k-z} and define $F'^{(2)}_z(\Delta) \in \mathbb{R}^{k-z+1,\psi_{n-1}(z)}$ by

$$F'^{(2)}_z(\Delta) := \prod_{\mu}^{k-z} L'^{\mu}_{k-z}F'^{(1)}_z(\Delta). \tag{35}$$

Then we form the matrices $F'^{(3)}_z(\Delta) \in \mathbb{R}^{k+1,\psi_{n-1}(z)}$ from $F'^{(2)}_z(\Delta)$ by permutation of the rows of $F'^{(2)}_z(\Delta)$ by the matrix S_{k-z}. Note that this procedure corresponds to the relation $R^{\mu}_{k-z} = S_{k-z}L'^{\mu}_{k-z}S_{k-z}$. In this way, (35) can be written as

$$F'^{(2)}_z(\Delta) = \prod_{\mu}^{k-z} R^{\mu}_{k-z}F^{(1)}_z(\Delta). \tag{36}$$

Then we append z rows of NaN entries to the bottom of $S_{k-z}F'^{(2)}_z(\Delta)$ or $F'^{(2)}_z(\Delta)$ if we use the matrices L'^{μ}_{k-z} and R^{μ}_{k-z}, respectively. Then we replace the columns in $\mathcal{B}(\Delta)$ by the corresponding resulting columns from $F'^{(3)}_z(\Delta)$. It is not hard to see that we get in this way $\mathcal{B}(\Delta^{[0]})$ and the Bernstein coefficients over $\Delta^{[0]}$ are the non-NaN entries of $\mathcal{B}(\Delta^{[0]})$.

We define $R^{\dagger}_{k-z} \in \mathbb{R}^{k-z+1,k-z+1}$ by

$$(R^{\dagger}_{k-z})_{ij} := \begin{cases} \binom{k-z-i+1}{j-i}\lambda_1^{j-i}(1-\lambda_1)^{k-z-j+1}, & j \geq i, \\ 0, & \text{otherwise.} \end{cases} \tag{37}$$

Then we get

$$R^{\dagger}_{k-z} = \prod_{\mu=1}^{k-z} R^{\mu}_{k-z}. \tag{38}$$

The matrix $R^{\dagger}_{k-z}$ can also be factorized as

$$R^{\dagger}_{k-z} = (1-\lambda_1)^{k-z}D_{k-z}(\frac{1}{\lambda_1})P'_{k-z}D_{k-z}(\frac{\lambda_1}{1-\lambda_1}), \tag{39}$$

where the matrix P'_{k-z} is defined as

$$(P'_{k-z})_{ij} := \begin{cases} \binom{k-z-i+1}{j-i}, & j \geq i, \\ 0, & \text{otherwise.} \end{cases} \tag{40}$$

By using (27) and the symmetry of the binomial coefficients, we can easily see that $P'_{k-z} = S_{k-z} P_{k-z} S_{k-z}$. By (4), we factorize P'_{k-z} as follows

$$P'_{k-z} = \prod_{\mu=1}^{k-z} K'_\mu \tag{41}$$

with the matrices $K'_\mu := S_{k-z} K_\mu S_{k-z}$, $\mu = 1, \ldots, k-z, z = 0, \ldots, k$. After substituting (39) in (35) by using (38), we get

$$F_z^{\prime(2)}(\Delta) = (1-\lambda_1)^{k-z} D_{k-z}(\frac{1}{\lambda_1}) P'_{k-z} D_{k-z}(\frac{\lambda_1}{1-\lambda_1}) F_z^{(1)}(\Delta). \tag{42}$$

It is easy to see that

$$P'_{k-z} = G'_{k-z} T_{k-z}^T G_{k-z}^{\prime -1}. \tag{43}$$

After substituting (43) in (39) and using (38), formula (35) can be written as

$$F_z^{\prime(2)}(\Delta) = D_{k-z}^{\prime(2)} T_{k-z}^T D_{k-z}^{\prime(1)} F_z^{(1)}(\Delta), \tag{44}$$

where $D_{k-z}^{\prime(2)} := (1-\lambda_1)^{k-z} D_{k-z}(\frac{1}{\lambda_1}) G'_{k-z}$ and $D_{k-z}^{\prime(1)} := G_{k-z}^{\prime -1} D_{k-z}(\frac{\lambda_1}{1-\lambda_1})$.

Now we are going to derive the amount of work needed by the presented methods for the simplicial subdivision. The number of arithmetic operations required for the multiplication of $F_z^{(1)}(\Delta)$ and $F_z^{\prime(1)}(\Delta)$ by the two diagonal matrices and the matrix P_{k-z} and P'_{k-z} in (32) and (42), (neglecting the multiplication by $(1-\lambda_1)^{k-z}$), respectively, is as follows:

For $z = 0, \ldots, k$, the matrix $F_z^{(1)}(\Delta)$ has $\psi_{n-1}(z) = \binom{n-2+z}{n-2}$ columns. The number of additions and multiplications for each of these columns is $\binom{k+1-z}{2}$ and $2(k-z)$, respectively. Summing over all columns and using [9, p. 222, 4.1.6.a] results in

$$\text{additions: } \sum_{z=0}^{k} \binom{k+1-z}{2}\binom{n-2+z}{n-2} = \binom{n+k}{n+1},$$
$$\text{multiplications: } 2\sum_{z=0}^{k}(k-z)\binom{n-2+z}{n-2} = 2\binom{n+k-1}{n}.$$

For the total number of arithmetic operations required to obtain the Bernstein patch over $\Delta^{[1]}$ by using (32), we have to add to the given number of additions and multiplications the amount of work for the computation of the two diagonal matrices, i.e., 1 addition and $2k-1$ multiplications/divisions. The number of arithmetic operations required to obtain the Bernstein patch over $\Delta^{[0]}$ by using (42) is presented in Table 2.

Table 2: Number of arithmetic operations required to obtain the Bernstein patch over $\Delta^{[0]}$ by using (42)

Calculation of	number of additions	number of multiplications/divisions
$(1-\lambda_1)^k D_k(\frac{1}{\lambda_1}), D_k(\frac{\lambda_1}{1-\lambda_1})$	1	$3k+2$
$D_{k-z}(\frac{\lambda_1}{1-\lambda_1})F_z^{\prime(1)}(\Delta)$, $(1-\lambda_1)^{k-z}D_{k-z}(\frac{1}{\lambda_1})(\ldots)$, $z=0,\ldots,k$	0	$\binom{n+k-1}{n} + \binom{n+k}{n}$
$P'_{k-z}(\ldots),\ z=0,\ldots,k$	$\binom{n+k}{n+1}$	0
total	$\binom{n+k}{n+1}+1$	$\binom{n+k-1}{n} + \binom{n+k}{n} + 3k + 2$

In Table 3 we assume that we want to compute the Bernstein patch over $\Delta^{[0]}$ as well as over $\Delta^{[1]}$ by using (32) and (42) together. Then we can reduce the amount of the arithmetic operations by noting that both formulae share common factors, e.g., $D_{k-z}\left(\frac{\lambda_1}{1-\lambda_1}\right)F_z^{(1)}(\Delta)$, and that the last row in $F_z^{(2)}(\Delta)$ and the first row of $F_z^{\prime(2)}(\Delta)$ are identical so that we can shorten the matrices $D_{k-z}(\frac{1}{\lambda_1})$, $D_{k-z}(\frac{\lambda_1}{1-\lambda_1})$, and P'_{k-z} in (42) by deletion of their first rows and columns.

Table 3: Number of arithmetic operations required to obtain the Bernstein patch over $\Delta^{[1]}$ and $\Delta^{[0]}$ by using (42) and (32) together

Calculation of	number of additions	number of multiplications/divisions
$D_k(1-\lambda_1), D_k(\frac{\lambda_1}{1-\lambda_1})$, $(1-\lambda_1)^k D_k(\frac{1}{\lambda_1})$	1	$4k+1$
$D_{k-z}(\frac{\lambda_1}{1-\lambda_1})F_z^{(1)}(\Delta)$, $D_{k-z}(1-\lambda_1)(\ldots)$, $z=0,\ldots,k$	0	$2\binom{n+k-1}{n}$
$P_{k-z}(\ldots),\ z=0,\ldots,k$	$\binom{n+k}{n+1}$	0
$P'_{k-z}(\ldots),\ z=0,\ldots,k$	$\binom{n+k-1}{n+1}$	0
$(1-\lambda_1)^{k-z} D_{k-z}(\frac{1}{\lambda_1})(\ldots)$, $z=0,\ldots,k$	0	$\binom{n+k-1}{n}$
total	$\binom{n+k}{n+1} + \binom{n+k-1}{n+1} + 1$	$3\binom{n+k-1}{n} + 4k + 1$

Table 4: Comparison between the de Casteljau algorithm and the new methods

Calculation of	number of additions	number of multiplications/divisions
de Casteljau algorithm	$\binom{n+k}{n+1} + 1$	$2\binom{n+k}{n+1}$
(32)	$\binom{n+k}{n+1} + 1$	$2\binom{n+k-1}{n} + 2k - 1$
(42)	$\binom{n+k}{n+1} + 1$	$\binom{n+k-1}{n} + \binom{n+k}{n} + 3k + 2$
(32) and (42) together	$\binom{n+k}{n+1} + \binom{n+k-1}{n+1} + 1$	$3\binom{n+k-1}{n} + 4k + 1$

In the following comparison of the amount of arithmetic operations for the several methods, see Table 4, we compare only the terms which are depending on k or n and neglect the trivial multiplications. The application of the de Casteljau algorithm, see Section 3, requires for the computation of both $\mathcal{B}(\Delta^{[1]})$ and $\mathcal{B}(\Delta^{[0]})$ $\binom{n+k}{n+1}$ additions and $2\binom{n+k}{n+1}$ multiplications, see [20]. We need the same amount in (28) and (35), respectively, if we carry out the

matrix multiplication from the right to the left. The amount of arithmetic operations for each of the computations of $\mathcal{B}(\Delta^{[1]})$ and $\mathcal{B}(\Delta^{[0]})$ by using the factorizations (4), (41) consists of $\binom{n+k}{n+1}$ additions and $2\binom{n+k-1}{n}+2k$ and $\binom{n+k-1}{n}+\binom{n+k}{n}+3k$ multiplications / divisions for (28) and (35), respectively. If only one of both patches is needed, (32) and (42) (in case $k=n$, if $n>4$) are superior to the de Casteljau algorithm for $k>2, n>1$ and $k>6, n>3$, respectively. If both patches are needed, the amount of arithmetic operations for both formulae consists then of $\binom{n+k}{n+1}+\binom{n+k-1}{n+1}$ additions and $3\binom{n+k-1}{n}+4k$ multiplications/ divisions. Therefore, for $k<n+2, n\leq 3$ the use of the de Casteljau algorithm is preferable, e.g., when the degree of at least two variables is at least two and the other degrees are at least 1, the use of (32) and (42) is superior. If instead (33) and (44) are employed, the amount of arithmetic operations can be reduced by using the FFT, see Subsection 4.1. In Table 5, we compare the different methods on the basis of our test cases, where we have chosen $\lambda_1 = 0.5$. In all cases, the de Casteljau algorithm is inferior to the use of (32) as well of (42); in only three cases the de Casteljau algorithm outperforms the joint use of (32) and (42) although the superiority is very small. This shows that the number of the arithmetic operations should not be the sole basis for the choice of the procedure.

Table 5: Time (in ms) required for the computation of the Bernstein patchs $\mathcal{B}(\Delta^{[1]})$ and $\mathcal{B}(\Delta^{[0]})$ given $\mathcal{B}(\Delta)$ by using the de Casteljau agorithm, (32), and (42)

Test case	n	k	de Casteljau algorithm	(32)	(42)	(32) and (42)
Booth	2	2	0.12	0.09	0.10	0.14
Ler1	2	2	0.12	0.09	0.10	0.14
Himmelblau	2	4	0.71	0.25	0.25	0.41
Rosenbrock	2	4	0.69	0.25	0.25	0.41
Ler2	2	4	0.66	0.24	0.25	0.41
Ler3	2	4	0.68	0.25	0.25	0.41
Camel 2	2	6	0.86	0.42	0.44	0.61
Ler4	2	6	0.84	0.43	0.42	0.60
Ler5	2	8	1.41	0.50	0.50	0.82
Trid 3	3	2	0.20	0.11	0.11	0.21
Schwefel	3	4	1.10	0.74	0.74	1.27
L. V. 4	4	3	1.82	0.39	0.40	0.52
Cap 4	4	4	11.85	0.71	0.75	1.33
Wrig 5	5	2	2.26	0.25	0.26	0.31
Cyc 5	5	4	26.15	1.50	1.51	2.58
Reim 5	5	6	152.96	5.50	5.66	6.19
Mag 6	6	2	4.19	0.61	0.61	1.00
But 6	6	3	32.25	2.35	2.37	3.75

Example 5.1. *Let $n = 2$ and $k = 4$ such that $\mathcal{B}(\Delta)$ has the following form*

$$\mathcal{B}(\Delta) = \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} & b_{04} \\ b_{10} & b_{11} & b_{12} & b_{13} & NaN \\ b_{20} & b_{21} & b_{22} & NaN & NaN \\ b_{30} & b_{31} & NaN & NaN & NaN \\ b_{40} & NaN & NaN & NaN & NaN \end{bmatrix}.$$

We want to illustrate how to calculate $\mathcal{B}(\Delta^{[1]})$ *by using* (32). *Starting from*

$$F_0(\Delta) = \begin{bmatrix} b_{00} \\ b_{10} \\ b_{20} \\ b_{30} \\ b_{40} \end{bmatrix}, F_1(\Delta) = \begin{bmatrix} b_{01} \\ b_{11} \\ b_{21} \\ b_{31} \\ NaN \end{bmatrix}, F_2(\Delta) = \begin{bmatrix} b_{02} \\ b_{12} \\ b_{22} \\ NaN \\ NaN \end{bmatrix}, F_3(\Delta) = \begin{bmatrix} b_{03} \\ b_{13} \\ NaN \\ NaN \\ NaN \end{bmatrix}, F_4(\Delta) = \begin{bmatrix} b_{04} \\ NaN \\ NaN \\ NaN \\ NaN \end{bmatrix},$$

and

$$F_0^{(1)}(\Delta) = \begin{bmatrix} b_{00} \\ b_{10} \\ b_{20} \\ b_{30} \\ b_{40} \end{bmatrix}, F_1^{(1)}(\Delta) = \begin{bmatrix} b_{01} \\ b_{11} \\ b_{21} \\ b_{31} \end{bmatrix}, F_2^{(1)}(\Delta) = \begin{bmatrix} b_{02} \\ b_{12} \\ b_{22} \end{bmatrix}, F_3^{(1)}(\Delta) = \begin{bmatrix} b_{03} \\ b_{13} \end{bmatrix}, F_4^{(1)}(\Delta) = \begin{bmatrix} b_{04} \end{bmatrix}$$

we use (32) *with the factorization of* P_{k-z} *to calculate* $F_z^{(2)}$, $z = 0, 1, 2, 3, 4$. *Then we add* z *rows with NaN entries to the matrices* $F_z^{(2)}$ *at their bottom. Finally, we arrange the columns in the same way that we have extracted them from* $\mathcal{B}(\Delta)$ *to get* $\mathcal{B}(\Delta^{[1]})$.

6. Evaluation of multivariate polynomials

In this section, we present matrix methods for the evaluation of a multivariate polynomial in both the power and the Bernstein representation.

6.1. Power representation

In [24], difficulties to implement variants of the multivariate Horner scheme, e.g., [20], are reported. In this subsection we present a matrical description for the evaluation of a multivariate polynomial by the Horner scheme which can very easily be implemented. Let p be an n-variate polynomial given in power form as

$$p(x) = \sum_{\hat{i}=0}^{l} a_{\hat{i}} \prod_{s=1}^{n} x_s^{\hat{i}_s} \tag{45}$$

with its coefficients arranged as in (19) in a matrix A. For $s = 1, \ldots, n$ and $\mu = 1, \ldots, l_s$, we define the matrices $H_\mu(\gamma) \in \mathbb{R}^{\mu,\mu+1}$ by

$$(H_\mu(\gamma))_{ij} := \begin{cases} 1, & i = j, \\ \gamma, & i = \mu,\ j = \mu + 1, \\ 0, & \text{otherwise.} \end{cases} \tag{46}$$

Then (45) can be written as

$$p(x) = \bigcirc_{s=1}^{n} (H_1(x_s) \cdots H_{l_s-1}(x_s) H_{l_s}(x_s))^c A, \tag{47}$$

where $\bigcirc$ denotes the composition of the induced linear mappings. After multiplication by $H_{l_{s-1}}(x_{s-1})$, we obtain a row vector w of length $\prod_{t=s}^{n}(l_t+1)$ which we order cyclically, i.e., we build from this vector a matrix C of order $(l_s + 1) \times \prod_{t=s+1}^{n}(l_t + 1)$ by

$$(C)_{ij} = w_{i+(j-1)(l_s+1)}. \tag{48}$$

Without loss of generality we assume that $l_s = \kappa,\ s = 1, \ldots, n$. The number of arithmetic operations required for the evaluation of $p(x)$ by using (47) is $(\kappa + 1)^n - 1$ additions and the same amount for multiplications which is identical to the amount of operations of the Horner scheme [17].

6.2. Bernstein representation

In this section, we present a matricial algorithm for the evaluation of a multivariate polynomial over the standard simplex.

Let p be an n-variate polynomial as given in (6). Its Bernstein representation (8) can be written as ($l' \leq k$)

$$p(x) = x_0^k \sum_{|i|=k} b_i \binom{k}{i} \prod_{s=1}^{n} \left(\frac{x_s}{x_0}\right)^{i_s}, \tag{49}$$

where $x_0 = 1 - \sum_{s=1}^{n} x_s$. We divide in the Bernstein patch each entry b_i by $\frac{k!}{i_0!}$, replace NaN entries, see Subsection 4.1, by zeros, and arrange the resulting patch in a matrix $\mathcal{B}'(\Delta)$ as in (19). Application of the representation (47) to (49) with A replaced by the matrix $\mathcal{B}'(\Delta)$ yields

$$p(x) = x_0^k \bigcirc_{s=1}^{n} (H_1(\theta_{s,1}) \ldots H_{k-1}(\theta_{s,k-1}) H_k(\theta_{s,k}))^c \mathcal{B}'(\Delta), \tag{50}$$

where $\theta_{s,\mu} := \frac{1}{\mu}\frac{x_s}{x_0}$ for $\mu = 1, \ldots, k,\ s = 1, \ldots, n$.

The number of arithmetic operations for the evaluation of a multivariate polynomial in Bernstein representation according to (50) is $\binom{n+k}{k}$ additions and $2\binom{n+k}{k}$ multiplications. If we assume that the matrix $\mathcal{B}'$ is precomputed, then we need $\binom{n+k}{k}$ additions and $\binom{n+k}{k}$ multiplications which are identical to the number of arithmetic operations of the de Boor evaluation algorithm [2], see also [4, Theorem 3.1].

Appendix 1: Some technical details of method 1

We start in the general multivariate case with the matrix $C(\Delta)$ formed in Subsection 4.1. For $t = 1, \ldots, n$, we collect in a $(k+1) \times \psi_{t-1}(z)$ matrix $C_{t-1}^{(z)}$ in the same order as they appear in C_{t-1} all its columns (except the columns whose entries are all NaN) which contain exactly z NaN entries, where $\psi_{t-1}(z)$ is the number of such columns in $C_{t-1},\ z = 0, \ldots, k$. Then we delete the last z rows of $C_{t-1}^{(z)}$ and name the resulting matrix $B_{t-1}^{(z)}$. For $z = 0, \ldots, k$, we put

$$D_{t-1}^{(z)} := P_{k-z} B_{t-1}^{(z)},\ t = 1, \ldots, n, \tag{51}$$

wherein we use the factorization (4) of P_{k-z}. Then we append to the bottom row of $D_{t-1}^{(z)}$ z rows containing only NaN entries and replace each column of C_{t-1} (except the columns whose entries are all NaN) by the corresponding column of the enlarged matrix $D_{t-1}^{(z)}$. Finally, we apply the cyclic ordering. It is not hard to see that the resulting matrix equals C_t.

We obtain $\Lambda(\Delta)$ from A by $\binom{n+k}{n}$ divisions by the binomial coefficients $\binom{k}{j}$ which we consider as precomputed. In Table 6 we present the number of arithmetic operations for the computation of $D_{t-1}^{(z)}$, $t = 1, \ldots, n,\ z = 0, \ldots, k$, by using method 1.

Table 6: Number of arithmetic operations required for the computation of (51) for $t-1$ by using method 1, $t = 1, \ldots, n$

Calculation of	number of columns	number of additions for each column
$D_{t-1}^{(0)}$	$\psi_{t-1}(0)$	$\binom{k+1-0}{2}$
$D_{t-1}^{(1)}$	$\psi_{t-1}(1)$	$\binom{k+1-1}{2}$
$\vdots$	$\vdots$	$\vdots$
$D_{t-1}^{(k-1)}$	$\psi_{t-1}(k)$	$\binom{k+1-(k-1)}{2}$

In total, we obtain the following number of additions, see the calculations in Subsection 4.1,

$$\begin{aligned}\sum_{t=1}^{n}\sum_{z=0}^{k-1}\psi_{t-1}(z)\binom{k+1-z}{2} &= \sum_{t=1}^{n}\sum_{z=0}^{k-1}\binom{n-2+z}{n-2}\binom{k+1-z}{2} \\ &= n\sum_{z=0}^{k-1}\binom{n-2+z}{n-2}\binom{k+1-z}{2} = n\binom{n+k}{n+1}.\end{aligned}$$

We note that $\psi_{n-1}(z) = \binom{n-2+z}{n-2}$. For fixed n, the complexity of method 1 is $O(\frac{n}{(n+1)!}k^{n+1})$. If n is also to allow to vary, the complexity is $O(\frac{n}{n+1}k^{n+1})$.

Appendix 2: Description of the test problems

In the following, we list the abbreviated and full names, the polynomials and the dimensionality of the problems used in our tests. Test cases nos. $1-4$, 10, and 11 are taken from [12], nos. $5-9$ from [14, Examples 5.8 - 5.11, 6.2], and nos. $12-18$ from [21], where the number appended to the abbreviation refers to the number of the variables chosen. In all cases we have chosen $k = l'$, see Tables 1 and 5, and Δ as the underlying region. This is the original domain in the examples given by Leroy, see Ler1- Ler5 below; in the other cases the polynomials are originally considered over a box which contains Δ in the test cases nos. $1-4$, $10-12$, $14-17$, or is related to Δ in the cases nos. 13 and 18.

1. **Booth:** Booth function, $n = 2$

$$p(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2.$$

2. **Himmelblau:** Himmelblau function, $n = 2$

$$p(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2.$$

3. **Rosenbrock:** Rosenbrock function, $n = 2$

$$p(x_1, x_2) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2.$$

4. **Camel 2:** Six hump camel back function, $n = 2$

$$p(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4.$$

The five Leroy functions:

5. **Ler1:** $n = 2$

$$p(x_1, x_2) = 625x_1^2 + 2550x_1x_2 + 2601x_2^2 - 1700x_1 - 3468x_2 + \frac{9249}{8}.$$

6. **Ler2:** $n = 2$

$$\begin{aligned} p(x_1, x_2) &= 25x_1^4 + 20x_1^3x_2 + 14x_1^2x_2^2 + 4x_1x_2^3 + x_2^4 + 20x_1^3 + 28x_1^2x_2 \\ &+ 12x_1x_2^2 + 4x_2^3 - 16x_1^2 - 8x_1 - 8x_2 + \frac{25}{6}. \end{aligned}$$

7. **Ler3:** $n = 2$

$$\begin{aligned} p(x_1, x_2) &= 2401x_1^4 - 1078x_1^3x_2 - 8993x_1^2x_2^2 + 2046x_1x_2^3 + 8649x_2^4 + 3822x_1^3 \\ &- 1642x_1^2x_2 - 7078x_1x_2^2 + 1488x_2^3 - 5045x_1^2 + 850x_1x_2 + 12526x_2^2 \\ &- 5226x_1 + 1072x_2 + \frac{35913}{8}. \end{aligned}$$

8. **Ler4:** $n = 2$

$$p(x_1, x_2) = -2x_1^2x_2 + 6x_1^5 + 4x_1^4x_2 - 3x_1^6 + 3x_1^3x_2^3 + 5x_2^6.$$

9. **Ler5:** $n = 2$

$$\begin{aligned} p(x_1, x_2) &= x_1^4x_2^4 - 4x_1^4x_2^3 + 4x_1^2x_2^5 + 8x_1^4x_2^2 - 8x_1^2x_2^4 + 4x_2^6 - 8x_1^4x_2 + 4x_1^2x_2^3 \\ &+ 4x_1^4 + 8x_1^2x_2^2 - 8x_2^4 - 8x_1^2x_2 + 4x_2^2 + \frac{1}{100}. \end{aligned}$$

10. **Trid 3:** 3-variable Trid function, $n = 3$

$$p(x_1, x_2, x_3) = (x_1 - 1)^2 + (x_2 - 1)^2 + (x_3 - 1)^2 - x_1 x_2 - x_2 x_3.$$

11. **Schwefel:** Schwefel function 2.25, $n = 3$

$$p(x_1, x_2, x_3) = (x_1 - x_2^2)^2 + (x_2 - 1)^2 + (x_1 - x_3^2)^2 + (x_3 - 1)^2.$$

12. **L. V. 4:** A neural network modeled by an adaptive Lotka-Volterra system, $n = 4$

$$p(x_1, x_2, x_3, x_4) = x_1 x_2^2 + x_1 x_3^2 + x_1 x_4^2 - 1.1 x_1 + 1.$$

13. **Cap 4:** Caprasse's system, $n = 4$

$$\begin{aligned} p(x_1, x_2, x_3, x_4) = \; & -x_1 x_3^3 + 4 x_2 x_3^2 x_4 + 4 x_1 x_3 x_4^2 + 2 x_2 x_4^3 \\ & + 4 x_1 x_3 + 4 x_3^2 - 10 x_2 x_4 - 10 x_4^2 + 2. \end{aligned}$$

14. **Wrig 5:** System of A. H. Wright, $n = 5$

$$p(x_1, x_2, x_3, x_4, x_5) = x_5^2 + x_1 + x_2 + x_3 + x_4 - x_5 - 10.$$

15. **Cyc 5:** Cyclic 5-roots problems, $n = 5$

$$p(x_1, x_2, x_3, x_4, x_5) = x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_5 + x_1 x_2 x_4 x_5 + x_1 x_3 x_4 x_5 + x_2 x_3 x_4 x_5.$$

16. **Reim 5:** The 5-dimensional system of Reimer, $n = 5$

$$p(x_1, x_2, x_3, x_4, x_5) = -1 + 2x_1^6 - 2x_2^6 + 2x_3^6 - 2x_4^6 + 2x_5^6.$$

17. **Mag 6:** System of magnetism in physics, $n = 6$

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = 2x_1^2 + 2x_2^2 + 2x_3^2 + 2x_4^2 + 2x_5^2 + x_6^2 - x_6.$$

18. **But 6:** Butcher's problem, $n = 6$

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = x_6 x_2^2 + x_5 x_3^2 - x_1 x_4^2 + x_4^3 + x_4^2 - \frac{1}{3} x_1 + \frac{4}{3} x_4.$$

References

[1] L.H. Bezerra, L.K. Sacht, On computing Bézier curves by Pascal matrix methods, Appl. Math. Comput. 217 (2011) 10118–10128.

[2] C. de Boor, Computational aspects of multivariate polynomial interpolation: indexing the coefficients, Adv. Comput. Math. 12 (4) (2000) 289–301.

[3] E. Chu, A. George, Inside the FFT Black Box, CRC Press, Boca Raton, London, New York, 2000.

[4] J. Czekansky, T. Sauer, The multivariate Horner scheme revisited, BIT Numer. Math. 55 (2015), 1043–1056.

[5] T. Dang, Approximate reachability computation for polynomial systems, in: J.P. Hespanha and A. Tiwari (Eds.), Hybrid Systems: Computation and Control, Proc. 9th Internat. Workshop HSCC 2006, Santa Barbara, CA, Lecture Notes in Computer Science, vol. 3927, Springer, 2006, pp. 138-152.

[6] R.T. Farouki, The Bernstein polynomial basis: A centennial retrospective, Comput. Aided Geom. Design 29 (2012) 379–419.

[7] J. Garloff, Convergent bounds for the range of multivariate polynomials, in: K. Nickel (Ed.), Interval Mathematics 1985, Lecture Notes in Computer Science, vol. 212, Springer, Berlin, Heidelberg, 1986, pp. 37–56.

[8] J. Garloff, A.P. Smith, Solution of systems of polynomial equations by using Bernstein expansion, in: G. Alefeld, S. Rump, J. Rohn, and T. Yamamoto (Eds.) , Symbolic Algebraic Methods and Verification Methods, Springer, 2001, pp. 87–97.

[9] J.L. Gross, Combinatorial Methods with Computer Applications, Chamman and Hall/CRC, Boca Raton, London, New York, 2007.

[10] R. Hungerbühler, J. Garloff, Bounds for the range of a bivariate polynomial over a triangle, Reliab. Comput. 4 (1998) 3–13.

[11] R. Hungerbühler, J. Garloff, Computation of the Bernstein coefficients on subdivided triangles, Reliab. Comput. 6 (2000) 115–121.

[12] M. Jamil, X.S. Yang, A literature survey of benchmark functions for global optimisation problems, J. Math. Model. Numer. Optim. 4 (2) (2013) 150-194.

[13] E. de Klerk, D. den Hertog, G. Elabwabi, On the complexity of optimization over the standard simplex, European J. Oper Res. 191 (2008) 773–785.

[14] R. Leroy, Certificats de positivité et minimisation polynomiale dans la base de Bernstein multivariée, Ph D Thesis, Université de Rennes 1, 2008.

[15] R. Leroy, Certificates of positivity in the simplicial Bernstein basis, preprint, available under http:hal.archives-ouvertes.fr/hal-00589945v1/, May 3, 2011.

[16] R. Leroy, Convergence under subdivision and complexity of polynomial minimization in the simplicial Bernstein basis, Reliab. Comput. 17 (2012) 11-21.

[17] S.K. Lodha, R. Goldman, A unified approach to evaluation algorithms for multivariate polynomials, Math. Comp. 66 (220) (1997) 1521–1553.

[18] P.S.V. Nataraj, M. Arounassalame. Constrained global optimization of multivariate polynomials using Bernstein branch and prune algorithm, J. Global Optim. 49 (2) (2011) 185-212.

[19] I. Novak, A branch-and-bound algorithm for computing the global minimum of polynomials by Bernstein-Bézier patches on simplices, preprint M-09, Brandenburgische Technische Universität Cottbus, 1996.

[20] J.M. Peña, T. Sauer, On the multivariate Horner scheme, SIAM J. Numer. Anal. 37 (4) (2000) 1186–1197.

[21] S. Ray, P.S.V. Nataraj, An efficient algorithm for range computation of polynomials using the Bernstein form, J. Global Optim. 45 (2009) 403–426.

[22] S. Ray, P.S.V. Nataraj, A matrix method for efficient computation of Bernstein coefficients, Reliab. Comput. 17 (2012) 40–71.

[23] M. Reuter, T.S. Mikkelsen, E.C. Sherbrooke, T. Maekawa, N.M. Patrikalakis, Solving nonlinear polynomial systems in the barycentric Bernstein basis, Visual Comput. 24 (2008) 187–200.

[24] G.S. Reynolds, Investigation of different methods of fast polynomial evaluation, Presentation for MSc in High Performance Computing, The University of Edinburgh, 2010.

[25] C. Sloth, R. Wisniewski, Robust stability of switched systems, Proc. IEEE 53rd Conference on Decision and Control. IEEE Press, 2014, pp. 4685–4690.

[26] C. Sloth, R. Wisniewski, Stability verification for energy-aware hydraulic pressure control via simplicial subdivision, Proc. European Control Conference, IEEE Press, 2015, pp. 2694–2699.

[27] Z. Tang, R. Duraiswami, N.A. Gumerov, Fast algorithm to compute matrix-vector products for Pascal matrices. UMIACS Tech. Rep. 2004-08, also issued as Computer Science Tech. Rep. CS-TR-#4563, University of Maryland, 2004.

[28] J. Titi, J. Garloff, Fast determination of the tensorial and simplicial Bernstein forms of multivariate polynomials and rational functions, Reliab. Comput. 25 (2017) 24-37.

[29] J. Titi, T. Hamadneh, J. Garloff, Convergence of the simplicial rational Bernstein form, in: Modelling, Computation and Optimization in Information Systems and Management Sciences, H. Thi, T. Dinh, and N. Nguyen (Eds.), Series Advances in Intelligent Systems and Computing, vol. 359, Springer, 2015, pp. 433-441.

[30] S.L. Yang, Explicit factorization of Pascal matrices, J. Math. Res. Exposition 24 (1) (2004) 73–82.