Topologically guaranteed univariate solutions of under-constrained polynomial systems via no-loop and single-component tests

Michael Bartoň*
Department of Computer Science, Technion 32000, Haifa, Israel barton@cs.technion.ac.il

Gershon Elber
Department of Computer Science, Technion 32000, Haifa, Israel gershon@cs.technion.ac.il

Iddo Hanniel†
Department of Computer Science, Technion 32000, Haifa, Israel ihanniel@solidworks.com

ABSTRACT
We present an algorithm which robustly computes the intersection curve(s) of an under-constrained piecewise polynomial system consisting of \( n \) equations with \( n + 1 \) unknowns. The solution of such a system is typically a curve in \( \mathbb{R}^{n+1} \). This work extends the single solution test of [11] for a set of algebraic constraints from zero dimensional solutions to univariate solutions, in \( \mathbb{R}^{n+1} \). Our method exploits two tests: a no–loop test (NLT) and a single component test (SCT) that together isolate and separate domains \( D \) where the solution curve consists of just one single component. For such domains, a numerical curve tracing is applied. If one of those tests fails, \( D \) is subdivided. Finally, the single components are merged together and, consequently, the topological configuration of the resulting curve is guaranteed. Several possible application of the solver, like 3D trisector curves or kinematic simulations in 3D are discussed.

Keywords
Underconstrained polynomial systems, trisector curves, univariate solution spaces, kinematic simulation

1. INTRODUCTION AND PREVIOUS WORK
Solving (piecewise) polynomial systems of equations is a crucial problem in many fields such as computer-aided design, manufacturing, robotics and kinematics. A robust and efficient solution is in strong demand. The symbolically oriented approaches such as Gröbner bases and similar elimination-based techniques [5] map the original system to a simpler one, preserving the solution set. Contrary to this, polynomial continuation methods start at roots of a suitable simple system and transform it continuously to the desired one [24]. These methods are very general and give global information about the solution set, regardless of the domain of interest. Typically, they operate in \( \mathbb{C}^n \) and when only real solutions are sought, these methods can hence be inefficient.

The other approach is represented by a family of subdivision based solvers, which typically treat the equations of the system as (parts of) hypersurfaces in \( \mathbb{R}^n \), and search for its (real) intersection points inside some particular domain, usually a box in \( \mathbb{R}^n \). The interval projected polyhedron algorithm [23] employs Bernstein-Bézier representations of polynomials and projects its control points into 2D subspaces where corresponding convex hulls are computed and intersected. If this domain-contracting step is not progressive, subdivision is applied.

In order to reduce the number of computationally demanding subdivision steps or improve the robustness of the subdivision process, local preconditioning may be applied [18]. The intersecting hypersurfaces are locally straightened which makes the contraction of the domain more efficient. This technique is considered only for well-constrained, or also squared, \((n \times n)\) systems of equal-degrees constraints. Various other methods for solving squared systems exist and many related references can be found in [17].

For such systems, whose solution is, in general, a zero-variate set, termination criterion is presented in [11]. This geometrically oriented scheme detects isolated roots and allows the application of techniques such as the multivariate Newton-Raphson method, which converges quickly to the isolated root.

The complexity of subdivision based solvers is exponential in the dimension of the problem, when tensor product representations are used. In [8], expression trees are employed, reducing the expected complexity to polynomial.

Clearly, if the polynomial system of \( n \) equations is under-constrained having \( n + 1 \) degrees of freedom, the solution set is, in general, a curve in \( \mathbb{R}^{n+1} \). In such a case, techniques that handle tracing of solution curve(s) are needed. For the last several decades, many tracing/marching methods were proposed, mostly inspired by the surface-surface intersection (SSI) problem, see [1, 16, 4, 10, 22, 19, 6, 9] and the literature cited herein.

SSI tracing techniques typically compute and isolate critical (boundary and/or singular) points of the solution curve a-priori. These critical points are then employed as start/end points and the numerical evaluation of each curve segment

*Corresponding author.
†Iddo Hanniel’s current affiliation: SolidWorks Corporation, 300 Baker Avenue, Concord, MA 01742 USA

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...$5.00.
is reached either by Newton iterations [1] or by solving a set of nonlinear ordinary differential equations as an initial value problem [19]. However, the topological configuration of the intersection curve is not guaranteed.

A different SSI-solving approach is presented in [10]. The parametric \((u, v)\)-space of the surface is split into parallel strips such that boundary lines of these panels pass through the turning/singular points of the preimage curve. Hence it is possible to detect, in advance, the correct starting and ending points and, consequently, the topology of the solution set is fully determined. Further, for every curve segment, spline collocation method is used to solve a two boundary value problem for ordinary differential equation of order two. Similarly, [13] presents an algorithm which splits the preimage of the solution curve into monotone segments with respect to both \((u \text{ and } v)\) directions which, again, results in a topologically consistent solution. Detailed survey on tracing methods is also given in [22].

Recently, a general solver for over/well/underconstrained systems of equations, relying on the representation of polynomials in the barycentric Bernstein basis and exploiting the projecting control polyhedron algorithm, has been presented [20]. In the case of the univariate solution set, the sequence of \(n\)-dimensional root-containing bounding simplices is sampled along the solution curve. If high accuracy is required, a vast number of simplices as well as subdivisions is to be expected.

In this paper, we focus on the underconstrained problem of \(n\) (piecewise) polynomial equations in \((n + 1)\) unknowns and present a generalization of the termination criterion of [11] for this class of systems. Our “divide and conquer” solver is based on two elementary tests:

1. A No Loop Test (NLT) which extends the idea of [21] for higher dimension and guarantees that the intersection curve has no loops, in the given domain.

2. A Single Component Test (SCT) which assures the sought curve consists of just one component inside the given domain.

If both tests are satisfied, a monotone single component is guaranteed and numerical tracing can be applied. In this curve tracing stage, we additionally assume that the hypersurfaces, represented as the zero sets of piecewise polynomial equations, possess \(C^1\) continuity. If this assumption is violated, one can a-priori split the initial system into \(C^1\) continuous sub-systems and, at the end of the process, merge the solution back together.

The rest of the paper is organized as follows. Section 2 briefly discusses curve tracing in \(\mathbb{R}^{n+1}\). Section 3 presents the no loop test (NLT) and the single component test (SCT), with detailed explanation of the needed wedge product and bounding hypercone’s computation, and Section 4 shows some examples where the presented solver may be applied. Finally, Section 5 identifies some possible future improvements of the presented method and concludes.

2. CURVE TRACING

While numerically tracing a (intersection) curve is not the aim of this work, we briefly review this step for completeness. Assuming we have a starting and ending point on the intersection curve in \(\mathbb{R}^{n+1}\), curve tracing is a numerically-oriented technique which tracks (and samples points that approximate) a given segment of the sought curve. Normally, such a method consists of two steps: 1) a prediction step, usually a step in the direction of the tangent vector \(\vec{t}\) pointing into the domain \(D\), b) correction step: point \(c\) is the solution of the linear system (4).

![Figure 1: Curve tracing of the solution curve of the system (1) for \(n = 1\): a) Prediction step: step in the direction of the tangent vector \(\vec{t}\) pointing into the domain \(D\), b) correction step: point \(c\) is the solution of the linear system (4).](mathworld.wolfram.com/WedgeProduct.html)
\[ \mathbf{a} + \lambda \mathbf{n}, \] where \( \lambda \) is the stepsize parameter, see Fig. 1. For now and unless otherwise stated, we assume the intersection problem is well defined, i.e. all \( f_i \) are independent.

2. correction step:
   - Evaluate \( f_i, i = 1, \ldots, n \) at point \( \mathbf{b} \), and compute the gradients \( \nabla f_i(\mathbf{b}) \) as well as the wedge product \( \mathbf{w} = \nabla f_1(\mathbf{b}) \wedge \nabla f_2(\mathbf{b}) \wedge \cdots \wedge \nabla f_n(\mathbf{b}). \)
   - Construct a \( t^{\text{th}} \)-order Taylor approximation, \( T^t_i \), the tangent plane of \( f_i \) at point \( \mathbf{b} \), for all \( i \).
   - Create a hyperplane \( P \) through \( \mathbf{b} \) with \( \mathbf{w} \) as its normal vector.
   - Solve the linear system
     \[
     T^t_i(\mathbf{x}) = 0, \\
     T^t_n(\mathbf{x}) = 0, \\
     P(\mathbf{x}) = 0,
     \]
     and use the solution as the corrected point.

The correction step is repeated until the improved point \( \mathbf{c} \) (solution of system (4)) satisfies all input constraints (1) to within numerical tolerance. The sequence of interleaved steps 1 and 2 is terminated when the correction point reaches the vicinity of the ending point.

The numerical tracer described above (or similar ones) poses some major potential drawbacks (can jump from one branch of the curve to another, looping, etc.) unless some strict conditions are satisfied. Two tests, that guarantee that such a curve tracing can be safely used, are presented in Section 3.

3. THE NO LOOP AND THE SINGLE COMPONENT TESTS

In this section, we formulate two subdivision termination criteria: a no loop test (NLT) and a single component test (SCT) which deal with solving underconstrained polynomial system (1). The first criterion guarantees the solution is a univariate and detects if the solution curve has a closed loop. The second test verifies that there exists just one connected component over a given domain.

If both requirements are met, this domain contains a single connected component and further, this component starts and ends on the boundary (i.e. not a closed loop).

3.1 The No Loop Test

**Definition 3.1.** Consider a polynomial function \( f : \mathbb{R}^{n+1} \rightarrow \mathbb{R} \) along with its gradient \( \nabla f = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}) \) and its normalized gradient \( \nabla f = \frac{\nabla f}{\| \nabla f \|} \), where \( \| \cdot \| \) denotes the Euclidean norm. Further, let us define the Gaussian unit hypersphere \( S^n \) as
\[
  x_1^2 + x_2^2 + \cdots + x_{n+1}^2 = 1, \quad x_1, x_2, \ldots, x_{n+1} \in \mathbb{R}. 
\]
Then, the zero set of \( f \) is known as a hypersurface in \( \mathbb{R}^{n+1} \) and \( \nabla f \subseteq S^n \) is the Gaussian image of \( f \).

**Definition 3.2.** The differentiable mapping \( \varphi : [0, 1] \rightarrow \mathbb{R}^{n+1} \) is a regular curve if \( \| \varphi'(t) \| \neq 0 \) for all \( t \in [0, 1] \). The tangential image \( \varphi_G \) of \( \varphi \) is a one-parameter subset of the unit Gaussian hypersphere (5) that contains the images of all unit tangent vectors \( \varphi_G(t) \).

**Remark 3.3.** If no misunderstanding can occur, we title both the mapping and its image by a curve.

**Definition 3.4.** We say that the solution of the polynomial system (1) contains a closed loop if there exist a regular curve \( \varphi : [0, 1] \rightarrow \mathbb{R}^{n+1} \) such that
\[
  \varphi(t) = \varphi(0), \quad \text{for all } t \in [0,1],
\]
\[
  \varphi(0) = \varphi(1). 
\]

**Lemma 3.5.** Let \( \varphi(t), t \in [0, 1] \) be a regular \( C^1 \) continuous curve in \( \mathbb{R}^{n+1} \) and let \( \varphi_G \) be its tangential image. If there exists a hyperplane \( \alpha \), passing through the center of the Gaussian hypersphere \( S^n \)
\[
  \alpha : a_1x_1 + a_2x_2 + \cdots + a_{n+1}x_{n+1} = 0, 
\]
then \( \varphi \) is not a closed loop.

**Proof.** Since \( \varphi(t) \) is a \( C^1 \) continuous curve, \( \varphi_G \) is also continuous. Further, the Gaussian hypersphere is split by \( \alpha \) into two hemispheres. We assume that the normal vector \( \vec{\mathbf{d}} = (a_1, a_2, \ldots, a_{n+1}) \) of \( \alpha \) is of unit size and its Gaussian image \( \vec{\mathbf{d}}_G \) lies in the same hemisphere as \( \varphi_G \) (if not, we simply apply \( -\vec{\mathbf{d}} \)). Without loss of generality we assume \( \vec{\mathbf{d}} \) is the first coordinate vector of the Cartesian coordinate system in \( \mathbb{R}^{n+1} \) and consequently \( \vec{\mathbf{d}} = (1, 0, \ldots, 0) \). Since \( \vec{\mathbf{d}}_G \) and \( \varphi_G \) lie in the same hemisphere, it holds \( (\vec{\mathbf{d}}, \varphi(t)) > 0 \) for all \( t \in [0, 1] \), where \( (\cdot, \cdot) \) denotes the Euclidean scalar product. Hence,
\[
  \varphi(t) = (1, 0, \ldots, 0) \cdot (\varphi_1(t), \varphi_2(t), \ldots, \varphi_{n+1}(t)) = (\vec{\mathbf{d}}, \varphi(t)) > 0,
\]
and the first coordinate \( \varphi_1(t) \) is increasing monotonously. Consequently \( \varphi(t) \neq \varphi(s) \) for any parameters \( t \neq s \in [0, 1] \).

Prior to the main part of this section, the SCT and the NLT, we pay special attention to explain how the wedge product and the bounding cones are efficiently computed, since both tools play a major role in our algorithms.

3.1.1 Computation of wedge product

Let vector \( \vec{\mathbf{w}} \in \mathbb{R}^{n+1} \) be the wedge product (recall Def. 2.2), of vectors \( \vec{\mathbf{v}}_i \in \mathbb{R}^{n+1}, i = 1, \ldots, n \).

Our aim is to find such a vector \( \vec{\mathbf{w}} \), when a set of linearly independent vectors \( \{\vec{\mathbf{v}}_1, \vec{\mathbf{v}}_2, \ldots, \vec{\mathbf{v}}_n\} \) is given. Since the wedge product is required for both the curve tracing and the NLT, and is computed during every iteration, its efficient evaluation is crucial. Hence, we now describe our approach in more detail.

Consider the application of a Gram-Schmidt orthogonalization process to the set of vectors \( \{\vec{\mathbf{v}}_1, \ldots, \vec{\mathbf{v}}_n\} \). In general, the result is an orthonormal basis \( \mathcal{M} = \{\vec{\mathbf{e}}_1, \ldots, \vec{\mathbf{e}}_n\} \). Generate a random unit vector \( \vec{\mathbf{r}} \in \mathbb{R}^{n+1} \), then the sought orthogonal complement, \( \mathcal{M}^+ \), is defined by vector
\[
  \vec{\mathbf{w}} = \vec{\mathbf{r}} - \sum_{i=1}^{n} (\vec{\mathbf{r}}, \vec{\mathbf{e}}_i) \vec{\mathbf{e}}_i. 
\]
We say that vector \( \vec{\mathbf{r}} \) is ineligible if \( \vec{\mathbf{r}} \in \text{span}\{\vec{\mathbf{v}}_1, \ldots, \vec{\mathbf{v}}_n\} \). In order to avoid sampling of such vectors, some a-posteriori

[See, e.g., http://en.wikipedia.org/wiki/Gram_schmidt]
verification of \( \vec{w} \) is required. As the last step of the process, the magnitude of \( \vec{w} \) is examined, (see Fig. 2). If this value is below some tolerance \( \varepsilon \) meaning \( \vec{r} \) was generated (almost) in span\{\( \vec{e}_1, \ldots, \vec{e}_n \)\}, the random sampling of \( \vec{r} \) is repeated.

The locus of ineligible endpoints of sampled unit vectors \( \vec{r} \) forms a hyperstrip

\[
T^n = \{ \vec{r} \in \mathbb{R}^{n+1} \mid \| \vec{r} - \sum_{i=1}^{n} (\vec{r}, \vec{e}_i)e_i \|_2 < \varepsilon \},
\]

(10)

on the Gaussian unit hypersphere, see Fig. 2 (a). The probability that the randomly generated point on \( S^n \) lies inside this hyperstrip corresponds to the ratio between its area and the area of the Gaussian unit hypersphere \( S^n \). Thus,

\[
P[\vec{r} \text{ lies outside } T^n] = \frac{A(S^n) - A(T^n)}{A(S^n)},
\]

(11)

where \( A() \) denotes the area of the particular hypersurface and is a function of the dimension \( n \) and the tolerance \( \varepsilon \). Since the area of the unit hypersphere \( S^n \) in \( \mathbb{R}^{n+1} \) may be expressed using the volume \( V() \) bounded by the unit sphere \( S^{n-2} \) as

\[
A(S^n) = 2\pi V(S^{n-2})
\]

(12)

and similar relation holds for the area of \( T^n \) with its adequate subvolume, the ratio (11) equals to ratio of corresponding subvolumes.

Note that the process of random sampling on \( S^n \) is non-trivial task. In order to simplify this process, \( \vec{r} \) is generated randomly inside the hypersphere \( \{-1,1\}^{n+1} \) via a random selection of all its coordinates inside the interval \([-1,1]\), only to be normalized.

**Example 3.6.** In the SSI case \( (n=3) \), the probability is given as the ratio between volumes of the grey sector and a 2-dimensional disc (bounded by \( S^1 \)), see Fig. 2 (c), and the combination of formulae (11) and (12) gives

\[
P = \frac{2(\arccos(|\varepsilon|) - \varepsilon\sqrt{1 - \varepsilon^2})}{\pi},
\]

(13)

Table 1 reflects the relative frequency of the random sampling inside the hypercube \([-1,1]^{n+1} \) and the theoretical probability (13) for various \( \varepsilon \). The wedge product computation was executed for \( 10^6 \) times. A choice of \( \varepsilon = 10^{-3} \) gives the small probability \( P \approx 10^{-6} \) that two consecutive samplings return ineligible vector. Hence, in the wedge product computation, we set this value which guarantees that the single random sampling is successful with the probability, Eq. (13), \( P \approx 0.9987 \). Using double floating point numbers, this setting also guarantees that the error of the wedge product computation is (at most) \( 10^{-13} \).

3.1.2 Computation of bounding hypercone

Consider hypersurface \( f_i(x) = 0 \) over some domain \( D \subseteq \mathbb{R}^{n+1} \). We define \( N_i \), the normal field of \( f_i(x) \), as a set of all normal vectors

\[
N_i = \{ \lambda \nabla f_i(x), x \in D, \lambda \in \mathbb{R} \},
\]

(14)

where \( \nabla f_i(x) = \left( \frac{\partial f_i}{\partial x_1}, \frac{\partial f_i}{\partial x_2}, \ldots, \frac{\partial f_i}{\partial x_{n+1}} \right) \) is the gradient of \( f_i \).

In order to obtain an easy-to-maintain bound on \( N_i \), we define a circular normal bounding hypercone of \( f_i(x) = 0 \), with the axis in the direction of unit vector \( \vec{e}_i \) and an opening angle \( \alpha_i \), as

\[
C_i^{N}(\vec{u}, \alpha_i) = \{ \vec{u}, (\vec{u}, \vec{u}) = ||\vec{u}|| \cos \alpha_i \},
\]

(15)
and holds
\[ \langle \vec{u}, \vec{v} \rangle \geq \| \vec{u} \| \cos \alpha_i, \quad \forall \vec{u} \in \mathcal{N}_i. \]  
(16)

We follow the computation of \([11]\), Eq. (3), where the axis vector \( \vec{v}_i \) is calculated as an average of the set of vectors obtained from computing partial derivatives of \( f_i(x) \) and \( \alpha_i \) is the maximum angle between \( \vec{v}_i \) and these vectors. Note that such a cone is not minimal, and an optimal solution is presented in \([2]\). The optimal bounding cones were not applied, since the numerical tests indicated minor improvement of the bounding angles \( \alpha_i \) compared to the increasing timing costs.

Once the circular normal bounding hypercone is computed, one can construct the complementary (or tangent) circular bounding hypercone
\[ \mathcal{C}^C_i(\vec{v}_i, \alpha_i) = \mathcal{C}^N_i(\vec{v}_i, 90^\circ - \alpha_i) \]  
(17)
which bounds the tangent space for all possible tangent directions of hypersurface \( f_i(x) = 0 \). Hypercones \( \mathcal{C}^C_i \) and \( \mathcal{C}^N_i \) share the same axis and have complementary angles, see \([11]\) for a more detailed explanation.

3.1.3 No loop termination criterion

**Theorem 3.7.** Let \( \mathbf{F}(x) = 0 \) be the polynomial system (1) defined in Definition 2.1. Denote by \( \mathcal{K} \) the intersection of all tangent hypercones \( \mathcal{C}^C_i \), \( i = 1, 2, \ldots, n \), of \( f_i \), and Gaussian hypersphere \( \mathbb{S}^n \)
\[ \mathcal{K} = \left( \bigcap_{i=1,2,\ldots,n} \mathcal{C}^C_i \right) \cap \mathbb{S}^n. \]  
(18)
If there exists a hyperplane \( \alpha \), passing through the center of the Gaussian hypersphere \( \mathbb{S}^n \), such that \( \alpha \cap \mathcal{K} = \emptyset \), then the dimension of the set of all real roots of (1) is at most one and these intersection curves have no closed loops.

**Proof.** Let us denote by \( \mathcal{I} \) the set of all real roots of system (1) and assume \( \mathcal{I} \) is a \( k \)-dimensional variety. Let \( \tau_y \) be the \( (k\text{-dimensional}) \) tangent space of \( \mathcal{I} \) at some point \( y \), \( y \in \mathcal{I} \) and \( \tau_y \) its tangential image on the Gaussian sphere. The set \( \mathcal{K} \), by its definition, is a set holding all feasible unit tangent vectors of the solution variety \( \mathcal{I} \). Hence,
\[ \tau_y \subseteq \mathcal{K} \quad \text{for all } y \in \mathcal{I} \]  
(19)
and by the assumption
\[ \tau_y \cap \alpha = \emptyset \quad \text{for all } y \in \mathcal{I}. \]  
(20)
Algorithm 1 (see Fig. 3) \{ No loop test (NLT) in $\mathbb{R}^{n+1}$ \}

1: INPUT: Coefficients of the system (1), domain $D$;
2: for $i = 1$ to $n$ do
3: \quad $C_i^d$ $\leftarrow$ generate the complementary tangent bounding hypercone of the hypersurface $f_i(x) = 0$ on domain $D$;
4: \quad $H_1^+, H_1^-$ $\leftarrow$ pair of hyperplanes that bounds $C_i^d$ in $\mathbb{S}^n$;
5: end for
6: $\vec{a} \leftarrow \vec{v}_1 \wedge \vec{v}_2 \wedge \cdots \wedge \vec{v}_n$, the wedge product in $\mathbb{R}^{n+1}$, where $\vec{v}_i$ are the axis-vectors of the tangent hypercones $C_i^d$, $i = 1, \ldots, n$;
7: $\alpha$ $\leftarrow$ hyperplane with normal vector $\vec{a}$ passing through the center of the Gaussian sphere $\mathbb{S}^n$;
8: $P_i$ $\leftarrow$ $\alpha \cap H_1^{+/-} \cap H_2^{+/-} \cap \cdots \cap H_n^{+/-}$, intersection points in $\mathbb{R}^{n+1}$, $i = 1, \ldots, 2^n$, of $\alpha$ with the bounding hyperstrip $\mathcal{P}$;
9: OUTPUT: TRUE if $P_i \subset \mathbb{S}^n$, $\forall i = 1, \ldots, 2^n$; FALSE otherwise.

Now, let $L_\alpha$ be the linear space related to the hyperplane $\alpha$ ($\dim(L_\alpha) = n$). Then, linear subspaces $\tau_\alpha$ and $L_\alpha$, of $\mathbb{R}^{n+1}$, are disjoint and hence $\dim(\tau_\alpha) \leq 1$ for all $y \in I$.

If not, $\dim(\tau_\alpha) \geq 2$, there exist a non-trivial intersection ($\tau_\alpha \cap L_\alpha \neq \emptyset$) of both subspaces as well as its non-trivial image on the Gaussian hypersphere ($\tau_\alpha^\tau \cap \alpha \neq \emptyset$) which violates the assumption (20) and proves the first part of the theorem.

Since the dimension of variety $\mathcal{I}$ is at most one, the Gaussian image of its tangent space

\[
\tau^\tau = \bigcup_{y \in \mathcal{I}} \tau^\tau_y
\]  

is composed of the finite number of continuous curves and isolated points. All these segments are bounded by $K$, so applying Lemma 3.5 on each solution component completes the proof.

\textbf{Remark 3.8.} Theorem 3.7 states that, if satisfied, the dimension of the root set $\mathcal{I}$ is at most one, in general. Note that the zero-dimensional roots ($\text{points in } \mathbb{R}^{n+1}$) can occur only at the boundary of some hypersurface $f_i(x) = 0$, $i = 1, \ldots, n$. If this point was interior for all $f_i(x)$, they would have a tangent hyperplane at common point at this point which violates the assumptions of the theorem. This means that the limit cases of closed loops are also detected.

Based on the result of the previous theorem, we formulate a no loop test (NLT) which guarantees that no closed loops appears in the (univariate) solution of the system (1). See Algorithm 1 and Fig. 3.

Some steps of the algorithm are now explained in some detail:

- In line 4, we construct a pair of hyperplanes that bound tangent hypercone $C_i^d$ in $\mathbb{S}^n$, following [11]. These hyperplanes are symmetric with respect to the origin, see Fig. 3c), their normal vector $\vec{v}_i$ coincides with the axis vector of $C_i^d$ and they intersect $\mathbb{S}^n$ in the same circles as $C_i^d$.

- The mutual intersections of $n$ pairs of bounding hyperplanes define a prismatic subset in $\mathbb{R}^{n+1}$, a \textit{bounding hyperstrip} $\mathcal{P}$, which is unbounded in the direction perpendicular to all axis vectors $\vec{v}_i$, $i = 1, \ldots, n$; this direction $\vec{a}$ is computed in line 6, see also Fig. 3d).

- In line 8, the intersection of the prismatic strip and a plane $\alpha$ is computed. There can be many planes that satisfy the condition in Theorem 3.7. However, we choose $\alpha$ to be perpendicular to $\vec{a}$, the direction of the strip $\mathcal{P}$, in order to minimize the maximum of distances between $P_1$ and the center of the Gaussian sphere.

In the surface-surface intersection literature, a similar idea of loop detection has been presented in [21], testing the mutual orientations of bounding cones of the two intersecting surfaces. However, we are aware of no generalization of that condition for a “no loop test” in the intersection of $n$ hypersurfaces in $\mathbb{R}^{n+1}$.

\textbf{3.2 Single Component Test}

In order to robustly solve underconstrained polynomial system (1) over some domain $D \subseteq \mathbb{R}^{n+1}$, we present a second test which examines the number of points lying together on the solution curve and on the boundary of $D$. Due to this criterion (and the NLT), we can completely classify the number of curve segments inside $D$. Once the number of intersections of $D$ with the solution curve is located, the curve is traced (two intersections), the domain is discarded (no intersections) or subdivided (more than two intersections). Moreover, the position of these points on the boundary gives us hints where to subdivide.

Consider the case where the numerical solver computes
all real boundary roots of polynomial systems $n \times n$ and the NLT returns a positive answer. Then, the single component test (SCT) is applied as in Algorithm 2. See also Fig. 4.

The SCT guarantees that just one segment can reside inside the box of interest. Nevertheless, curve tracing may potentially become unstable even if such a strong condition is satisfied. Consider a curve forming a “semi-loop” shape, as in Fig. 5 (a). An undesired jump can occur when numerically tracing this curve if two locations of the same segment are close to each other. Nevertheless, in the algorithm proposed here, SCT is called only if NLT returns a positive answer. Such a “semi-loop” is prevented by the NLT. After passing both the NLT and the SCT, only monotone curves are traced. See Fig. 5 (b). By monotone we mean that there exist a direction (vector $\vec{a} \in \mathbb{R}^{n+1}$) in which the inner product of the tangent of the intersection curve and $\vec{a}$ have the same sign throughout. This vector $\vec{a}$ is the normal vector of plane $\alpha$ from Theorem 3.7, constructed in line 6 of the NLT algorithm.

**Remark 3.9.** The distance of $K$, the patch on the Gaussian hypersphere that bounds all feasible tangent vectors of the solution curve, from the hyperplane $\alpha$ correlates with the monotonicity of the solution curve, see Fig. 3 d), e). More precisely, the more remote $K$ is from $\alpha$ on $S^n$, the smaller the deviation between the normal vector of plane $\alpha$, $\vec{a}$, and the tangent vector of the solution curve. This distance can be determined by the distance between the center of the Gaussian hypersphere $O$ and intersection points $P_i$. See Fig. 3 f). If desired, the user may prescribe the distance between $K$ and $\alpha$ by placing a tighter bound on $||P_iO||_2$, in order to get a high-pitched monotone curve segments and therefore a more robust curve tracing. Naturally, a higher number of
subdivisions will be the cost for this prescription.

Remark 3.10. In the general case, the SCT returns an even number of boundary points. Odd number of points may occur only in limit cases, having higher orders of contact, when some (segments of the) curve touches a boundary face of the domain. In order to avoid subdivision at such singular locations, if an odd number is reported, numerical perturbation is applied on the subdivision parameters of the hyperplanes.

Exploiting both tests, the SCT and the NLP, the complete underconstrained \( n \times (n + 1) \) subdivision-based solver is summarized in Algorithm 3.

4. APPLICATIONS OF THE ALGORITHM

In this section, we present several examples where the new algorithm may be applied. All examples were created using the GuIrit GUI user interface \(^4\) of the Irit solid modeling system \(^5\). The solver was implemented as a library of Irit.

4.1 Trisectors

The Voronoi diagrams and medial axis transforms have proven to be useful tools in a variety of application domains. Example of such applications include robot motion planning, molecular modeling and surface reconstruction, to name a few. An edge of a Voronoi diagram of geometric objects in \( \mathbb{R}^3 \) is formed out of the trisector curve of three objects (see for example [12]). Thus, a basic building block in the construction of Voronoi diagrams and medial axes is to compute the trisector between a triplet of objects, in 3-space.

Definition 4.1. Consider a given triplet of free-form objects (curves or surfaces) in \( \mathbb{R}^3 \). The trisector is the locus of centers of all spheres that possess (at least) first order contact with all three objects.

\(^4\)www.cs.technion.ac.il/~gershon/GuIrit
\(^5\)www.cs.technion.ac.il/~irit

Since both the tangency and the distance constraints can be expressed algebraically, we obtain in the curve case

\[
\langle C'(t), C(t) - O \rangle = 0, \quad (C(t) - O, C(t) - O) = r^2, \tag{22}
\]

and for surfaces

\[
\langle \frac{\partial S}{\partial u}, S(u,v) - O \rangle = 0, \quad \langle \frac{\partial S}{\partial v}, S(u,v) - O \rangle = 0, \quad (S(u,v) - O, S(u,v) - O) = r^2, \tag{23}
\]

where \( O \) is the center of the tangent sphere and \( r \) is its radius. Then, the (piecewise) polynomial system (1) is composed of the equations of type (22) and (23) whereas the dimension depends on the type of geometry. For instance, the curve-curve-surface trisector’s system in \( \mathbb{R}^3 \) consist of 6 equations (2 curve-tangency constraints, 2 surface tangency constraints and 2 distance-like constraints, which originated from the distance constraints whereas factor \( r \) was eliminated) with 7 unknowns (curves’ parameters \( t, s, \) surface’s parameters \( u, v \) and coordinates \( x, y, z \) of center \( O \)). The domain \( D \subset \mathbb{R}^7 \), where the system is solved, shares the range of the parametric subdomains in the directions of \( t, s, u \) and \( v \) axes. The coordinates of \( O \) are considered within a user-defined box of interest, typically a bounding box of all three input objects.

Seen from another perspective, looking for a trisector curve is equivalent to constructing a canal surface, when its three tangent objects are given. As shown at Fig. 6 c), the input curves define the “ribs”, which the desired canal surface needs to pass through.

4.2 Kinematic mechanisms

As another application example, the simulation of motions of planar/spatial kinematic mechanisms is considered. For many mechanical systems, the constraints between its components may be expressed algebraically. Then, searching for all feasible configurations of given mechanism, known as the kinematic synthesis, is accomplished by solving polynomial systems, where every root of the system corresponds to one
C(t)

(a)

C1(t)

C2(s)

S(u, v)

(b)

(c)

Figure 7: Kinematic mechanisms: a) Planar linkage consists of two anchored (black) and three movable (white) joints. The upper point is restricted to move along curve $C$. b) Motion of a rigid triangle in 3D. Two vertices are constrained to move along curves $C_1$ and $C_2$, the third is allowed to move on surface $S$. Color coding displays the solution paths of vertices during the motion. c) The spatial four-bar linkage with two fixed (black) and three movable points (white). Three bar-bar orthogonality constraints are preserved during the motion. Lower row: Several positions of mechanisms are shown.

4.3 Application’s summary

Table 2 summarizes the data of all presented examples and compares the timings of Algorithm 3 with those obtained by sampling points on the curve using the well-constrained solver from [11] and connecting them in a post-process. All presented data were tested with an 2.4 GHz Q6600 Intel machine running windows XP.

If the subdivision tolerance is low, the well-constrained solver [11] might outperform Algorithm 3 in timings, see Table 2 example 6(c). This phenomena is due to the fact that the presented method solves the boundary systems (see Fig. 4(c)) whereas [11] requires no extra computation during the subdivision. In such cases, the solution points are typically sparsely spread, giving no good indication on the topology of the solution curve. However, when a higher enough accuracy (subdivision tolerance) is requested, Algorithm 3 always dominates.

All the examples was executed mainly with a subdivision tolerance $\varepsilon_{\text{sub}} = 0.0005$ and the tracing step 0.01. If required, one can tune up these parameters in order to either improve the quality of the solution curve or reduce the timing costs.

The library dealing with kinematic simulations [3] exploits directly Algorithm 3 and hence a comparison with [11] is not available in Table 2.
5. CONCLUSION AND FUTURE WORK

In this work, we have presented a technique for robustly solving underconstrained (piecewise) polynomial system of \( n \) equations in \( n + 1 \) unknowns. A subdivision based solver that exploits two termination criteria, namely the no loop test (NLT) and the single component test (SCT), returns domains that contain a single monotone univariate solution. Segments are detected and isolated. The segmentation can also be used to safeguard the numerical tracing stage. Traced segments are afterwards merged together with the right neighbors (inverse process to subdivision) and thus correct topology of the intersection curve is guaranteed.

As a future work, an improved algorithm is intended, which better handles solution curves with singular points, or higher orders contact. In the current implementation, the solver subdivides only to a point where the subdivision tolerance is reached, which causes slow results. Similarly, if the constraint hypersurfaces have tangent contact, a special treatment similar to the one presented in [19], is desired.

Other geometric problems, whose solutions are described by \( n \times (n + 1) \) polynomial system, like medial axis computation, rounding of two surfaces or some other types of kinematic problems (e.g., a self-motion of Stewart platform), are also within the scope of our interest, and will be experimented with.

6. ACKNOWLEDGMENTS

This research was partly supported by the Israel Science Foundation (grant No. 346/07), and in part by the New York metropolitan research fund, Technion.

7. REFERENCES