A Brief Review of a Method for Bounds on Polynomial Ranges over Simplexes

Ralph Baker Kearfott and Dun Liu

University of Louisiana at Lafayette
rbk@louisiana.edu, dxl8898@louisiana.edu

Abstract. We are concerned with tools to find bounds on the range of certain polynomial functions of n variables. Although our motivation and history of the tools are from crisp global optimization, bounding the range of such functions is also important in fuzzy logic implementations. We review and provide a new perspective on one such tool. We have been examining problems naturally posed in terms of barycentric coordinates, that is, over simplexes. There is a long history of using Bernstein expansions to bound ranges of polynomials over simplexes, particularly within the computer graphics community for 1-, 2-, and 3-dimensional problems, with some literature on higher-dimensional generalizations, and some work on use in global optimization. We revisit this work, identifying efficient implementation and practical application contexts, to bound ranges of polynomials over simplexes in dimensions, 2, 3, and higher.

1 Introduction

Finding bounds on the range of a function

$$f(x) = f(x_0, \ldots, x_n) : \mathcal{D} \subseteq \mathbb{R}^{n+1} \to \mathbb{R}$$

of $n+1$ variables $x_0, \ldots, x_n$ over some region $\mathcal{D}$ is an important problem occurring in many forms and contexts. Indeed, evaluating the value of a function over fuzzy inputs involves finding its range over the appropriate non-fuzzy ("crisp") sets (namely, alpha-cuts); such evaluations are central to computing fuzzy outputs based on fuzzy inputs; see [6] for an excellent explanation of this. Finding the exact range (or good numerical approximation thereof), or equivalently, the global optimization problem, is known, for general continuous f and general compact region $\mathcal{D}$, to be NP-hard, and is not NP-hard only for certain special classes of f and $\mathcal{D}$. However, non-sharp bounds, if not too wide, can be a useful basic tool, even in global optimization algorithms. The literature is replete with discussion of the relationship between bounding the range of a function and fuzzy computations, as pointed out, say, in the overviews [8,11], in [24], or the more recent work [22]. (A plethora of references is omitted.) One survey on fuzzy optimization, including its relationship to crisp optimization techniques, is [18].

Various approaches to easily computing usable bounds on ranges, many depending on particular contexts and applications, have been proposed and implemented. A common context is where the region $\mathcal{D}$ is defined by lower and upper bounds $\underline{x}_i$ and $\overline{x}_i$ on each of the variables x_i, that is, $\underline{x}_i \le x_i \le \overline{x}_i$, $0 \le i \le n$. We speak of such regions as a *box*, or *interval vector*, and write

$$\boldsymbol{x} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n) = ([\underline{x}_0, \overline{x}_0], \ldots, [\underline{x}_n, \overline{x}_n]).$$

In this context, simple (or "naive") interval arithmetic, introduced and explained in numerous texts and reviews, such as our relatively recent work [10], is a possibility. However, such bounds are not guaranteed to be sufficiently close to the actual range to be useful; experts in interval arithmetic have extensively studied techniques to obtain the best possible bounds with the minimal amount of work.

1.1 Simplexes

In various applications, range bounds are required, not over a box, but over a *simplex* $\mathcal{S}$. In particular, to within an affine transformation, an n-dimensional simplex in $\mathbb{R}^{n+1}$ is characterized by:

$$\mathcal{S}_c = \left\{ (x_0, \ldots, x_n), \quad x_i \ge 0, \quad \sum_{i=0}^{n} x_i = 1 \right\}, \tag{1}$$

Thus, a simplex can be viewed as a box $\boldsymbol{x} \in \mathbb{R}^{n+1}$ subject to the additional (commonly occurring) constraint $\sum_{i=0}^{n} x_i = 1$. Alternatively, a simplex can be viewed as a volume in $\mathbb{R}^n$ bounded by affine constraints:

$$\mathcal{S} = \left\{ \sum_{i=0}^{n} x_i P_i, \quad x_i \ge 0, \quad P_i \in \mathbb{R}^q, \quad q \ge n, \quad \sum_{i=0}^{n} x_i = 1 \right\}. \tag{2}$$

The P_i in (2) are called the *vertexes* of $\mathcal{S}$, and the x_i are called the *barycentric coordinates* of points in $\mathcal{S}$. (In (1), the vertexes are the coordinate vectors in $\mathbb{R}^{n+1}$.) Simplexes are often specified in terms of their vertexes:

$$\mathcal{S} = \langle P_0, \ldots, P_q \rangle \quad \text{(and usually } q = n\text{)}. \tag{3}$$

When $n = 1$, a simplex corresponds to a line segment, when $n = 2$, a simplex corresponds to a triangle in $\mathbb{R}^3$, while simplexes with $n = 3$ correspond to tetrahedra.

Remark 1. In branch and bound algorithms for global optimization, the simplex $\mathcal{S}$ is partitioned or subdivided in various ways. That is,

Definition 1. *A* subdivision *of a simplex $\mathcal{S}$ is a set of simplexes $\{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ such that*

$$\mathcal{S} = \bigcup_{i=1}^{m} \mathcal{S}_i, \qquad \mathcal{S}_i \cap \mathcal{S}_j \textit{ lies on the boundary of } \mathcal{S}_i \textit{ and } \mathcal{S}_j \textit{ for each } i, j. \tag{4}$$

A common subdivision in this context is *bisection* into two simplexes, $\mathcal{S}_1$ and $\mathcal{S}_2$, by replacing P_i by $\frac{1}{2}(P_i + P_j)$ in $\mathcal{S}_1$ and replacing P_j by $\frac{1}{2}(P_i + P_j)$ in $\mathcal{S}_2$, for some suitably chosen i and j. If $\mathcal{S}$ is identified with $\mathcal{S}_c$ and $f(x_0, \ldots x_n)$ is defined on $\mathcal{S}_c$, we can identify each of $\mathcal{S}_1$ and $\mathcal{S}_2$ with $\mathcal{S}$, but then the domain of f (or coefficients of f if f is a polynomial in the barycentric coordinates) needs to be rescaled to the barycentric coordinates over $\mathcal{S}_1$ and over $\mathcal{S}_2$.

1.2 Bernstein Polynomials

Univariate Bernstein Polynomials If f is a polynomial, or in some cases a rational function, bounds on the ranges can be computed with *Bernstein polynomials*, both over boxes and simplexes. Bernstein polynomials have been analyzed by approximation theorists for over a century. The $d+1$ Bernstein polynomials of degree d form a basis for degree d polynomials over the interval $[0, 1]$, and are given by

$$B_i^{(d)}(t) = \binom{d}{i} t^i (1-t)^{d-i}, \quad 0 \le i \le d, \tag{5}$$

Following the notation in [2, p. 7], the properties of Bernstein polynomials that make them useful are:

$$\left.\begin{array}{ll} \displaystyle\sum_{i=0}^{d} B_i^{(d)}(t) \equiv 1 & \text{(partition of unity).} \\ B_i^{(d)}(t) \ge 0 \quad \text{for } t \in [0,1] & \text{(non-negativity).} \\ B_i^{(d)}(t) = (1-t)B_i^{(d-1)}(t) + tB_{i-1}^{(d-1)}(t) & \text{(recursion).} \end{array}\right\} \tag{6}$$

The general form (5) combined with the partition of unity property makes Bernstein polynomials suitable for representation of functions defined on simplexes. Observe, for $n = 1$, if $x_0 = 1 - t$ and $x_1 = t$, $B_i^{(1)} = x_i$. For a general function f defined on $[0, 1]$, the Bernstein approximation to f by a degree d polynomial is

$$f(t) \approx f_{B^{(d)}}(t) = \sum_{i=0}^{d} f\left(\frac{i}{d}\right) B_i^{(d)}(t). \tag{7}$$

For continuous f, the Bernstein approximation converges relatively slowly to f as the degree n increases, but the f_B are very smooth (without the overshoot in high degree polynomial interpolation and regression, or even in splines), and the convergence is uniform. Furthermore:

1. Because of the partition of unity property (6), $f_{B^{(d)}}(t)$ is a weighted average of the $f(t_i) = f(i/d)$, and therefore lies between the smallest and largest values of f at these $n+1$ equally spaced sample points t_i.
2. Suppose f is a homogeneous degree d polynomial of two variables:

$$f(x_0, x_1) = \sum_{i=0}^{d} a_i x_0^i x_1^{d-i} \quad \text{with the condition } x_0 + x_1 = 1.$$

Then

$$\left.\begin{aligned} &f(x_0,x_1)=\sum_{i=0}^{d}\frac{a_i}{\binom{d}{i}}B_i^{(d)}(t), \quad \text{where } x_0=t,\ x_1=1-t, \text{ and} \\ &\min_{\substack{x_0\in[0,1],\\ x_0+x_1=1}} f(x_0,x_1)=\min_{0\le i\le d}\frac{a_i}{\binom{d}{i}}, \quad \max_{\substack{x_0\in[0,1],\\ x_0+x_1=1}} f(x_0,x_1)=\max_{0\le i\le d}\frac{a_i}{\binom{d}{i}}. \end{aligned}\right\} \tag{8}$$

Property 2 concerns a homogeneous polynomial defined on a one-dimensional simplex ($n = 1$), and shows a quick way of computing exact bounds on the range of that polynomial. For example, a multi-dimensional analog of Property 2 can be used directly in algorithms for quadratic programming problems.

Property 1 can be used in conjunction with (7) for vector valued functions $\vec{f}(t)$, in particular for $\vec{f}(t) \in \mathbb{R}^2$ or $\mathbb{R}^3$. In that case, the resulting curve $\vec{f}(t)$, $0 \le t \le 1$ is called a *Bézier curve*, and the values $\vec{f}(i/d)$, usually given as discrete points in $\mathbb{R}^2$ or $\mathbb{R}^3$ rather than with reference to an underlying function, are called the *control points*. In 1959, Paul de Casteljau at Citroën (and independently, Pierre Bézier at Renault) at developed an ingenious algorithm, based on the above properties of Bernstein polynomials, to evaluate Bézier curves, for computer aided geometric design. The de Casteljau algorithm is now ubiquitous throughout the computer science literature and common in implementations. Furthermore, the de Casteljau algorithm is found in the literature on global optimization. Information about Bézier curves is available in numerous papers and course notes; a somewhat recent review is [3].

In addition to Bézier curves, the de Casteljau algorithm has been studied in the context of bounding ranges of functions. Thus, the plethora of literature on the de Casteljau algorithm within the computer aided geometric design literature is available to designers of global optimization algorithms.

Multivariate Bernstein Polynomials Computer-aided geometric design researchers, as well as global optimization experts and others, have examined generalizations of the definition (5), properties (6) and the approximation (7) to $n > 1$. Tensor products of the $B_i(d)$ are used, with generalizations to both boxes and simplexes. Here, we focus on such generalizations to homogeneous polynomials and n-simplexes in $\mathbb{R}^{n+1}$.

In much of the literature, the multi-dimensional forms are written down with the aid of multi-indexes. Loosely following the notation in [14] for the simplicial extension, we have

Definition 2. *A* multi-index $\vec{i}$ *is simply an* $(n+1)$*-vector of indexes:*

$$\vec{i} = (i_0,\dots,i_n), \quad |\vec{i}| = \sum_{j=0}^{n} i_j, \ \textit{and} \ \ x^{\vec{i}} = \prod_{j=0}^{n} x_j^{i_j}, \quad \binom{d}{\vec{i}} = \frac{d!}{\prod_{j=0}^{n}(i_j!)}.$$

The multi-dimensional simplicial Bernstein functions are defined on the canonical n-simplex[1] $\mathcal{S}_c \in \mathbb{R}^{n+1}$ of (1). With the notation in Definition 2, the n-

[1] not to be confused with the *standard simplex* in $\mathbb{R}^n$ of the literature, defined in terms of (2)

dimensional *simplicial Bernstein basis functions* corresponding to (5) are

$$B_{\vec{i}}^{(d,n)}(\vec{x}) = \binom{d}{\vec{i}} x^{\vec{i}} \quad \text{for } \vec{x} = (x_0, \dots x_n) \in \mathcal{S}_c \text{ and } |\vec{i}| = d, \tag{9}$$

where $\mathcal{S}_c$ is the canonical simplex (1). Observe that (9) corresponds to (5) for $n = 1$, $x_0 = 1 - t$, $x_1 = t$.

Definition 3. *A* homogeneous degree d polynomial *of* $n + 1$ *variables is a polynomial of the form* $f(x_0, \dots x_n) = \sum_{|\vec{i}|=d} a_{\vec{i}} x^{\vec{i}}$*, that is, a polynomial of* $n + 1$ *variables each of whose non-zero terms has total degree* d*.*

Remark 2. Corresponding to (8), if f is a homogeneous polynomial of $n + 1$ variables of the form in Definition 3 defined on $\mathcal{S}_c$, and $\vec{x} = (x_0, \dots, x_n)$, then

$$\left.\begin{array}{l} f(\vec{x}) = \displaystyle\sum_{|\vec{i}|=d} \frac{a_{\vec{i}}}{\binom{d}{\vec{i}}} B_{\vec{i}}^{(d,n)}(t), \quad \text{where } \vec{x} \in \mathcal{S}_c, \text{ and} \\ \displaystyle\min_{\vec{x}\in\mathcal{S}_c} f(\vec{x}) = \min_{|\vec{i}|=d} \frac{a_i}{\binom{d}{\vec{i}}}, \quad \max_{\vec{x}\in\mathcal{S}_c} f(\vec{x}) = \max_{|\vec{i}|=d} \frac{a_i}{\binom{d}{\vec{i}}}. \end{array}\right\} \tag{10}$$

In fact, arbitrary (non-homogeneous) polynomials can be represented in terms of the Bernstein basis; this is necessary for many important problems when using Bernstein techniques in global optimization. The conversion process, for various n has been presented and studied in the literature; for example, an algorithm for the $n = 2$ case is given in [23]. We analyze the conversion for an important case in global optimization in §2 below.

An advantage of the Bernstein polynomial representation is that the de Casteljau algorithm can simultaneously compute coefficients and bounds over each element $\mathcal{S}_i$ of a subdivision (4) of $\mathcal{S}$, with respect to the local barycentric coordinates for $\mathcal{S}_i$, by taking combinations of the coefficients over $\mathcal{S}$. In particular:

Remark 3. if $\mathcal{S}_1$ and $\mathcal{S}_2$ are formed from bisection and the coefficients are known for $\mathcal{S}$, then the coefficients (and hence bounds on f, via (10)) can be computed in $\binom{d+n}{n+1}$ add-and-shift operations; see [14, Lemma 3.2].

For example, for quadratic programming problems ($d = 2$) with the constraints $\sum_{i=0}^{n} x_i = 1$, $x_i \geq 0$, $0 \leq i \leq n$, this requires $n + 2$ adds and shifts, and may require less if many of the coefficients are non-zero.

Remark 4. Points to make: The multidimensional de Casteljau algorithm is most clearly implemented using recursion within a programming language, a technique that can be very inefficient in certain environments. On the other hand, implementation without recursion leads to confusing indexing schemes. Identification of important problems representing special cases where the algorithm can be simplified may thus be of use.

1.3 Alternatives and Previous Work

Garloff et al (see [19,20,21] and earlier works) Leroy (see [7]), Nataraj et al (see [13,15,16], etc.) and others have studied Bernstein polynomials in the context of global optimization, in particular, in the conjunction with interval arithmetic to supply mathematical guarantees on the results. Muñoz and Narkawicz (see [12]) consider efficient representation of Bernstein polynomials for symbolic computations to be incorporated in global optimization algorithms.

In [5], we analyze the interplay between interval arithmetic and the constraints defining a simplex, to obtain formulas that are superior to naive interval arithmetic.

Here, we identify applications for which the simplicial Bernstein form is natural and likely to be competitive. We look at sharpness of bounds on the range, and we count the number of operations. Exhaustive comparison of implementation efficiencies on appropriate problems will be in future work.

2 Quadratic Programming Problems

The quadratic programming problems naturally suited to benefiting from Bernstein representations can be written as

$$\left.\begin{aligned} \text{minimize } f(\vec{x}) &= \sum_{i=0}^{n}\sum_{j=0}^{n} a_{i,j}x_ix_j + \sum_{i=0}^{n} b_ix_i \\ \text{subject to } &\sum_{i=0}^{n} x_i = 1, \quad x_i \ge 0, \;\; i = 0,\ldots,n, \\ h_j(\vec{x}) &= \sum_{i=0}^{n} h_{i,j}x_i = r_j, \;\; j = 1,\ldots,m. \end{aligned}\right\} \tag{11}$$

The non-negativity conditions and the condition $\sum_{i=0}^{n} x_i = 1$ are common to many practical problems, and define the optimization to be over the unit simplex $\mathcal{S}$. However, the linear terms $\sum_{i=0}^{n} b_ix_i$ and $\sum_{i=0}^{n} h_{i,j}x_i$ in the objective function as well as in the m additional equality constraints are usually present, and cannot be easily eliminated. In the absence of these additional constraints, (11) would be of the form in (10) with $d = 2$, and no conversion would be required. However, for this $d = 2$ case, rewriting each x_k, $0 \le k \le n$ in terms of the basis functions $B_{\vec{i}}^{(2,n)}$ is relatively simple, so the objective function f and the constraints h_j are written in terms of barycentric coordinates over the canonical simplex $\mathcal{S}$. We have

$$x_k = 1 - \sum_{\substack{j=0\\ j\ne k}}^{n} x_j, \qquad \text{so} \qquad x_k\left(1 - \sum_{\substack{j=0\\ j\ne k}}^{n} x_j\right) = x_k - \sum_{\substack{j=0\\ j\ne k}}^{n} x_kx_j = x_k^2. \tag{12}$$

Adding the sum $\sum_{\substack{j=0\\j\neq k}}^{n} x_k x_j$ to both sides thus gives

$$x_k = \sum_{j=0}^{n} x_k x_j = \sum_{j=0}^{n} \frac{B^{(2,n)}_{\vec{i}_{j,k}}}{\binom{d}{\vec{i}_k}} = 2\sum_{\substack{j=0\\j\neq k}}^{n} B^{(2,n)}_{\vec{i}_{j,k}} + B^{(2,n)}_{\vec{i}_{k,k}}, \quad \text{where} \tag{13}$$

$\vec{i}_{j,k}$ is the multi-index with $n+1$ entries whose j-th and k-th entries are 1 and all of whose other entries are 0, and

$$B^{(2,n)}_{\vec{i}_{j,k}} = \begin{cases} 2x_k x_j,\ j \neq k, \\ \quad x_k^2,\ j = k. \end{cases} \tag{14}$$

Replacing linear terms in (11) using (13) and collecting terms gives the linear programming problem completely in terms of barycentric coordinates:

$$\left.\begin{aligned} &\text{minimize } f(\vec{x}) = \sum_{i=0}^{n}\sum_{j=0}^{n} \alpha_{i,j} B^{(2,n)}_{\vec{i}_{i,j}} \\ &\text{subject to } \sum_{i=0}^{n} x_i = 1, \quad x_i \geq 0,\ \ i = 0,\ldots,n, \\ &\quad h_j(x) = \sum_{i=0}^{n}\sum_{k=0}^{n} \gamma_{j,i,k} B^{(2,n)}_{\vec{i}_{i,k}} = r_j,\ \ j = 1,\ldots,m. \end{aligned}\right\} \tag{15}$$

The conversion process need be done only once, before beginning the branch and bound algorithm. Once the conversion is done, the de Casteljau algorithm may be applied separately to f and the h_j in (15), to obtain a barycentric representation of the quadratic program over each element of a subdivision of $\mathcal{S}$. As those elements of the subdivision are further subdivided, the process can be repeated.

Remark 5. Higher degree polynomials can be treated with this method, but the total number of adds and shifts for a bisection $\binom{d+n}{d+1}$ grows rapidly with the degree d, and implementation is not as simple. This may cause computations with high d to be impractical, although parallelization, and indeed, computations similar to fast Fourier transforms can possibly be used; see [1].

3 Examples

The following examples are applications that have appeared in the literature or in private correspondence.

Example 1. This example, namely, the Markowitz model of stock portfolio optimization, originates with [9]. It consists precisely of (11), with $b_i = 0$, $0 \leq i \leq n$, and $m = 1$. The objective f represents risk to be minimized, the $a_{i,j}$ are the correlations between holdings, and the constant r in the constraint c represents the required rate of return.

We will illustrate the use of Bernstein expansions in the de Casteljau algoritnm, as in [14], to facilitate computations associated with a subdivision process. In Example 1. f and the h_j have separate Bernstein expansions, and their ranges subject to the barycentric condition $\sum_{i=0}^{N} x_i = 1$ can be computed in parallel (e.g. in a vector computation). For brevity, we illustrate the technique with f alone. The objective function of Example 1 is exactly the objective function f in (2). We consider the case $d = 2,\ n = 2$. Then $\vec{x} = (x_0,\ x_1, x_2)$, and the objective function has 6 coefficients $a_{0,0}$, $a_{0,1}$, $a_{0,2}$, $a_{1,1}$, $a_{1,2}$, and $a_{2,2}$. In our illustration, set the coefficients the following:

$$a_{0,0} = 2,\ \ a_{0,1} = 6,\ \ a_{0,2} = 8,\ \ a_{1,1} = 4,\ \ a_{1,2} = 12,\ \ a_{2,2} = 8,$$

so the objective function is

$$f\left(\vec{x}\right) = 2x_0^2 + 6x_0x_1 + 8x_0x_2 + 4x_1^2 + 12x_1x_2 + 8x_2^2. \tag{16}$$

To apply the de Casteljau algorithm, the homogeneous polynomial (16) needs to be transformed into Bernstein form according to Remark 2. The corresponding Bernstein coefficients after conversion are

$$\alpha_{0,0} = 2, \alpha_{0,1} = 3, \alpha_{0,2} = 4, \alpha_{1,1} = 4, \alpha_{1,2} = 6, \alpha_{2,2} = 8.$$

The objective function in Bernstein form is given by

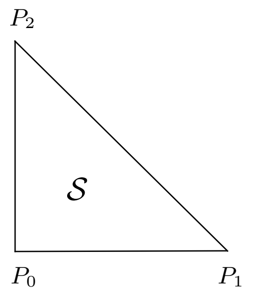
$$f\left(\vec{x}\right) = 2B_{\vec{i}_{0,0}}^{(2,2)} + 3B_{\vec{i}_{0,1}}^{(2,2)} + 4B_{\vec{i}_{0,2}}^{(2,2)} + 4B_{\vec{i}_{1,1}}^{(2,2)} + 6B_{\vec{i}_{1,2}}^{(2,2)} + 8B_{\vec{i}_{2,2}}^{(2,2)}. \tag{17}$$

We introduce additional notation, similar to that in [14], to denote Bernstein coefficients associated with subdivisions of the original simplex, as follows.

$$\begin{array}{lll} \alpha_{0,0} = c_{\vec{i}_{0,0}} = c_{2,0,0} = 2; & \alpha_{0,1} = c_{\vec{i}_{0,1}} = c_{1,1,0} = 3; & \alpha_{0,2} = c_{\vec{i}_{0,2}} = c_{1,0,1} = 4 \\ \alpha_{1,1} = c_{\vec{i}_{1,1}} = c_{0,2,0} = 4; & \alpha_{1,2} = c_{\vec{i}_{1,2}} = c_{0,1,1} = 6; & \alpha_{2,2} = c_{\vec{i}_{2,2}} = c_{0,0,2} = 8. \end{array}$$

The indexes for these coefficients correspond to scaled barycentric coordinates on edges of the simplex, as illustrated in Figure 1; they are related to the values of the polynomial at those points through (9). According to Remark 2, the objective function is bounded within [2,   8].

We now illustrate how these coefficients are reassigned and combined to obtain the coefficients for the barycentric representation over the two subsimplexes $\mathcal{S}_1$ and $\mathcal{S}_2$ obtained by bisecting the edge connecting P_1 and P_2. The corresponding computations may be done efficiently with the `EdgeDeCasteljau` algorithm in [14]. In particular, running `EdgeDecasteljau(`$b_{2,2}[\mathcal{S}]$`,`$\frac{1}{2}$`;...)` gives the coefficients in Figure 2(b). The de Casteljau algorithm computes these coefficients through a simple averaging process of adjacent coefficients. One reads off the Bernstein coefficients of the polynomials in barycentric form for $\mathcal{S}_1$ and $\mathcal{S}_2$ directly from this diagram, as illustrated in Figure 3 According to Remark 2, f is bounded over $\mathcal{S}_1$ by $[2, 8]$, and f is bounded over $\mathcal{S}_2$ by $[2, 6]$. If we continued the process within a branch and bound algorithm, $\mathcal{S}_1$ and / or $\mathcal{S}_2$ can replace $\mathcal{S}$, for the process to be repeated. Note that, since f is quadratic, the ranges $[2, 8]$ and [2.6] are exact ranges over $\mathcal{S}_1$ amd $\mathcal{S}_2$, respectively.

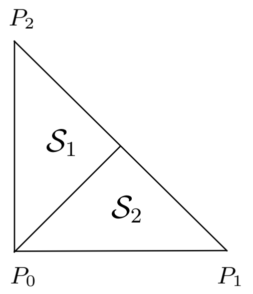


(a) The original simplex.

8 $c_{0,0,2}$		
4 $c_{1,0,1}$	6 $c_{0,1,1}$	
2 $c_{2,0,0}$	3 $c_{1.1.0}$	4 $c_{0,2,0}$

(b) Corresponding coefficient labeling.

Fig. 1. Simplex coefficients for Example 1.



(a) Bisected simplex.

8				
	7			
4		6		
	$\frac{7}{2}$		5	
2		3		4

(b) de Casteljau output.

Fig. 2. Bisecting the simplex.

8 $c_{0,0,2}$		
4 $c_{1,0,1}$	7 $c_{0,1,1}$	
2 $c_{2,0,0}$	$\frac{7}{2}$ $c_{1.1.0}$	6 $c_{0,2,0}$

(a) Coefficients for $\mathcal{S}_1$.

6 $c_{0,0,2}$		
$\frac{7}{2}$ $c_{1,0,1}$	5 $c_{0,1,1}$	
2 $c_{2,0,0}$	3 $c_{1.1.0}$	4 $c_{0,2,0}$

(b) Coefficients for $\mathcal{S}_2$.

Fig. 3. Bernstein coefficients for $\mathcal{S}_1$ and $\mathcal{S}_2$.

Example 2. The condition $\sum_{i=0}^{n} x_i^2 = 1$ is equivalent to requiring the optimum be on the unit $(n+1)$-sphere, a much-studied condition with many applications. As one of many examples of this, the work [17] deals with minimization of a homogeneous polynomial on a sphere. If each x_i only occurs to even powers in the objective and other constraints, a change of variables $\tilde{x}_i = x_i^2$ transforms the problem to optimization over the unit simplex.

Example 3. Minimization over the ℓ_1 sphere is minimization subject to the condition $\sum_{i=0}^{n} |x_i| = 1$. Minimization over the first orthant of the ℓ_1-sphere can thus be viewed as minimization over the standard simplex $\mathcal{S}$. If, for example, there is some symmetry across orthants, techniques for optimization over a simplex (and in particular, the Bernstein representation) can be used. This would happen, for example, if most of the individual variables only occur with even powers, or if there are a few variables that just occur with odd powers. However, associated branch and bound algorithms are likely to be practical only for relatively small n.

4 Summary

We have reviewed simplexes and the simplicial Bernstein form for multivariate polynomials in the context of global optimization and, indirectly, computing α-cuts of a fuzzy logic output. We have identified contexts commonly occurring in applications in which the simplicial Bernstein form and computations with it simplify and are likely to prove practical and competitive, and have presented some preliminary examples. Actual comparisons and computations, including within a fuzzy logic context, will appear in future work.

Acknowledgements

References

1. Licio Bezerra. Efficient computation of Bézier curves from their Bernstein-Fourier representation. *Applied Mathematics and Computation*, 220:235–238, 09 2013.
2. Wolfgang Böhm, Gerald Farin, and Jürgen Kahmann. A survey of curve and surface methods in CAGD. *Comput. Aided Geom. Des.*, 1(1):1–60, July 1984.
3. Rida T. Farouki. The Bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design*, 29(6):379–419, 2012.
4. Chenyi Hu, R. Baker Kearfott, and Andre de Korvin, editors. *Knowledge Processing with Interval and Soft Computing.* Advanced information and knowledge processing. Springer Verlag, Wien / New York, 2008.
5. Sam Karhbet and Ralph Baker Kearfott. Range bounds of functions over simplices, for branch and bound algorithms. *Reliable Computing*, 25:53–73, July 2017. Special volume containing refereed papers from SCAN 2016, guest editors Vladik Kreinovich and Warwick Tucker.
6. Vladik Kreinovich. *Relations between Interval and Soft Computing*, pages 75–97. In Hu et al. [4], 2008.

7. Richard Leroy. Convergence under subdivision and complexity of polynomial minimization in the simplicial Bernstein basis. *Reliable Computing*, 17(1):11–21, Dec 2012.
8. Weldon A. Lodwick and K. David Jamison. Special issue: interfaces between fuzzy set theory and interval analysis. *Fuzzy Sets and Systems*, 135(1):1–3, 2003. Interfaces between Fuzzy Set Theory and Interval Analysis.
9. Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.
10. R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. SIAM, Philadelphia, PA, 2009.
11. Ramon Moore and Weldon Lodwick. Interval analysis and fuzzy set theory. *Fuzzy Sets and Systems*, 135(1):5–9, 2003. Interfaces between Fuzzy Set Theory and Interval Analysis.
12. César Muñoz and Anthony Narkawicz. Formalization of a representation of Bernstein polynomials and applications to global optimization. *Journal of Automated Reasoning*, 51(2):151–196, August 2013.
13. P. S. V. Nataraj and M. Arounassalame. An interval newton method based on the Bernstein form for bounding the zeros of polynomial systems. *Reliable Computing*, 15(2):109–119, Jun 2011.
14. Jörg Peters. Evaluation and approximate evaluation of the multivariate Bernstein-Bézier form on a regularly partitioned simplex. *ACM Transactions on Mathematical Software*, 20(4):460–480, December 1994.
15. Shashwati Ray and P. S. V. Nataraj. A new strategy for selecting subdivision point in the Bernstein approach to polynomial optimization. *Reliable Computing*, 14(1):117–137, Jun 2010.
16. Shashwati Ray and P. S. V. Nataraj. A matrix method for efficient computation of Bernstein coefficients. *Reliable Computing*, 17(1):40–71, Dec 2012.
17. Anthony Man-Cho So. Deterministic approximation algorithms for sphere constrained homogeneous polynomial optimization problems. *Mathematical Programming*, 129(2):357–382, Oct 2011.
18. Jia Fu Tang, Ding Wei Wang, Richard Y. K. Fung, and Kai-Leung Yung. Understanding of fuzzy optimization: Theories and methods. *Journal of Systems Science and Complexity*, 17(1):117, 2004.
19. Jihad Titi and Jürgen Garloff. Fast determination of the tensorial and simplicial Bernstein forms of multivariate polynomials and rational functions. *Reliable Computing*, 25:24–37, 2017.
20. Jihad Titi and Jürgen Garloff. Matrix methods for the simplicial Bernstein representation and for the evaluation of multivariate polynomials. *Applied Mathematics and Computation*, 315:246–258, 2017.
21. Jihad Titi, Tareq Hamadneh, and Jürgen Garloff. Convergence of the simplicial rational Bernstein form. In Hoai An Le Thi, Tao Pham Dinh, and Ngoc Thanh Nguyen, editors, *Modelling, Computation and Optimization in Information Systems and Management Sciences*, pages 433–441, Cham, 2015. Springer International Publishing.
22. A. Ullah, J. Li, A. Hussain, and Yindong Shen. Genetic optimization of fuzzy membership functions for cloud resource provisioning. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Dec 2016.
23. Warren N. Waggenspack and David C. Anderson. Converting standard bivariate polynomials to Bernstein form over arbitrary triangular regions. *Computer-Aided Design*, 18(10):529–532, 1986.

24. G.William Walster and Vladik Kreinovich. Computational complexity of optimization and crude range testing: a new approach motivated by fuzzy optimization. *Fuzzy Sets and Systems*, 135(1):179–208, 2003. Interfaces between Fuzzy Set Theory and Interval Analysis.