Real Root Approximation Using Fat Spheres*

Sz. Béla†
Institute of Mathematics, Department of Geometry,
Budapest University of Technology and Economics,
Egry József út 1, 1111 Budapest, Hungary
belus@math.bme.hu

B. Jüttler
Institute of Applied Geometry, Johannes Kepler
University, Altenberger Str. 69, 4040 Linz, Austria
bert.juettler@jku.at

Abstract

We present a new algorithm to approximate real roots of multivariate polynomial systems. This algorithm combines the standard subdivision technique with a new domain reduction strategy. We introduce fat spheres as multidimensional quadratic enclosures for algebraic hypersurfaces. Then we present a local reformulation technique of the algebraic system, which provides a method to generate fat spheres. Based on the fat sphere generation we formulate the new domain reduction strategy. The iterative domain reduction generates a sequence of bounding boxes, which converge with order three to the single roots of a multivariate polynomial system.

Keywords: real root approximation, fat spheres, algebraic solver, subdivision
AMS subject classifications: 65D17, 65D18

1 Introduction

Finding real roots of multivariate polynomial systems is frequently needed in various algebraic and geometric computations. Therefore real root finding methods are under active development for a long time. Solvers presented in the literature are based on different algebraic and geometric approaches. A general overview of the multivariate root finding algorithms is given in [8, 18].

Algebraic approaches, such as the Gröbner-basis technique [4], resultant based methods or continuous fractions methods, assure exact and efficient solution algorithms. An algebraic solver, which uses the Gröbner-basis technique, was developed

*Submitted: July 29, 2011; Revised: December 7, 2011; Accepted: July 5, 2012.
†This work was supported by the Austrian Science Fund (FWF) through the Doctoral Program in Computational Mathematics, subproject 3.

for instance by Rouillier [15] for bivariate polynomial systems. Busé et al. considered resultant based methods in [5, 6]. In [9] an algebraic method is described, which uses Sturm-Habicht sequences. However, these algorithms frequently provide more information about the solutions than one needs in certain applications, especially in geometric computing. It is often unnecessary to compute all solutions. For instance CAD-systems usually require information only about real solutions, which lie in a certain domain. Moreover, symbolic methods are not well adapted to numerical computations.

Homotopy solvers compute a family of root-finding problems. These methods transform a simple problem to the original one in several steps, and compute the roots of each intermediate problem. The computed sequence of roots converges to the solutions of the original root-finding problem. Polynomial solvers based on homotopy methods can be found in [12, 13]. However, homotopy methods always compute all solutions of the polynomial system, which may lead to high memory requirements, even if only the solutions within a certain domain are needed for the application.

Geometric modeling is an important application area of real root approximation. In this field of application it is only required to compute real roots of polynomial systems in a bounded domain of n-dimensional space. Subdivision algorithms constitute an important family of the real root finding algorithms. These algorithms compute in a certain domain (usually in an axis-aligned box). They decompose the problem into several sub-problems. The decomposition terminates if suitable approximating primitives can be generated in each sub-problem [14]. In order to construct these approximating primitives, several domain reduction strategies can be combined with standard subdivision techniques. These domain reduction methods are usually based on interpolation, bounding region generation or least-squares approximation.

An essential tool of subdivision based algorithms is to represent the multivariate polynomials by their Bernstein-Bézier form. This representation form is numerically stable, and provides several advantageous properties. The de Casteljau algorithm provides fast a computational method for representing the polynomial system in different regions of the initial computational domain. The convex hull and the variation diminishing properties can be efficiently applied in several steps of the domain reduction strategies.

The first subdivision solvers were developed by Sederberg et al. for bivariate polynomials represented in Bernstein-Bézier tensor product form. They use clipping and subdivision techniques [16, 17]. Later on a family of algorithms was invented which uses projection techniques [19]. Garloff et al. [10] combined a subdivision technique with a pruning step based on the convex hull property of Bernstein polynomials and an existence test based on Miranda's theorem. The most recently developed solvers have been presented by Mourrain et al. [8] and Elber et al. [7].

In this paper we present a new domain reduction strategy, which is based on bounding region generation for algebraic hyper-surfaces. First we introduce a new type of quadratic enclosure and describe a method to bound the algebraic hyper-surfaces. Then we present a domain reduction algorithm, which generates intersecting enclosures to bound the real roots of the polynomial system. In Section 4 we show that the domain reduction strategy generates a sequence of bounding boxes, which converge with order three to the single roots of a multivariate polynomial system. Later on we present a hybrid algorithm, which uses the domain reduction strategy combined with the global subdivision process. Finally we demonstrate the behaviour of the algorithm by several examples.

2 Fat Spheres

In order to bound the real roots of multivariate polynomial systems we introduce a special, multi-dimensional enclosure, the so-called "fat sphere". We present an algebraic approach, which reformulates the algebraic system in order to bound algebraic hyper-surfaces by fat spheres.

2.1 Fat Spheres as Quadratic Enclosures

A segment of an algebraic hyper-surface is given as the zero set of a polynomial in an axis-aligned box $\Omega_0 \subset \mathbb{R}^n$. It allocates the point set

$$\mathcal{C}(f, \Omega_0) = \{\mathbf{x} : f(\mathbf{x}) = 0\} \cap \Omega_0.$$

We consider different segments $\mathcal{C}(f, \Omega)$ of this surface patch in different sub-domains of the initial computational domain $\Omega \subset \Omega_0$. All these sub-domains are assumed to be axis-aligned boxes as well. In order to generate a bounding region for a surface patch $\mathcal{C}(f, \Omega)$ we define fat spheres as follows.

Definition 2.1. *A fat sphere is defined in an axis-aligned box $\Omega \subset \mathbb{R}^n$ by*

- *a multi-dimensional sphere (median sphere) $\mathcal{S} \subset \mathbb{R}^n$,*
- *and a distance $\varrho \in \mathbb{R}$.*

Then the fat sphere is the point set in the box Ω

$$\mathcal{F}(\mathcal{S}, \varrho, \Omega) = \{\mathbf{x} : \exists \mathbf{x}_0 \in \mathcal{S},\ \|\mathbf{x} - \mathbf{x}_0\|_2 \leq \varrho\} \cap \Omega.$$

The fat sphere with one-dimensional median sphere is the so-called fat arc in $\mathbb{R}^2$ (see [2]). In this case the median sphere is a circle.

A multi-dimensional sphere $\mathcal{S}$ can always be defined as an algebraic set. It is the zero set of a special quadratic equation, which possesses the form

$$p = a\langle \mathbf{x}, \mathbf{x}\rangle + \langle \mathbf{b}, \mathbf{x}\rangle + c = 0, \quad a, c \in \mathbb{R},\ \mathbf{b} \in \mathbb{R}^n,$$

where not all coefficients vanish simultaneously. A whole sphere $\mathcal{S}$ is defined algebraically as

$$\mathcal{S} = \{\mathbf{x} : p(\mathbf{x}) = 0\}.$$

The median spheres can also be represented in parametric form with the help of rational functions. It is an advantageous property of arcs and spheres that they possess an exact rational parametric and implicit representation. The implicit representation provides a simple way to represent the offset of the spheres and to compute their intersection, while the parametric form simplifies the visualisation.

2.2 Algebraic Reformulation

We approximate the zero set of the polynomials $F = \{f_1, \ldots, f_n\}$ in the sub-domain $\Omega \subset \mathbb{R}^n$. The geometric interpretation of this problem is to find the intersection points of algebraic hyper-surfaces in the sub-domain Ω. In order to find these intersection points, we compute a new polynomial $\hat{f}$, which has a special Hessian matrix in the center point $\mathbf{c}$ of the sub-domain Ω,

$$\mathcal{H}(\hat{f})(\mathbf{c}) = \begin{pmatrix} \lambda & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda \end{pmatrix} = \lambda\, \mathbf{I}^{n\times n}, \quad \lambda \in \mathbb{R}. \tag{1}$$

We apply the same reformulation technique as presented in [3] to generate approximating arcs for algebraic space curves. We compute a polynomial $\hat{f}$ as the combination of the polynomials $f_i \in F$ with respect to the index set $i \in J \subseteq \{1, \ldots, n\}$ as

$$\hat{f} = \sum_{i \in J} k_i f_i \tag{2}$$

with the linear polynomials k_i defined as

$$k_i(\mathbf{x}) = u_i + \sum_{j=1}^{n} k_i^j (x^j - c^j), \quad k_i^j,\, u_i \in \mathbb{R}, \tag{3}$$

where $\mathbf{c} = (c^j)_{j=1}^n$ is the center point of the sub-domain Ω. The new polynomial $\hat{f}$ has to satisfy (1). The coefficients of k_i can be computed by solving a linear system. In order to obtain not only the trivial solution for the coefficients of the multipliers k_i, we choose for the constant terms u_i of the multipliers arbitrary but fixed, non-zero parameter values.

Lemma 2.2. *Let $N_{\mathrm{J}} = \#J$ denote the number of the elements of the index set $\mathrm{J} \subseteq \{1, \ldots, n\}$. We compute k_i for an arbitrary but fixed parameter vector $\mathbf{u} = (u_i)_{i \in \mathrm{J}}$, where $u_i \neq 0$. If the gradient vectors $\nabla f_i(\mathbf{c})$ are linearly independent and*

$$n - 1 \leq N_{\mathrm{J}}, \tag{4}$$

then a non-trivial polynomial $\hat{f}$ of the form (2) can always be found, which satisfies (1) in the center point $\mathbf{c}$ of a sub-domain Ω.

The proof is based on a comprehensive analysis of the linear system which has to be solved in order to find the linear multipliers k_i. Due to space limitations, we do not present the proof in this paper. The interested reader is referred to [1].

According to Lemma 2.2 we consider only the cases where the combination (2) involves $n - 1$ or n polynomials. In both cases the linear polynomials k_i can be computed for arbitrary but fixed parameters $u_i \neq 0$. In Table 1 we present the dimensions of the linear system which has to be solved in order to find the linear multipliers k_i. The solution space of the coefficients of k_i is at least one dimensional for polynomial systems given in three- or higher dimensional space. However, we need only one collection of coefficients, which defines the multipliers k_i. Therefore we compute the vector of coefficients k_i^j, which has the smallest l_2-norm. Then the multipliers k_i obtained by the construction are unique for each parameter vector $\mathbf{u}$.

Lemma 2.3. *Given the set of polynomials $F = \{f_i : i = 1, \ldots n\}$ over the domain $\Omega \subseteq \Omega_0$, we suppose that for any point $\mathbf{c} \in \Omega_0$ the vector set $\{\nabla f_i(\mathbf{c}) : f_i \in F\}$ is linearly independent. For an arbitrary but fixed vector of parameters $\mathbf{u}$, where $u^i \in \mathbb{R}\backslash\{0\}$, we compute the polynomial $\hat{f}$ with a special Hessian matrix using the linear multipliers with the coefficient vector with minimal l_2-norm. Then $\hat{f}$ depends continuously on the points of the domain Ω_0.*

The proof is available in [1].

After the algebraic reformulation the new polynomials possess a special quadratic Taylor expansion $p = T_{\mathbf{c}}^2 \hat{f}$. The quadratic polynomial defines the algebraic set

$$\mathcal{S} = \{\mathbf{x} : p(\mathbf{x}) = 0\},$$

Table 1: Comparison of strategies to construct polynomials with special Hessians for different numbers of variables. The table shows the number of coefficients and the dimension of their solution space in the construction of a new function $\hat{f}$. For each number of dimensions n, the first row shows the results if we combine $n-1$ polynomials, the second one if we combine n polynomials.

n	number of equations	N_J	number of coefficients	dimension of solution system
2	2	1	2	0
		2	4	2
3	5	2	6	1
		3	9	4
4	9	3	12	3
		4	16	7
5	14	4	20	10
		5	25	11
⋮				
100	5049	99	9900	4851
		100	10000	4951

which is a sphere. This sphere can be used as median sphere for approximating the algebraic hyper-surface $\hat{f} = 0$. The Bernstein-Bézier (BB) norm is the maximum absolute value of the coefficients in the BB-form of the polynomial represented in the domain Ω. With the help of this norm we can bound the difference of the polynomials $\hat{f}$ and p in Ω

$$\varepsilon = \|\hat{f} - p\|_{\mathrm{BB}}^{\Omega}. \tag{5}$$

Due to the convex hull property

$$|\hat{f}(\mathbf{x}) - p(\mathbf{x})| \leq \varepsilon, \quad \forall \mathbf{x} \in \Omega,$$

which implies that

$$p(\mathbf{x}) - \varepsilon \leq \hat{f}(\mathbf{x}) \leq p(\mathbf{x}) + \varepsilon, \quad \forall \mathbf{x} \in \Omega. \tag{6}$$

A fat sphere as bounding region can be defined in Ω for $\hat{f} = 0$ as

$$\mathcal{F}(p, \varepsilon, \Omega) = \{\mathbf{x} : |p(\mathbf{x})| \leq \varepsilon\} \cap \Omega.$$

The boundaries of this region are the offsets of the median sphere $p = 0$. This fat sphere bounds the zero level set of $\hat{f}$.

In the two-dimensional case the fat sphere generation is the same as the fat arc generation (see [2]). The zero level set of polynomials and their approximations are given by implicitly defined curves in $\mathbb{R}^2$. In the three-dimensional space we have two different strategies to generate modified polynomials. We can use either two or all three polynomials from F to generate a new polynomial $\hat{f}$. Then a fat sphere is defined as a thickened region of a three-dimensional sphere.

3 Domain Reduction

In order to find real roots of polynomial systems, we present here a new domain reduction strategy. This domain reduction strategy generates fat spheres to bound the zero set of the algebraic system.

3.1 Domain Reduction Algorithm

In order to bound the zero set of a polynomial system, we present here a domain reduction algorithm, which is applied to the sub-domains of the initial computational domain Ω_0. First it detects the empty sub-domains in the computational domain and eliminates them. Therefore it analyzes the sign changes of the BB-coefficients in the representation of the polynomials. If one of the polynomials has only negative or only positive BB-coefficients over the sub-domain, then no point of the sub-domain belongs to the solution set of the polynomial system. Such sub-domains can be neglected during further computations.

In order to bound the zero set of the polynomials $F = \{f_1, \ldots, f_n\}$, we compute a new system of polynomials with modified Taylor expansion. The technique described in Section 2.2 provides us with a method to compute polynomials $\hat{f}_i$ which possess a special Hessian matrix in the center point of the sub-domain. The set of modified polynomials $\hat{F} = \{\hat{f}_1, \ldots, \hat{f}_n\}$ has a zero set which contains the solution set of the polynomials F in the sub-domain Ω. The quadratic Taylor expansion of the modified polynomials about the center point $\mathbf{c}$ of Ω

$$p_i(\mathbf{x}) = T^2_{\mathbf{c}}(\hat{f}_i)(\mathbf{x}) = \hat{f}_i(\mathbf{c}) + \nabla \hat{f}_i(\mathbf{c})^{\mathrm{T}}(\mathbf{x}-\mathbf{c}) + \frac{1}{2}(\mathbf{x}-\mathbf{c})^{\mathrm{T}}\mathcal{H}(\hat{f}_i)(\mathbf{c})(\mathbf{x}-\mathbf{c})$$

has a zero level set, which is a part of a sphere. Each sphere is used as a median sphere to generate a fat sphere $\mathcal{F}_i(p_i, \varepsilon_i, \Omega)$. Such a fat sphere is the thickened neighbourhood of the median sphere $p_i = 0$, and it contains the zero set of $\hat{f}_i$ in the sub-domain Ω. If all the fat spheres intersect in Ω, then a min-max box is constructed around this intersection (see details in Section 3.2). The domain reduction algorithm returns this min-max box Ω^*as a bounding region of the zero set of the polynomials $f_i \in F$.

If the fat spheres have no intersection, then the sub-domain Ω does not contain any point of the zero set of $\hat{F}$, as well as any point of solution set of F. This implies that no real root of the polynomial system lies in the sub-domain Ω. Thus, such a sub-domain with non-intersecting fat spheres can be neglected in the further computations.

The two-dimensional real root finding algorithm approximates the solution of two bivariate polynomials. In this low dimensional case the median sphere is always a circular arc. In each sub-domain, which is not detected as a region without any root inside, the algorithm `DomainReduction` generates two fat arcs. These are the bounding regions of the two different algebraic curves. The first row of Figure 1 presents some examples of fat arcs and the bounding boxes around their intersections. In the second figure of the first row one can see that the fat arcs intersect each other, however the algebraic curves have no intersection point in the sub-domain. Such "false positive boxes" can be eliminated if we apply the domain reduction iteratively. In the three-dimensional case the algorithm `DomainReduction` bounds the intersection of three algebraic surfaces. The fat spheres are generated as thickened three-dimensional spheres. The second row of Figure 1 presents some examples of these fat spheres and the generated bounding boxes around their intersections.

Algorithm 1 `DomainReduction`(F, Ω)

Require: Each polynomial has a sign change in its BB-coefficients in Ω.

$\hat{f}_i$: modified polynomials with spherical quadratic Taylor expansion p_i
$\mathcal{S}_i = \{\mathbf{x} : p_i(\mathbf{x}) = 0\}$ {median spheres}
$\varepsilon_i = \|\hat{f}_i - p_i\|_{\mathrm{BB}}^{\Omega}$
$\mathcal{F}(p_i, \varepsilon_i, \Omega)$ {fat spheres}
$\mathcal{C} \leftarrow$ extremal points of fat sphere intersection
if $\mathcal{C} \neq \emptyset$ **then**
  $\Omega^* \leftarrow$ min-max box around the points $\mathcal{C}$ {new bounding domain}
  return Ω^*
end if
return $\emptyset$ {no bounding domain has been found}

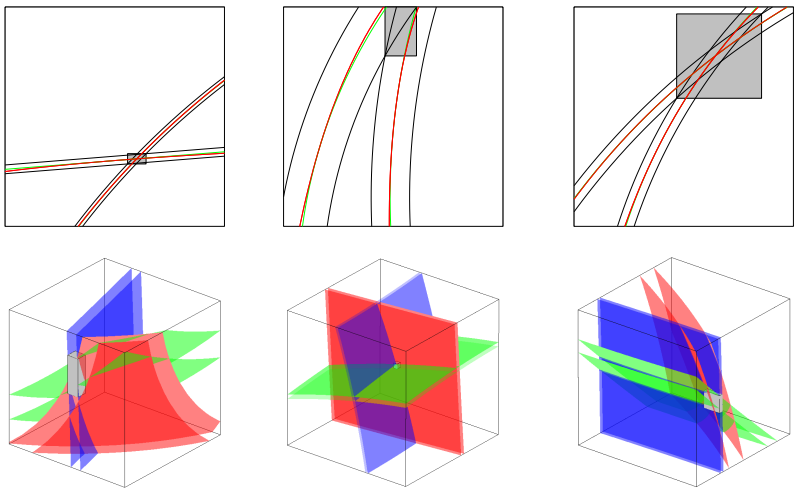

Figure 1: Examples generated by the algorithm `DomainReduction`. In the first row fat arcs are generated to bound the intersection of planar algebraic curves. The algebraic curves are red, the median circles are shown in green. The gray regions represent the generated bounding regions: the min-max boxes around the intersections of fat arcs. In the second row fat spheres are shown. The boundary patches of the three fat spheres are represented in red, green and blue. The gray regions represent the generated min-max boxes around the intersections of fat spheres.

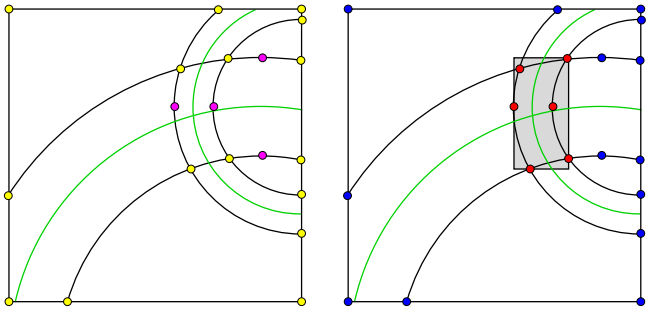

Figure 2: Extremal points of fat arc intersection. The fat arcs are represented by their bounding arcs (black) and the median arcs (green). In figure (a) the fat sphere corner points are marked by yellow dots and the fat sphere extreme points (fat sphere 1-extrema) by pink ones. The important fat sphere extrema are marked by red dots in figure (b). The bounding region of the fat arc intersection is the min-max box generated around the important fat sphere extrema (gray rectangle).

3.2 Min-max Box of the Intersection of Fat Spheres

In order to bound the zero set of the polynomials $\hat{F}$, we consider the intersection of the generated fat spheres. Each fat sphere

$$\mathcal{F}_i(p_i, \varepsilon_i, \Omega) = \{\mathbf{x} \,:\, |p_i(\mathbf{x})| \leq \varepsilon_i\} \cap \Omega$$

bounds the zero level set of the polynomial $\hat{f}_i$ in the sub-domain Ω. If the intersection of fat spheres is not empty,

$$\mathcal{I} = \bigcap_{i=1}^{n} \mathcal{F}_i(p_i, \varepsilon_i, \Omega) \neq \emptyset,$$

then it contains the zero set of the polynomials $\hat{f}_i$ in the sub-domain Ω. The region $\mathcal{I}$ is a "curved polytope", which is bounded by spherical patches and linear subspaces. The spherical patches are a part of the boundary surfaces of the fat spheres. The pair of bounding spheres of the fat sphere $\mathcal{F}_i(p_i, \varepsilon, \Omega)$ can be described as the point set

$$\mathcal{P}_i = \{\mathbf{x} \,:\, p_i(\mathbf{x}) = \pm\varepsilon_i\} \cap \Omega.$$

The segments of linear subspaces which bound the fat sphere intersection are a part of the boundaries of the sub-domain Ω. If the sub-domain is of the form $\Omega = \times_{i=1}^{n}[\alpha_i, \beta_i]$, then the ith boundary pair of the domain Ω is defined as

$$\partial\Omega_i = \{\mathbf{x} \,:\, x^i = \alpha_i \vee x^i = \beta_i\} \cap \Omega.$$

An example for a two-dimensional fat arc intersection is shown in Figure 2. Each fat arc is the intersection of the computational domain and an annulus. The intersection of two fat arcs is bounded by a curved polygon. The boundaries of the polygon are circular arcs.

In general it is not practical to use the intersection of fat spheres as computational domains in further domain reductions. In order to reduce iteratively the bounding regions, the output of the domain reduction has to be an axis-aligned box. Therefore we compute the min-max box, which bounds the fat sphere intersection $\mathcal{I}$. This box can be computed exactly, by finding the extremal points of the fat sphere intersection. For instance, in Figure 2(b) the extremal points of the fat arc intersection are marked by red dots. Four of these extrema are the intersection points of the fat arc boundaries, while another one is an extremal point of a boundary arc. In order to find the extremal points of the fat sphere intersection in general, we use the following definitions.

Definition 3.1. *Let N_S denote the number of the elements of an index set $S \subseteq \{1,\dots,n\}$. A point $\mathbf{x} \in \Omega$ is called*

(i) **fat sphere corner point** *if $I, J \subset \{1,\dots,n\}$, $N_I = k$, $N_J = n-k$:*

$$\mathbf{x} \in \mathcal{X} = \bigcap_{i\in I} \partial\Omega_i \bigcap_{j\in J} \mathcal{P}_j$$

(ii) **fat sphere m-extreme point** *if $I, J \subset \{1,\dots,n\}$, $N_I = k$, $N_J < n-k$*

$$\mathbf{x} \in \mathcal{Y} = \bigcap_{i\in I} \partial\Omega_i \bigcap_{j\in J} \mathcal{P}_j,$$

where $\mathcal{Y}$ is an m-dimensional algebraic object, and there exists $n-k-m$ different indexes $l \in \{1,\dots,n\} \setminus I$, such that each of them satisfies

$$\frac{\partial p_j}{\partial x_l} = 0.$$

All corner points $\mathbf{x}$ of the sub-domain Ω are fat sphere corner points for $k = n$

$$\mathbf{x} \in \bigcap_{i=1}^{n} \partial\Omega_i \subset \mathcal{X}.$$

All intersection points $\mathbf{x}$ of the fat sphere boundaries, which lie in the interior of the domain Ω, are in the point set

$$\mathbf{x} \in \bigcap_{j=1}^{n} \mathcal{P}_i \subset \mathcal{X},$$

These points are fat sphere corner points with $k = 0$.

Definition 3.2. *We call a fat sphere corner point or a fat sphere extreme point $\mathbf{x}$ an important fat sphere extrema, if it satisfies for all $i \in \{1,\dots,n\}$*

$$-\varepsilon_i \le p_i(\mathbf{x}) \le \varepsilon_i,$$

thus the point $\mathbf{x}$ belongs to the intersection $\mathcal{I}$ of the fat spheres.

According to these definitions, all fat sphere corner points and fat sphere extreme points are defined by an equation system with n equations in n variables, where all equations are linear or quadratic. The quadratic equations are the equations for spheres. Therefore each fat sphere corner point and fat sphere extreme point can be computed as the solution of an equation system consisting of $n-1$ linear equations

and a single quadratic equation. So all important fat sphere extrema can be computed by solving a finite number of algebraic systems, where each system consists of $n-1$ linear equations plus one linear or quadratic equation and at most $2n$ inequality tests. The min-max box around the region $\mathcal{I} = \bigcap_{i=1}^{n} \mathcal{F}_i(p_i, \varepsilon_i, \Omega) \neq \emptyset$, which is the fat sphere intersection in the sub-domain Ω, is the min-max box around the important fat sphere extrema. This axis-aligned box contains all points of the zero set of F, which lie in the sub-domain Ω. Therefore this box is a reduced bounding region of the zero set of F in the sub-domain Ω.

Figure 2(b) shows a two-dimensional fat arc intersection, where the fat sphere corner points and the fat sphere extreme points are marked by red and blue dots. The red ones denote the important fat sphere extrema. This computational method is sufficiently fast and effective to reduce bounding regions, for low dimensional examples according to our experiments. Nevertheless, later on the extremal point computation can cause problems, as the number of computed points increases exponentially with the number of dimensions n.

4 Convergence of Domain Reduction

We bound the zero sets of polynomials with the help of quadratic enclosures. Therefore we expect that the rate of convergence of the sequence of bounding regions is equal to three. This expectation is confirmed for single roots of polynomial systems in Theorem 4.6 at the end of this section.

If we assume that the polynomials F possess a single root $\mathbf{r}$ in a domain, then the gradient vectors of the polynomials are linearly independent in the point $\mathbf{r}$. Thus the implicitly defined hyper-surfaces, defined by the zero set of the polynomials, intersect each other transversely at the root. Moreover, there exists a domain Ω_0 around the root $\mathbf{r}$, such that for any point $\mathbf{x} \in \Omega_0$ it holds that

$$\det(J(F)(\mathbf{x})) \neq 0, \tag{7}$$

where J denotes the Jacobian matrix of the polynomials F. Thus the gradient vectors $\nabla f_1(\mathbf{x}), \nabla f_2(\mathbf{x}), \ldots, \nabla f_n(\mathbf{x})$ are linearly independent for all $\mathbf{x} \in \Omega_0$. Therefore we suppose that any point of the initial domain Ω_0 fulfills (7).

The algorithm `DomainReduction` computes first a set of modified polynomials $\hat{F}$. Each point of Ω_0 fulfills (7), so the gradient vectors $\nabla f_i(\mathbf{x})$ do not vanish. If we compute in a sufficiently small sub-domain of Ω_0, it can be shown that each modified polynomial has a positive lower bound on the gradient length. Therefore also the quadratic Taylor expansions of the modified polynomials are non-zero polynomials. The following lemma certifies that the gradient vectors of the modified polynomials are linearly independent in a sufficiently small sub-domain of Ω_0.

Lemma 4.1. *Suppose that the gradient vectors $\nabla f_1(\mathbf{x}), \nabla f_2(\mathbf{x}), \ldots, \nabla f_n(\mathbf{x})$ of the polynomials $f_i \in F$ are linearly independent for all $\mathbf{x} \in \Omega_0$. Consider a sub-domain $\Omega \subseteq \Omega_0$, which has a diameter $\delta_\Omega < \varepsilon$. We compute the set of modified polynomials $\hat{F}$ in the sub-domain Ω for the arbitrary but fixed vectors of constants $\mathbf{u}_i$, which are linearly independent. If ε is sufficiently small, then for all $\mathbf{x} \in \Omega$*

$$\det(J(\hat{F})(\mathbf{x})) \neq 0.$$

Proof. The gradient vectors of f_i are linearly independent in any point of Ω_0, therefore there exists a constant $K > 0$, such that all $\mathbf{x} \in \Omega_0$ satisfy

$$|\det(J(F)(\mathbf{x}))| \geq K > 0.$$

We compute the set of polynomials $\hat{F}_\mathbf{c}$ with special Hessians in a certain point $\mathbf{c} \in \Omega_0$ for the fixed vectors of constants $\mathbf{u}_i$. Then the gradient vectors of $\hat{f}_i \in \hat{F}_\mathbf{c}$ in the point $\mathbf{c}$ can be expressed as

$$\nabla \hat{f}_i(\mathbf{c}) = \sum_{j=1}^{n} u_i^j \nabla f_i(\mathbf{c}).$$

The vectors of constants $\mathbf{u}_i$ define the matrix $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$. Since the vectors $\mathbf{u}_i$ are linearly independent, the determinant of $\mathbf{U}$ is a positive constant U

$$|\det(\mathbf{U})| = U > 0.$$

Therefore the determinant of the Jacobian of $\hat{F}_\mathbf{c}$ at the point $\mathbf{c}$ satisfies

$$|\det(J(\hat{F}_\mathbf{c})(\mathbf{c}))| = \left|\det\left(\mathbf{U}^{\mathrm{T}} \cdot J(F)(\mathbf{c})\right)\right| = |\det(\mathbf{U})| \cdot |\det(J(F)(\mathbf{c}))| \geq UK > 0.$$

Suppose that $\Omega \subseteq \Omega_0$ is a sub-domain with center point $\mathbf{c}$. The set of new polynomials computed at a point $\mathbf{c}$ is $\hat{F}_\mathbf{c}$. Then there exists $\varepsilon_\mathbf{c} > 0$, such that if the diameter δ_Ω of the sub-domain Ω is smaller than $\varepsilon_\mathbf{c}$, for all $\mathbf{x} \in \Omega$

$$|\det(J(\hat{F}_\mathbf{c})(\mathbf{x}))| > 0.$$

Lemma 2.3 implies that for fixed vectors of constants $\mathbf{u}_i$ the system of polynomials $\hat{F}_\mathbf{c}$ depends continuously on the point $\mathbf{c}$. Thus there exists a general bound $\varepsilon > 0$, such that for any sub-domain $\Omega \subseteq \Omega_0$, which has the diameter $\delta_\Omega < \varepsilon$, any $\mathbf{x} \in \Omega$ satisfies

$$|\det(J(\hat{F})(\mathbf{x}))| > 0,$$

where $\hat{F}$ is the set of polynomials with special Hessians in the center of the sub-domain Ω. □

Corollary 4.2. *The median spheres are the zero sets of the quadratic Taylor expansions of $\hat{f}_i$ about the center of the sub-domain Ω*

$$p_i = T^2_\mathbf{c}(\hat{f}_i)(\mathbf{x}).$$

If the diameter of Ω is sufficiently small, then for all $\mathbf{x} \in \Omega$

$$\det(J(p_1, \ldots p_n)(\mathbf{x})) \neq 0.$$

Proof. The construction of p_i implies that

$$|\det(J(\hat{f}_1, \ldots, \hat{f}_n)(\mathbf{c}))| = |\det(J(p_1, \ldots, p_n)(\mathbf{c}))|.$$

The polynomials $\hat{f}_i$ depend continuously on the point $\mathbf{c}$, as their quadratic Taylor expansions p_i do. According to the proof of Lemma 4.1 there exists a general bound $\varepsilon > 0$, such that if the diameter δ_Ω of the sub-domain Ω is smaller than ε, then any $\mathbf{x} \in \Omega$ satisfies

$$|\det(J(p_1, \ldots, p_n)(\mathbf{x}))| > 0.$$

□

We computed the fat sphere boundaries as concentric spheres to the median sphere $p_i = 0$. These spheres are defined by the equations

$$p_i = \pm\varepsilon_i,$$

where ε_i is computed as

$$\varepsilon_i = \|\hat{f}_i - p_i\|_{\mathrm{BB}}^{\Omega}.$$

Lemma 4.3. *We compute a polynomial $\hat{f}_i$ with special Hessian at the center point of the sub-domain Ω. Let ε_i denote the bound*

$$\varepsilon_i = \|\hat{f}_i - T_{\mathbf{c}}^2(\hat{f}_i)\|_{\mathrm{BB}}^{\Omega}.$$

Then it satisfies

$$\varepsilon_i \leq C\mathrm{diam}(\Omega)^3.$$

Proof. The sub-domain Ω is an axis-aligned box. Since all norms are equivalent on finite dimensional vector spaces, there exists a constant C_1, such that

$$\varepsilon_i = \|\hat{f}_i - p_i\|_{\mathrm{BB}}^{\Omega} \leq C_1\|\hat{f}_i - p_i\|_{\infty}^{\Omega},$$

and C_1 does not depend on Ω. If the center point of Ω is denoted by $\mathbf{c}$, then

$$\|\hat{f}_i - p_i\|_{\infty}^{\Omega} = \|\hat{f}_i - T_{\mathbf{c}}^2(\hat{f}_i)\|_{\infty}^{\Omega} < \frac{1}{6}\underbrace{\max_{\mathbf{v}\in S^1,\mathbf{x}\in\Omega}\left|\frac{\mathrm{d}^3\hat{f}_i}{\mathrm{d}\mathbf{v}^3}(\mathbf{x})\right|}_{(*)}\mathrm{diam}(\Omega)^3.$$

Recall from Lemma 2.3 that $\hat{f}_i$ depends continuously on the points of the computational domain Ω_0 for each parameter vector of $\mathbf{u}$, where $u^j \neq 0$. Thus for all Ω a global upper bound C_2 can be given for $(*)$. Therefore we observe that

$$\varepsilon_i \leq \frac{1}{6}C_1C_2\mathrm{diam}(\Omega)^3 \leq C\mathrm{diam}(\Omega)^3.$$

□

In order to measure the longest diameter of the intersection of fat spheres we give a general lower bound on the gradient of a certain function.

Lemma 4.4. *Consider the function $h = \sqrt{\sum_{i=1}^n q_i^2}$ defined by the set of polynomials $Q = \{q_1, \ldots, q_n\}$. We assume that the Jacobian matrix is not singular in any $\mathbf{x} \in \Omega$*

$$\det(J(Q)(\mathbf{x})) \neq 0.$$

For all $\mathbf{x} \in \Omega$, which do not satisfy $q_i(\mathbf{x}) = 0$ for all $i = 1, \ldots, n$, there exists a positive constant L_Ω such that

$$\|\nabla h(\mathbf{x})\|^2 \geq L_\Omega > 0.$$

Proof. Since

$$\nabla h(\mathbf{x}) = \frac{\sum_{i=1}^n q_i\nabla q_i}{\sqrt{\sum_{i=1}^n q_i^2}},$$

we obtain

$$\|\nabla h(\mathbf{x})\|^2 = \left\langle \frac{\sum_{i=1}^n q_i\nabla q_i}{\sqrt{\sum_{i=1}^n q_i^2}}, \frac{\sum_{i=1}^n q_i\nabla q_i}{\sqrt{\sum_{i=1}^n q_i^2}} \right\rangle =$$

$$= \frac{\mathbf{q}(\mathbf{x})}{\|\mathbf{q}(\mathbf{x})\|}^{\mathrm{T}} J(Q)(\mathbf{x}) J(Q)(\mathbf{x})^{\mathrm{T}} \frac{\mathbf{q}(\mathbf{x})}{\|\mathbf{q}(\mathbf{x})\|} \geq \min_{\|\mathbf{v}\|=1} \mathbf{v}^{\mathrm{T}} \mathrm{Gram}(\nabla q_1(\mathbf{x}), \ldots, \nabla q_n(\mathbf{x}))\mathbf{v},$$

where $\mathbf{q}(\mathbf{x})^{\mathrm{T}} = (q_1(\mathbf{x}), \ldots, q_n(\mathbf{x}))$. We assumed that $J(Q)(\mathbf{x})$ is not singular, therefore $\mathrm{Gram}(\nabla q_1(\mathbf{x}), \ldots, \nabla q_n(\mathbf{x}))$ is also non-singular. Moreover it is symmetric. Thus for all $\mathbf{x} \in \Omega$

$$\|\nabla h(\mathbf{x})\|^2 \geq \lambda(\mathbf{x}) > 0,$$

where $\lambda(\mathbf{x})$ is the minimal eigenvalue of the Gram matrix. Since the Gram matrix is not singular, and it depends continuously on the points of Ω, there exists a positive lower bound L_Ω depending on Ω, such that

$$\lambda(\mathbf{x}) \geq L_\Omega > 0.$$

□

Lemma 4.5. *Consider the sub-domains $\Omega_{\mathbf{c}} \subset \Omega_0$, where $\mathbf{c}$ is the center point of the sub-domain. A set of polynomials $Q_{\mathbf{c}}$ is given for each sub-domain $\Omega_{\mathbf{c}}$. We assume that the Jacobian matrix of the polynomial system $Q_{\mathbf{c}}$ is not singular in any $\mathbf{x} \in \Omega_{\mathbf{c}}$. We consider the function*

$$h_{\mathbf{c}} = \sqrt{\sum_{i=1}^{n} q_i^2}$$

defined by the polynomials $q_i \in Q_{\mathbf{c}}$. For all $\mathbf{x}$ from the sub-domain $\Omega_{\mathbf{c}}$, which do not satisfy $q_i(\mathbf{x}) = 0$ for $i = 1, \ldots, n$, there exists a general positive constant L such that

$$\|\nabla h_{\mathbf{c}}(\mathbf{x})\|^2 \geq L > 0.$$

Proof. Each polynomial $q_i \in Q_{\mathbf{c}}$ depends continuously on the choice of the point $\mathbf{c}$. According to Lemma 4.4 there exists a lower bound of $\|\nabla h_{\mathbf{c}}(\mathbf{x})\|^2$ for all $\mathbf{x} \in \Omega_{\mathbf{c}}$, which bounds the minimal eigenvalue of the Gram matrix of $q_i \in Q_{\mathbf{c}}$. Therefore for all $\Omega_{\mathbf{c}}$, where $\det(J(Q_{\mathbf{c}})(\mathbf{x})) \neq 0$, there exists a general positive lower bound L, such that any $\mathbf{x} \in \Omega_{\mathbf{c}}$ satisfies

$$\|\nabla h(\mathbf{x})\|^2 \geq L > 0,$$

if $\mathbf{x}$ does not satisfy $q_i(\mathbf{x}) = 0$ for all $q_i \in \Omega_{\mathbf{c}}$. □

Theorem 4.6. *Suppose that the gradient vectors of the polynomials $f_i \in F$ are linearly independent for all points $\mathbf{x} \in \Omega_0$. Consider a sub-domain $\Omega \subseteq \Omega_0$, which is sufficiently small and contains a single root $\mathbf{r}$ of the polynomials f_i. We compute the set of polynomials $\hat{F}$ with special Hessians in the center point of the domain Ω for the arbitrary but fixed vectors of constants $\mathbf{u}_i$, which are linearly independent. If we apply the algorithm* `DomainReduction` *to the sub-domain Ω, then there exists a constant C, such that the generated bounding region Ω^* satisfies*

$$\mathrm{diam}(\Omega^*) \leq C \mathrm{diam}(\Omega)^3.$$

Proof. Suppose that Ω is a sub-domain of Ω_0, which contains a single root $\mathbf{r}$. We compute the set of polynomials $\hat{F}$ with special Hessians in the center point $\mathbf{c}$ of Ω. The fat spheres are defined by the point sets $\mathcal{F}_i(p_i, \varepsilon_i, \Omega)$. The fat sphere intersection is denoted by $\mathcal{I} = \bigcap_{i=1}^{n} \mathcal{F}_i$. Each fat sphere bounds the hyper-surface $\hat{f}_i = 0$, thus the single root $\mathbf{r} \in \Omega$ is contained in the fat sphere intersection

$$\mathbf{r} \in \mathcal{I} \cap \Omega.$$

We define the function

$$h(\mathbf{x}) = \sqrt{\sum_{i=1}^{n} q_i^2},$$

where $q_i(\mathbf{x}) = p_i(\mathbf{x}) - p_i(\mathbf{r})$. We consider the integral curves defined by the vector field $-\nabla h/\|\nabla h\|$ in Ω. If Ω has a sufficiently small diameter, according to Corollary 4.2 all $\mathbf{x} \in \Omega$ satisfy

$$\det(J(p_1, \dots p_n)(\mathbf{x})) \neq 0.$$

Since $\nabla p_i(\mathbf{x}) = \nabla q_i(\mathbf{x})$, for all $\mathbf{x} \in \Omega$

$$\det(J(q_1, \dots q_n)(\mathbf{x})) \neq 0. \tag{8}$$

Together with Lemma 4.4 this implies that the integral curves are regular in the inner points of $\Omega \setminus \{\mathbf{r}\}$.

Suppose that $\mathbf{x}$ is an arbitrary point of the fat sphere intersection $\mathcal{I}$ computed in a sufficiently small domain Ω. Such a point $\mathbf{x} \in \mathcal{I} \cap \Omega$ fulfills for all $i = 1, \dots n$

$$|p_i(\mathbf{x})| \leq \varepsilon_i.$$

We consider the integral curve $\mathbf{u}(s)$ with the starting point $\mathbf{u}(0) = \mathbf{x} \in \mathcal{I}$, which is regular on $\Omega \setminus \{\mathbf{r}\}$. We assume that the curve is parameterised by arc length. Since $h(\mathbf{x}) \geq 0$ and the tangent vectors of the curve $\mathbf{u}(s)$ always point in the direction of steepest decent on h, there exists a parameter value s^* such that for $s < s^*$

$$\lim_{s \to s^*} \mathbf{u}(s) = \mathbf{r}.$$

According to the mean value theorem there exists $\xi \in (0, s^*)$ such that

$$\frac{h(\mathbf{u}(s^*)) - h(\mathbf{u}(0))}{s^*} = \nabla h(\mathbf{u}(\xi)) \cdot \dot{\mathbf{u}}(\xi) = -\|\nabla h(\mathbf{u}(\xi))\|.$$

Since $h(\mathbf{u}(s^*)) = 0$

$$s^* = \frac{h(\mathbf{u}(0))}{\|\nabla h(\mathbf{u}(\xi))\|} = \frac{h(\mathbf{x})}{\|\nabla h(\mathbf{u}(\xi))\|} \leq \sqrt{\frac{2\sum_{i=1}^{n} \varepsilon_i^2}{L_\Omega}}.$$

We supposed that $\mathbf{u}(s)$ is arc length parametrized, therefore $\mathbf{x} \in \mathcal{I}$ satisfies

$$\|\mathbf{x} - \mathbf{r}\| = \|\mathbf{u}(0) - \mathbf{u}(s^*)\| \leq \sqrt{\frac{2\sum_{i=1}^{n} \varepsilon_i^2}{L_\Omega}}.$$

Thus any point of $\mathcal{I}$ is closer to $\mathbf{r}$ than $\sqrt{\frac{2\sum_{i=1}^{n} \varepsilon_i^2}{L_\Omega}}$. So the min-max box $\Omega^* \subset \Omega$, which contains $\mathcal{I}$, has a diameter

$$\operatorname{diam}(\Omega^*) \leq 2\sqrt{\frac{2n\sum_{i=1}^{n} \varepsilon_i^2}{L_\Omega}}.$$

In Lemma 2.3 we have shown that the system of polynomials $\hat{F}$ depends continuously on the choice of the domain Ω. Therefore also each p_i and q_i depend continuously on the choice of Ω. The lower bound L_Ω of $\|\nabla h(\mathbf{x})\|^2$ bounds the minimal eigenvalue

of the Gram matrix of q_i. According to Lemma 4.4 there exists a general positive lower bound L, such that any $\mathbf{x} \in \Omega$ satisfies

$$\|\nabla h(\mathbf{x})\|^2 \geq L > 0.$$

We have also shown in Lemma 4.3 that there exists a constant D, which does not depend on the choice of Ω, such that

$$\varepsilon_i \leq D\mathrm{diam}(\Omega)^3.$$

Therefore the diameter of the min-max box Ω^* satisfies

$$\mathrm{diam}(\Omega^*) \leq 2\sqrt{\frac{2n\sum_{i=1}^n \varepsilon_i^2}{L}} \leq \frac{2\sqrt{2}Dn}{\sqrt{L}}\mathrm{diam}(\Omega)^3 = C\mathrm{diam}(\Omega)^3,$$

where C does not depend on the choice of Ω. □

In this result we used the condition that the polynomials possess a single root in the computational domain. However, the `DomainReduction` algorithm can approximate double roots of polynomial systems, too. Later we show an example (see Example 5.3) for double root approximation. As Table 3 shows, the algorithm reduces the bounding region around a double root faster than the ordinary subdivision. Our further numerical experiments indicate that the `DomainReduction` algorithm combined with iterative subdivision provides super-linear convergence rate to double roots of a polynomial system.

5 Subdivision Method

In this section we present an algorithm, which combines the `DomainReduction` with a standard subdivision technique. It is an iterative domain reduction, which reduces the bounding regions either by subdivision or by bounding the intersection of fat spheres.

5.1 Hybrid Algorithm

The global root approximation algorithm (see Algorithm 2) is an iterative domain reduction, which bounds the roots of a multivariate polynomial system F within a prescribed tolerance bound ε. The algorithm computes a set of axis-aligned boxes with the help of hierarchical subdivision and fat sphere intersection. Each root of the system is approximated via a nested sequence of domains, which have decreasing diameters. The algorithm reduces the domains, until each list of nested domains has an element with sufficiently small diameter. Then the algorithm returns the last element of the lists.

Each sub-domain is analyzed, until it is detected as an empty region or it has a sufficiently small diameter. We detect empty sub-domains via the convex hull property. A sub-domain is also empty, if the algorithm `DomainReduction` generates fat spheres which do not intersect. Then the algorithm does not analyze these domains any further. Nevertheless, it can happen that a sub-domain without a root is computed with small diameter, but it is not detected as an empty region. Thus the output can also contain empty sub-domains (false positive boxes).

It is also important to separate the real roots of polynomials into different bounding domains. In some cases we can certify whether a domain in the output contains only

Algorithm 2 `Hybrid Algorithm`(F, Ω, ε)

```
1: A ← DomainReduction(F, Ω)                          {domain reduction}
2: if 2 · diam(A) ≤ diam(Ω) then
3:    if diam(A) > ε then
4:       Hybrid Algorithm(F, A, ε)                    {recursive call}
5:    else
6:       B = B ∪ A
7:    end if
8: else
9:    if diameter of Ω > ε then
10:      subdivide the domain Ω to Ω_i                 {subdivision}
11:      Hybrid Algorithm(F, Ω_i, ε)                   {recursive call}
12:   else
13:      B = B ∪ Ω
14:   end if
15: end if
16: return B
```

one single root, although this is not always possible. If two real roots have smaller distance than the tolerance ε, they may share their bounding region in the output of the algorithm. Therefore clearly the number of bounding regions in the output is not necessarily equal to the number of real roots of the polynomial system.

As we described already in Section 2.2 we compute polynomials with special Hessians as the combination of $n-1$ or n different polynomials from the original set of polynomials F. If we only combine $n-1$ polynomials from F, then we can choose the set of polynomials in the construction of each new polynomial $\hat{f}_i$ differently. However, we approximate the zero set of all polynomials in F, so we have to use all polynomials at least once in the computation of $\hat{f}_i$. Otherwise we only approximate the solution set of a certain subset of F. This problem does not appear if we use all the polynomials in F to compute $\hat{f}_i$. According to our experiments, this strategy reduces the size of the bounding domains faster, although we have to handle larger linear systems to find polynomials with special Hessians.

In order to compute each new polynomial $\hat{f}_i$, we have to choose an arbitrary but fixed vector of constants $\mathbf{u}_i$. These vectors are chosen a priori and they are kept fixed during each subdivision and domain reduction step. We have seen in Lemma 4.1 that the choice of the vectors $\mathbf{u}_i$ is important. These vectors have to be linearly independent in order to provide the third order convergence of the bounding regions for single roots. It is also important that the choice of these constants does not decrease the numerical accuracy of the computations. Therefore the coordinates of the vectors $\mathbf{u}_i$ should be of the same order of magnitude as the coefficients of the original polynomials. Lemma 4.1 also shows that the choice of the vectors $\mathbf{u}_i$ influences the gradient direction of the new polynomials. In addition to the fact that we need to generate new polynomials with linearly independent gradients, a useful further condition would be to create a new polynomial system with an orthogonal system of gradients. This increases the numerical stability of the computations. In [1] we show that such a polynomial system can be computed using the linear combinations of the new polynomials $\hat{f}_i$. With the help of this linear transformation we can re-compute the system of new polynomials, therefore we do not need to find an ideal choice of the vectors $\mathbf{u}_i$.

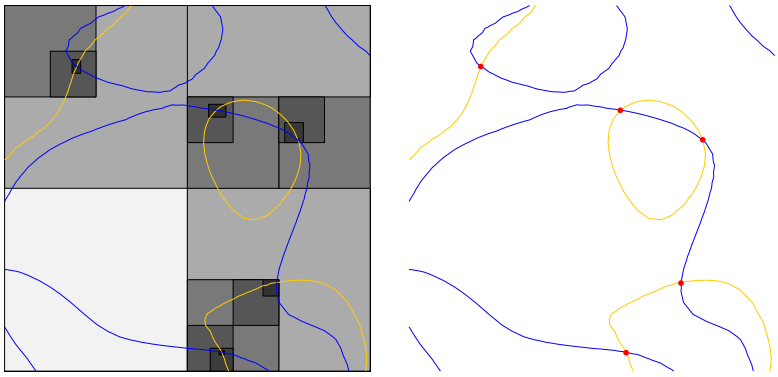

Figure 3: Approximation of the intersection points of implicitly defined curves given by the zero level set of polynomials with bi-degree $(9, 8)$ and $(6, 9)$. On the left: domains generated during the domain reduction steps, on the right: the center points of the bounding domains are marked by red dots.

5.2 Examples

We present here several examples, which show the behaviour of the root-finding algorithm `Hybrid Algorithm` for polynomial systems in two or three variables.

Intersection Points of Planar Algebraic Curves

Example 5.1. First we present a two-dimensional example to show the behaviour of the `Hybrid Algorithm`. The two implicitly defined curves are defined by polynomials with bi-degree $(9, 8)$ and $(6, 9)$. They are represented in the unit box. The intersection points of the curves are approximated within the tolerance $\varepsilon = 10^{-4}$. The curves have five intersection points in the domain. After three subdivision steps all roots are separated into different sub-domains. Then four or five domain reduction steps are made in order to achieve the prescribed accuracy around each intersection point. The output is represented in Figure 3. On the left one can see the domains generated during the domain reduction steps (either with subdivision or with the help of fat arc intersection). They are shown in different shades of gray. On the right the center point of each bounding domain from the output is marked as a red dot.

Example 5.2. This example appears in the paper of Elber et al. [11]. They present a strategy to approximate the intersection points of implicitly defined curves. Their algorithm purges away empty domains and identifies domains with single solutions more efficiently than the subdivision method. We compare here the `Hybrid Algorithm` with the simple subdivision via this example.

The two bi-cubic curves are the reflection of each other along the $x = y$ line (see Figure 4). They intersect each other along the reflection line at five different points and also in two other points in the domain. We represent the curves in the unit square $[0, 1]^2$, and approximate the roots using different tolerances. In Table 2 we compare the total number of bounding domains in the output. The hybrid algorithm returns for small tolerance a number of bounding domains, which is equal to the number of the

intersection points, while the subdivision method returns a large number of bounding boxes. The hybrid algorithm eliminates efficiently the empty sub-domains. Moreover it speeds up the convergence and uses fewer subdivision steps. In Figure 4 we show the output of the hybrid algorithm and the subdivision algorithms. The intersection points of the curves are marked by black crosses, while the generated bounding domains are represented by their center points marked by red dots. In the first row of the figure we represent the outputs of the hybrid algorithm, while in the second row the outputs of the simple subdivision method are shown. Table 2 shows the number of generated bounding boxes in the output for four different tolerances.

Both algorithms – the hybrid algorithm and the simple subdivision – were implemented by using the computer algebra system Maple. Therefore the computational time of these programs are not competitive with other implementations written in programming languages such as C or C++. Thus we present here the running time of the hybrid algorithm and the simple subdivision compared to each other. In the last row of Table 2 we show the ratio of the running times used by the hybrid and the subdivision algorithms for the four different tolerances.

Table 2: Approximating intersection of implicitly defined curves. The first two rows show the number of used bounding regions for the seven intersection points of the curves in Figure 4. The last row presents the relative running time of the two algorithms; it is computed as the running time of the hybrid algorithm divided by the running time used by the simple subdivision.

Algorithm	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$	$\varepsilon = 0.0001$
Hybrid Algorithm	15	14	7	7
Subdivision	22	40	68	71
Relative Timing (hybrid/subdivision)	3.32	0.6	0.34	0.2

Example 5.3. Example 5.2 indicates that the hybrid algorithm separates the different roots well. Therefore we present an example where the roots of the polynomials change from two single roots to one double root with the translation of one of the curves. The curves are represented by the zero set of

$$f(x,y) = -(0.95 + 10^{-k}) + 0.2x + 0.4y + x^2 + y^2,$$
$$g(x,y) = -0.48 + 0.2x + 0.1y + xy.$$

We set the tolerance to $\varepsilon = 10^{-8}$ and compute the approximation in the unit box for the values of $k = 2, 5$ and 10. The distance between the exact roots (denoted by δ) is given in the first row of Table 3 for each value of k. At the top of the table we show the results obtained by using the hybrid algorithm, while at the bottom the outputs of the simple subdivision method are shown. In each column the diameters of the bounding domains are given, which were generated step by step during the approximation methods. The bounding regions are reduced either until their diameter is smaller than the tolerance or after at most up to eight steps. In the last column we show the reduction of the bounding regions for one double root. The last row of the table presents the relative running times of the two algorithms. Finally we present a figure where the bounding domains are shown in the cases of $k = 2, 5$ and for the double root. The shrinking regions are represented in different shades of gray (see Figure 5).

Table 3: Approximating intersection points of implicitly defined curves, which are translated in three steps ($k = 2, 5, 10$) from two single roots to one double root. We present here the diameters of bounding boxes in each step of the bounding region generation. In the cases of two single roots we mark the level of domain reduction where the algorithms separate the roots. The distance of the two roots is given in the first row of the table (δ). The last row presents the relative running time of the two algorithms it is computed as the running time of the hybrid algorithm divided by the running time used by the simple subdivision.

$k = 2$ ($\delta = 1.41\,10^{-1}$)		$k = 5$ ($\delta = 4.47\,10^{-3}$)		$k = 10$ ($\delta = 1.41\,10^{-5}$)		Double root
Fat arc generation						
root	separation	0.707		0.707		0.707
0.707	0.707	$9.65\,10^{-2}$		$9.64\,10^{-2}$		0.164
0.128	0.151	$1.55\,10^{-2}$		$1.49\,10^{-2}$		$2.35\,10^{-2}$
$1.85\,10^{-3}$	$3.00\,10^{-3}$	$4.57\,10^{-3}$		$9.11\,10^{-4}$		$1.28\,10^{-3}$
$5.49\,10^{-9}$	$2.35\,10^{-8}$	root	separation	$1.97\,10^{-5}$		$1.62\,10^{-5}$
		$2.28\,10^{-3}$	$2.28\,10^{-3}$	root	separation	$2.30\,10^{-8}$
		$3.34\,10^{-7}$	$3.34\,10^{-7}$	$9.86\,10^{-6}$	$9.86\,10^{-6}$	
		$1.05\,10^{-18}$	$1.05\,10^{-18}$	$8.48\,10^{-12}$	$8.48\,10^{-12}$	
Subdivision						
root	separation	0.707		0.707		0.707
0.707	0.707	0.353		0.353		0.353
0.353	0.353	0.176		0.176		0.176
0.176	0.176	$8.88\,10^{-2}$		$8.88\,10^{-2}$		$8.88\,10^{-2}$
$8.88\,10^{-2}$	$8.88\,10^{-2}$	$4.41\,10^{-2}$		$4.41\,10^{-2}$		$4.41\,10^{-2}$
$4.41\,10^{-2}$	$4.41\,10^{-2}$	root	separation	$2.20\,10^{-2}$		$2.20\,10^{-2}$
$2.20\,10^{-2}$	$2.20\,10^{-2}$	$2.20\,10^{-2}$	$2.20\,10^{-2}$	$1.10\,10^{-2}$		$1.10\,10^{-2}$
$1.10\,10^{-2}$	$1.10\,10^{-2}$	$1.10\,10^{-2}$	$1.10\,10^{-2}$	$5.52\,10^{-3}$		$5.52\,10^{-3}$
Relative Timing (hybrid/subdivision)						
0.32		0.028		0.0029		0.0025

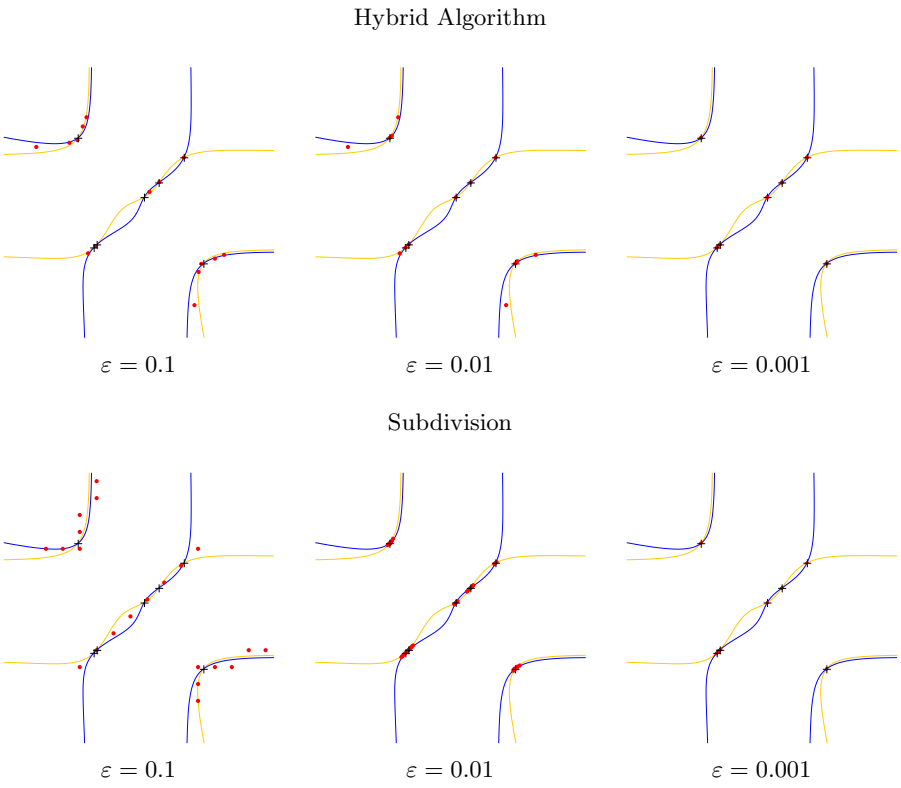

Figure 4: Comparison of root approximation, computed by the hybrid algorithm and subdivision. In the first row we present the outputs of the hybrid algorithm, while in the second row the outputs of the simple subdivision method is shown for different tolerances. The intersection points of the curves are marked by black crosses, while the generated bounding domains are represented by their center points marked by red dots.

Intersection Points of Algebraic Surfaces

Example 5.4. This example corresponds to the two-dimensional one in Example 5.2. It compares the simple subdivision method with the hybrid algorithm in a three-dimensional root-finding problem. The problem is given by the equation system

$$\begin{aligned} 0.4(x^2+y^2+z^2) - 0.88(x+y+z) - 4xyz + 1.452 &= 0, \\ 104(x^3+y^3+z^3) - 141(x^2+y^2+z^2) + 61.875(x+y+z) - 27.978125 &= 0, \\ x^2+y^2+z^2 + 0.4(x+y+z) - 1.58 &= 0, \end{aligned}$$

with respect to the unit cube. The system has six different roots in the computational domain. These roots are situated pairwise relatively close to each other. If we approximate such roots with simple subdivision, usually the root separation process is slow,

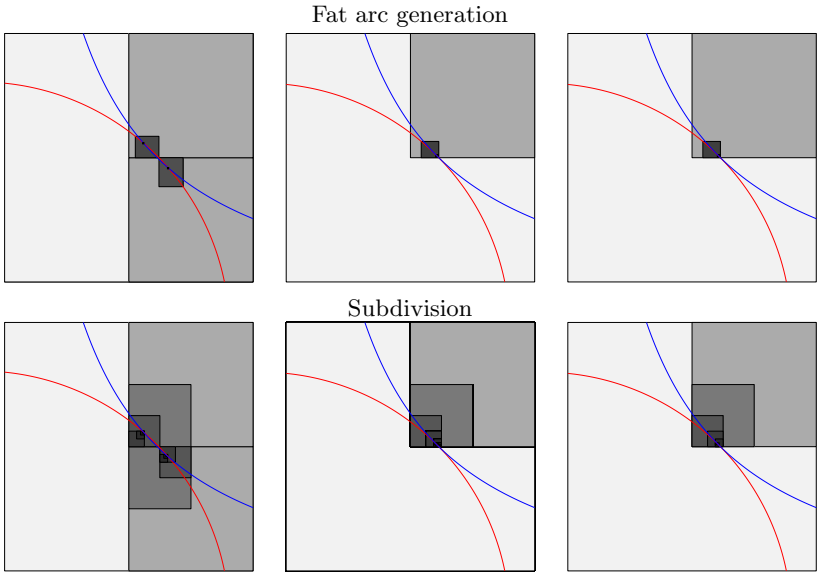

Figure 5: Reduction of bounding boxes in the cases of $k = 2, 5$ and for the double root. In the first row we used the hybrid algorithm, while in the second row simple subdivision was used.

and it uses a high number of bounding domains in the output. According to our experiments in Example 5.2, we expect that the hybrid algorithm will use less subdivision steps and generate fewer bounding regions in the output. In Table 4 we compare the total number of computed bounding domains. The hybrid algorithm returns for small tolerances a number of bounding domains, which is equal to the number of the roots. Moreover it uses less subdivision steps (see the columns for $\#l$), while the subdivision method returns a large number of bounding boxes. In Figure 6 we show the output of the hybrid algorithm and the subdivision algorithm. The generated bounding domains are represented by their center point marked by red dots. In the first row we present the outputs of the fat sphere generation, while in the second row the outputs of simple subdivision method are shown. The last row presents the relative running time of the two algorithms, computed as the running time of the hybrid algorithm divided by the running time used by the simple subdivision.

Example 5.5. We can approximate the ordinary singular points of an implicitly defined surface with the help of the hybrid algorithm. In this example we present two different algebraic surfaces given by an implicit equation $f(x, y, z) = 0$ with ordinary singularities. These singularities can be found by computing the zero set of the partial derivatives

$$f_x = 0, \quad f_y = 0, \quad f_z = 0.$$

A singular point of the surface also satisfies the equation $f = 0$.

In Figure 7 the dots mark the approximate solution points of the system of partial derivatives. The red ones are the solutions, which lie close to the implicitly defined surfaces $f = 0$. The first surface in the figure is called the Cayley-cubic. It has four

Table 4: Approximating intersection of implicitly defined surfaces. We present the number of used bounding regions and the number of domain reduction steps (denoted by $\#l$). This number shows the maximal depth of the domain reduction or subdivision tree, which is traversed by the algorithm during the root approximation. The last row presents the relative running time of the two algorithms, computed as the running time of the hybrid algorithm divided by the running time used by the simple subdivision.

Algorithm	$\varepsilon = 0.1$	$\#l$	$\varepsilon = 0.01$	$\#l$	$\varepsilon = 0.001$	$\#l$
Hybrid Algorithm	42	2	6	5	6	5
Subdivision	78	5	78	8	66	11
Relative Timing (hybrid/subdivision)	0.97		0.67		0.81	

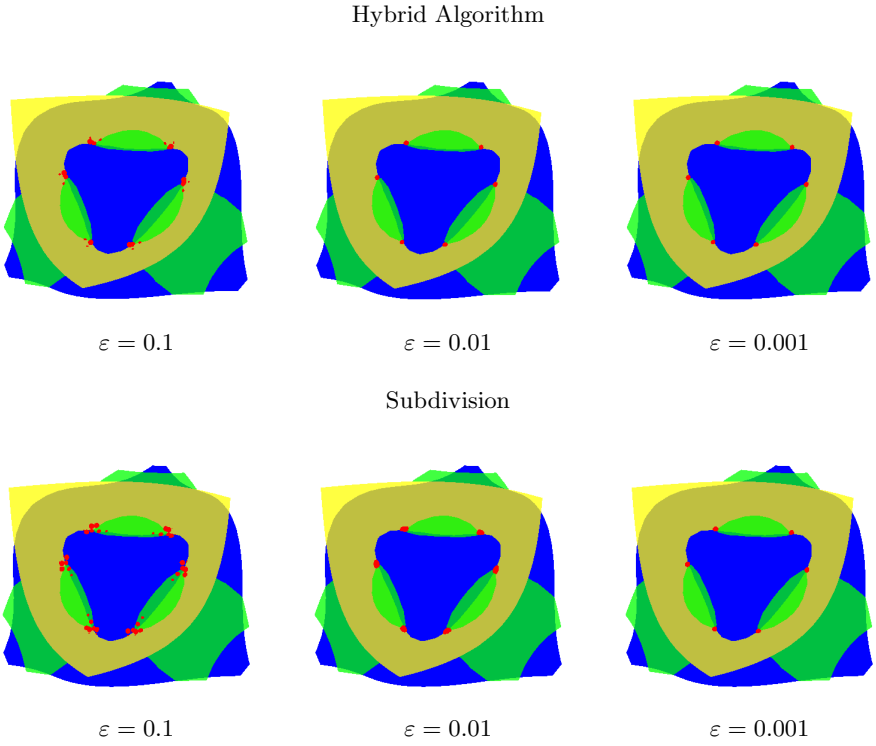

Figure 6: Comparison of approximate roots computed with the hybrid algorithm and subdivision. In the first row we represent the outputs of the hybrid algorithm, while in the second row the outputs of simple subdivision method are shown for different tolerances.

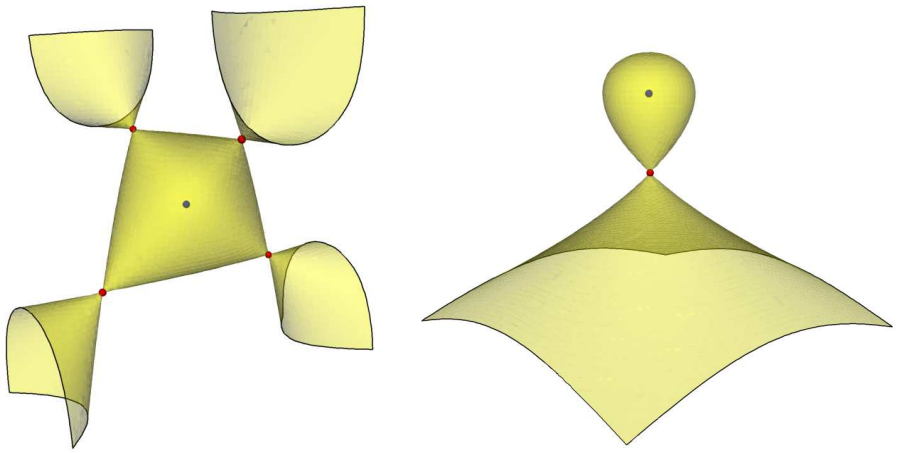

Figure 7: Ordinary singularities on implicitly defined surfaces.

ordinary singularities, which are computed in the unit cube as the solution of the system

$$\begin{aligned} -250xz + 175x + 125.5z - 87.85 &= 0, \\ 250yz - 75y - 124.95z + 37.485 &= 0, \\ -125x^2 + 125y^2 + 125.5x - 124.95y + 50z - 25.275495 &= 0. \end{aligned}$$

The second surface is the Ding-dong surface, which has one ordinary singularity. It is computed in the unit cube as the solution of the system

$$\begin{aligned} 18x - 9.06 &= 0, \\ 18y - 8.994 &= 0, \\ 81z^2 - 100.08z + 29.9136 &= 0. \end{aligned}$$

The tolerance during the computations was set to 0.01 in both examples.

The examples were computed by using the computer algebra system Maple. Therefore only relative timings were investigated during the comparison of the hybrid algorithm and the simple subdivision approach.

Currently we are working on the C++ implementation of the algorithm. We plan to embed our real root finding method into the library "Mathemagix", developed by the GAALAD group, at INRIA, Sophia Antipolis. It would improve the efficiency of the computations and enable the computation of examples in higher dimensions. This implementation will also pave the way for certified computing, e.g. by using interval arithmetic.

6 Conclusion

We presented an algorithm to approximate real roots of multivariate polynomial systems. This algorithm combines a standard subdivision technique with a new domain reduction strategy. The domain reduction is based on a special bounding region generation method. This method reformulates the algebraic system in order to bound the zero set of each new polynomial. The bounding primitive is the so-called fat sphere, which consists of an approximating spherical patch and its thickened neighbourhood.

Subdivision combined with the domain reduction method provides a fast and efficient method to approximate real roots of polynomial systems. The structure of this hybrid algorithm carries two main messages. First of all, that analyzing geometric properties of algebraic objects leads to stable techniques on real algebraic set approximation. This stability is certified by the Bernstein-Bézier polynomials. In addition, the use of local domain reduction strategies is advantageous. Although fat sphere computation requires extra computational time compared with other bounding primitives, the generated bounding regions converge faster. Computing with quadratic bounding regions provides faster termination of the algorithm and reduces the depth of the subdivision tree. Moreover, this hybrid algorithm performs better than pure subdivision in the case of double roots and single roots which are close to each other.

References

[1] Sz. Béla. *Fat Arcs and Fat Spheres for Approximating Algebraic Curves and Solving Polynomial Systems*. PhD thesis, Johannes Kepler University, Linz, 2011. `http://www.math.bme.hu/~belus/thesis.pdf`.

[2] Sz. Béla and B. Jüttler. Fat arcs for implicitly defined curves. In *Mathematical Methods for Curves and Surfaces*, volume 5862 of *Lecture Notes in Computer Science*, pages 26–40. Springer, 2009.

[3] Sz. Béla and B. Jüttler. Approximating algebraic space curves by circular arcs. In J.-D. Boissonnat et al, editor, *Curves and Surfaces, Avignon, June 24-30, 2010*, Lecture Notes in Computer Science. Springer, to appear.

[4] B. Buchberger. Gröbner-bases: An algorithmic method in polynomial ideal theory. In N.K. Bose, editor, *Multidimensional Systems Theory - Progress, Directions and Open Problems in Multidimensional Systems*, pages 184–232. Reidel Publishing Company, The Netherlands, 1985.

[5] L. Busé, M. Elkadi, and B. Mourrain. Generalized resultants for unirational algebraic varieties. *Journal of Symbolic Computation*, 59:515–526, 2000.

[6] L. Busé, M. Elkadi, and B. Mourrain. Residual resultant of complete intersection. *Journal of Pure and Applied Algebra*, 164:35–57, 2001.

[7] G. Elber and M. 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, 2001.

[8] M. Elkadi and B. Mourrain. Symbolic-numeric tools for solving polynomial equations and applications. In I.Z. Emiris and A. Dickenstein, editors, *Algorithms and Computation in Mathematics*, volume 14, pages 125–168. Springer-Verlag, 2005.

[9] I.Z. Emiris and E.P. Tsigaridas. Real solving of bivariate polynomial systems. In E.W. Mayr V.G. Ganzha and E.V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing, LNCS*, pages 150 –161. Springer-Verlag, 2005.

[10] J. Garloff and A. P. Smith. Solution of systems of polynomial equations by using bernstein expansion. Alefeld, Götz (ed.) et al., Symbolic algebraic methods and verification methods. Wien: Springer. 87-97 (2001)., 2001.

[11] I. Hanniel and G. Elber. Subdivision termination criteria in subdivision multivariate solvers using dual hyperplanes representations. *Comput. Aided Des.*, 39:369–378, 2007.

[12] T.Y. Li. Numerical solution of multivariate polynomial systems by homotopy continuation methods. *Acta Numerica*, 6:399–436, 1997.

[13] A. Morgan and A. Sommese. A homotopy for solving general polynomial systems that respects m-homogeneous structures. *Applied Mathematics and Computation*, 24:101–113, 1987.

[14] B. Mourrain and J.-P. Pavone. Subdivision methods for solving polynomial equations. *Journal of Symbolic Computation*, 44(3):292–306, 2009.

[15] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Appl. Algebra Engrg. Comm. Comput.*, 9(5):433–461, 1999.

[16] T. W. Sederberg and T. Nishita. Curve intersection using Bézier clipping. *Computer-Aided Design*, 22(9):538–549, 1990.

[17] T. W. Sederberg and S. R. Parry. Comparison of three curve intersection algorithms. *Computer-Aided Design*, 18(1):58–63, 1986.

[18] E. C. Sherbrooke. *Computation of the Solutions of Nonlinear Polynomial Systems*. PhD thesis, Massachusetts Institute of Technology, 1993.

[19] E.C. Sherbrooke and N.M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, 10(5):379–405, 1993.