Bernstein basis and its applications in solving geometric constraint systems

Sebti Foufou* (soufou@u-bourgogne.fr, sfoufou@qu.edu.qa)
CSE Department, CENG, Qatar University, P.O. Box 2713, Doha, Qatar.
Le2i, UMR CNRS 5158, Université de Bourgogne, BP 47870, 21078 Dijon, France.

Dominique Michelucci (dmichel@u-bourgogne.fr)
Le2i, UMR CNRS 5158, Université de Bourgogne, BP 47870, 21078 Dijon, France.

Abstract. This paper reviews the properties of Tensorial Bernstein Basis (TBB), then discusses the use of this basis and interval analysis for solving systems of non-linear univariate or multivariate equations resulting from geometric constraints. TBB are routinely used in computerized geometry: geometric modelling for CAD-CAM, or computer graphics. They provide sharp enclosures of polynomials and their derivatives. They are used to reduce domains while preserving roots of polynomial systems, to prove that domains do not contain roots, and to make existence and uniqueness tests. They are compatible with standard preconditioning methods and fit linear programming techniques. However, current Bernstein-based solvers are limited to small algebraic systems. The paper presents the Bernstein polytopes and shows how combining them with Linear Programming permit to solve big systems as well. The paper also gives a generalization of Bernstein polytopes to higher degrees and a comparison of polytopes-based versus TBB-based polynomial bounds.

Keywords: Tensorial Bernstein basis, Algebraic systems, Univariate and multivariate polynomials, Geometric constraint solving. Bernstein polytope

1. Introduction

This text intents to be a gentle introduction for using Tensorial Bernstein Basis (TBB) to compute sharp ranges for the values of a multivariate polynomial inside a box, *i.e.* a vector of intervals, and for solving well-constrained systems of polynomial equations. It also summarizes former results in this topic and presents Bernstein polytopes and their use in solving geometric constraint systems. The content of this paper should be useful to people in interval analysis, since several reference textbooks on interval analysis (or on the resolution of polynomial systems) do not mention Bernstein basis.

In the remainder of this paper, the word interval refers to a 1D interval, while domain and box refer to 2D, 3D, ... nD intervals. The range refers to the interval that encloses the image of a polynomial inside an interval or a box.

* Corresponding author

The paper is organized as follows: Section 2 recalls the definition and the basic properties of Bernstein polynomials, and the Bernstein basis and discusses their use in computer graphics and geometric constraint solving. Section 3 gives a brief presentation of our solver which is based on interval arithmetic and Bernstein basis, and then presents the fundamental ingredients and algorithms used to build this solver. The Bernstein classical approach (the one based on TBB) expresses all polynomials in the TBB, thus it is exponential time, and it becomes unpractical with systems of more than six variables. Section 4 presents the notion of Bernstein polytopes and shows how they can be computed and used, resorting to linear programming, to improve Bernstein basis-based solvers. This approach does not need to express polynomials in the TBB, it is polynomial time and thus it solves systems of arbitrary size. Section 5 extends the idea of Bernstein polytopes to higher degree. Section 6 gives a short comparison between Bernstein polytopes-based and TBB-based polynomial bounds. Section 7 concludes and presents possible future extensions.

2. Bernstein polynomials and Bernstein basis

2.1. Bernstein polynomials definitions and properties

The $d+1$ Bernstein polynomials $B_i^{(d)}$ of degree d, also written $B_i(t)$ for fixed d, constitute a basis for degree d polynomials:

$$B_i^{(d)}(x) = \binom{d}{i} x^i (1-x)^{d-i}$$

The conversion between this basis and the canonical basis: $(x^0, x^1, \dots x^d)$ is a linear mapping. Classical formulas are [5]:

$$\begin{aligned} x^k &= (1/\binom{k}{d}) \sum_{i=k}^{d} \binom{k}{i} B_i^{(d)}(x) \\ x &= (1/d) \times \sum_{i=0}^{d} i\, B_i^{(d)}(x) \\ x^0 &= 1 = \sum_{i=0}^{d} B_i^{(d)}(x) \end{aligned} \tag{1}$$

Bernstein polynomials have two main properties: their sum equals 1 (see eq. 1), and every $B_i^{(d)}(x)$ is positive for $x \in (0,1)$. These two properties imply that for $0 \leq x \leq 1$, $p(x) = \sum p_i B_i(x)$ is a linear convex combination of the coefficients p_i. For a polynomial p, each $p_i \in \mathbb{R}$ and $p(x \in [0,1])$ lies in $[\min p_i, \max p_i]$. This enclosure is tight, and the min or max bound is exact if it occurs at $i = 0$ or $i = d$. When p_i lies in 2D (or 3D), $p(x)$ describes a 2D (or 3D) Bézier curve, and the arc $p(x), x \in [0,1]$ lies inside the convex hull of its so called control points p_i.

Example: since $x = 0\,B_0(x) + 1/d\,B_1(x) + 2/d\,B_2(x) + \ldots d/d\,B_d(x)$, the polynomial curve $(x, y = p(x))$, with $x \in [0,1]$, lies in the convex hull of its control points $(i/d, p_i)$.

Contrarily to coefficients in the usual basis: $(1, x, x^2, \ldots x^d)$, control points depend on the x interval. The classical de Casteljau method provides the control points of $p(x), x \in [0,t]$, and of $p(x), x \in [t,1]$.

For multivariate polynomials, the TBB is the tensorial product:

$$(B_0^{(d_1)}(x_1), \ldots B_{d_1}^{(d_1)}(x_1)) \times (B_0^{(d_2)}(x_2), \ldots B_{d_2}^{(d_2)}(x_2)) \times \ldots$$

The convex hull properties and the de Casteljau method extend to the TBB, which provide sharp enclosure of multivariate polynomials $p(x), x \in [0,1]^n$.

2.2. Bernstein polynomials and computer graphics

In a nutshell, tensorial Bernstein basis provides sharp enclosures for the value of a multivariate polynomial inside a box. The superiority of Bernstein basis over the naive interval arithmetic has been illustrated in [16] and in [15, 17] where implicit algebraic curves $f(x,y) = 0$ are displayed with the classical subdivision method using both the naive interval arithmetic and a more optimized interval arithmetic that relies on the tensorial Bernstein basis for bound computations. These studies clearly revealed that the last method needs much less subdivision to cull domains which are not crossed by the curve. Bernstein basis is optimal in dealing with the problem of inaccuracy and numerical instability, which has been proved by several authors, in particularly by Farouki's theorem on condition number [5, 6]. In comparison, the canonical basis is terribly unstable, as illustrated by Wilkinson's polynomials. Last but not least, Bernstein basis brings geometric insight and intuition on algebraic and numerical problems. Examples of naive interval and Bernstein-based arithmetic applied on different polynomials are shown in Figure 1.

On one hand, there is no textbook that exposes Bernstein-based solutions, while several textbooks expose interval analysis [13] and homotopy methods [24]. On the other hand most of the methods presented here are all well known and widely used in computer graphics, computer geometry, CAD-CAM; all principles were laid down in CAD-

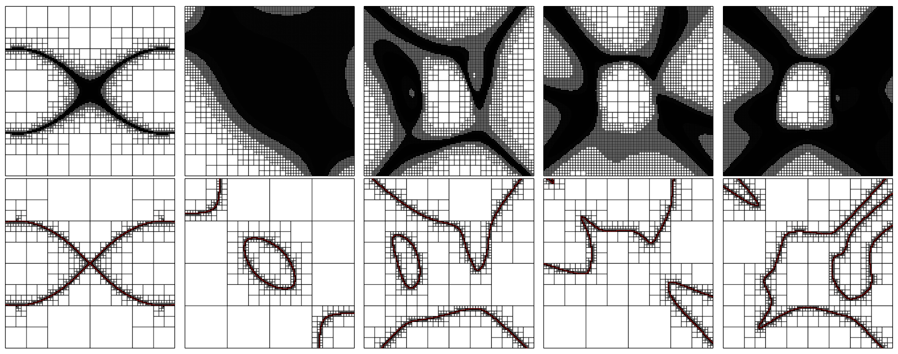

Figure 1. top: naive interval arithmetic. *bottom*: Bernstein-based arithmetic. *Left to right columns*: Cassini oval $C_{2,2}(x, y) = 0$ in $[-2, 2] \times [-2, 2]$ where $C_{a,b}(x, y) = ((x + a)^2 + y^2) \times ((x - a)^2 + y^2) - b^4$, the curve $f(x, y) = 15/4 + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$ on the square $[0, 1] \times [0, 1]$, and three random algebraic curves with total degree 10, 14, 18.

CAM with the de Casteljau's work in 1959 (in industry at Citroën), which have been covered until 1975 when W. Böhm made them public. Meanwhile, P. Bézier published his work on UNISURF in the sixties, in Renault, another French car company. Arguably, Bernstein basis and Bézier curves and surfaces, and their offspring (spline basis for piecewise algebraic functions, and the today fashionable subdivision curves and surfaces or volumes) are at the heart of CAD-CAM.

2.3. Bernstein basis and Geometric Constraint Solving

Geometric constraint-based modelling includes three connected (and sometimes iterative) phases: (i) constraint specifications, (ii) constraint analyses and decompositions, (iii) constraint solving. This paper concerns the third phase and aims at showing the appropriateness of the Bernstein Basis for solving algebraic systems of equations resulting from the analysis and decomposition phase. Solving methods for these systems can be classified in two main categories: iterative numerical methods such as Newton-Raphson and continuation (homotopy) [14, 24], and interval numerical methods which combine the advantages of interval arithmetic for correct numerical computations and rigorous searches of solutions [12, 13]. We note here that continuation methods do not use Bernstein basis [24], and that Newton-Raphson-based methods may use interval arithmetic [12], but, to our knowledge, never Bernstein basis.

The principle of current Bernstein-based solvers [20, 11, 18] is as follows. To find the roots of an algebraic system inside an initial box, contract the box, while preserving the roots, until it is no more possible to significantly reduce it; then try to prove that it does not contain any root (these 2 procedures can be used in the other order). Otherwise, bisect the studied box (for instance along its longest side) and study recursively the two halves. To quote a few, Patrikalakis and Maekawa [20], Garloff and Smith [11], Mourrain and Pavone [18], and Elbert and Kim [4] have proposed variants of this method. The refinement is stopped either when some accuracy is reached, or when it is no more possible to refine depending on the available finite numerical accuracy. To understand this last point, consider that numbers (*i.e.* Bernstein coefficients) are represented with intervals with integer bounds (for instance, 1 means 10^{-6} meter). When an interval width is equal to one, bisecting it is no more possible: the left half (and the right half) of $[0, 1]$ is itself; actually, all intervals with width 1 are "atomic", and can no more be bisected. The same kind of thing occurs with floating point numbers, instead of integers, due to discreteness.

3. Resolution of algebraic systems

This section presents a solver relying on the tensorial Bernstein properties. This solver explicitly represents polynomials in the tensorial Bernstein basis [11, 18, 20]. This is the classical approach in Computer Graphics and CAD-CAM. This solver also uses some Linear Programming (see Section 3.4).

Our approach performs some simple symbolic operations such as sums, linear combinations, and derivatives on polynomials, but it does not need expensive operations used in computer algebra, such as resultants, greatest common divisors and standard basis. The coefficients of polynomials are represented by intervals to account for rounding errors: thus these intervals remain narrow during computations. Classically, the use of interval arithmetic is two-folds: on one hand, intervals enclose rounding errors of floating point arithmetic, and on the other hand, interval arithmetic is used to compute the range of functions on large domains where the width can be huge. Our approach uses interval arithmetic only to account for rounding errors. Indeed, sharp ranges of polynomials inside boxes are computed relying on properties of Bernstein basis (and not on the classical interval arithmetic, naive or centered).

The solver starts by expressing the polynomials of the equations in the Bernstein basis using classical formulas for the conversion between canonical and Bernstein basis [5]. The initial box is recursively subdivided until one of the three following terminating cases is reached:

(i) there is no solution within the box (see Section 3.4), (ii) the box contains one unique solution (see Section 3.5), and (iii) the box is so small that it cannot be subdivided any more. This process may be accelerated with two independent optimizations: preconditioning of the equations in the multivariate case, and contraction of boxes (Section 3.2). Box contraction uses the computation of 2D convex hulls, for any number of equations and unknowns, as a subroutine (Section 3.3). A variant of box contraction used by Mourrain [18] relies on the resolution of univariate polynomials (Section 3.1).

3.1. Isolating real roots of univariate polynomials

Bernstein basis enables a simple and fast algorithm to enclose real roots of an univariate polynomial in an interval, *e.g.* $[0, 1]$. Convert the polynomial in the Bernstein basis. If all Bernstein coefficients have the same sign, the polynomial has no root inside the interval. Otherwise study recursively the two halves of the interval. The Bernstein coefficients within the two halves are computed using the classical de Casteljau algorithm [5]. When the Bernstein coefficients increase (decrease) monotonously, and are negative at one end and positive at the other end, then the interval contains a single, regular root, and the standard Newton method is guaranteed to converge to the root. Intervals without roots are culled quickly; it is known that the convergence of Bernstein-based subdivision methods is quadratic [18].

To find roots of $f(x) = 0$ in $[1, +\infty)$, define $g(x) = x^d f(1/x)$: if $f(x) = \sum f_i x^i$ in the canonical basis, then $g(x) = \sum f_i x^{d-i}$, *i.e.* g is a polynomial with the same coefficients than f but in reverse order in the canonical basis; and the roots of g inside $[0, 1]$ are clearly the inverses of the roots of f inside $[1, +\infty)$. This way all positive roots are found. To find negative roots of $f(x) = 0$, compute the positive roots of the polynomial $h(x) = f(-x)$.

This kind of algorithm is used in computer graphics to ray trace algebraic implicit surfaces $f(x, y, z) = 0$ with total degree d, *i.e.* to compute the intersection points between a ray (a half line) and the surface. The ray is parameterized with: $x = x_0 + at, y = y_0 + bt, z = z_0 + ct$, where the origin (x_0, y_0, z_0), and the direction (a, b, c) of the ray are known. Replacing x, y, z by their values in t gives an univariate algebraic equation in t, with degree d, which can be solved by some variant of the previous method.

3.2. Preconditioning and box contractions

As it is well known in interval analysis, contraction methods, *i.e.* methods which contract the considered domain while preserving the roots it contains, are interesting since they can avoid useless and costly branching (bisections). Bernstein basis enables to very efficiently contract domains around their contained roots.

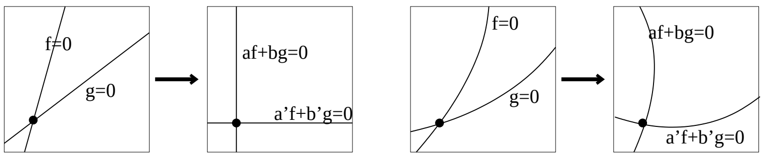

Figure 2. Effect of preconditioning for a linear system (left), and a non-linear one (right).

First the polynomial system is preconditioned (see Figure 2 for two 2D examples), so that its jacobian is the identity matrix at the center x_0 of the studied box. Let $f(x) = 0$ be the studied system, the preconditioned system is $g = Mf$ for some matrix M, so that $g'(x_0) = I_{nn}$; of course, f and g have the same roots. Since $g'(x_0) = Mf'(x_0) = I_{nn}$, it turns out that $M = f'(x_0)^{-1}$ is the inverse of the jacobian at x_0. After preconditioning, the k^{th} equation in $g(x) = 0$ can be close to an hyperplane having equation $x_k = c_k$, where c_k is some constant. Now, each hypersurface $z = g_k(x) = 0$ lies inside the convex hull of its control points. The convex hull is a polytope in high dimension, which is not convenient. But this polytope lies inside a prism, the base of which is the $2D$ convex hull of the projections of the control points on the (x_k, z) plane, as exemplified in Figure 3. For a system of n unknowns and equations, n 2D convex hulls have to be computed, in the planes (x_i, z), $i \in [1, n]$. Computing a $2D$ convex hull is cheap and easy (section 3.3).

If all equations have the same degree, the Bernstein coefficients of a linear combination $\sum a_i f_i(x)$ are just the linear combination of the Bernstein coefficients of the f_is. This makes possible to not really compute the jacobian inverse, but to rather solve several linear systems. Finally, this contraction method also partially applies when the jacobian has not full rank, *i.e.* is not invertible.

A variant of box contraction used by Mourrain [18] relies on the resolution of univariate polynomials *e.g.* in right most part of Figure 3, the three points having maximal (minimal) z coordinates in $x_1 = 0, 1/2$, and 1 give control points of an univariate maximal (minimal) polynomial. Computed roots of these polynomials permit box contractions.

3.3. 2D convex hull computations

Computing the convex hull of a set of 2D points is a basic problem of computerized geometry. Let (x_i, z_i) be a set of points in the plane x, z. We only explain how to compute the lower part of the convex hull of the (x_i, z_i) points. The upper part can be computed in a similar way. First, sort the points by increasing x_i: $x_0 < x_1 < \dots x_d$. Note that there is only one point for each x_i value (the one with the smallest z). Initialize the lower convex hull with the two leftmost points $p_0 = (x_0, z_0), p_1 = (x_1, z_1)$, and $h = 1$. Then scan the (x_k, z_k) points from left to right (*i.e.* with increasing x), for $k = 2, \dots d$, and update the lower convex hull as follows. Let p_h be the rightmost point of the lower convex hull, and p_{h-1} the point just before. While $p_{h-1}p_hp_k$ "turns right" (the angle $p_{h-1}p_hp_k$ is concave), remove the point p_h from the lower hull. Then add point p_k at the end of the lower convex hull. Three points p, q, r "turn right" when the determinant of $((p_x, p_z, 1), (q_x, q_z, 1), (r_x, r_z, 1))^t$ is negative. When it vanishes, points p, q, r are aligned; when it is positive, they turn left.

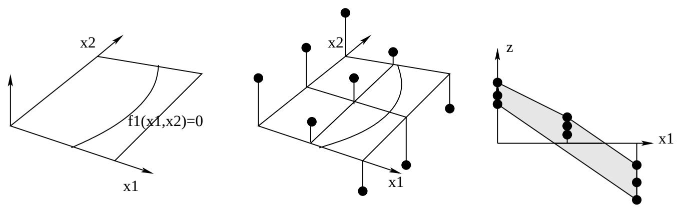

Figure 3. Equation $z = f_1(x_1, x_2) = 0$ has degree 2 in x_1 and x_2, and a grid of 3×3 control points. This curve is seen as the intersection between the zero level set $z = 0$ and the surface $z = f_1(x_1, x_2)$. This surface lies inside the convex hull of its control points $(i/2, j/2, b_{i,j}), i = 0, 1, 2, j = 0, 1, 2$. It is easy to compute the 2D convex hull of the projection on the x_1, z plane of the control points. The intersection of this $2D$ convex hull with the x_1 axis encloses all points of the curve.

This algorithm is in $O(d \log d)$ if d is the number of points, the $d \log d$ factor is due to the sorting stage, which is useless here, so the method is linear in the number of points (the number of Bernstein coefficients).

3.4. Proving a domain does not contain roots

Sometimes, the previous reduction method cannot detect quickly that a box does not contain roots. For instance, with 2 circles with the same center and close radius, the method has to subdivide all along the 2 circles, until it separates them. The following test detects very quickly this kind of situation.

Let $f_1(x) = f_2(x) = \ldots = f_n(x) = 0$ be a system of equations. Assume that all f_i have equal degrees. Then, if there are numbers $\lambda_i \in \mathbb{R}^n$ such that $g(x) = \sum_{i=1}^{n} \lambda_i f_i(x)$ has only positive Bernstein coefficients, then $g(x)$ is always positive in the studied domain; since g vanishes at all common roots of f_i, $i = 1 \ldots n$, it proves that the f_i have no common root in the domain. The Bernstein coefficients of g are just λ_i linear combinations of the Bernstein coefficients of the f_i, so such λ_i exist if the corresponding linear programming problem has feasible solutions.

This idea straightforwardly extends to systems of equations and inequalities. Assume the problem is to find x in some box of $\mathbb{R}^n$ such that $f_1(x) = \ldots = f_e(x) = 0$ and $g_1(x) \le 0, \ldots, g_s(x) \le 0$. If there are $\lambda_i \in \mathbb{R}^e$ and $\mu_j \ge 0$ such that $\sum_i \lambda_i f_i + \sum_j \mu_j g_j > 0$ (say $\sum_i \lambda_i f_i + \sum_j \mu_j g_j \ge 1$) for all points x in the box B, then the system has no solution. Such λ_i and μ_j exist iff the corresponding linear programming problem is feasible, *i.e.* this question reduces to linear programming. This idea is illustrated in Figure 4 where the problem is to find x such that $f(x) = 0$ and $g(x) \le 0$. Since the Bernstein coefficients of (say) $g - 2f$ are all strictly positive, then $g - 2f$ is strictly positive on the considered interval, thus $f(x) = 0 \Rightarrow g(x) > 0$, and the system has no solution in the interval.

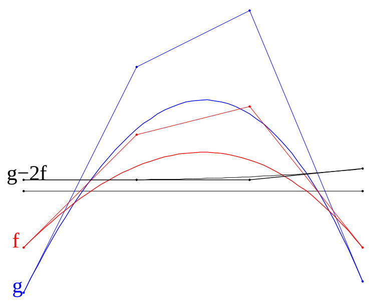

Figure 4. A system with no solution in the given interval

3.5. Proving existence and uniqueness

The solver terminates with a set of small boxes it cannot eliminate. There are very few spurious boxes (*i.e.* boxes without roots) due to the optimality of Bernstein basis. To prove that a resulting box contains a root, or contains a unique root, all tests available in interval analysis apply, of course. Moreover, their computation time is not an issue, due to the small number of spurious boxes; their computation time is an issue only for solvers which are driven by such tests.

Even if all tests apply, some fit the Bernstein scheme in a very natural way. For instance, according to Miranda (or Poincaré-Miranda) theorem, if n continuous functions f_k from $\mathbb{R}^n$ to $\mathbb{R}$ are such that the k^{th} function has constant sign on the hyperface $x_k = a_k$ of the hypercube $a_k \leq x_k \leq b_k$ and constant but opposite sign on the opposite face $x_k = b_k$, for every $k = 1, \dots d$, then the system $f_1(x) = \dots = f_n(x) = 0$ with $x \in \mathbb{R}^n$ has a common root inside the hypercube, see the left half of Figure 5. Now, after preconditioning, in a box containing a regular root $r = (r_1, \dots r_d)$, the hypersurface of the k^{th} equation is very close to the hyperplane $x_k = r_k$; the hyperfaces $x_k = a_k$ and $x_k = b_k$ are on opposite sides of the hyperplane $x_k = r_k$; the fact that f_k has constant sign on an hyperface $x_k = a_k$ or $x_k = b_k$ is proved as long as the Bernstein coefficients on the hyperface have constant sign, and Miranda theorem then applies. This approach is smart because it is simple and uses only available data.

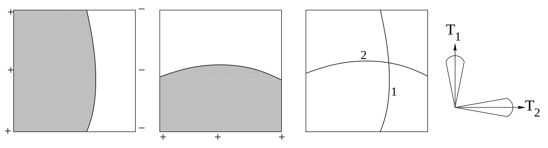

Figure 5. Left: Poincaré-Miranda theorem in 2D proves existence of a root. Right: the uniqueness test, tangent cones are disjoint.

Elbert and Kim [4] propose an uniqueness test, which fits nicely with Bernstein basis (and with B-splines basis, which extend Bernstein basis to piecewise algebraic functions). Let T_i be the cone of vectors tangent to the hypersurface $z = f_i(x)$. Elbert and Kim enclose such cones with a central vector (the average of the generators) and an interval of angles; in passing, the same central vector is used for the cone bounding the normal vectors of this hypersurface $z = f_i(x)$. When the null vector is the only vector common to all T_i, then there is at most one common root in the studied box, due to the mean value theorem. After preconditioning, when the box contains a unique regular root, this condition very likely holds since preconditioning favors orthogonal surfaces (see Figure 5). Elbert and Kim stop the reduction process as soon as existence and uniqueness are proved, and resort to a Newton-Raphson iteration, or some variant. This improvement may speed up the solver, though its complexity is unchanged: in this case, both the reduction process and Newton iteration have quadratic convergence.

Actually, the theorem also holds for weaker conditions on f: it suffices f to be G_1 continuous; thus the theorem applies to function f

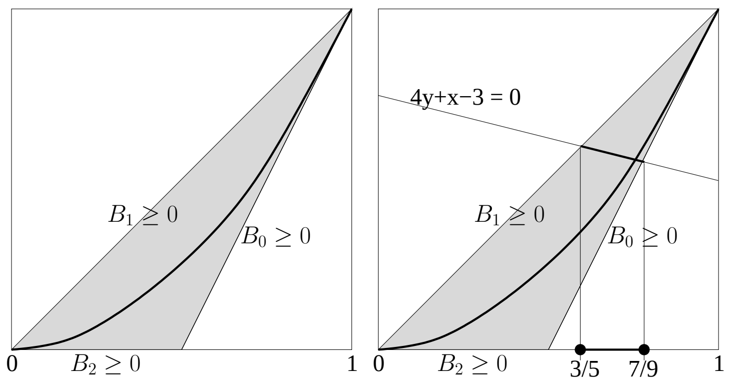

Figure 6. Left: The Bernstein polytope encloses the curve: $(x, y = x^2)$, for $(x, y) \in [0,1]^2$. Its limiting sides are: $B_0(x) = (1-x)^2 = y - 2x + 1 \geq 0$, $B_1(x) = 2x(1-x) = 2x - 2y \geq 0$, $B_2(x) = x^2 = y \geq 0$. Right: Solving $4x^2 + x - 3 = 0$, with $x \in [0,1]$, is equivalent to intersecting the line $4y + x - 3 = 0$ with the curve (x, x^2). Linear programming gives the intersection between the line and the Bernstein polytope.

which are piecewise polynomial, or piecewise rational and well defined everywhere in the box (no component of f has a pole in the box B).

In the univariate case, if $f(x)$ has degree d and control values F_k, then the d control values of its derivative $f'(x)$ are $F'_k = (F_{k+1} - F_k)/d$, for $k = 0, \dots d-1$. This straightforwardly extends to the multivariate case, and to derivatives of higher order.

4. The Bernstein polytope

A difficulty of TBBs is that they have an exponential numbers of elements (it also applies for the canonical basis, but very often polynomials are sparse in the canonical basis). The size is $(d_1+1)(d_2+1)\dots(d_n+1)$ where n is the number of variables and each d_i, $i = 1 \dots n$ is the partial degree of the polynomial in the variable number i. Current Bernstein-based solvers compute all control points, which prevents them to be used beyond a small number of unknowns (6 or 7 without optimization, a dozen with optimization such as [23])

This section presents a polynomial time method to bypass the difficulty due to the exponential number of the Bernstein basis functions [9, 8]. The key remark is that existing Bernstein-based solvers compute all control points, *i.e.* all coefficients in the TBB, and only the smallest and the greatest ones are needed. Resorting to linear programming permits to bypass the above mentioned difficulty. TBB bounds and the bounds provided by linear programming are compared in section 6.

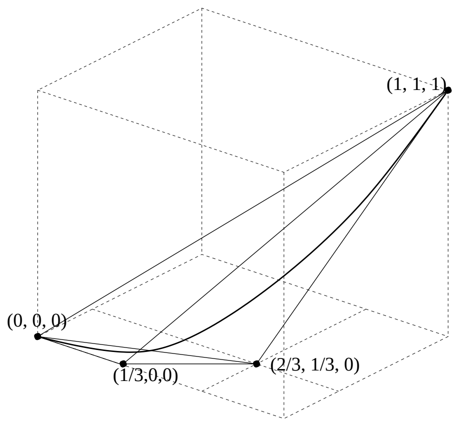

Figure 7. The Bernstein polytope, a tetrahedron, enclosing the curve $(x, y = x^2, z = x^3)$ with $x \in [0,1]$. Its vertices are $v_0 = (0,0,0)$, $v_1 = (1/3,0,0)$, $v_2 = (2/3,1/3,0)$ and $v_3 = (1,1,1)$. v_0 lies on $B_1 = B_2 = B_3 = 0$, v_1 on $B_0 = B_2 = B_3 = 0$, etc. $B_0(x) = (1-x)^3 = 1-3x+3x^2-x^3 \geq 0 \Rightarrow 1-3x+3y-z \geq 0$, $B_1(x) = 3x(1-x)^2 = 3x - 6x^2 + 3x^3 \geq 0 \Rightarrow 3x - 6y + 3z \geq 0$, $B_2(x) = 3x^2(1-x) = 3x^2 - 3x^3 \geq 0 \Rightarrow 3y - 3z \geq 0$, $B_3(x) = x^3 \geq 0 \Rightarrow 3z \geq 0$.

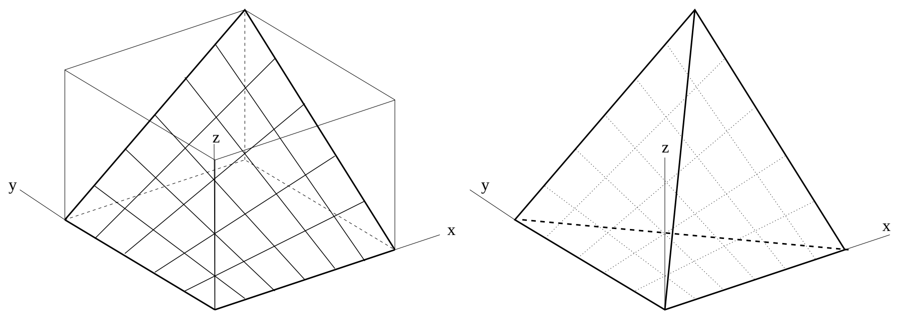

Figure 8. The Bernstein polytope enclosing the surface patch: $(x, y, z = xy)$. Inequalities of delimiting planes are: $B_i(x)B_j(y) \geq 0$, with $i = 0,1$ and $B_0(t) = 1 - t$ and $B_1(t) = t$

We need to define the Bernstein polytope. For univariate polynomials with degree d, the Bernstein polytope is a simplex in dimension d, *i.e.* d coordinates $(x_1, x_2, \ldots x_d)$ are needed to represent its vertices. The Bernstein polytope encloses the arc of curve: $(x_i = x^i), x \in [0,1], i = 1 \ldots d$. The hyperplanes of its faces are given by the inequalities: $B_i^{(d)}(x) \geq 0, i = 0, \ldots d$, where each monomial x^i is changed in x_i, for $i = 1, \ldots d$. Figure 6 shows the Bernstein polytope (a triangle) for degree $d = 2$, which encloses the arc of curve (x_i, x_i^2), where $0 \leq x_i \leq 1$: for convenience, the points (x_i, x_i^2) are renamed $(x, y = x^2)$ on the figure. Figure 7 shows the Bernstein polytope (a tetrahedron) for degree $d = 3$; this tetrahedron encloses the curve (x_i, x_i^2, x_i^3), where $0 \leq x_i \leq 1$:

for convenience, the points (x_i, x_i^2, x_i^3) are renamed $(x, y = x^2, z = x^3)$ on the figure.

This definition of Bernstein polytope can then be extended to multivariate polynomials. For simplicity, consider first quadratic polynomials, *i.e.* the total degree of monomials is at most 2 with the tensorial canonical basis. The Bernstein polytope enclosing the quadratic surface patch $(x_i, x_j, x_i \times x_j)$, or rather $(x, y, z = xy)$ to use more convenient and intuitive notations, is illustrated in Figure 8. Its hyperplanes are defined by $B_0^{(1)}(x) \times B_0^{(1)}(y) \geq 0 \Rightarrow (1-x)(1-y) \geq 0 \Rightarrow 1-x-y+z \geq 0$, $B_0^{(1)}(x) \times B_1^{(1)}(y) \geq 0 \Rightarrow (1-x)y \geq 0 \Rightarrow y - z \geq 0$, $B_1^{(1)}(x) \times B_0^{(1)}(y) \geq 0 \Rightarrow x(1-y) \geq 0 \Rightarrow x - z \geq 0$, $B_1^{(1)}(x) \times B_1^{(1)}(y) \geq 0 \Rightarrow xy \geq 0 \Rightarrow z \geq 0$. This tetrahedron is optimal: it is the convex hull of the algebraic patch. Notice that to the monomial xy is attached a LP (linear programming) variable z. More generally, for a quadratic system, every monomial $x_i x_j$, with $i < j$ is attached to a LP variable x_{ij}, every monomial x_i^2 is attached to a LP variable q_i, every monomial is attached to a LP variable x_i (itself). The monomial 1 is attached to no LP variable. Replacing the monomials x_i^2, $x_i x_j$, x_i with the corresponding LP variables in the inequalities: $B_i^{(2)}(x_i) \geq 0, i = 0, 1, 2$, and in the inequalities: $B_s^{(1)}(x_i) \times B_t^{(1)}(x_j) \geq 0, s = 0, 1, t = 0, 1$ provide the linear inequalities (in the LP variables) which bound the Bernstein polytope. For instance, $B_{11}(x_1, x_2) = B_1(x_1) \times B_1(x_2) = x_1 x_2 = x_{12}$. Clearly, the Bernstein polytope for a quadratic system in n unknowns has $O(n^2)$ hyperplanes.

For systems of higher degree, either auxiliary equations and unknowns are used to reduce them to quadratic systems, or the Bernstein polytope is extended: for a total degree d system, the Bernstein polytope has $O(n^d)$ hyperplanes. The Bernstein polytope has an exponential number of vertices [1], but it has only a polynomial number of hyperplanes. Therefore, in the multivariate case, the Bernstein polytope is not a simplex.

Computing a lower and an upper bounds of a polynomial $p(x)$, see an example in section 4.1, reduces to find the vertex of the Bernstein polytope which minimizes or maximizes the linear objective function $p(x)$ which is obtained after replacing the monomials not equal to 1 with the corresponding LP variables. This is a linear programming problem with polynomial size [22]. It is theoretically soluble in (weak) polynomial time with some interior point method or with the ellipsoid method; in practice, the simplex method is very competitive.

Linear programming can also be used to contract the box, while preserving roots, as illustrated in Figure 6-right and in section 4.2. Each equation of the system provides an equality in the LP variables. It is also possible to account for algebraic inequalities in a very straightforward way: each inequality provides a linear inequality in the LP variables. These equalities and inequalities are used together with the inequalities defining the Bernstein polytope, see [10] for further explanation. The new solver that exploits the LP programming, is simpler than former solvers and encompasses them; for instance, the method illustrated in section 3.4 becomes useless: in such cases, the feasible set of the LP problem is empty, which is detected by the simplex algorithm.

The Bernstein polytope and linear programming permit to use Bernstein-based solvers with big algebraic systems. The simplex method, as well as other methods for Linear Programming, may suffer reliability problems due to rounding errors of floating point numbers. This inaccuracy issue is studied and solved in [19, 7].

An advantage of this approach is that it extends to non-algebraic functions; for instance, for equations involving transcendentals like $y = \exp x$ or $y = \cos x$, it suffices to enclose the curve $(x, y = \exp x)$ or the curve $(x, y = \cos x)$ in 2D convex polygons.

4.1. Bounding: $4x^2 + x - 3, 0 \le x \le 1$

To compute a lower and an upper bound of the polynomial $p(x) = 4x^2 + x - 3$, for $x \in [0, 1]$, minimize, and maximize, the linear objective function: $4y + x - 3$ on the Bernstein polytope (the triangle in Figure 6) enclosing the curve $(x, y = x^2), x \in [0, 1]$. It is an LP problem, after replacing x^2 with y:

$$\begin{array}{l}
\min p = 4y + x - 3, \max p = 4y + x - 3 \\
B_0 = y - 2x + 1 \\
B_1 = -2y + 2x \\
B_2 = y \\
x \ge 0, y \ge 0, B_0 \ge 0, B_1 \ge 0, B_2 \ge 0
\end{array}$$

Notations are self-explanatory. The simplex algorithm [3, 21] provides the enclosure $[-3, 2]$:

$$\begin{array}{ll}
\min p = -3 + x + 4y & \max p = 2 - 5B_0 - 9/2B_1 \\
B_0 = 1 - 2x + y & x = 1 - B_0 - B_1/2 \\
B_1 = 2x - 2y & y = 1 - B_0 - B_1 \\
B_2 = y & B_2 = 1 - B_0 - B_1
\end{array}$$

Remember that variables on the right side ("not in base") have values 0. Only the max part is commented. In $\max p = 2 - 5B_0 - 9/2B_1$, the value of p cannot be greater than 2: variables on the right side B_0 and B_1 are zero, and increasing one of them or both will only decrease p due to the negative coefficients $-5B_0 - 9/2B_1$ in the objective function.

Consequently, at vertex $v_0 = (0,0)$ where $B_1 = B_2 = 0$, the polynomial value is $p_0 = p(0) = -3$; at vertex $v_1 = (1/2, 0)$ where $B_0 = B_2 = 0$, the polynomial value is $p_1 = p(1/2) = -3/2$; at vertex $v_2 = (1,1)$ where $B_0 = B_1 = 0$, the polynomial value is $p_2 = p(1) = 2$. These values p_0, p_1, p_2 are the coefficients in the Bernstein basis, or control points, of $p(x)$: $p(x) = p_0 B_0(x) + p_1 B_1(x) + p_2 B_2(x)$. This property extends to all univariate polynomials. It is a consequence of the definition of Bernstein polytopes. This property does not extend to multivariate polynomials otherwise the Bernstein polytope would be bounded by an exponential number of hyperplanes, see section 6.

Using other inequalities (we use: $B_1^{(2)}(x) \leq 1/2$) often provides tighter bounds. This feature is not compatible with the standard approach of TBB.

4.2. Solving: $4x^2 + x - 3 = 0, x \in [0,1]$

This section shows how the solver reduces intervals or boxes containing roots, for the simple equation $4x^2 + x - 3 = 0$ for $x \in [0,1]$. Solving is equivalent to finding the intersection points between the line $4y + x - 3 = 0$, and the curve $(x, y = x^2)$. This curve is enclosed in its Bernstein polytope: the triangle of Figure 6. Intersecting the line and the triangle, *i.e.* finding the min and max value of x, will reduce the interval for x; it is the same LP problem as above, except we minimize and maximize x. Solutions are:

$$\begin{array}{ll}
\min x = 3/5 + 2/5B_1 & \max x = 7/9 - 4/9B_0 \\
x = 3/5 + 2/5B_1 & x = 7/9 - 4/9B_0 \\
y = 3/5 - 1/10B_1 & y = 5/9 + 1/9B_0 \\
B_0 = 2/5 - 9/10B_1 & B_1 = 4/9 - 10/9B_0 \\
B_2 = 3/5 - 1/10B_1 & B_2 = 5/9 + 1/9B_0
\end{array}$$

Thus the interval $[0,1]$ for x has been reduced to $[3/5, 7/9]$. To further reduce this interval, apply a scaling to map $x \in [3/5, 7/9]$ to $X \in [0,1]$: $x = 3/5 + (7/9 - 3/5)X = b + aX$, and the equation in X is: $4a^2X^2 + (8ab + b)X + (b - 3) = 0$. LP is used again to reduce the interval. Convergence around a regular root is very fast, but is not discussed for conciseness. A thorough treatment about convergence is done in [18].[18,19].

Notice that if the line does not cut the Bernstein polytope, then the LP problem is not feasible and the interval $[0,1]$ contains no roots.

5. Generalization of Bernstein polytopes for higher degree

This section explains the generalization of the Bernstein polytopes for higher degrees. The main idea is to consider all inequalities

$$0 \le B_{i_1}^{(d_1)}(x_1)B_{i_2}^{(d_2)}(x_2)\dots B_{i_k}^{(d_k)}(x_k) \le B_{i_1}^{(d_1)}(i_1/d_1)B_{i_2}^{(d_2)}(i_2/d_2)\dots B_{i_k}^{(d_k)}(i_k/d_k)$$

which are relevant to the problem at hand, and to translate them into the tensorial canonical basis. A LP variable is then associated to each monomial in the tensorial canonical basis, and the two previous inequalities provide two linear inequalities in the LP variables in a straightforward way. The following examples illustrate this process.

In a first example, $k = 2, d_1 = d_2 = 2, x_1 = x, x_2 = y$. Then, using symmetry to omit some inequalities for conciseness:

$$\begin{array}{lll}
0 \le B_0(x)B_0(y) = (1-x)^2 \times (1-y)^2 & \le B_0(0)B_0(1) & = 1 \\
0 \le B_0(x)B_1(y) = (1-x)^2 \times 2y(1-y) & \le B_0(0)B_1(1/2) & = 1/2 \\
0 \le B_0(x)B_2(y) = (1-x)^2 \times y^2 & \le B_0(0)B_2(1) & = 1 \\
0 \le B_1(x)B_1(y) = 2x(1-x) \times 2y(1-y) & \le B_1(1/2)B_1(1/2) & = 1/4 \\
0 \le B_1(x)B_2(y) = 2x(1-x) \times y^2 & \le B_1(1/2)B_2(1) & = 1/2 \\
0 \le B_2(x)B_2(y) = x^2y^2 & \le B_1(1)B_2(1) & = 1
\end{array}$$

and expanding polynomials in each row, we get the following set of inequalities:

$$\begin{array}{ll}
0 \le x^2y^2 - 2xy^2 - 2x^2y + x^2 + y^2 + 4xy - 2x - 2y + 1 & \le 1 \\
0 \le -2x^2y^2 + 2x^2y + 4xy^2 - 4xy - 2y^2 + 2y & \le 1/2 \\
0 \le x^2y^2 - 2xy^2 + y^2 & \le 1 \\
0 \le 4(x^2y^2 - x^2y - xy^2 + xy) & \le 1/4 \\
0 \le -2x^2y^2 + 2xy^2 & \le 1/2 \\
0 \le x^2y^2 & \le 1
\end{array}$$

In a second example, $k = 3, d_1 = 3, d_2 = 2, d_3 = 1, x_1 = x, x_2 = y, x_3 = z$. Then:

$$\begin{array}{llll}
0 \leq B_0^3(x)B_0^2(y)B_0^1(z) = (1-x)^3 \times (1-y)^2 \times (1-z) & \leq B_0^3(0)B_0^2(0)B_0^1(0) & = 1 \\
0 \leq B_0^3(x)B_0^2(y)B_1^1(z) = (1-x)^3 \times (1-y)^2 \times z & \leq B_0^3(0)B_0^2(0)B_1^1(1) & = 1 \\
0 \leq B_0^3(x)B_1^2(y)B_0^1(z) = (1-x)^3 \times 2y(1-y) \times (1-z) & \leq B_0^3(0)B_1^2(1/2)B_0^1(0) & = 1/2 \\
0 \leq B_0^3(x)B_1^2(y)B_1^1(z) = (1-x)^3 \times 2y(1-y) \times z & \leq B_0^3(0)B_1^2(1/2)B_1^1(1) & = 1/2 \\
0 \leq B_0^3(x)B_2^2(y)B_0^1(z) = (1-x)^3 \times y^2 \times (1-z) & \leq B_0^3(0)B_2^2(1)B_0^1(0) & = 1 \\
0 \leq B_0^3(x)B_2^2(y)B_1^1(z) = (1-x)^3 \times y^2 \times z & \leq B_0^3(0)B_2^2(1)B_1^1(1) & = 1 \\
0 \leq B_1^3(x)B_0^2(y)B_0^1(z) = 3x(1-x)^2 \times (1-y)^2 \times (1-z) & \leq B_1^3(1/3)B_0^2(0)B_0^1(0) & = 4/9 \\
0 \leq B_1^3(x)B_0^2(y)B_1^1(z) = 3x(1-x)^2 \times (1-y)^2 \times z & \leq B_1^3(1/3)B_0^2(0)B_1^1(1) & = 4/9 \\
0 \leq B_1^3(x)B_1^2(y)B_0^1(z) = 3x(1-x)^2 \times 2y(1-y) \times (1-z) & \leq B_1^3(1/3)B_1^2(1/2)B_0^1(0) & = 2/9 \\
0 \leq B_1^3(x)B_1^2(y)B_1^1(z) = 3x(1-x)^2 \times 2y(1-y) \times z & \leq B_1^3(1/3)B_1^2(1/2)B_1^1(1) & = 2/9 \\
0 \leq B_1^3(x)B_2^2(y)B_0^1(z) = 3x(1-x)^2 \times y^2 \times (1-z) & \leq B_1^3(1/3)B_2^2(1)B_0^1(0) & = 4/9 \\
0 \leq B_1^3(x)B_2^2(y)B_1^1(z) = 3x(1-x)^2 \times y^2 \times z & \leq B_1^3(1/3)B_2^2(1)B_1^1(1) & = 4/9 \\
0 \leq B_2^3(x)B_0^2(y)B_0^1(z) = 3x^2(1-x) \times (1-y)^2 \times (1-z) & \leq B_2^3(2/3)B_0^2(0)B_0^1(0) & = 4/9 \\
0 \leq B_2^3(x)B_0^2(y)B_1^1(z) = 3x^2(1-x) \times (1-y)^2 \times z & \leq B_2^3(2/3)B_0^2(0)B_1^1(1) & = 4/9 \\
0 \leq B_2^3(x)B_1^2(y)B_0^1(z) = 3x^2(1-x) \times 2y(1-y) \times (1-z) & \leq B_2^3(2/3)B_1^2(1/2)B_0^1(0) & = 2/9 \\
0 \leq B_2^3(x)B_1^2(y)B_1^1(z) = 3x^2(1-x) \times 2y(1-y) \times z & \leq B_2^3(2/3)B_1^2(1/2)B_1^1(1) & = 2/9 \\
0 \leq B_2^3(x)B_2^2(y)B_0^1(z) = 3x^2(1-x) \times y^2 \times (1-z) & \leq B_2^3(2/3)B_2^2(1)B_0^1(0) & = 2/9 \\
0 \leq B_2^3(x)B_2^2(y)B_1^1(z) = 3x^2(1-x) \times y^2 \times z & \leq B_2^3(2/3)B_2^2(1)B_1^1(1) & = 2/9 \\
0 \leq B_3^3(x)B_0^2(y)B_0^1(z) = x^3 \times (1-y)^2 \times (1-z) & \leq B_3^3(1)B_0^2(0)B_0^1(0) & = 1 \\
0 \leq B_3^3(x)B_0^2(y)B_1^1(z) = x^3 \times (1-y)^2 \times z & \leq B_3^3(1)B_0^2(0)B_1^1(1) & = 1 \\
0 \leq B_3^3(x)B_1^2(y)B_0^1(z) = x^3 \times 2y(1-y) \times (1-z) & \leq B_3^3(1)B_1^2(1/2)B_0^1(0) & = 1/2 \\
0 \leq B_3^3(x)B_1^2(y)B_1^1(z) = x^3 \times 2y(1-y) \times z & \leq B_3^3(1)B_1^2(1/2)B_1^1(1) & = 1/2 \\
0 \leq B_3^3(x)B_2^2(y)B_0^1(z) = x^3 \times y^2 \times (1-z) & \leq B_3^3(1)B_2^2(1)B_0^1(0) & = 1 \\
0 \leq B_3^3(x)B_2^2(y)B_1^1(z) = x^3 \times y^2 \times z & \leq B_3^3(1)B_2^2(1)B_1^1(1) & = 1
\end{array}$$

There are other possible bounds for polynomials. In his PhD [2], Olivier Beaumont bounds $f(x) = x^k - x^{k+1}$, for $x \in [0, 1]$, as follows: since $f'(x) = kx^{k-1} - (k+1)x^k = x^{k-1}(k - (k+1)x)$ vanishes at $x = k/(k+1)$, then $f(x)$ is maximum at $x = k/(k+1)$ and is equal to $k^k/(k+1)^{k+1}$. To correlate the x^i (each x^i is represented with a LP-variable x_i in a LP problem) Beaumont uses the constraints $x^k - x^{k+1} \leq k^k/(k+1)^{k+1}$; each of these constraints is represented with the linear constraint $0 \leq x_k - x_{k+1} \leq k^k/(k+1)^{k+1}$. The obtained bounds are related to Tchebychev polynomials. For multivariate polynomials, Beaumont uses the tensorial product, like for the Bernstein polytope. Beaumont's polytope is less tight than the Bernstein polytope; they have the same complexity (number of hyperplanes). Of course, it is possible to use both Bernstein and Beaumont inequalities.

6. Polytope-based versus TBB-based bounds

For a multivariate polynomial, the classical TBB bounds are its smallest and its greatest coefficient in the TBB. These bounds can also be expressed as two Linear Programming problems, with, unfortunately, exponential size. Pose $x = (x_1, \dots x_n)$, $\mathcal{D}$ is the set of multi-degrees: $\mathcal{D} = [0, d_1] \times \dots [0, d_n]$ and the multivariate polynomial is

$$p(x) = \sum_{\alpha \in \mathcal{D}} c_\alpha x^\alpha = \sum_{\alpha \in \mathcal{D}} p_\alpha B_\alpha^{(\mathcal{D})}(x)$$

with the usual notation $x^\alpha = x_1^{\alpha_1} \dots x_n^{\alpha_n}$, *i.e.* its coefficients in the tensorial canonical basis are c_α and its coefficients in the TBB are p_α. Then the TBB simplex is the set of points in $\mathbb{R}^{|\mathcal{D}|}$ with coordinates b_α and constrained with: $\sum_{\alpha \in \mathcal{D}} b_\alpha = 1$ and $0 \leq b_\alpha$, for all $\alpha \in \mathcal{D}$ (it implies that $b_\alpha \leq 1$). The b_α are LP unknowns corresponding to the $B_\alpha^{(\mathcal{D})}(x)$. Then $\min \sum_{\alpha \in \mathcal{D}} p_\alpha b_\alpha$, and $\max \sum_{\alpha \in \mathcal{D}} p_\alpha b_\alpha$ are obviously equal to the classical TBB bounds. The TBB simplex has an exponential number of vertices and hyperfaces: $|\mathcal{D}| = \prod_{i=1}^n (1 + d_i)$. It may seem at first glance that expressing the classical TBB bounds as two LP problems on the TBB simplex brings nothing. However, it has two advantages: First, it becomes apparent that the TBB simplex can be tightened, because all except one b_α are strictly less than 1, and it is easy to compute the greatest value M_α of b_α, *i.e.* the greatest value of $B_\alpha^{(\mathcal{D})}(x)$ for $x \in [0, 1]^n$. Thus it is possible to find better bounds than the classical TBB bounds; of course the TBB simplex is no more a simplex after all these clippings $b_\alpha \leq M_\alpha$. Second, this polytope-based formulation permits to unify the classical TBB bounds with other enclosing methods which are polytope-based and resort to Linear Programming, and thus to compare pros and cons of all these polytopes. The study in [8] concluded that Bernstein polytope, or some variant, is the best trade off: it is defined with $O(n^d)$ hyperplanes where $d = \max |\alpha| = \max \sum_{i=1}^n \alpha_i$ is the maximal partial degree of involved monomials x^α (for quadratic system, $d = 2$, and there is $O(n^2)$ hyperplanes), while the TBB simplex is defined with $(d+1)^n$ hyperplanes (*e.g.* 3^n hyperplanes for quadratic systems, *i.e.* when $d = 2$), or, dually, as the convex hull of its $(d+1)^n$ vertices which has the same exponential complexity; though the definition of the Bernstein polytope is polynomial space, it still has an exponential number of vertices (see [8]), which is required to obtain sharp bounds. Moreover in some cases, the Bernstein polytope provides sharper bounds than the classical TBB bounds.

7. Conclusion

This paper discussed the use of tensorial Bernstein basis in geometric constraint solving. Current Bernstein-based solvers are very competitive for small systems and are routinely used in Computer Graphics, for instance for ray tracing implicit algebraic surfaces $f(x, y, z) = 0$ (*e.g.* torii), and for ray tracing polynomial or rational patches. Current Bernstein solvers compute all coefficients in the tensorial Bernstein basis; since the number of these coefficients is exponential, these solvers cannot be used with more than six unknowns. This paper has shown how the Bernstein polytopes and linear programming make possible to compute an enclosure (the lower and upper bounds) of the values of a multivariate polynomial over $[0, 1]^n$. The Bernstein polytope and linear programming also make possible to reduce boxes while preserving contained roots, and to solve polynomial systems with arbitrary size. Moreover, these new solvers are simpler and can be extended to handle transcendental functions.

Acknowledgements

This research work has been funded by NPRP grant number NPRP $09-906-1-137$ from the Qatar National Research Fund (a member of the Qatar Foundation).

References

1. Faiz A. Al-Khayal and James E. Falk. Jointly constrained biconvex programming*. *Mathematics of Operations Research*, 8(2):273–286, May, 1986.
2. O. Beaumont. *Algorithmique pour les intervalles: comment obtenir un résultat sûr quand les données sont incertaines?* PhD thesis, Université Rennes I, 1999.
3. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms.* MIT Press, Cambridge, MA, second edition, 2001.
4. G. Elber and M-S. Kim. Geometric constraint solver using multivariate rational spline functions. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 1–10, New York, NY, USA, 2001. ACM Press.
5. G. Farin. *Curves and Surfaces for CAGD: A Practical Guide.* Academic Press Professional, Inc., San Diego, CA, 1988.
6. R. T. Farouki and V. T. Rajan. On the numerical condition of polynomials in Bernstein form. *Comput. Aided Geom. Des.*, 4(3):191–216, 1987.
7. Ch. Funfzig, D. Michelucci, and S. Foufou. Nonlinear systems solver in floating-point arithmetic using linear programming reduction. In *SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pages pp. 123–134, San Francisco, CA, USA, 2009.
8. Ch. Funfzig, D. Michelucci, and S. Foufou. Polytope-based computation of polynomial ranges. *Computer Aided Geometric Design*, 29:18–29, Jan. 2012.
9. Ch. Funfzig, D. Michelucci, and S. Foufou. Optimizations for tensorial Bernstein-based solvers by using polyhedral bound. *International Journal of Shape Modeling (IJSM)*, 16(1-2):109–128, Dec. 2010.
10. Ch. Funfzig, D. Michelucci, and S. Foufou. Polytope-based computation of polynomial ranges. In *SAC '10 Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1247–1252, Sierre, Switzerland, March 22 - 26, 2010.

11. J. Garloff and A. P. Smith. Solution of systems of polynomial equation by using Bernstein expansion. In *Symbolic Algebraic Methods and Verification Methods*, pages 87–97. Springer, 2001.
12. R. B. Kearfott. *Rigorous Global Search: Continuous Problems.* Nonconvex Optimization and its Applications, Vol. 13. Springer, 1996.
13. R. B. Kearfott and V. Kreinovich, editors. *Applications of Interval Computations.* Applied Optimization, Vol. 3. Springer, 1996.
14. H. Lamure and D. Michelucci. Solving constraints by homotopy. In *Proc. of the Symp. on Solid Modeling Foundations and CAD/CAM Applications*, pages 263–269, May 1995.
15. R. R. Martin, H. Shou, I. Voiculescu, A. Bowyer, and G. Wang. Comparison of interval methods for plotting algebraic curves. *Computer Aided Geometric Design*, 19(7):553–587, 2002.
16. D. Michelucci, S. Foufou, L. Lamarque, and D. Ménégaux. Bernstein based arithmetic featuring de Casteljau. In *Proc. of the 17th Canadian Conference on Computational Geometry*, pages 212–215, University of Windsor, Canada, August 2005.
17. D. Michelucci, S. Foufou, L. Lamarque, and P. Schreck. Geometric constraints solving: some tracks. In *ACM Symp. on Solid and Physical Modelling*, pages 185–196, 2006.
18. B. Mourrain and J-P. Pavone. Subdivision methods for solving polynomial equations. *Journal of Symbolic Computation*, 44(3):292–306, March, 2009.
19. Arnold Neumaier and Oleg Shcherbina. Safe bounds in linear and mixed-integer programming. *Math. Prog*, 99:283–296, 2004.
20. N. M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing.* Springer Verlag, 2002.
21. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, the Art of Scientific Computing.* Cambridge University Press, 1992.
22. Hanif Sherali and Leo Liberti. Reformulation-linearization technique for global optimization. In P. Pardalos and C. Floudas, editors, *Encyclopedia of Optimization*, pages 3263–3268. springer, Berlin, 2008.
23. Andrew Smith. Fast construction of constant bound functions for sparse polynomials. *Journal of Global Optimization*, 43:445–458, 2009. 10.1007/s10898-007-9195-4.
24. A. J. Sommese and C. W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science.* World Scientific, 2005.