Linear Programming for Bernstein Based Solvers

Dominique Michelucci, Christoph Fünfzig

LE2I, UMR CNRS 5158, 9 av Alain Savary, BP 47870, 21078 Dijon cedex, France
Dominique.Michelucci@u-bourgogne.fr

Abstract. Some interval Newton solvers rely on tensorial Bernstein bases to compute sharp enclosures of multivariate polynomials $p(x_1, \dots x_n)$ where $x_i \in [0, 1]$. Unfortunately, polynomials become exponential size in tensorial Bernstein bases. This article gives the first polynomial time method to solve this issue. It proposes to resort to Linear Programming, for computing tight ranges of muti-variate polynomials on a given box (interval vector), and for reducing a box while preserving included roots. The underlying Bernstein polytope is defined, it is the feasible set of the LP problems. Its defining inequalities are given by the positivity of relevant Bernstein polynomials. It has an exponential number of vertices but only a polynomial number of hyperplanes and hyperfaces.

1 Introduction

Especially in 3D, geometric constraints solving eventually requires solving systems of non-linear equations, typically algebraic. Typically, irreducible systems are solved with numerical methods, for instance homotopy [1–3], Newton-Raphson iterations, or some Newton interval methods [4] including Bernstein based solvers.

Computer Graphics, CAD-CAM, and some people in numerical analysis, use properties of TBB[1], and Bernstein based solvers, for computing intersection between algebraic non-linear surfaces and curves, and for numerically solving algebraic systems [5–10]. TBB provide sharp enclosures of multivariate polynomials over a box, *i.e.* a vector of intervals. The range of a multivariate polynomial $p(x), x = (x_1, \dots x_n)$ over the unit box $x \in [0, 1]^n$ is the interval given by the smallest and the greatest coefficients of the polynomial $p(x)$ expressed in the TBB. This property, with the de Casteljau algorithm or other subdivision methods, are used in Computer Graphics and CAD-CAM to compute tight covers of implicit algebraic curves and surfaces [11].

However polynomials are no more sparse and become exponential size in the TBB. For instance the monomial 1 is written $(B_0^{(d_1)}(x_1) + \dots B_{d_1}^{(d_1)}(x_1)) \times \dots \times (B_0^{(d_n)}(x_n) + \dots B_{d_n}^{(d_n)}(x_n))$ in the TBB. Even a linear polynomial $p(x_1, \dots x_n)$ has an exponential number 2^n of coefficients in the TBB, while it has linear size $O(n)$ in the canonical base. A quadratic polynomial $p(x_1, \dots x_n)$ has exponential size 3^n in the TBB, while it has a quadratic number $O(n^2)$ of coefficients (for monomials: x_i^2, $x_i x_j$, x_i, and 1) in the canonical base.

[1] TBB: tensorial Bernstein bases

Clearly this feature makes current Bernstein based solvers impracticable for systems with more than $n = 6$ or 7 unknowns. Geometric constraints, especially in 3D, often yield much bigger irreducible systems. For instance, the regular icosahedron, and non-regular ones (20 triangular faces, 12 vertices, 30 edges) can be specified, up to location and orientation in 3D space, with the length of their edges; similarly their duals: the regular dodecahedron, or non-regular ones (12 pentagonal faces, 20 vertices, 30 edges) can be specified with the length of their edges and coplanarity conditions for each of their faces. In passing, these systems of equations are quadratic, *i.e.* the degree of their monomials is at most 2, since (using usual notations and conventions) equations are: $a_k^2 + b_k^2 + c_k^2 = 1$, $a_k x_i + b_k y_i + c_k z_i + d_k = 0$, $(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 = D_{ij}^2$, where (x_i, y_i, z_i) are the coordinates of vertex i, $a_k x + b_k y + c_k z + d_k = 0$ is the equation of the plane of face k, and D_{ij} is the length of edge ij. To get a well constrained system, 3 points are arbitrarily fixed: a vertex is fixed at the origin, one of its neighbor on the x axis, and another neighbor of the first fixed vertex is fixed on the Oxy plane. These systems are huge, they are roughly irreducible, and they can not be solved with current TBB solvers, due to the exponential size of the representation of the polynomials.

This article proposes the first polynomial time algorithm to overcome this difficulty. Until now, the only solution in the litterature was to dismiss TBB, *e.g.* to resort to the simplicial Bernstein bases, which have polynomial cardinality, as [9], or to resort to more basic interval analysis. In this latter approach, we will mention Yamamura and Fujioka [12], because they also use linear programming, like us. The difference with our work is that they use simpler enclosing polytopes: boxes, provided by interval arithmetic.

The main idea is to resort to Linear Programming. We call the underlying polytope, *i.e.* the feasible set, the Bernstein polytope, because its bounding hyperplanes are provided by the non negativity of relevant Bernstein polynomials. The Bernstein polytope (§3) has an exponential number of vertices, but only a polynomial number of bounding hyperplanes. This Bernstein polytope encloses the quadratic algebraic patch

$$\mathcal{Q}_n = \{(x_1, \dots x_n, x_1^2, \dots x_n^2, x_1x_2, \dots x_{n-1}x_n) \mid 0 \le x_i \le 1\}$$

which is reminiscent to the Veronese map. $\mathcal{Q}_n$ is more conveniently defined as:

$$x = (x_1, \dots x_n) \in \mathbb{R}^n, \quad \phi(x) = (x_1, \dots x_n, x_1^2, \dots x_n^2, x_1x_2, \dots x_{n-1}x_n) \in \mathbb{R}^N$$
$$\mathcal{Q}_n = \{\phi(x) \mid x \in [0,1]^n\} \subset \mathbb{R}^N,\ N = n(n+1)/2 - 1$$

Every polynomial $p(x)$ can then be expressed as $p(x) = L(\phi(x))$ with $L : X \in \mathbb{R}^N \to \mathbb{R}$ a linear function in $\mathbb{R}^N$. Then linear programming algorithms, like the simplex algorithm, can be used to compute the vertex $X \in \mathbb{R}^N$ of this Bernstein polytope which minimizes or maximizes this linear objective function L: it provides a lower and an upper bound of $p(x), x \in [0,1]^n$. Linear programming methods are also able to reduce the box $[0,1]^n$, or any box after some scaling, while preserving contained roots, and thus to solve non linear systems of non-linear equations and inequalities.

It is known that the simplex method is not polynomial time in the worst case; thus, from a theoretical point of view, it is better to invoke a polynomial time method, like the ellipsoid method, or an interior point method. However in practice the simplex algorithm is very competitive.

Since a lot of systems of geometric constraints yield to systems of quadratic equations (like the examples above), and since anyway all algebraic systems can be reduced with polynomial overhead to quadratic systems using auxilliary equations and variables and using iterated squaring, this paper only considers, for simplicity, systems of quadratic equations. Anyway, the main idea of the algorithm: define the Bernstein polytope through its hyperplanes rather than as the convex hull of its vertices, can straightforwardly be extended to systems with higher degrees.

§2 reminds standard notations. §3 define the Bernstein polytope and its bounding hyperplanes. §4.1 and 4.2 explain how computing a range of a multivariate polynomial and reducing a domain while preserving its roots reduces to linear programming problems, considering the Bernstein polytope. Computing tight ranges of multivariate polynomials can be used in interval Newton solvers, the principle of which is presented in §5. Actually, the Bernstein polytope can be used to propose a new kind of solver, which no more refers to Newton's method, and which is presented in §6. Some technical problems, scaling and inaccuracy, can only be mentioned in §7, for conciseness. The first solver, by Dominique Michelucci, bypasses the inaccuracy issue using rational arithmetic; Christoph Fünfzig then implemented the first robust floating-point CPU variant of this solver (§7.4). §7.3 mentions some theorems used by solvers to certify that a box contains no root, or at least one root, or a unique regular root. §8 is a post-scriptum section which summarizes empirical results detailed elsewhere [13–15]. §9 concludes and lists future works.

2 TBB. Notations. Definitions. Main properties

TBB stands for tensorial Bernstein base. LP stands for linear programming.

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

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

where the binomial coefficient $C(i,d) = \binom{d}{i}$ is the number of i-subsets of a d-set.

The conversion with the canonical base: $(x^0, x^1, \ldots x^d)$ is a linear mapping. Classical formulas are [16]:

$$x^k = (1/C(k,d)) \sum_{i=k}^{d} C(k,i)\, B_i^{(d)}(x)$$

$$x = (1/d) \times \sum_{i=0}^{d} i\, B_i^{(d)}(x)$$

$$x^0 = 1 = \sum_{i=0}^{d} B_i^{(d)}(x) : \text{Their sum equals 1}$$

Main properties are: their sum equals 1, and every $B_i^{(d)}(x)$ is positive for $x \in [0,1]$.

It means 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 minimal or maximal 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: let $p(x)$ be a polynomial in $x \in \mathbb{R}$. 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 $(i/d, p_i)$, where $p(x) = \sum_i p_i B_i(x)$.

Contrarily to coefficients in the usual base: $(1, x, x^2, \ldots x^d)$, 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 $p(x), x \in [0,1]^n$. For this reason, TBB are routinely used in CAD-CAM and Computer Graphics for computing tight covers (*e.g.* voxelizations) of implicit algebraic curves or surfaces (see [11] for instance) in low dimension (2D, 3D, 4D).

3 Definition of the Bernstein polytope

Each monomial with total degree 1 or 2: $x_k, x_i x_j, x_k^2$ is attached a variable of a LP (linear programming) problem. Non linear dependences between monomials x_k and x_k^2, or between monomials $x_i, x_j, x_i x_j$, will be represented resorting to the Bernstein polytope, through linear inequalities constraining corresponding LP variables.

3.1 Univariate case

For univariate polynomials with degree d, the Bernstein polytope is a convex polyhedron which encloses the arc of the curve $(x, x^2, \ldots x^d)$ in $\mathbb{R}^d$, with $x \in [0,1]$. Its hyperplanes and halfspaces are given by $B_i^{(d)}(x) \geq 0, i = 0, \ldots d$; then every monomial x^k is replaced with the corresponding LP variable.

For the degree $d = 2$, the Bernstein polytope is a triangle, in Fig. 1; x and y are the LP variables representing monomials x and x^2; counting multiplicities, each triangle side meets the curve $(x, y = x^2), 0 \leq x \leq 1$, in 2 points. For degree $d = 3$, the Bernstein polytope is a tetrahedron, in Fig. 2; x, y, z are the LP variables representing monomials x, x^2, x^3; counting multiplicities, each plane meets the curve $(x, y = x^2, z = x^3), 0 \leq x \leq 1$ in 3 points. Generalization to higher degrees is obvious.

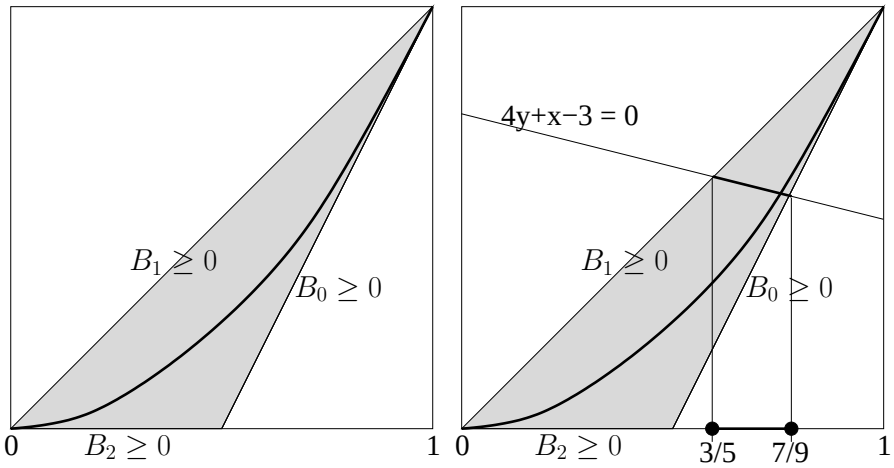

Fig. 1. Left: The Bernstein polytope encloses the curve: $(x, y = x^2)$, for $(x, y) \in [0, 1]^2$. Its limiting sides are: $B_0(x) = (1-x)^2 = y-2x+1 \geq 0$, $B_1(x) = 2x(1-x) = 2x-2y \geq 0$, $B_2(x) = x^2 = y \geq 0$. Right: solving $4x^2 + x - 3 = 0$, with $x \in [0, 1]$, is equivalent to intersecting the line $4y + x - 3 = 0$ with the curve (x, x^2). Linear programming gives the intersection between the line and the Bernstein polytope: the x interval is reduced from $[0, 1]$ to $[3/5, 7/9]$.

3.2 Multivariate polynomials

We extend the Bernstein polytope to multivariate polynomials as follows. The inequalities for multivariate polynomials are obtained as the relevant products of the inequalities for univariate polynomials. We consider only quadratic polynomials, with monomials $x_i, x_i^2, x_i x_j, x_j^2$. The hyperplanes for x_i, x_i^2 have been given. So consider now the non-linear dependences between monomials $x_i, x_j, x_i x_j$. Let us rename them $x, y, z = xy$ to get more usual and intuitive notations. As usual, all variables have values inside the unit interval $[0, 1]$. The surface patch $(x, y, z = xy)$ is enclosed in a convex polyhedron, shown in Fig. 3, whose halfspaces are:

$$B_0^{(1)}(x) \times B_0^{(1)}(y) \geq 0 \Rightarrow (1 - x)(1 - y) \geq 0 \Rightarrow 1 - x - y + z \geq 0$$

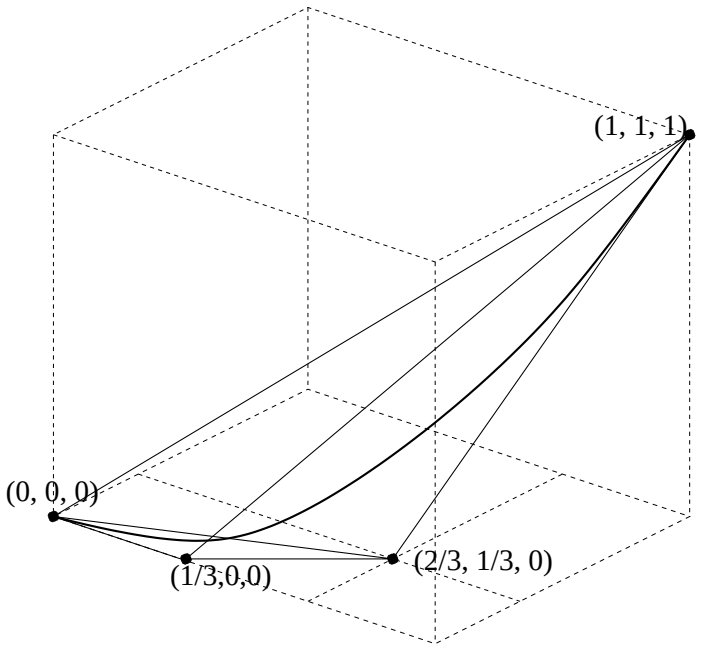

Fig. 2. The Bernstein polytope, a tetrahedron, enclosing the curve $(x, y = x^2, z = x^3)$ with $x \in [0, 1]$. Its vertices are $v_0 = (0, 0, 0)$, $v_1 = (1/3, 0, 0)$, $v_2 = (2/3, 1/3, 0)$ and $v_3 = (1, 1, 1)$. v_0 lies on $B_1 = B_2 = B_3 = 0$, v_1 on $B_0 = B_2 = B_3 = 0$, etc. $B_0(x) = (1-x)^3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

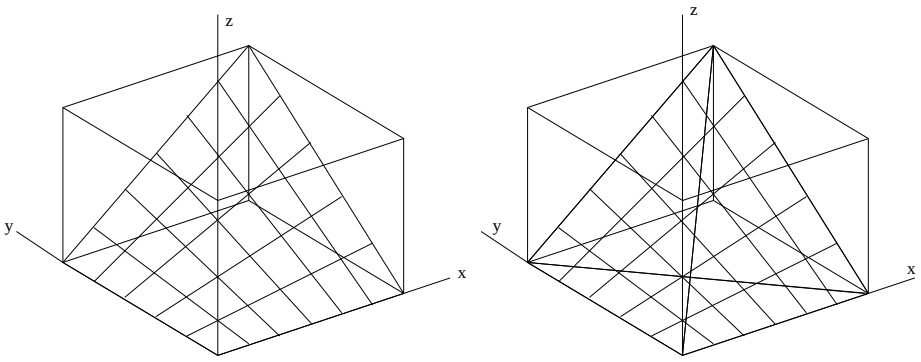

Fig. 3. The Bernstein polytope enclosing the surface patch: $(x, y, z = xy)$. Inequalities of delimiting planes are: $B_i(x)B_j(y) \geq 0$, where $i = 0, 1$ and $B_0(t) = 1-t$ and $B_1(t) = t$.

$$B_0^{(1)}(x) \times B_1^{(1)}(y) \geq 0 \Rightarrow (1-x)y \geq 0 \Rightarrow y - z \geq 0$$
$$B_1^{(1)}(x) \times B_0^{(1)}(y) \geq 0 \Rightarrow x(1-y) \geq 0 \Rightarrow x - z \geq 0$$
$$B_1^{(1)}(x) \times B_1^{(1)}(y) \geq 0 \Rightarrow xy \geq 0 \Rightarrow z \geq 0$$

This tetrahedron is the convex hull of the patch, thus it is optimal. Each of these non-linear inequality $B_i^{(1)}(x) \times B_j^{(1)}(y) \geq 0$ in x and y gives a linear inequality in the LP variables x, y, z. For a quadratic system in n unknowns $x_1, \ldots x_n$, their number is polynomial: $O(n^2)$. It is the same complexity as the number of coefficients of the quadratic polynomials in the canonical base. The extension to higher degrees is easy, but left to the reader for conciseness.

4 Using Linear Programming

The method resorts to LP (Linear Programming) to bypass the problem due to the exponential cardinality of the TBB. This section shows how computing a range for a polynomial $p(x), x = (x_1, \ldots x_n), x \in [0,1]^n$ reduces to solving a linear programming problem on the Bernstein polytope.

4.1 Computing range of polynomials

The method for computing the range of a polynomial is illustrated with a simple example. 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 Fig. 1) enclosing the curve $(x, y = x^2), x \in [0,1]$. It is a LP problem, after replacing x^2 with y. From left to right, the LP tableau of the initial problems, the LP tableau for the minimum, the LP tableau for the maximum, are:

$\min, \max : p = 4y + x - 3$	$\min p = -3 + x + 4y$	$\max p = 2 - 5B_0 - 9/2B_1$
$0 \leq B_0 = y - 2x + 1$	$B_0 = 1 - 2x + y$	$x = 1 - B_0 - B_1/2$
$0 \leq B_1 = -2y + 2x$	$B_1 = 2x - 2y$	$y = 1 - B_0 - B_1$
$0 \leq B_2 = y$	$B_2 = y$	$B_2 = 1 - B_0 - B_1$

The simplex method due to Dantzig performs Gauss pivoting operations on the rows of the initial tableau, to reach the two last tableaux which exhibit the minimum and the maximum. Variables on the left side of the tableaux are basic variables, variables on the right side are non-basic variables and have values 0, with the standard convention in Linear Programming. Let us comment the max tableau. In $\max p = 2 - 5B_0 - 9/2B_1$, the value of p can not be greater than 2, because non-basic variables B_0 and B_1 have values 0, and increasing their values can only decrease p due to their negative coefficients $-5B_0 - 9/2B_1$ in the objective function. The same kind of comments hold for the min part. Thus the polynomial $p(x \in [0,1])$ lies in the range $[-3, 2]$. The minimum occurs at $x = 0$ (x is a non-basic variable for the minimum tableau) so it is exact. The maximum occurs at $x = 1$ (x is a basic variable at line: $x = 1 - B_0 - B_1/2$ in the rightmost tableau) so it is exact too.

Observe that 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 base, or control points, of $p(x)$: $p(x) = p_0 B_0(x) + p_1 B_1(x) + p_2 B_2(x)$.

This feature trivially extends to all univariate polynomials, by definition of the Bernstein polytope.

Remark 1: the Bernstein polytope for univariate polynomials can be tightened, *e.g.* with $B_1^{(2)}(x_i) \leq 1/2$, as in Fig. 4. The number of hyperplanes is still polynomial, and tighter ranges are obtained. It can also improve the reduction of domains. Of course, with this supplementary halfspaces, the minimum and maximum may no more correspond to coefficients in the TBB. Since inequalities for multivariate polynomials are just products of inequalities for univariate ones, the Bernstein polytope for multivariate polynomials (with degree greater than 2) can also be tightened. Adding halfspaces, is not possible with the current TBB solvers which use the primal definition.

Remark 2: A forerunner of this approach is Olivier Beaumont [17]: he used Chebychev polynomials and LP for enclosing multivariate polynomials in his PhD. In the univariate case, Chebychev inequalities are obtained as follows: the monomial x^d is interpolated at the d Gauss points with a degree $d-1$ polynomial $T(x)$; then the error is bounded, which gives inequalities $b_0 \leq x^d - T(x) \leq b_1$. Actually, any approximation scheme, using its own interpolation points, will give inequalities *e.g.* the minimax polynomial. Inequalities in the multivariate case $(x_1, x_2 \ldots x_n)$ are again given by relevant products of univariate inequalities. Chebychev polynomials, or the minimax polynomial, provide other inequalities and other halfspaces, which can be used in place of, or together with, the Bernstein polytope.

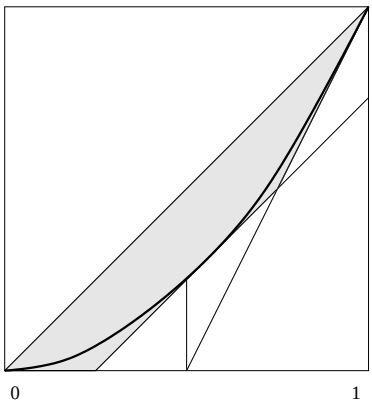

Fig. 4. It is possible to tighten the Bernstein polytope, *e.g.* with the inequality: $x - y \leq 1/4$.

4.2 Domain reduction, preserving roots

This section shows how the solver reduces intervals or boxes, preserving the contained 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 Fig. 1. 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 this time we minimize and maximize x. LP tableaux are:

$\min, \max : x$	$\min x = 3/5 + 2/5B_1$	$\max x = 7/9 - 4/9B_0$
$0 \leq B_0 = y - 2x + 1$	$y = 3/5 - 1/10B_1$	$y = 5/9 + 1/9B_0$
$0 \leq B_1 = -2y + 2x$	$B_0 = 2/5 - 9/10B_1$	$B_1 = 4/9 - 10/9B_0$
$0 \leq B_2 = y$	$B_2 = 3/5 - 1/10B_1$	$B_2 = 5/9 + 1/9B_0$

Thus the interval $[0, 1]$ for x is reduced to $[3/5, 7/9]$, and no root is lost. To further reduce this interval, use the scaling in §7.1 which maps $x \in [3/5, 7/9]$ to $X \in [0, 1]$: $x = 3/5 + (7/9 - 3/5)X = b + aX$, and the equation in X is: $4a^2X^2 + (8ab + b)X + (b - 3) = 0$. Convergence around a regular root is quadratic (like Newton's method) but this concern is not discussed for conciseness.

Remark: if the line does not cut the Bernstein triangle, more generally if the LP problem is not feasible, then it proves that the domain contains no root.

5 Application with interval Newton solvers

TBB based solvers are interval Newton solvers, which rely on TBB properties to compute tight ranges of multivariate polynomials.

This section presents the principle of interval Newton methods, which isolates real roots of a well constrained system $f(x) = 0$, $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$, inside a given initial box B of $\mathbb{R}^n$. Push B on a stack of boxes to be studied. First try to reduce B: compute with some interval method a range B' (an enclosing box) of $N(B)$, where $N(x) = x - f(x)M$, where M is the inverse of the jacobian of f at the center of the box B; as usual, a floating point approximation of the inverse is sufficient, and some LU decomposition can be used instead of an inverse computation. Roots inside B are located in $B \cap B'$. If $B \cap B'$ is empty, B contains no root. Otherwise if $B \cap B'$ is significantly smaller than B, try to reduce $B \cap B'$ again, or, if some Kantorovich test guarantees that there is a unique root inside and that Newton iterations are going to converge (see §7.3), polish the center of $B \cap B'$ with the classical Newton's method, and add the resulting root to a list of solutions. If $B \cap B'$ is not significantly smaller than B, bissect $B \cap B'$ for instance along its longest side, or the side which is the less reduced in the current iteration, and push the two halves on the stack. Actually, a set of residual boxes is typically handled: a box is residual when the box is small and can no more be divided because of the finite accuracy of floating point arithmetic, but the method can not decide on the status of the box: for instance,

it contains a singular root, or very close regular roots, or a "quasi root" (*e.g.* 0 is a "quasi root" of $x^2 + \epsilon = 0$ with ϵ a positive very small number).

The main difficulty in this algorithm, and one topic of this article, is to compute a tight range of $N(B)$.

Computing an ϵ approximation of the exact range is NP-hard, thus researchers in the interval analysis community have proposed several methods to compute in polynomial time a superset of the exact range, with some trade off between time complexity and accuracy. It turns out that TBB provide sharp enclosures.

If all equations are quadratic, it is easy to symbolically compute $N(x)$: it is a set of n quadratic polynomials $P_i(x)$. Each polynomial is defined by $O(n^2)$ coefficients, represented with floating point values, or better with intervals with floating point bounds. We can also in polynomial time apply some scaling (§7.1), so that the studied box B is $[0,1]^n$, and the main problem is then to compute a range of a quadratic polynomial $p(x)$ with $x \in [0,1]^n$. To do that, the method relying on LP is explained in §4.1.

6 A new solver

The principle of the Bernstein polytope can be used to compute tight ranges of multivariate polynomials, and apply to classical interval Newton methods. However the Bernstein polytope, or tightened Bernstein polytopes, make possible new solvers which no more refer to Newton's method.

In this new method, the Bernstein polytope enclosing the quadratic algebraic patch $(x_1, \dots x_n, x_1^2, \dots x_n^2, x_1x_2, \dots x_{n-1}x_n)$, where $x_i \in [0,1]$, is defined as before[2]. Moreover all equations of $f(x) = 0$ are translated into n linear constraints in the LP variables. In passing, quadratic inequalities can also be translated into linear inequalities in the LP variables: this approach deals very easily with inequalities, contrarily to other solvers, say homotopy solvers. Then $2n$ linear optimizations are performed: minimize x_i for $i = 1, \dots n$ and maximize x_i for $i = 1, \dots n$. These problems are independent and can be solved in parallel.

Fig. 1 Right shows this method applied to the equation: $4x^2 + x - 3 = 0$. Let y be the LP variable representing x^2, and x stands for itself. The intersection of the convex polygon and the line $4y + x - 3 = 0$ gives an interval for x: $[3/5, 7/9]$ which encloses the root of the equation: $4x^2 + x - 3 = 0$. This interval is then mapped to $[0,1]$: $x = 3/5 + (7/9 - 3/5)X$, $X \in [0,1]$ with the scaling in §7.1. The same method is then applied to the resulting equation in X. The convergence is quadratic when there is only a regular root. When the box is not significantly reduced (for instance for the equation $x^2 - x = 0$), a bisection is performed as usual. Empirically, almost all bisections separate roots; note that bisections are the only way to separate roots, domain reduction can not.

An advantage of this solver is that preconditionning the system (multiplying equations with the inverse of the jacobian at the center of the box), or inversing

[2] An anonymous reviewer remarks that this patch is reminiscent to the Veronese map.

the jacobian is useless, or more precisely all the work is performed by the simplex method.

Indeed current TBB solvers often need and use some specific procedure to detect as early as possible that the studied box contains no root [6, 10] (§7.3), in order to avoid an exponential number of bisections (*e.g.* to separate two close and "parallel" curves in 2D). A recent article [10] proposed a procedure which also takes into account inequalities $g_i(x) \leq 0$ in the system: $f(x) = 0, g(x) \leq 0$. Its principle is to search with linear programming a polynomial $h(x) = \sum_j \alpha_j f_j(x) + \sum \beta_i g_i(x)$, with $\alpha_j \in \mathbb{R}$ and $\beta_i \geq 0$ such that $h(x)$ is always greater than 1 in the studied box, *i.e.* its smallest coefficient in the TBB is 1: if such an h exists, then the studied box contains no root. It turns out that the new solver proposed in this article early detects boxes containing no root without call to specific procedures: the LP problem solved by the new solver is not feasible.

7 Technicalities

7.1 Scaling

After reduction, the reduced box is no more $[0,1]^n$. A scaling maps the box $[u,v]$, where $u_i \leq v_i$, to the unit hypercube $[0,1]^n$: define $x_i = u_i + (v_i - u_i)X_i$, $w_i = v_i - u_i$, $X \in [0,1]^n$. Then $x_i^2 = w_i^2 X_i^2 + 2u_i w_i X_i + u_i^2$, $x_i x_j = w_i w_j X_i X_j + u_i w_j X_j + u_j w_i X_i + u_i u_j$. Scaling is a linear mapping in the space of the LP variables.

Another approach scales the Bernstein polytope, leaving unchanged equations and inequalities of the system $f(x) = 0, g(x) \leq 0$. For a box $x_i = [u_i, v_i]$ with width w_i, the hyperplanes of the Bernstein polytope are changed as follows (as usual, the monomial x_i^2 is represented with some LP variable q_i in the LP problems, and the monomial $x_i x_j$ with some LP variable x_{ij}).

$B_0^{(2)}(x_i) = (1-x_i)^2 \geq 0$ becomes: $(v_i - x_i)^2 = x_i^2 - 2v_i x_i + v_i^2 = q_i - 2v_i x_i + v_i^2 \geq 0$.

$B_1^{(2)}(x_i) = 2x_i(1-x_i) \geq 0$ becomes: $2(x_i - u_i)(v_i - x_i) = 2(-q_i + (u_i + v_i)x_i - u_i v_i) \geq 0$.

$B_2^{(2)}(x_i) = x_i^2 \geq 0$ becomes: $(x_i - u_i)^2 = q_i - 2u_i x_i + u_i^2 \geq 0$.

$B_0^{(1)}(x_i)B_0^{(1)}(x_j) = (1-x_i)(1-x_j) \geq 0$ becomes: $(v_i - x_i)(v_j - x_j) = x_{ij} - v_i x_j - v_j x_i + v_i v_j \geq 0$.

$B_0^{(1)}(x_i)B_1^{(1)}(x_j) = (1-x_i)x_j \geq 0$ becomes: $(v_i - x_i)(x_j - u_j) = -x_{ij} + u_j x_i + v_i x_j - v_i u_j \geq 0$.

$B_1^{(1)}(x_i)B_1^{(1)}(x_j) = x_i x_j \geq 0$ becomes: $(x_i - u_i)(x_j - u_j) = x_{ij} - u_j x_i - u_i x_j + u_i u_j \geq 0$.

Michelucci's solver uses the first scaling, and Fünfzig's the second scaling.

7.2 Inaccuracy issue

The Bernstein polytope encloses very tightly the underlying algebraic quadratic[3] patch: $(x_1, \dots x_n, x_1^2, \dots x_n^2, x_1x_2, \dots x_{n-1}x_n), x_i \in [0,1]$ thus with a naive floating point implementation, some roots are missed because of rounding errors. For instance, when solving $x^2 - x = 0$ with $x \in [0,1]$, the line $y - x = 0$ is considered (see Fig. 1); if this line becomes $y - x = \epsilon$ with $\epsilon > 0$ due to inaccuracy, the two roots are missed.

For conciseness, we will only mention the principle of three solutions: the first and the simplest one is to resort to an exact rational arithmetic; unfortunately it is terribly slow. Michelucci's solver uses this first solution. We then considered a second solution. It resorts to interval arithmetics [17, 4]; intervals bounds are floating point numbers, and intervals are rounded outwards at each operation. The used intervals are typically some ULPs (Units in the Last Place) large: they only account for the rounding inaccuracy. However, the simplex algorithm has to be modified, so it is impossible to use a pre-existing LP solver. For this reason, we prefered a third approach: interval arithmetics in the second approach is replaced with some error analysis à la Wilkinson. Fünfzig's solver used this approach [15].

7.3 Guarantees and theorems

Interval solvers use procedures to prove that the studied box contains no root, or contains at least one root, or contains a unique regular root. These tests rely on mathematical theorems, *e.g.* Miranda or Kantorovich. This section presents mathematical theorems which fit well with TBB solvers, including the new solver in §6. Details will be given elsewhere in a forthcoming article.

Some solvers require an existence test. Miranda's theorem, also called Poincaré-Miranda's theorem, can be used in TBB solvers [10] to prove that a given box contains at least one root of a system of equation. This theorem states that, under mild assumptions (*e.g.* the continuity of the functions f_i, which is fulfiled in our context), if n continuous functions from $\mathbb{R}^n$ to $\mathbb{R}^n$ are such that each function $f_i(x)$ is always negative on the hyperface $x_i = 0$ of the hypercube $[0,1]^n$ and $f_i(x)$ is always positive on the opposite hyperface $x_i = 1$ for $i = 1, \dots n$, then the system $f_1(x) = \dots = f_n(x) = 0$ has at least one root in the hypercube $[0,1]^n$. The hypothesis of Miranda's theorem is more likely to hold if the system is preconditionned: instead of solving the initial system $f(x) = 0$, a linear combination of the f_i is considered, so that its jacobian is (approximately) the identity matrice at the center of the studied box; this preconditionned system is $g(x) = J(x_c)^{-1} f(x) = 0$.

Some solvers require an uniqueness test. Kantorovich's theorem (also called Newton-Kantorovich's theorem) can be used to prove that a box contains a unique regular root of a system of non-linear equations. This theorem is especially convenient for algebraic quadratic systems, where second derivatives are

[3] This quadratic patch is reminiscent to the Veronese map. But Veronese map is homogeneous and unbounded.

constant. A second computable condition for uniqueness is given by Kim and Elbert [8]: they prove that if the null vector is the only common tangent vector to hypersurfaces $f_i(x) = 0$, then the uniqueness of the root is guaranteed. An equivalent condition is that all enclosing cones of normals of the n hypersurfaces $f_i(x) = 0$ are disjoint; after preconditionning, this condition becomes likely for a small enough box enclosing a unique regular root r: in this case, preconditionning makes hypersurfaces close to orthogonal planes passing through r. A third computable condition which guarantees uniqueness considers the Newton's map: $n(x) = x - Mf(x)$, where $f(x) = 0$ is the system to be solved and where M is close to the inverse of the jacobian of f at the center of the studied box; it also considers the norm of its jacobian: $n'(x) = I - Mf'(x)$: when for some norm $||n'(x)||$ is smaller than 1 in the studied box, then $n(x)$ is guaranteed to be contractant in the studied box, which proves that the root is unique. An upper bound of the max and infinite norms can be computed with interval analysis. In passing, the approach proposed in this article is also able to compute such upper bounds for matrice norms.

Several methods have been proposed to detect quickly that a studied box contains no root [6, 10]. A recent one [10] also takes into account inequalities $g_i(x) \leq 0$ in the system. Its principle is to search with linear programming a polynomial $h = \sum_j \alpha_j f_j + \sum \beta_i g_i$, with $\alpha_j \in \mathbb{R}$ and $\beta_i \geq 0$ such that h is greater than 1 in the studied box (the smallest coefficient in the TBB is 1): if such h exists, then the studied box contains no root. It turns out that the new solver in §6 straightforwardly supersedes this method: the studied box contains no root when the feasible set of the LP problem is empty.

7.4 Michelucci's solver, Fünfzig's solver

Dominique Michelucci implemented the first variant of the new solver in §6 in May 2009. He used exact rational arithmetic to avoid errors due to numerical inaccuracy, and a straightforward simplex solver [18] with rational arithmetic, in Ocaml. His program proves the correctness and the feasibility of the LP reduction approach, but this solver is too slow in practice.

Then Christoph Fünfzig implemented the first floating-point variant of this solver, in January-June 2009, during his post doc in Dijon [15]. For solving LP problems, Christoph's solver relies on the freely available revised simplex solver SoPlex 1.4.1 developped by Roland Wunderling [19, 20] in his PhD thesis. Christoph's solver needs only floating point arithmetic, and routinely solves non-linear systems with several dozens of non-linear algebraic (quadratic or higher degrees) equations and unknowns, which previous Bernstein based solvers are not able to solve. Fünfzig et al [15] generate quadratic systems with arbitrary size from circle-packing representations of planar and completely triangulated graphs. [15] gives other examples from 3D geometric constraints, like the molecule problem (for instance the Stewart platform, also called the octahedron problem), or the computation of the 3D lines tangent to 4 given spheres.

2D examples can be displayed, and permit to visually compare the new solver with a standard interval Newton solver. Figure 5 (from [15]) shows the compu-

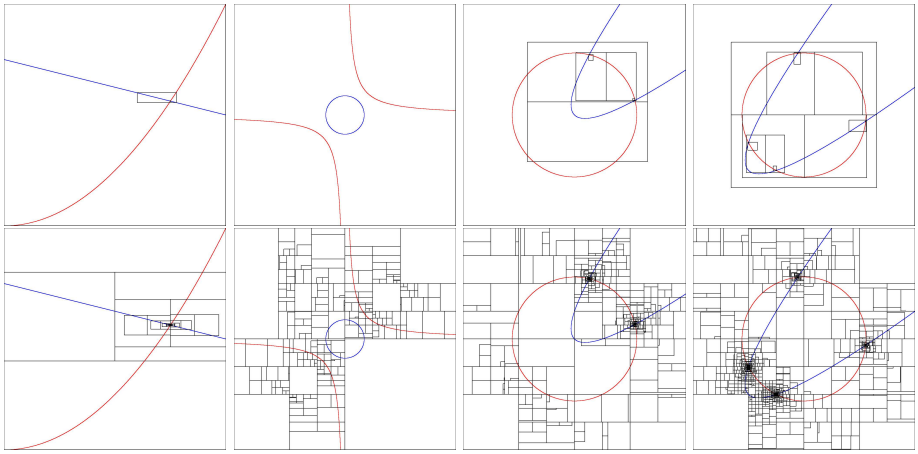

Fig. 5. The same 2D examples are used above and below. Above: the sequence of boxes computed with the new solver; the convergence is super-quadratic around each (regular) solution. Below: the sequence of boxes computed with a standard interval Newton solver. Clearly, the new solver detects much earlier empty boxes, and its convergence rate is greater.

tation on intersection points between two conics, above with the new solver, and below with a standard interval Newton solver.

Both solvers run on CPU. We are considering the project of GPU implementations: for each box reduction, the $2n$ LP problems can be solved in parallel. Moreover, there is some intrinsic parallelism in the simplex method (and in interior-point LP solvers). Thus GPU may divide running times by more than $2n$, where n is the number of unknowns.

8 Empirical results

For completeess, this post-scriptum section summarizes some empirical results detailed elsewhere [13–15]. Fünfzig et al [13] define and compare several Bernstein related polytopes. They also give their volume, their number of vertices, and their number of hyperfaces, for a number of unknowns $n < 5$. These data were computed with David Avis software. Fünfzig et al show that current TBB solvers implicitely use a TBB simplex, which has as many vertices as hyperfaces (simplices): 3^n, which grows exponentially with the dimension n. The conclusion of [13] is that the Bernstein polytope defined in §3 achieves the best trade-off: first it provides polynomial time methods, while methods based on TBB simplex are exponential time; moreover the width of ranges provided by the Bernstein polytope is only 10 or 15 % larger than the ranges provided by the TBB simplex

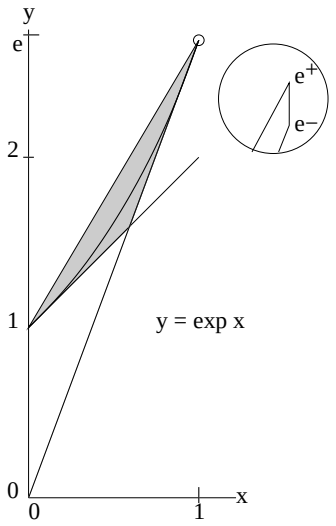

Fig. 6. A convex polygon enclosing the arc of curve $(x, y = \exp x), 0 \leq x \leq 1$. e is enclosed in $[e^-, e^+]$.

(*i.e.* by the smallest and the greatest coefficients in the TBB) for n smaller than 10. For $n > 10$, the TBB becomes terrily slow and unusable: it has to compute 3^n coefficients for enclosing a quadratic polynomial in n unknowns ($3^{10} = 59049$, $3^{11} = 177147$). Fünfzig et al [13] also compare with the interval arithmetic; the latter gives much larger ranges than methods based on the Bernstein polytope or the TBB simplex, and the difference quickly grows with n, the number of unknowns. It is already known [21] that, for computing covers of algebraic curves or surfaces implicitely defined by their equation $f(x, y) = 0$ or $f(x, y, z) = 0$, space subdivision methods which use interval arithmetics are outperformed by methods which rely on TBB: for computing ranges of polynomials, interval arithmetic is faster than Bernstein based methods, but since ranges are much less accurate, more subdivisions are required.

Fünfzig et al [14] compare the new solver relying on the Bernstein polytope with a solver similar to Mourrain et al [6]. For small quadratic systems ($n = 2$ equations and unknowns) equivalent to the computation of intersection points between two conics, Mourrain's solver needs to compute only $3^2 = 27$ coefficients and is about 3 times faster than Fünfzig's solver. In 3 dimensions, Mourrain's solver needs to compute $3^3 = 81$ coefficients, and running times are closer. For $n \geq 7$, Mourrain's solver becomes very slow, due to the exponential number of basis functions in the TBB. In comparison, Fünfzig's solver still works for arbitrarily large systems: Fünfzig generate quadratic systems with arbitrary size from circle-packing representations of planar and completely triangulated graphs. See [13] for details of running times. The problem with 20 unknown circles is solved in less than 8 minutes, which is encouraging for a first imple-

mentation; this problem has almost 60 unknowns: for each circle i in $1, \dots 20$, the unknowns are x_i, y_i the coordinates of the center and r_i the radius. Some optimizations are likely possible. Moreover a GPU program can divide running times by more than $2n$, due to intrinsic parallelism in the simplex method.

Fünfzig et al [15] give other examples from 3D geometric constraints, like the molecule problem (for instance the Gough-Stewart platform, also called the octahedron problem), or the computation of the 3D lines tangent to 4 given spheres. We detail the case of the Gough-Stewart platform. This problem can be expressed in two ways: either with cartesian coordinates, or in a coordinate free way relying on Cayley-Menger determinants [22]. With cartesian coordinates, there are 9 unknowns $(x_i, y_i, z_i), i = 1, 2, 3$, which is too big for TBB solvers like Mourrain's; Fünfzig's solver solves it in 10-16 seconds, depending on the number of roots. For the coordinate free system provided by Cayley-Menger determinants, the system has 2 unknowns and 2 equations with degree 4, and both solvers can solve it. An extra variable is introduced to make the system quadratic; we feed both solvers with the same quadratic system; the TBB solver needs 0.015-0.55 seconds and is about 2-4 times faster than Fünfzig's solver. Clearly the formulation has a strong influence on the running time.

9 Conclusion and future works

This article has proposed the first polynomial time method to overcome the difficulty due to the exponential cardinality of the TBB. It has defined the Bernstein polytope. Here are possible future works and concerns which could not be discussed for conciseness:

Examples of geometric constraints solving, implementations, and implementation issues such as accounting for inaccuracy in the simplex method, are detailed elsewhere [15, 14].

GPU implementations of the new solver are possible and planned for the near future.

The Kantorovich theorem (also called Newton Kantorovich) provides a simple and convenient test to prove the uniqueness of a regular root in a box, especially for quadratic systems where all second derivatives are constant. This concern, not specific to TBB solvers, could not be detailed here, as well as tests deciding the existence of at least one root in a given box, *e.g.* the test relying on Miranda theorem [10].

The Bernstein polytope should be defined for higher degrees, and for other geometric bases, *e.g.* splines bases. It will extend the scope of the geometric solver by Kim and Elbert [8].

The new solver can be generalized in order to manage non-algebraic equations, using for instance transcendental functions $\cos, \exp$, etc. It suffices to compute a convex polygone enclosing the 2D curve: $(x, \cos x)$ and the 2D curve $(x, \exp x)$ for $x \in [a, b]$. Figure 6 shows a possible convex polygon for $(x, \exp x)$ for $x \in [0, 1]$. This feature is a great advantage when compared to other solvers, *e.g.* homotopy.

The new solver applies without modification to over-constrained systems (again, inaccuracy is a serious issue), but more work is needed to extend it to under-constrained systems, and to compute in a certified manner the topology of semi-algebraic or semi-analytic sets defined by a system of equations and inequalities [23, 24]. However we already use the two solvers to compute sharp covers of constrained curves and surfaces [15].

Acknowledgements

We gratefully acknowledge the "Conseil Régional de Bourgogne" (Regional Council of Burgundy) which is funding the post doc position of C. Fünfzig in LE2I, Dijon. This funding has been essential.

References

1. Michelucci, D.: Solving geometric constraints by homotopy. In: IEEE Trans on Visualization and Computer Graphics. (1996) 28–34
2. Durand, C.B.: Symbolic and Numerical Techniques for Constraint Solving. PhD thesis, Purdue University (1998)
3. Sommese, A., Wampler, C.: Numerical solution of polynomial systems arising in engineering and science. World Scientific Press, Singapore (2005)
4. Kearfott, R.: Rigorous Global Search: Continuous Problems. Kluwer, Dordrecht, Netherlands (1996)
5. Garloff, J., Smith, A.P.: Investigation of a subdivision based algorithm for solving systems of polynomial equations. Journal of nonlinear analysis : Series A Theory and Methods **47**(1) (2001) 167–178
6. Mourrain, B., Pavone, J.P.: Subdivision methods for solving polynomial equations. Technical Report RR-5658, INRIA (August 2005)
7. Sherbrooke, E.C., Patrikalakis, N.M.: Computation of the solutions of nonlinear polynomial systems. Comput. Aided Geom. Des. **10**(5) (1993) 379–405
8. Elber, G., Kim, M.S.: Geometric constraint solver using multivariate rational spline functions. In: SMA'01: Proc. of the 6th ACM Symp. on Solid Modeling and Applications, New York, NY, USA, ACM Press (2001) 1–10
9. Reuter, M., Mikkelsen, T.S., Sherbrooke, E.C., Maekawa, T., Patrikalakis, N.M.: Solving nonlinear polynomial systems in the barycentric Bernstein basis. Vis. Comput. **24**(3) (2008) 187–200
10. Michelucci, D., Foufou, S.: Bernstein basis for interval analysis: application to geometric constraints systems solving. In Bruguera, Daumas, eds.: 8th Conference on Real Numbers and Computers. (July 2008) 37–46
11. Martin, R., Shou, H., Voiculescu, I., Bowyer, A., Wang, G.: Comparison of interval methods for plotting algebraic curves (2002)
12. Yamamura, K., Fujioka, T.: Finding all solutions of nonlinear equations using the dual simplex method. J. Comput. Appl. Math. **152**(1-2) (2003) 587–595
13. Fünfzig, C., Michelucci, D., Foufou, S.: Polytope-based computation of polynomial ranges. In: ACM SAC, 25th Symposium On Applied Computing. (Sierre, Switzerland, 2010)

14. Fünfzig, C., Michelucci, D., Foufou, S.: Optimizations for Bernstein-based solvers using domain reduction. In: Proceedings of International Symposium on Tools and Methods of Competitive Enginee ring (TMCE). (April 2010)
15. Fünfzig, C., Michelucci, D., Foufou, S.: Nonlinear System Solver in Floating-Point Arithmetic using LP Reduction. In: SPM'09: Proc. of the ACM Symp. on Solid and Physical Modeling, San-Francisco, California, october 5-8, 2009. to be published
16. Farin, G.: Curves and Surfaces for CAGD: A Practical Guide. Academic Press Professional, Inc., San Diego, CA (1988)
17. Beaumont, O.: Algorithmique pour les intervalles. PhD thesis, IRISA, projet Aladin (1999)
18. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. Second edn. MIT Press, Cambridge, MA (2001)
19. Wunderling, R.: Paralleler und objektorientierter Simplex-Algorithmus. PhD thesis, Technische Universität Berlin (1996)
20. PhD thesis http://www.zib.de/Publications/abstracts/TR-96-09/.
21. Michelucci, D., Foufou, S., Lamarque, L., Schreck, P.: Geometric constraints solving: some tracks. In: ACM Symp. on Solid and Physical Modelling. (2006) 185–196
22. Michelucci, D., Foufou, S.: Using Cayley-Menger determinants for geometric constraint solving. In: SM '04: Proceedings of the ninth ACM symposium on Solid modeling and applications, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association (2004) 285–290
23. Delanoue, N., Jaulin, L., Cottenceau, B.: Guaranteeing the homotopy type of a set defined by nonlinear inequalities. Reliable Computing **13**(5) (2007) 381–398
24. N. Delanoue, L.J., Cottenceau, B.: Using interval arithmetic to prove that a set is path-connected. Theoretical Computer Science, Special issue: Real Numbers and Computers **351**(1) (2006) 119–128