Universität Konstanz

# Matrix Methods for the Tensorial Bernstein Form

Jihad Titi

Jürgen Garloff

# Matrix Methods for the Tensorial Bernstein Form

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 box (tensorial Bernstein representation) are considered. A new matrix method for the computation of the polynomial coefficients with respect to the Bernstein basis, the so-called Bernstein coefficients, are presented and compared with existing methods. Also matrix methods for the calculation of the Bernstein coefficients over subboxes generated by subdivision of the original box are proposed. 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. In the case that the coefficients of the polynomial are due to uncertainties and can be represented in the form of intervals it is shown that the developed methods can be extended to compute the set of the Bernstein coefficients of all members of the polynomial family.

*Keywords:* Bernstein coefficient, tensorial Bernstein form, range enclosure, subdivision, interval polynomial

---

## 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. 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 enclosures for the range of the

objective function and the constraint functions 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, e.g., [4]. Then the minimum and maximum of the coefficients of this expansion, the so-called Bernstein coefficients, provide an enclosure for the range of the polynomial over the search region. This property is employed for the unconstrained minimization of a polynomial over an (axis-aligned) box [13], [14], [16] as well as for the constrained minimization, where the constraints are provided by polynomials [8], [9], [15]. These include mixed-integer nonlinear programming problems [18], [19]. Besides boxes also simplices [7], [11], [28] are considered as underlying regions for polynomial global optimization using the Bernstein approach. Many other applications of the range enclosing property of the Bernstein coefficients can be found in [4, Section 9] and [10].

A drawback of the traditional method for the computation of the Bernstein coefficients, see, e.g., [5], [7], [24], [32], is that the number of the Bernstein coefficients over a box which has to be computed is exponential with respect to the number of the variables. In [25], a method for the implicit representation of the Bernstein coefficients is introduced by which the computational complexity becomes nearly linear with respect to the number of the terms of the polynomial. Furthermore, three tests for the determination of the minimum Bernstein coefficient are presented for the case that the given box is restricted to a single orthant[1]; the determination of the maximum coefficient is analogous. The examples in [25] document that the use of the implicit Bernstein form and the three tests can dramatically reduce the number of the Bernstein coefficients that need to be computed explicitly, thereby speeding up the computation of the range enclosure provided by the interval spanned by the minimum and maximum Bernstein coefficients by up to several orders of magnitude. It should be noted that all the polynomials tested in [25] are sparse and of low degree, with few or no terms involving more than a single variable. This seems to be typical of the types of polynomials encountered in global optimization problems. In such cases, the three mentioned tests are much more likely to succeed, compared to a polynomial,

---

[1]If the given box spans multiple orthants of $\mathbb{R}^n$, then it should be subdivided around the origin into two or more subboxes and the Bernstein enclosure for each subbox computed seperately.

2

where each variable appears in many terms.

If a sufficiently often differentiable function is to be minimized over a box $\boldsymbol{x}$ the Bernstein approach can be applied in the way that $f$ is approximated by its Taylor polynomial. Over $\boldsymbol{x}$, the remainder part of the approximation of $f$ can be enclosed into an interval; this can be accomplished by several methods [16]. However, the Taylor polynomial of $f$ will in general be not sparse so that the approach presented in [25] does not have an advantage. In this paper, we consider the computation of the Bernstein coefficients in the case of non-sparse polynomials. Before we detail the topics of our paper, we mention three points: The fast determination of the Bernstein enclosure for polynomials over a simplex and for multivariate rational functions based in part on the three tests in [25] is presented in [28]. The parallel computation of the Bernstein coefficients is first considered in [6] and in [3] combined with the implicit Bernstein form [25]. Extensions of the Bernstein enclosure for multivariate complex polynomials and rational functions may be found in [30].

In [7], the second author proposed a difference table method for the computation of the Bernstein coefficients of a bivariate polynomial over the unit box. Ray and Nataraj introduced in [23], see also [22], a matrix method for the calculation of the Bernstein coefficients of a multivariate polynomial over a box. Using the same matrix representation of the array of the Bernstein coefficients as in [23], we present in this paper a new method for the calculation of the Bernstein coefficients over the unit box and over a general box. Our method uses as [23] solely matrix operations such as multiplication, transposition, and reshaping. It relies on the use of the lower triangular Pascal matrix and is based on the fact that this matrix can be represented as a product of bidiagonal matrices [26], [31]. This method has the same complexity as the one in [23] but the number of the required arithmetic operations is smaller.

In practical problems, the coefficients of a polynomial may be corrupted by uncertainties due to, e.g., rounding errors or data variability. If we can embrace these uncertainties by upper and lower bounds then we have a polynomial with coefficients varying in given intervals, i.e., a family of polynomials. The question arises, how to compute the Bernstein coefficients in the presence of variation of the polynomial coefficients, i.e., of all members of the

3

interval family. We show that in many cases the performance of the developed methods in interval arithmetic yields this set of the Bernstein coefficients. In the other cases a reformulation of the representation can be used to avoid overestimation[2]. So we are able to compute the Bernstein coefficients of all members of the interval family without any overestimation.

The organization of the paper is as follows: In the next section, we introduce the notation and some matrices which are used throughout the paper. In Section 3, we briefly recall the most important properties of the Bernstein expansion over a box. In Section 4, we derive the proposed matrix method for the computation of the Bernstein coefficients over the unit box and over a general box and compare it with existing methods. We present matrix methods for the calculation of the Bernstein coefficients on subboxes generated by subdivision in Section 5. The related problems for interval polynomials are considered in Section 6. Conclusions are drawn in the last section.

## 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 $(0, \ldots, 0)$. Comparison between and arithmetic operations with multi-indices are defined 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 $\sum_{i=0}^{d} := \sum_{i_1=0}^{d_1} \ldots \sum_{i_n=0}^{d_n}$ and $\binom{d}{i} := \prod_{s=1}^{n} \binom{d_s}{i_s}$.

Let $\mathbb{IR}$ be the set of compact, nonempty real intervals $[x] = [\underline{x}, \overline{x}]$, $\underline{x} \leq \overline{x}$. A *box* (also termed *interval vector*) is a vector and an *interval matrix* is a matrix with components from $\mathbb{IR}$. Interval quantities (with the exception of intervals denoted by brackets) are written in bold.

A general comment on the numbering of indices: It turned out that most of the formulae in the presentation of the matrix methods become simpler when the indices of matrices are starting running at 0 instead of running at 1.

---

[2]The explanation of the formulation 'without overestimation' is given in Section 6.

Therefore, we are using this unusual numbering. We introduce the following matrices[3] of $\mathbb{R}^{l_s+1,l_s+1}$ ($l_s$ will be introduced in the next section). The lower triangular Toeplitz matrix $T_s$ is defined as

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

$P_s$ is the *lower triangular Pascal matrix*, i.e.,

$$(P_s)_{ij} := \begin{cases} \binom{i}{j}, & \text{if } j \leq i, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The bidiagonal matrices $K_\mu^s$, $\mu = 1, \ldots, l_s$, are given by

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

We will make use of the following factorization, e.g., [26, Lemma 2.4],

$$P_s = \prod_{\mu=1}^{l_s} K_\mu^s. \tag{4}$$

The matrices $D_s(t)$ and $G_s$ are the diagonal matrices $D_s(t) := \text{diag}(1, t, t^2, \ldots, t^{l_s})$, and $G_s := \text{diag}(1, 1, 2!, \ldots, l_s!)$. The matrix $P_s$ can also be factorized as

$$P_s = G_s T_s G_s^{-1}, \tag{5}$$

see, e.g., [1], [26, 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 unit box

In this section, we present fundamental properties of the Bernstein expansion over a box that are employed throughout the paper. For simplicity

---

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

5

we consider the unit box $\boldsymbol{u} := [0,1]^n$, since any compact nonempty box $\boldsymbol{x}$ of $\mathbb{R}^n$ can be mapped affinely onto $\boldsymbol{u}$. Let $p$ be an $n$-variate polynomial with the power representation

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

We expand $p$ into Bernstein polynomials of degree $d$, $d \geq l$, over $\boldsymbol{u}$ as

$$p(x) = \sum_{i=0}^{d} b_i^{(d)} B_i^{(d)}(x), \tag{7}$$

where $B_i^{(d)}$ is the *i-th Bernstein polynomial of degree d*, defined as

$$B_i^{(d)}(x) := \binom{d}{i} x^i (1-x)^{d-i}, \tag{8}$$

and $b_i^{(d)}$ is the *i-th Bernstein coefficient of $p$ of degree $d$ over $\boldsymbol{u}$* which is given by

$$b_i^{(d)} = \sum_{j=0}^{i} \frac{\binom{i}{j}}{\binom{d}{j}} a_j, \quad 0 \leq i \leq d, \tag{9}$$

with the convention that $a_j := 0$ for $j \geq l$, $j \neq l$. We call (7) the *Bernstein representation of $p$* (more precisely, the *tensorial* Bernstein representation to distinguish it from the representation with respect to the basis of Bernstein polynomials over a simplex, see, e.g., [11], [29], called the *simplicial* Bernstein representation).

The Bernstein coefficients can be arranged in a multidimensional array $B(\boldsymbol{u}) = (b_i^{(d)})_{0 \leq i \leq d}$, the so-called *Bernstein patch*.

The Bernstein coefficients provide lower and upper bounds for the range of $p$ over $\boldsymbol{u}$,

$$\min_{i=0}^{d} b_i^{(d)} \leq p(x) \leq \max_{i=0}^{d} b_i^{(d)}, \quad \text{for all } x \in \boldsymbol{u}. \tag{10}$$

This property is called the *range enclosure property*. The interval spanned by the minimum and maximum Bernstein coefficients is called the *Bernstein*

*form of p over $\boldsymbol{u}$.* Equality holds in the left or the right inequalities of (10) if and only if the minimum or the maximum, respectively, is attained at a vertex of $B(\boldsymbol{u})$, i.e., if $i_s \in \{0, d_s\}$, $s = 1, \ldots, n$. This condition is known as the *vertex condition.* From the representation (9) the *linearity* of the Bernstein coefficients follows immediately: Let $p_1$ and $p_2$ be polynomials with the power representations (6) with $l = l^{(1)}$ and $l = l^{(2)}$, respectively, and let $l := \max \left\{ l^{(1)}, l^{(2)} \right\}$. If $p = \alpha p_1 + \beta p_2$, $\alpha, \beta \in \mathbb{R}$, then

$$b_i^{(d)}(p) = \alpha b_i^{(d)}(p_1) + \beta b_i^{(d)}(p_2), \text{ for all } 0 \leq i \leq d, \tag{11}$$

where $b_i^{(d)}(p_1)$ and $b_i^{(d)}(p_2)$ are the $i$-th coefficients of the Bernstein expansions of degree $d$ of $p_1$ and $p_2$, respectively, $d \geq l$.

We can tighten the *Bernstein form of p over $\boldsymbol{u}$* (10) by elevating the degree $d$ of the Bernstein expansion or subdividing $\boldsymbol{u}$. 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 $\boldsymbol{u}$, in contrast to linear convergence when degree elevation is applied, see [5], [7]. So we put $d = l$; the methods investigated in this paper easily extend to $d > l$. In the sequel, we suppress the upper index $l$. Subdivision of $\boldsymbol{u}$ in the $s$-th coordinate direction, $s \in \{1, \ldots, n\}$, produces two subboxes $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$,

$$\begin{aligned}
\boldsymbol{u}_1 &:= [0,1] \times \ldots \times [0, \lambda] \times \ldots \times [0,1], \\
\boldsymbol{u}_2 &:= [0,1] \times \ldots \times [\lambda, 1] \times \ldots \times [0,1],
\end{aligned} \tag{12}$$

for some $\lambda \in (0,1)$.

The Bernstein coefficients over $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, denoted by $b_i(\boldsymbol{u}_1)$ and $b_i(\boldsymbol{u}_2)$, respectively, can be computed from the Bernstein coefficients $b_i$ over $\boldsymbol{u}$ by using the de Casteljau algorithm, see [4], [21]. This algorithm is summarized as follows:

1. Put $b_i^{[0]} := b_i^{(l)}$, $0 \leq i \leq l$.
2. For $\nu = 1, \ldots, l_s$:

$$b_i^{[\nu]} := \left\{ \begin{array}{ll} b_i^{[\nu-1]}, & \text{if } i_s < \nu \\ (1-\lambda)b_{i_1,\ldots,i_s-1,\ldots,i_n}^{[\nu-1]} + \lambda b_i^{[\nu-1]}, & \text{if } i_s \geq \nu \end{array} \right\}, \quad 0 \leq i \leq l. \tag{13}$$

3. $b_i^{(l)}(\boldsymbol{u}_1) := b_i^{[l_s]}$; $b_i^{(l)}(\boldsymbol{u}_2) := b_{i_1,\ldots,l_s,\ldots,i_n}^{[l_s-i_s]}$, $0 \leq i \leq l$.

7

The Bernstein coefficients $b_i(\boldsymbol{u}_2)$ on the neighbouring subbox $\boldsymbol{u}_2$ are obtained as intermediate values in this computation, since for $\nu = 0, \ldots, l_s$ the following relation holds

$$b^{(l)}_{i_1,\ldots,l_s-\nu,\ldots,i_n}(\boldsymbol{u}_2) = b^{[\nu]}_{i_1,\ldots,l_s,\ldots,i_n}. \tag{14}$$

## 4. Methods for computing the Bernstein coefficients over a box

### 4.1. Garloff's method

In [7], the second author proposed a method for computing the Bernstein coefficients of a bivariate polynomial $p$ given by (6) over the unit box $\boldsymbol{u}$ by using the following relation of a forward difference operator $\Delta$

$$\Delta_{j_1,j_2} b_{0,0} = \frac{1}{\binom{l_1}{j_1}\binom{l_2}{j_2}} a_{j_1,j_2}, \ j_s = 0, \ldots, l_s, \ s = 1, 2. \tag{15}$$

Then, the Bernstein coefficients can be computed from (15) by using the following recurrence relations:

$$\begin{aligned}
\Delta_{0,0} b_{i_1,i_2} &= b_{i_1,i_2}, \\
\Delta_{j_1+1,j_2} b_{i_1,i_2} &= \Delta_{j_1,j_2} b_{i_1+1,i_2} - \Delta_{j_1,j_2} b_{i_1,i_2}, \\
\Delta_{j_1,j_2+1} b_{i_1,i_2} &= \Delta_{j_1,j_2} b_{i_1,i_2+1} - \Delta_{j_1,j_2} b_{i_1,i_2},
\end{aligned}$$

where $0 \leq i_s + 1 \leq l_s, \ s = 1, 2$, and

$$\Delta_{j_1,j_2} b_{0,0} := 0 \ \text{if} \ j_1 > l_1 \ \text{or} \ j_2 > l_2.$$

This can be done by performing sequentially the following steps in the $(l_1 + 1) \times (l_2 + 1)$ matrix with the scaled polynomial coefficients given by (15) as entries:

- Step $(\nu)$: Starting from the $\nu$-th column, add each column to the following one, $\nu = 1, \ldots, l_2$.

- Step $(\nu')$: Starting from the $\nu'$-th row, add each row to the following one, $\nu' = 1, \ldots, l_1$.

An extension of this method to multivariate polynomials was used in [32].

## 4.2. Ray and Nataraj's method

In [23], Ray and Nataraj proposed a matrix method for computing the Bernstein coefficients of a multivariate polynomial over the unit and a general box. Their method involves only matrix operations such as multiplication, inversion, transposition, and reshaping. The method is obtained by equating the matrix representations of a polynomial in the power representation and in the Bernstein representation and solving for the Bernstein coefficients. In contrast, our matrix method relies on the matrix representation of Garloff's method and the use of an affine transformation to circumvent the costly use of the Horner's scheme, see, e.g., [20], for the conversion of a polynomial over a general box to the unit box, see Subsection 4.4. We make, however, use of the matrix representation of the Bernstein patch and the operation of cyclic reordering on which the matrix method by Ray and Nataraj [23] relies.

## 4.3. Matrix method for the unit box

In the next subsection, we present a method for the computation of the Bernstein coefficients of a given multivariate polynomial over a general box which is superior to the two former methods. In this subsection, we present the proposed method over the unit box $\boldsymbol{u}$.

a) Bivariate case

We start with a matrical description of Garloff's method that is presented in Subsection 4.1. We arrange the coefficients of $p$ in a matrix $A$ as follows:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \ldots & a_{0,l_2} \\ a_{1,0} & a_{1,1} & \ldots & a_{1,l_2} \\ \vdots & \vdots & \ldots & \vdots \\ a_{l_1,0} & a_{l_1,1} & \ldots & a_{l_1,l_2} \end{bmatrix}. \tag{16}$$

Let the entries of the matrix $\Lambda(\boldsymbol{u}) = (\lambda_{j_1,j_2})$ represent the right-hand side of (15), i.e.,

$$\lambda_{j_1,j_2} := \frac{a_{j_1,j_2}}{\binom{l_1}{j_1}\binom{l_2}{j_2}}, \tag{17}$$

where $j_s = 0, \ldots, l_s$, $s = 1, 2$. Then steps $\nu$, $\nu = 1, \ldots, l_2$, and $\nu'$, $\nu' = 1, \ldots, l_1$, see Subsection 4.1, can be represented by using the matrices $K_\mu^s$, $\mu =$

$1, \ldots, l_s, \ s = 1, 2$, see (3). Then the Bernstein patch is given as

$$B(\boldsymbol{u}) = K_1^1 \cdots K_{l_1}^1 \Lambda(\boldsymbol{u}) \ {K_{l_2}^2}^T \cdots {K_1^2}^T. \tag{18}$$

By using the associativity property of the matrix multiplication and transposition, (18) can be written as

$$B(\boldsymbol{u}) = (K_1^2 \cdots K_{l_2}^2 (K_1^1 \cdots K_{l_1}^1 \Lambda(\boldsymbol{u}))^T)^T. \tag{19}$$

Then by (4), (19) can be represented as

$$B(\boldsymbol{u}) = (P_2 (P_1 \Lambda(\boldsymbol{u}))^T)^T. \tag{20}$$

b) Multivariate case

We extend the procedure introduced in a) to the $n$-variate polynomial $p$ given in (6). 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 (see Figure 1 for an illustration in the trivariate case). So after $n$ cyclic orderings the sequence of the indices is again in its initial order. Note that in the bivariate case the cyclic ordering is just the usual matrix transposition. In this way, the matrix multiplication in formulae like (24) and (29) is well defined.

The coefficients of $p$ are arranged in an $(l_1 + 1) \times l^\star$ matrix $A$, where $l^\star := \prod\limits_{s=2}^{n} (l_s + 1)$. The correspondence between the coefficients $a_{i_1, \ldots, i_n}$ of $p$ and the entry of $A$ in row $i$ and column $j$ is as follows:

$$i = i_1, \tag{21a}$$

$$j = i_2 + \sum_{s=3}^{n} i_s (l_2 + 1) \cdot \ldots \cdot (l_{s-1} + 1), \tag{21b}$$

10

where $i = 0, \ldots, l_1$, $j = 0, \ldots, l^\star - 1$. Then $A$ is the matrix

$$
\begin{bmatrix}
a_{0,0,0,\ldots,0} & a_{0,1,0,\ldots,0} \cdots & a_{0,l_2,0,\ldots,0} & a_{0,0,1,\ldots,0} & \cdots & a_{0,l_2,1,\ldots,0} & \cdots & a_{0,0,l_3,\ldots,0} & \cdots & a_{0,l_2,l_3,\ldots,0} \cdots \\
a_{1,0,0,\ldots,0} & a_{1,1,0,\ldots,0} \cdots & a_{1,l_2,0,\ldots,0} & a_{1,0,1,\ldots,0} & \cdots & a_{1,l_2,1,\ldots,0} & \cdots & a_{1,0,l_3,\ldots,0} & \cdots & a_{1,l_2,l_3,\ldots,0} \cdots \\
\vdots & \vdots \quad \cdots & \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \quad \cdots \\
a_{l_1,0,0,\ldots,0} & a_{l_1,1,0,\ldots,0} \cdots & a_{l_1,l_2,0,\ldots,0} & a_{l_1,0,1,\ldots,0} & \cdots & a_{l_1,l_2,1,\ldots,0} & \cdots & a_{l_1,0,l_3,\ldots,0} & \cdots & a_{l_1,l_2,l_3,\ldots,0} \cdots
\end{bmatrix}
$$

$$
\begin{bmatrix}
\cdots & a_{0,0,l_3,\ldots,l_n} & a_{0,1,l_3,\ldots,l_n} & \cdots & a_{0,l_2,l_3,\ldots,l_n} \\
\cdots & a_{1,0,l_3,\ldots,l_n} & a_{1,1,l_3,\ldots,l_n} & \cdots & a_{1,l_2,l_3,\ldots,l_n} \\
\cdots & \vdots & \vdots & \cdots & \vdots \\
\cdots & a_{l_1,0,l_3,\ldots,l_n} & a_{l_1,1,l_3,\ldots,l_n} & \cdots & a_{l_1,l_2,l_3,\ldots,l_n}
\end{bmatrix}.
\tag{22}
$$

Let $\lambda_{j_1,j_2}$, $0 \leq j_1 \leq l_1$, $0 \leq j_2 \leq l^\star - 1$, be the entry of the matrix $\Lambda(\boldsymbol{u})$, which is obtained by dividing the corresponding entry of $A$ by $\binom{l}{i}$, i.e.,

$$
\lambda_{j_1,j_2} := \frac{a_i}{\binom{l}{i}}, \quad i = 0, \ldots, l,
\tag{23}
$$

where $j_1$ equals $i_1$ and $j_2$ can be determined by the right-hand side of (21b). We put $\Lambda_0 := \Lambda(\boldsymbol{u})$ and define for $s = 1, \ldots, n$

$$
\Lambda_s := (P_s \Lambda_{s-1})^c.
\tag{24}
$$

Then for $s = 1, \ldots, n$ the entry in position $(v_1, v_2)$ in $\Lambda_{s-1}$ becomes $(v_1', v_2')$ in $\Lambda_s$, where

$$
\begin{aligned}
v_1' &= v_2 \bmod (l_{s+1} + 1), \\
v_2' &= \left\lfloor \frac{v_2}{l_{s+1} + 1} \right\rfloor + v_1 \prod_{\substack{m=1, \\ m \neq s, s+1}}^{n} (l_m + 1).
\end{aligned}
$$

To explain in which way we get the Bernstein patch $B(\boldsymbol{u})$, we define for $i_1 = 0, \ldots, l_1$ $\Lambda_{i_1}(\boldsymbol{u}) := \Lambda(\boldsymbol{u})$, and for $i_s = 0, \ldots, l_s$, $s = 2, \ldots, n$, the $(l_s + 1) \times \prod_{\substack{w=1, \\ w \neq s}}^{n} (l_w + 1)$ matrix

$$
\Lambda_{i_s}(\boldsymbol{u}) := (\Lambda_{i_{s-1}}(\boldsymbol{u}))^c.
\tag{25}
$$

11

The $i$-th Bernstein coefficient given in (9) can be explicitly written as

$$b^{(l)}_{i_1,i_2,\dots,i_n} = \sum_{j_n=0}^{i_n} \binom{i_n}{j_n} \cdots \sum_{j_2=0}^{i_2} \binom{i_2}{j_2} \sum_{j_1=0}^{i_1} \binom{i_1}{j_1} \frac{a_{j_1,j_2,\dots,j_n}}{\binom{l}{j}}. \tag{26}$$

From (26), we make the following observation: The first inner-most sum can be obtained by multiplying the $i_1$-th row vector of the Pascal matrix of order $l_1 + 1$ by the first $i_1$ rows of $\Lambda_{i_1}(\boldsymbol{u})$. Then we get a row vector of length $\prod_{w=2}^{n} (l_w + 1)$. After that, we apply the cyclic ordering and obtain the matrix $\Lambda_{i_2}(\boldsymbol{u})$, see (25). Now, the second inner-most sum is performed by multiplying $\Lambda_{i_2}(\boldsymbol{u})$ by the $i_2$-th row of the Pascal matrix of order $l_2 + 1$ which yields a row vector of length $\prod_{w=3}^{n} (l_w + 1)$. Continuing in this way and observing that in each column of $\Lambda_{i_s}(\boldsymbol{u})$, $s = 1, \dots, n$, the indices of the entries (except the first index) are fixed due to cyclic ordering, we obtain by $\Lambda_n$ the Bernstein patch $B(\boldsymbol{u})$ arranged accordingly in an $(l_1 + 1) \times l^*$ matrix, denoted by $\mathcal{B}(\boldsymbol{u})$, and thus as

$$\mathcal{B}(\boldsymbol{u}) = (P_n(\cdots(P_2(P_1\Lambda(\boldsymbol{u}))^c)^c \cdots)^c)^c.$$

In Subsection 4.5, we investigate a method for the computation of $\mathcal{B}(\boldsymbol{u})$ according to (24) which relies on the factorization (4) of the matrices $P_s$, $s = 1, \dots, n$.

*4.4. Matrix method for a general box*

Firstly, we affinely map a given box $\boldsymbol{x}$,

$$\boldsymbol{x} = ([\underline{x}_s, \overline{x}_s])_{s=1}^{n}, \text{ with } \underline{x}_s < \overline{x}_s, \ s = 1, \dots, n,$$

to the unit box $\boldsymbol{u}$ by

$$z_s = \frac{x_s - \underline{x}_s}{\overline{x}_s - \underline{x}_s}, \quad s = 1, \dots, n. \tag{27}$$

We sequentially transform $x_s$, $s = 1, \dots, n$. By substituting (27) in (6) for one $x_s$, $s = 1, \dots, n$, at a time, we obtain a polynomial $p^\star$ over $\boldsymbol{u}$. The coefficients of $p^\star$ arranged in an $(l_1 + 1) \times l^\star$ matrix, say $A^\star$, can be derived as follows from the matrix $A$ of the coefficients of $p$ given in (22): For $s = 1, \dots, n$ define

$$Q_s := \begin{cases} D_s(\frac{\overline{x}_s - \underline{x}_s}{\underline{x}_s})P_s^T D_s(\underline{x}_s), & \underline{x}_s \neq 0, \\ D_s(\overline{x}_s), & \underline{x}_s = 0. \end{cases} \tag{28}$$

Then $A^\star$ can be represented as

$$A^\star = (Q_n(\cdots(Q_2(Q_1 A)^c)^c \cdots)^c)^c. \tag{29}$$

By applying the procedure that is presented in Subsection 4.3 to the matrix $A^\star$, we obtain the Bernstein patch of $p$ over $\boldsymbol{x}$.

### 4.5. Amount of arithmetic operations

For simplicity and ease of comparison between the existing methods, our standing assumption here is that $l_1 = \cdots = l_n = \kappa$. Therefore, we suppress the subscript of the matrices that are used below; they all belong to $\mathbb{R}^{\kappa+1,\kappa+1}$. Also, we will suppress the upper index of the matrices $K_\mu^s$, $s = 1, \ldots, n$. Here and in the sequel we assume that some basic quantities like factorials and binomial coefficients are precomputed since the number of operations required for their calculation is of lower order compared to the computation of the other quantities. Even if we present for procedures the numbers of their additions and multiplications/ divisions separately, we will not make a difference between them when counting the arithmetic operations.

a) Unit box

In this subsection, we present a method by which we carry out the matrix multiplication in (24). We sequentially multiply in (24) $\Lambda(\boldsymbol{u})$ by $P$ according to the factorization (4), i.e., we firstly multiply $K_\kappa$ and $\Lambda(\boldsymbol{u})$ and multiply the resulting matrix by $K_{\kappa-1}$ and so on. The main advantage of using factorization (4) of the Pascal matrix is that it allows us to get rid of the multiplication operations. This method requires $n\frac{\kappa(\kappa+1)^n}{2}$ additions and $n(\kappa+1)^n$ multiplications/ divisions. Since this method is merely a matrical representation of Garloff's method, the amount of operations is identical for both methods. Ray and Nataraj report for their method in [23] also $n\frac{\kappa(\kappa+1)^n}{2}$ additions but $\frac{\kappa(\kappa-1)}{2} + n\frac{\kappa(\kappa+1)^n}{2}$ multiplications which are more than required for our method if $\kappa > 1$. So our method is superior to Ray and Nataraj's method if $\kappa > 1$.

b) General box

In [23, Section 6], Ray and Nataraj compare their method for computing the Bernstein coefficients over a general box with Garloff's method. Since

13
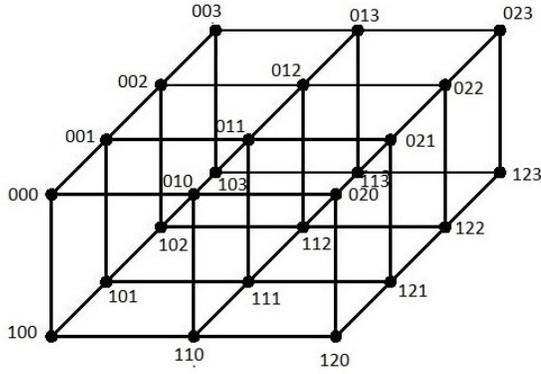
(a) $\Lambda(\boldsymbol{u})$
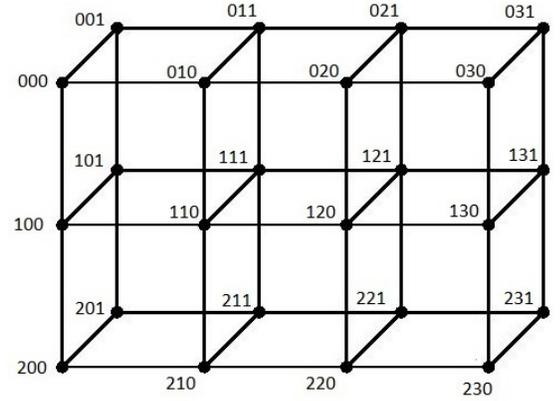
(b) $(P_1\Lambda(\boldsymbol{u}))^c$
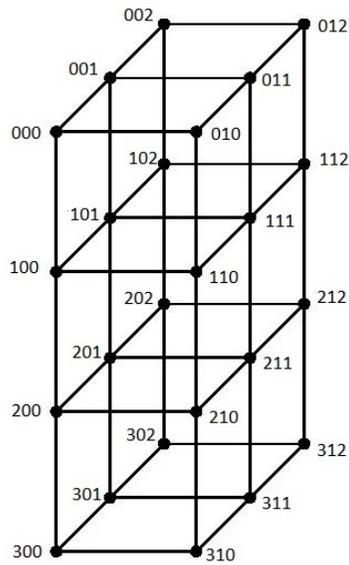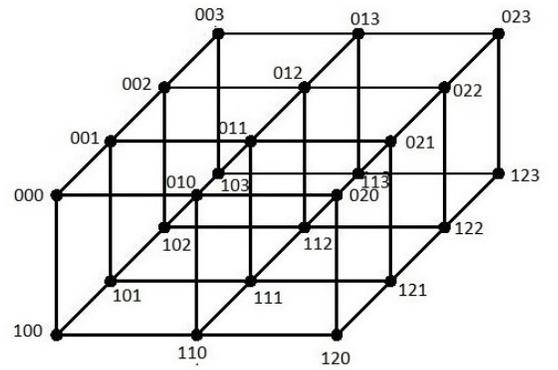
(c) $(P_2(P_1\Lambda(\boldsymbol{u}))^c)^c$

(d) $(P_3(P_2(P_1\Lambda(\boldsymbol{u}))^c)^c)^c$ providing

$\mathcal{B}(\boldsymbol{u})$

Figure 1: Cyclic ordering of a three-dimensional array with $l_1 = 1$, $l_2 = 2$, and $l_3 = 3$

14

the latter one uses the multivariate Horner's scheme to transform the box to $\boldsymbol{u}$ it has the complexity of this scheme, viz. $O(\kappa^{2n})$. We will not consider this method further because the other two methods have lower complexity. We extend now our method to a general box.

In (29), the multiplication by the matrix $P^T$ is performed by sequential multiplication by the matrices $K_\mu^T$, $\mu = 1, \ldots, \kappa$, see (3). In Table 1 the number of the operations of this method and Ray and Nataraj's method (as reported in [23]) is presented. We see that both methods have the same complexity $O(n\kappa^{n+1})$ but Ray and Nataraj's method requires for $\kappa > 1$ more arithmetic operations.

Table 1: Number of arithmetic operations required to obtain the Bernstein patch over a general box by our method and Ray and Nataraj's method (assuming that $\underline{x}_s \neq 0$, $s = 1, \ldots, n$)

| Calculation of | number of additions | number of multiplications/div. |
|---|---|---|
| $D_s(\underline{x}_s)$, $D_s(\frac{\overline{x}_s - \underline{x}_s}{\underline{x}_s})$, $s = 1, \ldots, n$ | $n$ | $2n(\kappa - 1) + n$ |
| $A^\star$ | $n\kappa \frac{(\kappa+1)^n}{2}$ | $2n(\kappa + 1)^n$ |
| our method (starting from $A^\star$) | $n\kappa \frac{(\kappa+1)^n}{2}$ | $n(\kappa + 1)^n$ |
| total number of our method | $n\kappa(\kappa + 1)^n + n$ | $3n(\kappa + 1)^n + 2n(\kappa - 1) + n$ |
| Ray and Nataraj's method | $n + \frac{n\kappa(\kappa+1)(2\kappa+1)}{6} + n\kappa(\kappa+1)^n$ | $\frac{\kappa(\kappa-1)}{2} + \kappa n(\kappa + 3) + \frac{n(\kappa+1)(\kappa+2)(2\kappa+3)}{6} + n(\kappa + 1)^{n+1}$ |

**Remark 4.1.** *If we have $\varsigma$ terms such that $\underline{x}_s = 0$, see (28), then the number of additions and multiplications / divisions given in Table 1 for our method is $\varsigma(\kappa \frac{(\kappa+1)^n}{2} + 1)$ and $\varsigma((\kappa + 1)^n + \kappa)$, respectively, less.*

**Remark 4.2.** *Since the denominator of $\lambda_i$ depends on $i_s$, $i_s = 0, \ldots, l_s$, $s = 1, \ldots, n$, i.e., on the power of $x_s$, we define $D'_s := \operatorname{diag}(1, \frac{1}{\binom{l_s}{1}}, \frac{1}{\binom{l_s}{2}}, \ldots, \frac{1}{\binom{l_s}{l_s}})$, $s = 1, \ldots n$. If we replace now in (29) $Q_s$ by $D'_s Q_s$, $s = 1, \ldots, n$, then we get*

15

$\Lambda(\boldsymbol{u})$ *instead of* $A^{\star}$, *i.e.,*

$$\Lambda(\boldsymbol{u}) = (D'_n Q_n(\cdots(D'_2 Q_2(D'_1 Q_1 A)^c)^c \cdots)^c)^c. \tag{30}$$

*By combining the two diagonal matrices* $D'_s$, $D_s(\frac{\overline{x}_s - \underline{x}_s}{\underline{x}_s})$ *as one diagonal matrix* $(D'_s, D_s(\overline{x}_s)$ *if* $\underline{x}_s = 0)$, $s = 1, \ldots, n$, *we can reduce the number of multiplications of our method by* $n(\kappa + 1)^n - n\kappa$. *In passing we note that by the combination of* (24) *and* (30) *identity (63) in [23] can be derived which is fundamental for the approach in [23].*

We illustrate the performance of our method by a pseudocode and Example 1.

---

**Algorithm 1** Method for the computation of the Bernstein coefficients over a general box

---

1: Input: The box $\boldsymbol{x}$ and the coefficients of the power representation of $p$ over $\boldsymbol{x}$.
2: Output: The matrix $\mathcal{B}(\boldsymbol{u})$ containing the Bernstein coefficients of $p$ over $\boldsymbol{x}$.
3: Step 1: Arrange the coefficients of $p$ in matrix $A$.
4: Step 2: Compute $\Lambda(\boldsymbol{u})$.
5: Put $Q_0^{\star} := A$.
6: **for** $s = 1, \ldots, n$ **do**
7:     Compute $Q_s^{\star} := D'_s Q_s Q_{s-1}^{\star}$, where $Q_s$ is given in (28).
8: **end for**
9: Step 3: Computation of $\mathcal{B}(\boldsymbol{u})$
10: Put $\Lambda_1^1 := Q_n^{\star}$.
11: **for** $s = 1, \ldots, n$ **do**
12:     **for** $\mu = 1, \ldots, l_s$ **do**
13:         Compute $\Lambda_{\mu+1}^s := K_{l_s-\mu+1}^s \Lambda_\mu^s$.
14:     **end for**
15:     Put $\Lambda_1^{s+1} := (\Lambda_{l_s+1}^s)^c$.
16: **end for**
17: Put $\mathcal{B}(\boldsymbol{u}) := \Lambda_1^{n+1}$.
18: Step 4: End of the algorithm.

---

**Example 4.1.** *Let $p$ be the Himmelblau function, see Appendix 1. The coef-*

*ficient matrix A of p is given as follows*

$$A = \begin{bmatrix} 170 & -22 & -13 & 0 & 1 \\ -14 & 0 & 2 & 0 & 0 \\ -21 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{31}$$

*The coefficients of p after the affine mapping of $\boldsymbol{x} = [-5, 5] \times [-5, 5]$ in the direction of $x_1$ can be obtained from (rounded to five decimal places)*

$$D_1' Q_1 A = \begin{bmatrix} 340 & 28 & -23 & 0 & 1 \\ -760 & -50 & 5 & 0 & 0 \\ 2150 & 33.33333 & 0 & 0 & 0 \\ -5000 & 0 & 0 & 0 & 0 \\ 10000 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{32}$$

*After multiplication of the transposed matrix in (32) by $D_2' Q_2$ and transposition of the resulting matrix, cf. (30), we get $\Lambda(\boldsymbol{u})$ as*

$$\Lambda(\boldsymbol{u}) = \begin{bmatrix} 250 & -605 & 2116.66667 & -5000 & 10000 \\ -385 & -250 & 83.33333 & 0 & 0 \\ 1983.33333 & 83.33333 & 0 & 0 & 0 \\ -5000 & 0 & 0 & 0 & 0 \\ 10000 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

*The application of our method with respect to $x_1$, i.e., performance of (24) for $s = 1$, yields*

$$P_1 \Lambda(\boldsymbol{u}) = \begin{bmatrix} 250 & -605 & 2116.66667 & -5000 & 10000 \\ -135 & -855 & 2200 & -5000 & 10000 \\ 1463.33333 & -1021.66667 & 2283.33333 & -5000 & 10000 \\ 45 & -1105 & 2366.66667 & -5000 & 10000 \\ 610 & -1105 & 2450 & -5000 & 10000 \end{bmatrix}.$$

*Finally, the Bernstein coefficient matrix is given by*

$$B(\boldsymbol{u}) = \left( P_2 \left( P_1 \Lambda(\boldsymbol{u}) \right)^T \right)^T = \begin{bmatrix} 250 & -355 & 1156.66667 & -215 & 530 \\ -135 & -990 & 355 & -1100 & -355 \\ 1463.33333 & 441.66667 & 1703.33333 & 248.33333 & 1076.66667 \\ 45 & -1060 & 201.66667 & -1170 & -175 \\ 610 & -495 & 850 & -355 & 890 \end{bmatrix}.$$

17

## 4.6. Use of an alternative factorization of the Pascal matrix

In the case of the unit box, we may use the factorization (5) instead of (4) of the Pascal matrix. The factorials $i_s!$ appearing in $\binom{\kappa}{i_s}^{-1}$ and on the diagonal of $G^{-1}$ cancel out, $s = 1, \ldots, n$. Therefore, if we multiply $a_{i_1,\ldots,i_n}$ by $\frac{(\kappa-i_1)!}{\kappa!} \cdots \frac{(\kappa-i_n)!}{\kappa!}$ and name the resulting matrix $\Lambda'(\boldsymbol{u})$ then

$$\underbrace{(GT(\cdots(GT(GT\,\Lambda'(\boldsymbol{u}))^c)^c\cdots)^c)^c}_{n \ times} \tag{33}$$

yields the matrix $\Lambda_n$. The formulation 'n times' includes here and in the sequel the $n$-fold cyclic ordering.

In the case of a general box $\boldsymbol{x}$, we assume without loss of generality that $\underline{x}_s \neq 0$, $s = 1, \ldots, n$. After substituting (5) in (28) and defining

$$F_s := GD(\underline{x}_s) \quad \text{and} \quad E_s := D(\frac{\overline{x}_s - \underline{x}_s}{\underline{x}_s})G^{-1}, \quad s = 1, \ldots, n,$$

we obtain

$$Q_s = E_s T^T F_s.$$

The matrix $A^\star$ in (29) can then be written as

$$A^\star = (E_n T_n^T F_n(\cdots(E_2 T_2^T F_2(E_1 T_1^T F_1 A)^c)^c \cdots)^c)^c. \tag{34}$$

We proceed then with the matrix $A^\star$ as with $A$ in the case of the unit box, see (33).

The multiplication of the matrix $T$ by a vector can be carried out by using the FFT [26, Theorem 1.6], see [2, Chapter 13]. Then using (33), the total number of operations required for the computation of $B(\boldsymbol{u})$ can be reduced to $O(n\kappa^n \log_2 \kappa)$. At a first glance, it seems that the number of operations required for the computation of the Bernstein coefficients could significantly be reduced. However, in order to apply the FFT the size of the Bernstein patch has to be properly extended. Specifically, the multiplication of $\Lambda_{s-1}$ by $P_s$ in (24), $s = 1, \ldots, n$, according to the factorization (4), i.e., by the matrices $K_\mu$, $\mu = 1, \ldots, n$, requires $n(\kappa+1)^{n-1}\frac{\kappa(\kappa+1)}{2}$ arithmetic operations,

18

whereas the application of the FFT needs $n(\kappa + 1)^{n-1}\left(15m\log_2 m + m + \kappa\right)$ operations [2, Section 3.1], where $m$ is the smallest natural number greater than or equal to $2\kappa - 2$ such that $m = 2^\sigma$ for some $\sigma$ [2, Chapter 13]. Thus, the use of the FFT is superior to our method when $n(\kappa + 1)^{n-1}\frac{\kappa(\kappa+1)}{2} > n(\kappa + 1)^{n-1}\left(15m\log_2 m + m + \kappa\right)$, i.e., when $\frac{\kappa(\kappa+1)}{2} > 15\sigma 2^\sigma + 2^\sigma + \kappa$. This inequality is fulfilled only for $\kappa \in [826, 1025]$ and for all $\kappa \geq 1219$. Therefore, our method is superior for $\kappa \in [1, 825] \cup [1026, 1218]$. Since the use of the FFT is advantageous only for polynomials of extremely high degree we will not pursue this approach further[4].

*4.7. Numerical examples*

We have run the 11 at least trivariate test problems listed in Appendix 1. In the first three columns of Table 2 the abbreviated name of the example, the number $n$ of the variables, and the multi-index $l$, see (6), are displayed. In the examples no degree elevation of $l$ to $\kappa$ is applied. In columns 6 and 9 we report the *percent reduction* of our method which is provided by

$$\frac{\text{time taken by R.N. method} - \text{time taken by our method}}{\text{time taken by R.N. method}} \times 100,$$

see [23, p.66]. Both the results for the unit box and general boxes clearly document the superiority of our method to the method by Ray and Nataraj except for the three test functions by Reimer. These test functions are formed as a sum of univariate monomial terms. This structure is taken into account by a variant of our method which is explained in Appendix 2. By using this variant, the timings for the Reimer examples are smaller than the ones when Ray and Nataraj′s method is used. However, this does not apply to the other test examples with a similar structure, viz. R. D. 3, Wrig 5, and Mag 6/7. The results in Table 2 as well as in Tables 5 and 6 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.

---

[4]The use of the factorization (5) and of the FFT in Subsections 5.2 and 6.1 can be found in [27]

Table 2: Time (in ms) required for the computation of the Bernstein matrix $\mathcal{B}(\boldsymbol{u})$ by using our method and Ray and Nataraj's (R.N.) method

| Test case | $n$ | $l$ | unit box domain | | | general box domain | | |
|---|---|---|---|---|---|---|---|---|
| | | | our method | R.N. method | reduction | our method | R.N. method | reduction |
| L. V. 3 | 3 | $(1, 2, 2)$ | 0.05 | 0.54 | 90.74 | 0.06 | 0.66 | 90.91 |
| R. D. 3 | 3 | $(1, 2, 1)$ | 0.05 | 0.46 | 89.13 | 0.05 | 0.63 | 92.06 |
| L. V. 4 | 4 | $(1, 2, 2, 2)$ | 0.11 | 1.47 | 92.52 | 0.12 | 2.20 | 94.55 |
| Cap 4 | 4 | $(1, 1, 3, 3)$ | 0.13 | 2.95 | 95.59 | 0.20 | 3.47 | 94.24 |
| Wrig 5 | 5 | $(1, 1, 1, 1, 2)$ | 0.11 | 1.46 | 92.47 | 0.12 | 1.89 | 93.65 |
| Reim 5 | 5 | $(6, 6, 6, 6, 6)$ | 82.18 | 63 | -30.44 | 84.02 | 69.75 | -20.46 |
| Mag 6 | 6 | $(2, 2, 2, 2, 2, 2)$ | 0.64 | 6.34 | 89.91 | 0.66 | 6.76 | 90.24 |
| But 6 | 6 | $(1, 2, 2, 3, 1, 1)$ | 0.29 | 5.17 | 94.39 | 0.29 | 5.32 | 94.55 |
| Reim 6 | 6 | $(7, 7, 7, 7, 7, 7)$ | 1715.98 | 498.22 | -244.42 | 1802.06 | 507.26 | -255.25 |
| Mag 7 | 7 | $(2, 2, 2, 2, 2, 2, 2)$ | 2.12 | 6.30 | 66.35 | 2.43 | 7.30 | 66.71 |
| Reim 7 | 7 | $(8, 8, 8, 8, 8, 8, 8)$ | 34309.29 | 9859.62 | -247.98 | 38829.24 | 9725.18 | -299.27 |

## 5. Tensorial subdivision in matrix form

### 5.1. A matricial representation of the de Casteljau algorithm

We assume that $n \geq 2$ and we are given a polynomial $p$ as in (6) with its Bernstein coefficients over the unit box $\boldsymbol{u}$ arranged in a matrix $\mathcal{B}(\boldsymbol{u})$ as in (22). The box $\boldsymbol{u}$ is now subdivided in the $s$-th coordinate direction, $s \in \{1, \ldots, n\}$, into the subboxes $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. We arrange their Bernstein coefficients likewise in matrices $\mathcal{B}(\boldsymbol{u}_1)$ and $\mathcal{B}(\boldsymbol{u}_2)$ of the form (22). We perform the de Casteljau

algorithm, see Section 3, with the matrix $\mathcal{B}(\boldsymbol{u})$ instead of the Bernstein patch $B(\boldsymbol{u})$ and get then $l_s+1$ matrices $\mathcal{B}^{[\nu]}(\boldsymbol{u})$. In the $\nu$-th step, $\mathcal{B}^{[\nu]}(\boldsymbol{u})$ is obtained as follows: The first $\nu$ rows are identical with the first $\nu$ rows in the $(\nu-1)$-th step and the remaining rows are obtained as a convex combination of two consecutive rows. The Bernstein coefficient matrix $\mathcal{B}(\boldsymbol{u}_1)$ over the subbox $\boldsymbol{u}_1$ is provided by the last matrix $\mathcal{B}^{[l_s]}(\boldsymbol{u})$. The $i$-th row, $i = 1, \ldots, l_s + 1$, of the Bernstein coefficient matrix $\mathcal{B}(\boldsymbol{u}_2)$ over the subbox $\boldsymbol{u}_2$ is equal to the last row in $\mathcal{B}^{[l_s-i+1]}(\boldsymbol{u})$. In matrix language, these manipulations are described by pre-multiplication of the matrix $\mathcal{B}(\boldsymbol{u})$ by the following matrices. Without loss of generality we subdivide $\boldsymbol{u}$ in the direction perpendicular to the first coordinate direction $(s = 1)$. In the case of subdivision in a general direction $s$, $s \in \{2, \ldots, n\}$, we firstly apply $(s-1)$-fold cyclic ordering that is defined in Subsection 4.3. For simplicity we assume that $l_1 = \cdots = l_n = \kappa$. For $\mu = 1, 2, \ldots, \kappa$ we define the bidiagonal matrices $L_\mu$ and $R_\mu \in \mathbb{R}^{\kappa+1,\kappa+1}$ ($\delta$ denotes the Kronecker delta) as follows

$$(L_\mu)_{ij} := \left\{ \begin{array}{ll} \delta_{i,j}, & \text{if } i = 0, \ldots, \kappa - \mu, \\ 1 - \lambda, & \text{if } j = i - 1, \\ \lambda, & \text{if } j = i, \\ 0, & \text{otherwise,} \end{array} \right\} \text{for } i = \kappa - \mu + 1, \ldots, \kappa, \tag{35}$$

and

$$(R_\mu)_{ij} := \left\{ \begin{array}{ll} 1 - \lambda, & \text{if } j = i, \\ \lambda, & \text{if } j = i + 1, \\ 0, & \text{otherwise,} \\ \delta_{i,j}, & \text{if } i = \mu, \ldots, \kappa. \end{array} \right\} \text{for } i = 0, \ldots, \mu - 1, \tag{36}$$

Let $S \in \mathbb{R}^{\kappa+1,\kappa+1}$ be the permutation matrix defined by

$$(S)_{ij} := \begin{cases} 1, & \text{if } i = \kappa - j + 1, \\ 0, & \text{otherwise.} \end{cases} \tag{37}$$

Then we obtain the Bernstein coefficient matrices over $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ by

$$\begin{align} \mathcal{B}(\boldsymbol{u}_1) &= L_1 L_2 \cdots L_\kappa \mathcal{B}(\boldsymbol{u}), \tag{38} \\ \mathcal{B}(\boldsymbol{u}_2) &= R_1 R_2 \cdots R_\kappa \mathcal{B}(\boldsymbol{u}). \tag{39} \end{align}$$

Equation (39) can be derived from equation (38) as follows: First we map $x_1$ to $1 - x_1$ and keep the other variables. By the identity $B_i^{(d)}(1 - x_1) = B_{d-i}^{(d)}(x_1)$, see (8), this corresponds to forming $S\mathcal{B}(\boldsymbol{u})$. Now we apply (38) with the subdivision point $1 - \lambda$ and get back (again by multiplying the resulting matrix by the matrix $S$). Note that this procedure corresponds to the relation $R_\mu(\lambda) = SL_\mu(1 - \lambda)S$, $\mu = 1, \ldots, \kappa$.

### 5.2. A novel method for the calculation of the Bernstein coefficients on sub-boxes

a) On the left subbox

We define the lower triangular matrix $L^\dagger \in \mathbb{R}^{\kappa+1,\kappa+1}$ by

$$(L^\dagger)_{ij} := \begin{cases} \binom{i}{j}\lambda^j(1 - \lambda)^{i-j}, & i \geq j, \\ 0, & \text{otherwise.} \end{cases} \tag{40}$$

Then the factorization

$$L^\dagger = \prod_{\mu=1}^{\kappa} L_\mu \tag{41}$$

holds [1, p.10121]. It is easy to see that

$$L^\dagger = D(1 - \lambda)PD(\frac{\lambda}{1 - \lambda}). \tag{42}$$

Then by using (38), (41), and (42), $\mathcal{B}(\boldsymbol{u}_1)$ can be represented as

$$\mathcal{B}(\boldsymbol{u}_1) = D(1 - \lambda)PD(\frac{\lambda}{1 - \lambda})\mathcal{B}(\boldsymbol{u}). \tag{43}$$

b) On the right subbox

We define the upper triangular matrix $R^\dagger \in \mathbb{R}^{\kappa+1,\kappa+1}$ by

$$(R^\dagger)_{ij} := \begin{cases} \binom{\kappa-i}{j-i}\lambda^{j-i}(1 - \lambda)^{\kappa-j}, & j \geq i, \\ 0, & \text{otherwise.} \end{cases} \tag{44}$$

Then, we get

$$R^\dagger = \prod_{\mu=1}^{\kappa} R_\mu. \tag{45}$$

The matrix $R^\dagger$ can also be factorized as

$$R^\dagger = (1-\lambda)^\kappa D(\frac{1}{\lambda}) P' D(\frac{\lambda}{1-\lambda}), \tag{46}$$

where the upper triangular matrix $P'$ is defined as

$$(P')_{ij} := \begin{cases} \binom{\kappa-i}{j-i}, & j \geq i, \\ 0, & \text{otherwise.} \end{cases} \tag{47}$$

By using the permutation matrix $S$, see (37), and the symmetry of the binomial coefficients, we easily see that $P' = SPS$. By (4), we factorize $P'$ as follows

$$P' = \prod_{\mu=1}^{\kappa} K'_\mu, \tag{48}$$

with the matrices $K'_\mu := SK_\mu S$, $\mu = 1, \ldots, \kappa$. After substituting (46) in (39) and employing (45), we get

$$\mathcal{B}(\boldsymbol{u}_2) = (1-\lambda)^\kappa D(\frac{1}{\lambda}) P' D(\frac{\lambda}{1-\lambda}) \mathcal{B}(\boldsymbol{u}). \tag{49}$$

*5.3. Comparison of the amount of arithmetic operations for the several methods*

In the following comparison of the amount of arithmetic operations for the several methods, cf. Tables 3 and 4, we are giving only the terms which are depending on $\kappa$ or $n$ and neglect the trivial multiplications. The application of the de Casteljau algorithm, see Section 3, requires for the computation of both $\mathcal{B}(\boldsymbol{u}_1)$ and $\mathcal{B}(\boldsymbol{u}_2)$ $\frac{\kappa(\kappa+1)^n}{2}$ additions and $\kappa(\kappa+1)^n$ multiplications. The same amount needs the calculation of $\mathcal{B}(\boldsymbol{u}_1)$ by (38) or of $\mathcal{B}(\boldsymbol{u}_2)$ by (39) because they are merely matrical representations of this algorithm. The amount of arithmetic operations for each of the computations of $\mathcal{B}(\boldsymbol{u}_1)$ and $\mathcal{B}(\boldsymbol{u}_2)$ according to (43) and (49) consists of $\frac{\kappa(\kappa+1)^n}{2}$ additions for the factorizations (4), (48) and $2\kappa(\kappa+1)^{n-1} + 2\kappa$ and $(\kappa+1)^{n-1}(2\kappa+1) + 3\kappa$ multiplications / divisions for the computation of the diagonal matrices and forming the products with them in (43) and (49), respectively. If only one of both patches is needed, (43) and (49) are superior to the de Casteljau algorithm for $\kappa > 1$ (if $\kappa = 2$, (49) for $n \geq 3$), see Table 4. If both patches are

23

needed the amount can be reduced by noting that both formulae share common factors, e.g., $D\left(\frac{\lambda}{1-\lambda}\right)\mathcal{B}(\boldsymbol{u})$, and that the last row in $\mathcal{B}(\boldsymbol{u}_1)$ and the first row of $\mathcal{B}(\boldsymbol{u}_2)$ are identical so that we can shorten the matrices $D(\frac{1}{\lambda})$, $D(\frac{\lambda}{1-\lambda})$, and $P'$ in (49) by deletion of their first rows and columns. The amount of arithmetic operations for both formulae consists then of $\kappa^2(\kappa+1)^{n-1}$ additions and $3\kappa(\kappa+1)^{n-1}+4\kappa$ multiplications / divisions. Therefore, the use of the de Casteljau algorithm is preferable for $\kappa \leq 3$, whereas for $\kappa \geq 4$ the use of (43) and (49) is superior (if $\kappa = 4$, then $n \geq 3$), see Table 4.

Table 3: Comparison between de Casteljau algorithm and the computation according to (43) and (49)

| Calculation of | number of additions | number of multiplications/div. |
|---|---|---|
| de Casteljau algorithm | $\kappa\frac{(\kappa+1)^n}{2}$ | $\kappa(\kappa+1)^n$ |
| (43) | $\kappa\frac{(\kappa+1)^n}{2}$ | $2\kappa(\kappa+1)^{n-1}+2\kappa$ |
| (49) | $\kappa\frac{(\kappa+1)^n}{2}$ | $(\kappa+1)^{n-1}(2\kappa+1)+3\kappa$ |
| (43) and (49) together | $\kappa^2(\kappa+1)^{n-1}$ | $3\kappa(\kappa+1)^{n-1}+4\kappa$ |

Table 4: The superiority of using (43) and (49) over the use of the de Casteljau algorithm for the computation of the Bernstein patches over $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$

| Calculation of | $\kappa$ | number of variables |
|---|---|---|
| (43) | $\kappa \geq 2$ | $n \geq 2\ (\kappa = 2),\ \forall\, n\ (\kappa > 2)$ |
| (49) | $\kappa \geq 2$ | $n \geq 3\ (\kappa = 2),\ \forall\, n\ (\kappa \geq 3)$ |
| (43) and (49) together | $\kappa \geq 4$ | $n \geq 3\ (\kappa = 4),\ \forall\, n\ (\kappa > 4)$ |

**Remark 5.1.** *If $\boldsymbol{u}'$ is a subbox of $\boldsymbol{u}$ then we can use repeated subdivision and the above methods to compute $B(\boldsymbol{u}')$. We consider first the case that $\boldsymbol{u}'$ is contained in the interior of $\boldsymbol{u}$. Then we are needing two subdivisions in each*

*coordinate direction, where in each coordinate direction the Bernstein patches over exactly one left and one right subbox are required. So we may use* (43) *as well as* (49) *in each direction. This procedure (and therefore the de Casteljau algorithm, too) is inferior to the use of our method for the computation of $B(\boldsymbol{u}')$. If $\boldsymbol{u}'$ has common faces with $\boldsymbol{u}$ then fewer subdivisions are required and the superiority of our method may be lost. In the extreme case that $\boldsymbol{u}'$ has n faces of dimension $n-1$ in common with $\boldsymbol{u}$, only one subdivision in each coordinate direction is necessary. In this case, our method is superior to the de Casteljau algorithm for $\kappa \geq 4$ (taking Remark 4.2 into account); otherwise the de Casteljau algorithm is preferable. The use of* (43) *and* (49) *is always superior to our method.*

### 5.4. Numerical examples

We have run all the 13 test problems listed in Appendix 1 (but in contrast to the results presented in Table 2 without the variant designed in Appendix 2 for the three Reimer problems). The results in Table 5 are in close agreement with our findings in Table 4. However, the single use of (43) and (49) is not superior to the use of the de Casteljau algorithm to such an extend like the superiority of our method to Ray and Nataraj′s method.

Table 5: Time (in ms) required for the computation of the Bernstein matrices $\mathcal{B}(\boldsymbol{u}_1)$ and $\mathcal{B}(\boldsymbol{u}_2)$ given $\mathcal{B}(\boldsymbol{u})$ by using the de Casteljau algorithm, (43), and (49)

| Test case | $n$ | $l$ | de Casteljau algorithm | (43) | (49) | (43) and (49) |
|---|---|---|---|---|---|---|
| Booth | 2 | $(2,2)$ | 0.0531 | 0.0455 | 0.0736 | 0.0780 |
| Himmelblau | 2 | $(4,4)$ | 0.1391 | 0.1038 | 0.1047 | 0.1845 |
| L. V. 3 | 3 | $(1,2,2)$ | 0.0365 | 0.0331 | 0.0336 | 0.0588 |
| R. D. 3 | 3 | $(1,2,1)$ | 0.0358 | 0.0351 | 0.0357 | 0.0564 |
| L.V. 4 | 4 | $(1,2,2,2)$ | 0.0836 | 0.0565 | 0.0584 | 0.0845 |
| Cap 4 | 4 | $(1,1,3,3)$ | 0.0470 | 0.0461 | 0.0465 | 0.0736 |
| Wrig 5 | 5 | $(1,1,1,1,2)$ | 0.0465 | 0.0435 | 0.0462 | 0.0691 |
| Reim 5 | 5 | $(6,6,6,6,6)$ | 0.3627 | 0.1881 | 0.1915 | 0.3380 |
| Mag 6 | 6 | $(2,2,2,2,2,2)$ | 0.2104 | 0.1931 | 0.1971 | 0.3127 |
| But 6 | 6 | $(1,2,2,3,1,1)$ | 0.0779 | 0.0736 | 0.0784 | 0.1086 |
| Reim 6 | 6 | $(7,7,7,7,7,7)$ | 0.7500 | 0.3030 | 0.3273 | 0.6151 |
| Mag 7 | 7 | $(2,2,2,2,2,2,2)$ | 0.5243 | 0.5035 | 0.5137 | 0.7664 |
| Reim 7 | 7 | $(8,8,8,8,8,8,8)$ | 0.9572 | 0.4210 | 0.4522 | 0.5321 |

## 5.5. Changing the underlying box

We turn to the situation that we have to change the box over which we want to find an enclosure for the range of the given polynomial $p$. For the computation of the Bernstein patch over the new box we can use the Bernstein patch over the original box without recourse to the power representation of $p$ and apply again the procedure in Subsection 4.4. The procedure described below applies also to the case that we know the Bernstein patch

$B(\boldsymbol{u})$ of $p$ over $\boldsymbol{u}$ and want to find the power representation of $p$. Without loss of generality we assume that we change the domain of $x_s, s \in \{2, \ldots, n-1\}$, from $[0,1]$ to $[\underline{x}_s, \overline{x}_s]$. Then the new box is $\boldsymbol{u}^\star := [0,1]^{s-1} \times [\underline{x}_s, \overline{x}_s] \times [0,1]^{n-s}$. To obtain $\mathcal{B}(\boldsymbol{u}^\star)$, we distinguish the following cases:

Case (i): $\underline{x}_s > 0$[5]: Firstly, we affinely map $[0,1]$ to $[0, \overline{x}_s]$ by introducing the new variable

$$\overline{x}_s x_s. \tag{50}$$

After substituting (50) in (6), we get a polynomial, $p_s$ say, over $\tilde{\boldsymbol{u}} := [0,1]^{s-1} \times [0, \overline{x}_s] \times [0,1]^{n-s}$. By some arrangements, using the binomial expansion, equating the resulting formula with the Bernstein representation of $p$ over $[0,1]^n$, and solving for the Bernstein coefficients, its Bernstein coefficient matrix $\mathcal{B}(\tilde{\boldsymbol{u}})$ is obtained as (we first apply the cyclic ordering $s-1$ times starting from $\mathcal{B}(\boldsymbol{u})$)

$$\mathcal{B}(\tilde{\boldsymbol{u}}) = D_s\left(\frac{\overline{x}_s - 1}{\overline{x}_s}\right) P_s D_s\left(\frac{1}{\overline{x}_s - 1}\right) \mathcal{B}(\boldsymbol{u}). \tag{51}$$

By using (49) (starting from $\mathcal{B}(\tilde{\boldsymbol{u}})$) with $\lambda = \frac{\underline{x}_s}{\overline{x}_s}$ we get $\mathcal{B}(\boldsymbol{u}^\star)$.

Case (ii): $\overline{x}_s < 0$: We replace in (50) and (51) $\overline{x}_s$ by $\underline{x}_s$ and affinely map in this way $[0,1]$ to $[\underline{x}_s, 0]$ and compute $\mathcal{B}(\tilde{\boldsymbol{u}})$ from which we get $\mathcal{B}(\boldsymbol{u}^\star)$ by using (43) with $\lambda = \frac{\overline{x}_s}{\underline{x}_s}$.

Case (iii): $\underline{x}_s < 0 < \overline{x}_s$: The matrix $\Lambda(\boldsymbol{u})$ is obtained from $A^\star$ as

$$\Lambda(\boldsymbol{u}) = (D'_n(\cdots (D'_2(D'_1 A^\star)^c)^c \cdots )^c)^c, \tag{52}$$

where $D'_s$ is defined in Remark 4.2. Let $\mathcal{B}'_s(\boldsymbol{u}) = (b'_i)$ be the $(l_s + 1) \times \prod\limits_{\substack{w=1, \\ w \neq s}}^{n} (l_w + 1)$ matrix which is computed by using (24) and (52) as

$$\mathcal{B}'_s(\boldsymbol{u}) := (D'_s)^{-1} P_s^{-1} \mathcal{B}(\boldsymbol{u}). \tag{53}$$

The coefficients $b'_i$ appear also in the representation of the polynomial $p$ as

$$p(x) = \sum_{i^{(s)}=0}^{l^{(s)}} B_{i^{(s)}}(x^{(s)}) \sum_{i_s=0}^{l_s} b'_{i_1, \ldots, i_s, \ldots, i_n} x_s^{i_s}, \tag{54}$$

---

[5]If $\overline{x}_s = 1$ we may apply subdivision with $\lambda = \underline{x}_s$ and obtain $\mathcal{B}(\boldsymbol{u}^\star)$ from (49) as $\mathcal{B}(\boldsymbol{u}_2)$.

where the superscript $(s)$ denotes that in $i, l,$ and $x$ the $s$-th component is put as zero. Then $\mathcal{B}(\boldsymbol{u}^\star)$ is given by

$$\mathcal{B}(\boldsymbol{u}^\star) = P_s D'_s D_s \left(\frac{\overline{x}_s - \underline{x}_s}{\underline{x}_s}\right) P_s^T D_s(\underline{x}_s) \mathcal{B}'_s(\boldsymbol{u}). \tag{55}$$

Let $D_s^\star := D_s(\underline{x}_s)(D'_s)^{-1}$ and $D_s^{\star\star} := D'_s D_s\left(\frac{\overline{x}_s - \underline{x}_s}{\underline{x}_s}\right)$, then $B(\boldsymbol{u}^\star)$ is obtained by using (53) and (55) as follows

$$\mathcal{B}(\boldsymbol{u}^\star) = P_s D_s^{\star\star} P_s^T D_s^\star P_s^{-1} \mathcal{B}(\boldsymbol{u}). \tag{56}$$

Let the bidiagonal matrix $K_\mu^{\star s}$, $\mu = 1, \dots, l_s$, be given by

$$(K_\mu^{\star s})_{ij} := \begin{cases} 1, & \text{if } i = j, \\ -1, & \text{if } i = j + 1, \ l_s - \mu \le j \le l_s - 1, \\ 0, & \text{otherwise.} \end{cases} \tag{57}$$

Then $P_s^{-1}$ allows the factorization

$$P_s^{-1} = \prod_{\mu=1}^{l_s} K_\mu^{\star s}. \tag{58}$$

By substituting (4) and (58) in (56) the number of the arithmetic operations can be reduced.

## 6. Extension to interval polynomials

In this section, we extend our results to polynomials with intervals as coefficients. For this purpose, we recall the definition of addition, subtraction, and multiplication of intervals (division is not used here).

**Definition 6.1** (Interval arithmetic operations). *Let $[x] = [\underline{x}, \overline{x}]$ and $[y] = [\underline{y}, \overline{y}] \in \mathbb{IR}$. Then we have*

$$\begin{aligned} [x] + [y] &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}], \\ [x] - [y] &= [\underline{x} - \overline{y}, \overline{x} - \underline{y}], \\ [x] * [y] &= [\min\{\underline{x} * \underline{y}, \underline{x} * \overline{y}, \overline{x} * \underline{y}, \overline{x} * \overline{y}\}, \max\{\underline{x} * \underline{y}, \underline{x} * \overline{y}, \overline{x} * \underline{y}, \overline{x} * \overline{y}\}]. \end{aligned}$$

For properties of the interval arithmetic operations and the extension to interval vectors and matrices the reader is referred to [12] and [17].

Let $[a_i] = [\underline{a}_i, \overline{a}_i] \in \mathbb{IR}, i = 0, \ldots, l$, be given. Then the set of polynomials

$$\boldsymbol{p}(x) = \sum_{i=0}^{l} [a_i] x^i \tag{59}$$

is called an *interval polynomial.* In the robust control area there is a vast literature on the origin and use of interval polynomials, see [4], [32] and the references therein.

Replacing in an algorithm real numbers by intervals and the real arithmetic operations by the corresponding interval arithmetic operations results in general in gross overestimation due to the so-called dependency problem of interval arithmetic, see [12, p.38], [17, pp.16-17]. Rather surprisingly, the performance of the algorithms developed in this paper for the computation of the Bernstein coefficients (with the exception of the computation of the Bernstein coefficients over a general box not restricted to the nonnegative orthant of $\mathbb{R}^n$) provides the set of the Bernstein coefficients of all members of the family without overestimation. The reason is that the real matrices involved in the methods are (entry-wise) nonnegative, so that in each step the infimum and supremum of the resulting interval matrices are attained by respective point problems, see [17, Proposition 3.1.5, (15d)] in this regard. This means that the sequence of interval computations can be reduced to two sequences of real computations involving the endpoints of the coefficient intervals. Since this argument applies to most of the problems we can keep the discussion of the performance of the methods in interval arithmetic to a minimum. The above formulation 'without overestimation' means that we obtain the smallest interval matrix which contains the Bernstein coefficient matrices of all members of the family. So, in general, not each coefficient matrix contained in the resulting interval matrix is realizable as the Bernstein coefficient matrix of any polynomial in the interval family, see Example 6.1 below.

*6.1. Calculation of the Bernstein coefficients of an interval polynomial*

Let $\boldsymbol{p}$ be the interval polynomial given in (59) and $\boldsymbol{x}$ be a box in $\mathbb{R}^n$. To apply the method introduced in Section 4 we proceed as follows. Firstly, we

replace the matrix $A$ on the right-hand side of (29) by the interval matrix $\boldsymbol{A}$ comprising the interval coefficients of $p$. Then we perform (29) in interval arithmetic yielding the matrix $\boldsymbol{A}^{\star}$. This step does not suffer from overestimation if $\boldsymbol{x}$ is restricted to the nonnegative orthant of $\mathbb{R}^n$ because in this case all the matrices being involved are nonnegative. If this assumption is not fulfilled, we are able to compute the Bernstein coefficients of all members of the family without overestimation by direct computation after a suitable reformulation, see below. Now we perform our method in interval arithmetic starting from $\boldsymbol{A}^{\star}$. This step itself produces no overestimation since all the matrices involved continue to be nonnegative. So if $\boldsymbol{x}$ is restricted to the nonnegative orthant we have all together no overestimation and obtain from the entries of the resulting interval matrix exactly the set of the Bernstein coefficients of all members of the interval family.

We now turn to the case that the box $\boldsymbol{x}$ is not restricted to the nonnegative orthant. The representation (7) can be extended to a representation of $p$ over the box $\boldsymbol{x}$. Then the $i$-th Bernstein coefficient of $p$ is given by

$$
b_i^{(l)} = \sum_{j=0}^{i} \frac{\binom{i}{j}}{\binom{l}{j}} (\overline{x} - \underline{x})^j \sum_{t=j}^{l} \binom{t}{j} \underline{x}^{t-j} a_t, \ 0 \le i \le l. \tag{60}
$$

If in the case of an interval polynomial we replace the real coefficients by the respective intervals we will be faced by overestimation of the set of the Bernstein coefficients of all members of the interval family since each coefficient appears more than once in the representation. Following a reformulation of (60) in the univariate case [24], we may represent the Bernstein coefficients in the following form

$$
b_{i_1,\ldots,i_n}^{(l)} = \sum_{t_1=0}^{l_1} \sum_{t_2=0}^{l_2} \cdots \sum_{t_n=0}^{l_n} a_{t_1,\ldots,t_n} \chi_{t_1}^{i_1} \chi_{t_2}^{i_2} \cdots \chi_{t_n}^{i_n}, \tag{61}
$$

where

$$
\chi_{t_s}^{i_s} := \sum_{k_s=0}^{\min(i_s,t_s)} \underline{x}_s^{t_s-k_s} \frac{(\overline{x}_s - \underline{x}_s)^{k_s} \binom{i_s}{k_s} \binom{t_s}{k_s}}{\binom{l_s}{k_s}} \tag{62}
$$

and $t_s = 0, \ldots, l_s, \ s = 1, \ldots, n.$

The advantage of this representation is that now each polynomial coefficient appears only once in the formula of each Bernstein coefficient. If we replace now the real coefficients by the respective interval coefficients then no overestimation will occur, see, e.g., [12, Theorem 6.2], [17, Corollary 1.4.4].

We define for $s = 1, \ldots, n$ the matrices $\mathcal{U}_s \in \mathbb{R}^{l_s+1, l_s+1}$ by

$$(\mathcal{U}_s)_{ij} := \chi_j^i, \; i, j = 0, \ldots, l_s. \tag{63}$$

Then by using the matrices $D'_s$, see Remark 4.2, and $Q_s$, see (28), the matrices $\mathcal{U}_s$ can also be represented as

$$\mathcal{U}_s = P_s D'_s Q_s. \tag{64}$$

The Bernstein patch of $p$ over $\boldsymbol{x}$ (arranged as in (22)) can be represented as

$$(\mathcal{U}_n(\ldots (\mathcal{U}_2(\mathcal{U}_1 A)^c)^c \ldots)^c)^c. \tag{65}$$

We then replace in (65) the real matrix $A$ by the interval matrix $\boldsymbol{A}$ and obtain the set of the Bernstein coefficients of all the member of the interval family.

**Example 6.1.** *Let $\boldsymbol{p}$ be the bivariate interval polynomial (59) with $l = (1, 2)$ and coefficients arranged in the interval matrix $\boldsymbol{A}$ as follows*

$$\boldsymbol{A} = \begin{bmatrix} [-1, 1] & [1, 3] & [-1, 0] \\ [-1, 2] & [1, 2] & [0, 2] \end{bmatrix}. \tag{66}$$

*The application of our method in interval arithmetic yields the interval matrix*

$$\boldsymbol{\mathcal{B}}(\boldsymbol{u}) := \begin{bmatrix} [-1, 1] & [\frac{-1}{2}, \frac{5}{2}] & [-1, 4] \\ [-2, 3] & [-1, \frac{11}{2}] & [-1, 10] \end{bmatrix}. \tag{67}$$

*Each interval entry of this matrix provides the exact range of the respective Bernstein coefficient for all members of the interval family. Denote by q the polynomial which has the Bernstein patch*

$$\begin{bmatrix} -1 & \frac{-1}{2} & 4 \\ & & \\ 3 & -1 & 10 \end{bmatrix} \tag{68}$$

*which is contained in $\mathcal{B}(\boldsymbol{u})$. Then q is (uniquely) represented in the Bernstein basis over $\boldsymbol{u}$ and in the power basis as*

$$\begin{aligned} q(x) &= -(1-x_1)(1-x_2)^2 + \frac{-1}{2}2(1-x_1)x_2(1-x_2) + 4(1-x_1)x_2^2 \\ &\quad + 3x_1(1-x_2)^2 - 2x_1x_2(1-x_2) + 10x_1x_2^2 \\ &= -1 + x_2 + 4x_2^2 + 4x_1 - 9x_1x_2 + 11x_1x_2^2. \end{aligned}$$

*The representation in the power basis shows that the polynomial q is not a member of the interval family $\boldsymbol{p}$. So not all points in $\mathcal{B}(\boldsymbol{u})$ are realizable as Bernstein coefficients of a polynomial contained in $\boldsymbol{p}$.*

*6.2. Tensorial subdivision for an interval polynomial*

The Bernstein coefficient matrices $\mathcal{B}(\boldsymbol{u}_1)$ and $\mathcal{B}(\boldsymbol{u}_2)$ of $\boldsymbol{p}$ over the subboxes $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ can be obtained by performance of (43) and (49) in interval arithmetic. Since all matrices involved are nonnegative we get the Bernstein coefficients over both subboxes of all members of the polynomial family without overestimation.

## 7. Conclusions

We have investigated new methods for the computation of the Bernstein coefficients over boxes. Our analysis of the amount of arithmetic operations as well as the test examples show that our method outperforms two methods known from literature to a large extend. The numerical examples further show that for the computation of the Bernstein coefficients on subboxes generated by subdivision the superiority of the new methods (based on formulae (43) and (49)) to the de Casteljau algorithm is not so pronounced. Also, we have considered the case that the coefficients of the given polynomial are subject to uncertainties. In this case, we were able to compute the Bernstein coefficients of all members of the polynomial family without overestimation. In this paper, we have compared the diverse methods on the basis of their

computational work. The comparison with respect to stability against rounding errors will be investigated in a future paper. The results of Section 4 are applied in [30] to compute an upper bound for the modulus and an enclosure of the range of a multivariate complex polynomial over a complex box. A matrix method for the evalaution of multivariate polynomials in the tensorial Bernstein representation can be found in [27].

## Appendix 1: Description of the test problems

In the following, we list the abbreviated and full names, the polynomials, the boxes $\boldsymbol{x}$, and the dimensionality of the problems used in our tests. Test cases nos. $1, 2$ are taken from [14], nos. $3 - 11$ from [23], where the number appended to the abbreviation refers to the number of the variables chosen.

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

$$p(x_1, x_2) = 74 - 38x_1 - 34x_2 + 5x_1^2 + 5x_2^2 + 8x_1x_2,$$

where $x_i \in [-10, 10]$, $i = 1, 2$.

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

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

where $x_i \in [-5, 5]$, $i = 1, 2$.

3. **L. V. 3:** A neural network modeled by an adaptive Lotka-Volterra system, $n = 3$,

$$p(x_1, x_2, x_3) = x_1x_2^2 + x_1x_3^2 - 1.1x_1 + 1,$$

where $x_i \in [-1.5, 2]$, $i = 1, 2, 3$.

4. **R. D. 3:** A 3-dimentional reaction diffusion problem, $n = 3$,

$$p(x_1, x_2, x_3) = x_1 - 2x_2 + x_3 + 0.835634534x_2(1 - x_2),$$

where $x_i \in [-5, 5]$, $i = 1, 2, 3$.

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

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

where $x_i \in [-2, 2]$, $i = 1, \ldots, 4$.

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

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

where $x_i \in [-0.5, 0.5]$, $i = 1, \ldots, 4$.

,

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

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

where $x_i \in [-5, 5]$, $i = 1, \ldots, 5$.

8. **Reim 5-7:** The $n$-dimensional system of Reimer, $n = 5, 6, 7$,

$$p(x_1, \ldots, x_n) = -1 - 2 \sum_{i=1}^{n} (-1)^i x_i^{n+1},$$

where $x_i \in [-1, 1]$, $i = 1, \ldots, n$, for $n = 5, 7$ and $x_i \in [-5, 5]$, $i = 1, \ldots, 6$, if $n = 6$.

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

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

where $x_i \in [-5, 5]$, $i = 1, \ldots, 6$.

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

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

where $x_1 \in [-1, 0], x_2 \in [-0.1, 0.9], x_3 \in [-0.1, 0.5], x_4 \in [-1, -0.1]$, $x_5 \in [-0.1, -0.05], x_6 \in [-0.1, -0.03]$.

11. **Mag 7:** System of magnetism in physics, $n = 7$,

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_4^2 + 2x_5^2 + 2x_6^2 + 2x_7^2 - x_1,$$

where $x_i \in [-5, 5]$, $i = 1, \ldots, 7$.

## Appendix 2: A variant of our method for polynomials which are linear combinations of univariate monomials

We assume that the polynomial $p$ is a sum of univariate monomial terms and explain the procedure on the basis of the Reimer problems. We firstly compute the Bernstein coefficients of each univariate polynomial, e.g.,

$$g(x_1) = 2x_1^w, w = 5, 6, 7.$$

Then, we generate $w$ matrices, which all contain the Bernstein coefficients of the polynomial

$$g_{new}(x) = g(x_1)$$

in such a way that $b_{i_1,\ldots,i_n}(g_{new}) = b_{i_1}(g)$, for all $0 \leq i_s \leq \kappa, s = 1, \ldots, n$. Multiply the coefficients of the matrices which result for $s = 1, \ldots, n$ by $-1$ if the coefficient of the corresponding monomial term has a negative sign. After that the order of the variables in the matrix of the term that contains $x_s$ only, is $(x_s, x_{s+1}, \ldots, x_n, x_1, x_2, \ldots, x_{s-1})$, $s = 2, \ldots, n$. By applying the cyclic ordering $n - s + 1$ times we get the original order of the variables, i.e., $(x_1, x_2, \ldots, x_n)$. By the linearity of the Bernstein coefficients, see (11), we obtain the Bernstein coefficient matrix of the given polynomial $p$. In Table 6 we present the results of the application of the variant to the Reimer problems.

Table 6: Time (in ms) required for the computation of the Bernstein matrix $\mathcal{B}(\boldsymbol{u})$ of the Reimer problems by using the variant of our method and Ray and Nataraj′s (R.N.) method

| Test case | $n$ | $\kappa$ | unit box domain | | | general box domain | | |
|---|---|---|---|---|---|---|---|---|
| | | | our method | R.N. method | reduction | our method | R.N. method | reduction |
| Reim 5 | 5 | $(6, 6, 6, 6, 6)$ | 19.06 | 63 | 69.75 | 19.54 | 64.59 | 69.75 |
| Reim 6 | 6 | $(7, 7, 7, 7, 7, 7, 7)$ | 332.38 | 498.22 | 33.29 | 344.81 | 507.26 | 32.02 |
| Reim 7 | 7 | $(8, 8, 8, 8, 8, 8, 8)$ | 7108.88 | 9859.62 | 27.90 | 7338.59 | 9725.18 | 24.54 |

## References

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

[2] E. Chu, A. George, Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms, CRC Press, Boca Raton, London, New York, 2000.

[3] P.S. Dhabe, P.S.V. Nataraj, A parallel Bernstein algorithm for global optimization based on the implicit Bernstein form, Int. J. Syst. Assur. Eng. Manag. 8 (Suppl. 2) (2017) 1654–1671.

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

[5] H.C. Fischer, Range computation and applications, in: Ullrich C. (Ed.), Contributions to Computer Arithmetic and Self-Validating Numerical Methods, J.C. Baltzer, Amsterdam, 1990, pp. 197–211.

[6] Z.A. Garczarczyk, Parallel schemes of computation for Bernstein coefficients and their application, in: Proc. Int. Conf. Parallel Comput. Electr. Eng. (PARELEC'02), IEEE, Los Alamitos, CA, 2002, pp. 334–339.

[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, C. Jansson, A.P. Smith, Lower bound functions for polynomials, J. Comput. Appl. Math. 157 (1) (2003) 207–225.

[9] J. Garloff, A.P. Smith, Rigorous affine lower bound functions for multivariate polynomials and their use in global optimization, in: Lecture Notes in Management Sciences, vol. 1, Tadbir Institute for Operational Research, System Design and Financial Services, 2008, pp. 199–211; also: Konstanzer Schriften in Mathematik und Informatik vol. 250, University of Konstanz, Konstanz, Germany. http://nbn-resolving.de/urn:nbn:de:bsz:352-opus-58558, 2008.

[10] J. Garloff, A.P. Smith, Preface to the Special Issue on the Use of Bernstein Polynomials in Reliable Computing, Reliab. Comput. 17 (2012) i-vii.

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

[12] R.E. Moore, R.B. Kearfott, M.J. Cloud, Introduction to Interval Analysis, SIAM, Philadelphia, 2009.

[13] C. Muñoz, A. Narkawicz, Formalization of Bernstein polynomials and applications to global optimization, J. Automat. Reason. 51 (2013) 151–196.

[14] P.S.V. Nataraj, M. Arounassalame, A new subdivision algorithm for the Bernstein polynomial approach to global optimization, Int. J. Automat. Comput. 4 (4) (2007) 342–352.

[15] 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.

[16] P.S.V. Nataraj, K. Kotecha, An improved interval global optimization algorithm using higher-order inclusion function forms, J. Global Optim. 32 (1) (2005) 35–63.

[17] A. Neumaier, Interval Methods for Systems of Equations, Encyclopedia Math. Appl. 37, Cambridge University Press, Cambridge, UK, 1990.

[18] B.V. Patil, P.S.V. Nataraj, S. Bhartiya, Global optimization of mixed-integer nonlinear (polynomial) programming problems: the Bernstein polynomial approach, Computing 94 (2-4) (2012) 325–343.

[19] B.V. Patil, P.S.V. Nataraj, An improved Bernstein global optimization algorithm for MINLP problems with application in process industry, Math. Comput. Sci. 8 (2014) 357–377.

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

[21] H. Prautzsch, W. Boehm, M. Paluszny, B*é*zier and B-Spline Techniques, Springer, Berlin, Heidelberg, 2002.

[22] S. Ray, P.S.V. Nataraj, An efficient algorithm for range computation of polynomials using the Bernstein form, J. Global Optim. 45 (3) (2009) 403–426.

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

[24] J. Rokne, Optimal computation of the Bernstein algorithm for the bound of an interval polynomial, Computing. 28 (3) (1982) 239–246.

[25] A.P. Smith, Fast construction of constant bound functions for sparse polynomials, J. Global Optim. 43 (2-3) (2009) 445–458.

[26] 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.

[27] J. Titi, Matrix methods for the tensorial and simplicial Bernstein forms with application to global optimization, University of Konstanz, Konstanz, Germany (2018).

[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, J. Garloff, Matrix methods for the simplicial Bernstein representation and for the evaluation of multivariate polynomials, Appl. Math. Comput. 315 (2017) 246–258.

[30] J. Titi, J. Garloff, Bounds for the range of a complex polynomial over a rectangular region, submitted.

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

[32] M. Zettler, J. Garloff, Robustness analysis of polynomials with polynomial parameter dependency using Bernstein expansion, IEEE Trans. Automat. Control 43 (1998) 425–431.