Topologically guaranteed solution of surface–surface intersections 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 computes the intersection curve(s) of two free-form surfaces in 3D. The surface-surface intersection (SSI) problem is equivalent to solving an underconstrained polynomial system of three equations with four unknowns, over some domain $D \in \mathbb{R}^4$. The solution of such a system is a curve in 4-space. This work extends the single solution test for a set of algebraic constraints from zero dimensional solutions to univariate solutions, in $\mathbb{R}^n$. 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 Newton-Raphson 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. This extension is mostly discussed in $\mathbb{R}^{n+1}$ for n equations with $n+1$ unknowns while examples are provided for the SSI problem.

Keywords

Underconstrained polynomial systems, surface-surface intersection, univariate solution spaces

1. INTRODUCTION AND PREVIOUS WORK

Solving (piece-wise) 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 like *Gröbner bases* and similar elimination-based techniques [4] 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 [19]. These methods are very general and give global information about the solution set, regardless 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 polyhedra* algorithm [18] 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 [13]. 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. For such systems, whose solution is in general a zero-variate set, the *termination criterion* is presented in [9]. This geometrically oriented scheme detects isolated roots and allows application of techniques like multivariate Newton-Raphson which converges quickly to the root.

The complexity of subdivision based solvers is exponential in the dimension of the problem, when tensor product representations are used. In [6], expression trees are employed, reducing the expected complexity to polynomial.

Clearly, if the polynomial system of n equations is *underconstrained* 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, 12, 3, 8] and the literature cited herein. SSI solving methods may be classified into three main groups: lattice evaluation, subdivision based solvers, and tracing methods, where the latter two "are the most widely used because of their generality" [17].

Subdivision techniques exploit a divide-and-conquer approach that intersection of sub-patches is usually simpler than intersection of original surfaces [11]. This process is recursively repeated until these sub-patches are sufficiently flat and the solution is interpolated by a linear segment, or there is a guarantee of no intersection curve. Such a process is demanding and hence is often combined with tracing

*Iddo Hanniel's current affiliation: SolidWorks Corporation, 300 Baker Avenue, Concord, MA 01742 USA

SIAM '09 San Francisco, California USA

methods [5, 1, 8].

An alternative subdivision method is based on *affine* or *interval arithmetic* [7]. For both domains of the parametric surfaces, an estimation of the range of values of corresponding parametric functions is computed. As a result of this estimation, two axes-aligned 3D bounding boxes, that contain the corresponding parametric surfaces, are obtained. If the bounding boxes do not intersect, there is no intersection curve. Otherwise, surfaces are subdivided and the process is repeated.

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 is reached either by Newton iterations [1] or by solving a set of nonlinear ordinary differential equations as an initial value problem [14]. However, the topological configuration of the intersection curve is not guaranteed.

A different approach is presented in [8]. 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 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, [10] presents an algorithm which splits the preimage of the solution curve into monotone segments with respect to both (u and v) directions which, again, results in topologically consistent solution. Detailed survey on tracing methods is also given in [17].

Recently, a general over/well/underconstrained solver, relying on the representation of polynomials in the *barycentric Bernstein basis* and exploiting the projecting control polyhedra algorithm, has been presented [15]. In the case of the univariate solution set, the sequence of n-dimensional root-containing bounding *simplices* is returned. If high accuracy is required, the vast number of simplices as well as the number of subdivision is to be expected.

In this paper, we focus on the *underconstrained* problem of n (piece-wise) polynomial equations in $(n + 1)$ unknowns and present a generalization of the termination criterion of [9] 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 [16] for higher dimension and guarantees that the intersection curve has no loops.

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 the Newton-Raphson numerical tracing can be applied. In this curve tracing stage, we additionally assume that the hypersurfaces, represented as the zero sets of piece-wise 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. While the presented tests are general, in this work and due to a limited space, solutions are demonstrated only on the SSI problem.

The rest of the paper is organized as follows. Section 2 briefly discusses the SSI problem and curve tracing. Section 3 presents the no loop test (NLT) and the single component test (SCT), with detailed explanation of the wedge product and bounding hypercone's computation, and section 4 shows some examples of SSI. Finally, Section 5 identifies some possible future improvements of the presented method and concludes.

2. PRELIMINARIES

In order to make this paper more self-contained, the basic notions in surface–surface intersection (SSI) and curve tracing are recalled. Section 2.1 prescribes the SSI problem as an underconstrained polynomial system and section 2.2 briefly summarizes the numerical tracing stage.

2.1 Surface–surface intersection

Surface–surface intersection (SSI) is one of the most fundamental problems in geometric modeling. Let us consider two parametric surfaces $P(u, v)$ and $Q(s, t)$ in Euclidean space $\mathbb{E}^3$

$$\begin{array}{rcl} P(u,v) & = & [x_P(u,v), y_P(u,v), z_P(u,v)], \\ Q(s,t) & = & [x_Q(s,t), y_Q(s,t), z_Q(s,t)], \end{array} \tag{1}$$

where all the coordinate functions $x_P(u, v), \ldots, z_Q(s, t)$ are either polynomial or rational and pairs of variables (u, v) and (s, t) belong to *parameter domains* D_P, $D_Q \subseteq \mathbb{R}^2$, respectively. The *intersection* of P and Q, represented as a solution of an underconstrained polynomial system, is a set of all quadriplets $(u, v, s, t) \in D_P \times D_Q$ such that

$$\begin{array}{rcl} x_P(u,v) & = & x_Q(s,t), \\ y_P(u,v) & = & y_Q(s,t), \\ z_P(u,v) & = & z_Q(s,t), \end{array} \tag{2}$$

Note that for rationally parametrized surfaces, the system (2) remains polynomial, albeit of a higher degree.

2.2 Curve Tracing

While tracing a (intersection) curve numerically 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, *curve tracing* is a numerically–oriented technique which tracks (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 of the curve and 2) *a correction step* which puts the approximation point closer to the intersection curve. The correction phase is repeated until the approximation point is close enough to the curve – which typically means until some predefined numerical tolerance is reached. Steps 1 and 2 are interleaved and once the proximity of the endpoint is achieved, the curve segment's approximation is completed and represented, for example either by some polyline or a B-spline curve).

Apparently, the application of the curve tracing is not limited to tracing curves in Euclidean 3-space ($\mathbb{E}^3$) but may be equally applied for tracing a solution curve of an underconstrained polynomial system of size n in $\mathbb{R}^{n+1}$:

DEFINITION 2.1. *Consider the mapping* $\mathcal{F} : \mathbb{R}^{n+1} \to \mathbb{R}^n$, *such that each component* f_i, $i = 1, \ldots, n$ *of* $\mathcal{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})]$ *is a polynomial function in variables* $\mathbf{x} =$

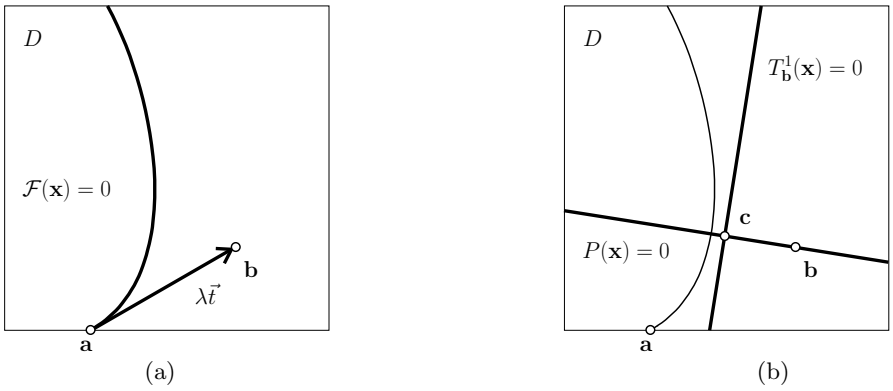


Figure 1: Curve tracing of the solution curve of the system (3) for $n = 1$: a) Prediction step: step in the direction of the tangent vector $\vec{t}$ pointing into the domain D, b) correction step: point c is the solution of the linear system (6).

$(x_1, x_2, \ldots, x_{n+1})$. *Then, every solution* $\mathbf{x}$ *of the system,*

$$\mathcal{F}(\mathbf{x}) = 0, \tag{3}$$

is called a root *of* $\mathcal{F}$ *and the set of all roots is known as the* zero set *of the mapping* $\mathcal{F}$.

DEFINITION 2.2. *Vector* $\vec{w} \in \mathbb{R}^{n+1}$ *is the* wedge product[1] *of* n *linearly independent vectors* $\vec{v}_i \in \mathbb{R}^{n+1}$, $i = 1, \ldots, n$

$$\vec{w} = \vec{v}_1 \wedge \vec{v}_2 \wedge \cdots \wedge \vec{v}_n, \tag{4}$$

if and only if $\langle \vec{w}, \vec{v}_i \rangle = 0, \forall i = 1, \ldots, n$.

Let us assume system (3) has a univariate zero set in some domain $D \subseteq \mathbb{R}^{n+1}$ and $\mathbf{a} = (a_1, a_2, \ldots, a_{n+1}) \in D$ is a root. Based on the multivariate Newton–Raphson method, a univariate solution tracer may be described as follows.

1. ***prediction step***: Compute the tangent direction of the solution curve,

$$\vec{t} = \nabla f_1(\mathbf{a}) \wedge \nabla f_2(\mathbf{a}) \wedge \cdots \wedge \nabla f_n(\mathbf{a}), \tag{5}$$

   where $\vec{t}$ is the *wedge product*, see Def. 2.2, in $\mathbb{R}^{n+1}$ of $\nabla f_i(\mathbf{a})$, the gradients of f_i at point $\mathbf{a}$. Let $\mathbf{b} = \mathbf{a} + \lambda\vec{t}$, where λ is the *stepsize* parameter, see Fig. 1. For now and unless stated otherwise, 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 $\vec{w} = \nabla f_1(\mathbf{b}) \wedge \nabla f_2(\mathbf{b}) \wedge \cdots \wedge \nabla f_n(\mathbf{b})$.
   - Construct a 1st-order Taylor approximation, $T^i_{\mathbf{b}}$, the tangent plane of f_i at point $\mathbf{b}$.
   - Create a hyperplane P through $\mathbf{b}$ with $\vec{w}$ as its normal vector.
   - Solve the linear system

$$\begin{aligned} T^1_{\mathbf{b}}(\mathbf{x}) &= 0, \\ &\vdots \\ T^n_{\mathbf{b}}(\mathbf{x}) &= 0, \\ P(\mathbf{x}) &= 0. \end{aligned} \tag{6}$$

[1]See, e.g., `mathworld.wolfram.com/WedgeProduct.html`.

The correction step is repeated until the improved point $\mathbf{c}$ (solution of system (6)) satisfies all input constraints (3) 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 (3). The first criterion guarantees the one-dimensionality of the solution and detects if the solution curve has a closed loop. The second test verifies that there exist just one 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 loop).

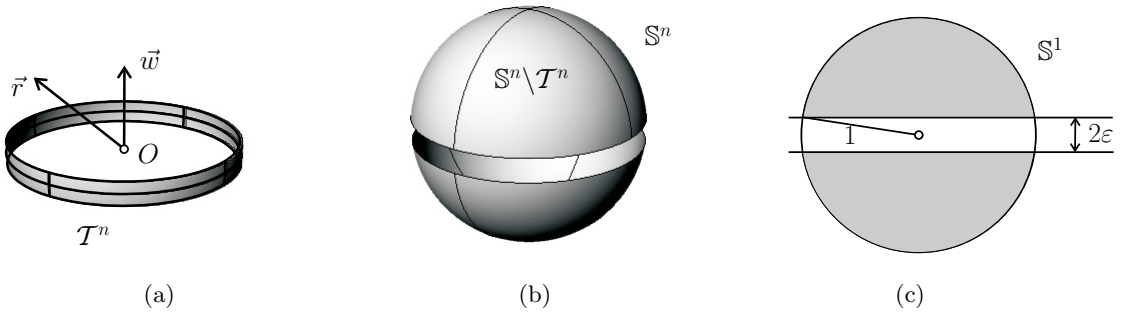
3.1 The No Loop Test

DEFINITION 3.1. *Consider a polynomial function* $f : \mathbb{R}^{n+1} \to \mathbb{R}$ *along with its gradient* $\nabla f = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_{n+1}})$ *and its normalized gradient* $\nabla_N f = \frac{\nabla f}{\|\nabla f\|_2}$, *where* $\|.\|_2$ *denotes the Euclidean norm. Further, let us define the* Gaussian unit hypersphere $\mathbb{S}^n$ *as*

$$x_1^2 + x_2^2 + \cdots + x_{n+1}^2 = 1, \quad x_1, x_2, \ldots, x_{n+1} \in \mathbb{R}. \tag{7}$$

Then, the zero set of f *is known as a* hypersurface *in* $\mathbb{R}^{n+1}$ *and the* $\nabla_N f \subseteq \mathbb{S}^n$ *is its* Gaussian image.

DEFINITION 3.2. *The differentiable mapping* $\varphi : [0, 1] \to \mathbb{R}^{n+1}$ *is a regular curve if* $\|\varphi'(t)\|_2 \neq 0$ *for all* $t \in [0, 1]$. *The* tangential image φ_G *of* φ *is a one-parametric subset of the unit Gaussian hypersphere (7) that contains the images of all unit tangent vectors* $\frac{\varphi'(t)}{\|\varphi'(t)\|_2}$.



(a) (b) (c)

Figure 2: Wedge product via random vector $\vec{r}$: a) A unit random vector $\vec{r}$ is suitable if its endpoint lies on the Gaussian unit hypersphere (center O) outside the strip $\mathcal{T}^n$: $|r_{n+1}| > \varepsilon$. b) The probability that a random vector is suitable corresponds to the ratio between areas of the complement of $\mathcal{T}^n$ and $\mathbb{S}^n$. c) In the SSI case, $n = 3$, this probability is equivalent to the ratio between the volume of the gray sector and volume bounded by $\mathbb{S}^1$.

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 (3) contains a* closed loop *if there exist a regular curve $\varphi : [0, 1] \to \mathbb{R}^{n+1}$ such that*

$$\begin{aligned} &\mathcal{F}(\varphi(t)) = 0, \quad \text{for all } t \in [0, 1], \\ &\varphi(0) = \varphi(1). \end{aligned} \tag{8}$$

LEMMA 3.5. *Let $\varphi(t)$, $t \in [0, 1]$ be a regular C^1 continuous curve in $\mathbb{R}^{n+1}$ and let φ_G be its tangential image. If there exists a hyperplane α, passing through the center of the Gaussian hypersphere $\mathbb{S}^n$*

$$\alpha : \ a_1x_1 + a_2x_2 + \cdots + a_{n+1}x_{n+1} = 0, \tag{9}$$

such that $\alpha \cap \varphi_G = \emptyset$, then φ is not a closed loop.

Proof. Since $\varphi(t)$ is a C^1 continuous curve, φ_G is also continuous. Further, the Gaussian hypersphere is split by α into two hemispheres. We assume that the normal vector $\vec{a} = (a_1, a_2, \ldots, a_{n+1})$ of α is of unit size and its Gaussian image $\vec{a}_G$ lies in the same hemisphere as φ_G (if not, we apply $-\vec{a}$). Without loss of generality we assume $\vec{a}$ is the first coordinate vector of the Cartesian coordinate system in $\mathbb{R}^{n+1}$ and consequently $\vec{a} = (1, 0, \ldots, 0)$. Since $\vec{a}_G$ and φ_G lies in the same hemisphere, it holds $\langle \vec{a}, \varphi'(t) \rangle > 0$ for all $t \in [0, 1]$, where $\langle \cdot \rangle$ denotes the Euclidean scalar product. Hence,

$$\varphi_1'(t) = (1, 0, \ldots, 0) \cdot (\varphi_1'(t), \ldots, \varphi_{n+1}'(t)) = \langle \vec{a}, \varphi'(t) \rangle > 0, \tag{10}$$

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 $\vec{w} \in \mathbb{R}^{n+1}$ be the wedge product, see Def. 2.2, of vectors $\vec{v}_i \in \mathbb{R}^{n+1}$, $i = 1, \ldots, n$.

Our aim is to find such a vector $\vec{w}$, when a set of linearly independent vectors $\{\vec{v}_1, \vec{v}_2, \ldots \vec{v}_n\}$ 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[2] orthogonalization process to the set $\{\vec{v}_1, \ldots, \vec{v}_n\}$. In general, the result is an orthonormal basis $\mathcal{M} = \{\vec{e}_1, \ldots, \vec{e}_n\}$. Generate a random unit vector $\vec{r} \in \mathbb{R}^{n+1}$, then the sought orthogonal complement, $\mathcal{M}^\perp$, is defined by vector

$$\vec{w} = \vec{r} - \sum_{i=1}^{n} \langle \vec{r}, e_i \rangle e_i. \tag{11}$$

We say that vector $\vec{r}$ is *ineligible* if $\vec{r} \in \texttt{span}\{\vec{v}_1, \ldots, \vec{v}_n\}$. In order to avoid sampling of such points, some *a-posteriori* 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 ε (meaning $\vec{r}$ was generated almost in $\texttt{span}\{\vec{v}_1, \ldots, \vec{v}_n\}$), the random sampling of $\vec{r}$ is repeated.

The locus of unfavorable endpoints of sampled unit vectors $\vec{r}$ forms a hyperstrip

$$\mathcal{T}^n = \{\vec{r} \in \mathbb{R}^{n+1} \mid \|\vec{r} - \sum_{i=1}^{n} \langle \vec{r}, e_i \rangle e_i\|_2 < \varepsilon\}, \tag{12}$$

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

$$P[\vec{r} \text{ lies outside } \mathcal{T}^n] = \frac{A(\mathbb{S}^n) - A(\mathcal{T}^n)}{A(\mathbb{S}^n)}, \tag{13}$$

where $A()$ denotes the area of the particular hypersurface and is a function of the dimension n and the tolerance ε. Since the area of the unit hypersphere $\mathbb{S}^n$ in $\mathbb{R}^{n+1}$ may be expressed using the volume $V()$ bounded by the unit sphere $\mathbb{S}^{n-2}$ as [3]

$$A(\mathbb{S}^n) = 2\pi V(\mathbb{S}^{n-2}) \tag{14}$$

[2] See, e.g., `http://en.wikipedia.org/wiki/Gram_schmidt`.
[3] See, e.g., `http://en.wikipedia.org/wiki/Hypersphere`.

Table 1: N represents the number of repetition of the random vector sampling process necessary to obtain eligible wedge product vector vs. the width ε of the strip $\mathcal{T}^3$ (see Fig. 2). The relative frequency of random sampling inside the hypercube $[-1,1]^{n+1}$ is displayed, for the event (wedge product computation) that was repeated 10^6 times. P is the theoretical probability given by formula (15).

N \ ε	10^{-2}	10^{-3}	10^{-4}
0	986325	998610	999878
1	13467	1389	122
2	205	1	0
P	0.9872	0.9987	0.9998

and similar relation holds for the area of $\mathcal{T}^n$ with its adequate subvolume, the ratio (13) equals to ratio of corresponding subvolumes.

Note that the process of random sampling on $\mathbb{S}^n$ is nontrivial task. In order to simplify this process, $\vec{r}$ is generated randomly inside the hypercube $[-1,1]^{n+1}$ by sampling every its coordinate inside the interval $[-1,1]$ and 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 $\mathbb{S}^1$), see Fig. 2 c), and the combination of formulae (13) and (14) gives*

$$P = \frac{2(\arccos(\varepsilon) - \varepsilon\sqrt{1-\varepsilon^2})}{\pi}. \tag{15}$$

Table 1 reflects the relative frequency of the random sampling inside the hypercube $[-1,1]^{n+1}$ and the theoretical probability (15) for various ε. The wedge product computation was executed for 10^6 times. A choice of $\varepsilon = 10^{-3}$ gives an almost zero 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. (15), $P \approx 0.9987$.

3.1.2 Bounding hypercone computation

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

$$\mathcal{N}_i = \{\lambda \nabla f_i(\mathbf{x}), \mathbf{x} \in D, \lambda \in \mathbb{R}\}, \tag{16}$$

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

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

$$\mathcal{C}_i^N(\vec{v}_i, \alpha_i) = \{\vec{u}, \langle \vec{u}, \vec{v}_i \rangle = \|\vec{u}\| \cos\alpha_i\}, \tag{17}$$

and holds

$$\langle \vec{u}, \vec{v}_i \rangle \geq \|\vec{u}\| \cos\alpha_i, \quad \forall \vec{u} \in \mathcal{N}_i. \tag{18}$$

We follow the computation of [9], Eq. (3), where the axis vector $\vec{v}_i$ is calculated as an average of the set of vectors

Algorithm 1 (see Fig. 3) { No loop test (NLT) in $\mathbb{R}^{n+1}$}

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

obtained from computing partial derivatives of $f_i(\mathbf{x})$ and α_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 α_i compared to the increasing timing costs.

Once the circular normal bounding hypercone is computed, one can easily construct the *complementary* (or *tangent*) *circular bounding hypercone*

$$\mathcal{C}_i^C(\vec{v}_i, \alpha_i) = \mathcal{C}_i^N(\vec{v}_i, 90^\circ - \alpha_i) \tag{19}$$

which bounds the tangent space for all possible tangent directions of hypersurface $f_i(\mathbf{x}) = 0$. Hypercones $\mathcal{C}_i^C$ and $\mathcal{C}_i^N$ share the same axis and have complementary angles, see [9] for a more detailed explanation.

3.1.3 No loop termination criterion

THEOREM 3.7. *Let $\mathcal{F}(\mathbf{x}) = 0$ be the polynomial system (3) defined in Definition 2.1 and denote by $\mathcal{K}$ the intersection of all tangent hypercones $\mathcal{C}_i^C$, $i = 1, 2, \ldots, n$, of f_i, and Gaussian hypersphere $\mathbb{S}^n$*

$$\mathcal{K} = \Big(\bigcap_{i=1,2,\ldots,n} \mathcal{C}_i^C\Big) \cap \mathbb{S}^n. \tag{20}$$

If there exists a hyperplane α, 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 (3) 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 (3) and assume $\mathcal{I}$ is a k-dimensional variety. Let $\tau_{\mathbf{y}}$ be the (k-dimensional) tangent space of $\mathcal{I}$ at some point $\mathbf{y}$, $\mathbf{y} \in \mathcal{I}$ and $\tau_{\mathbf{y}}^G$ 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,

$$\tau_{\mathbf{y}}^G \subseteq \mathcal{K} \quad \text{for all } \mathbf{y} \in \mathcal{I} \tag{21}$$

and by the assumption

$$\tau_{\mathbf{y}}^G \cap \alpha = \emptyset \quad \text{for all } \mathbf{y} \in \mathcal{I}. \tag{22}$$

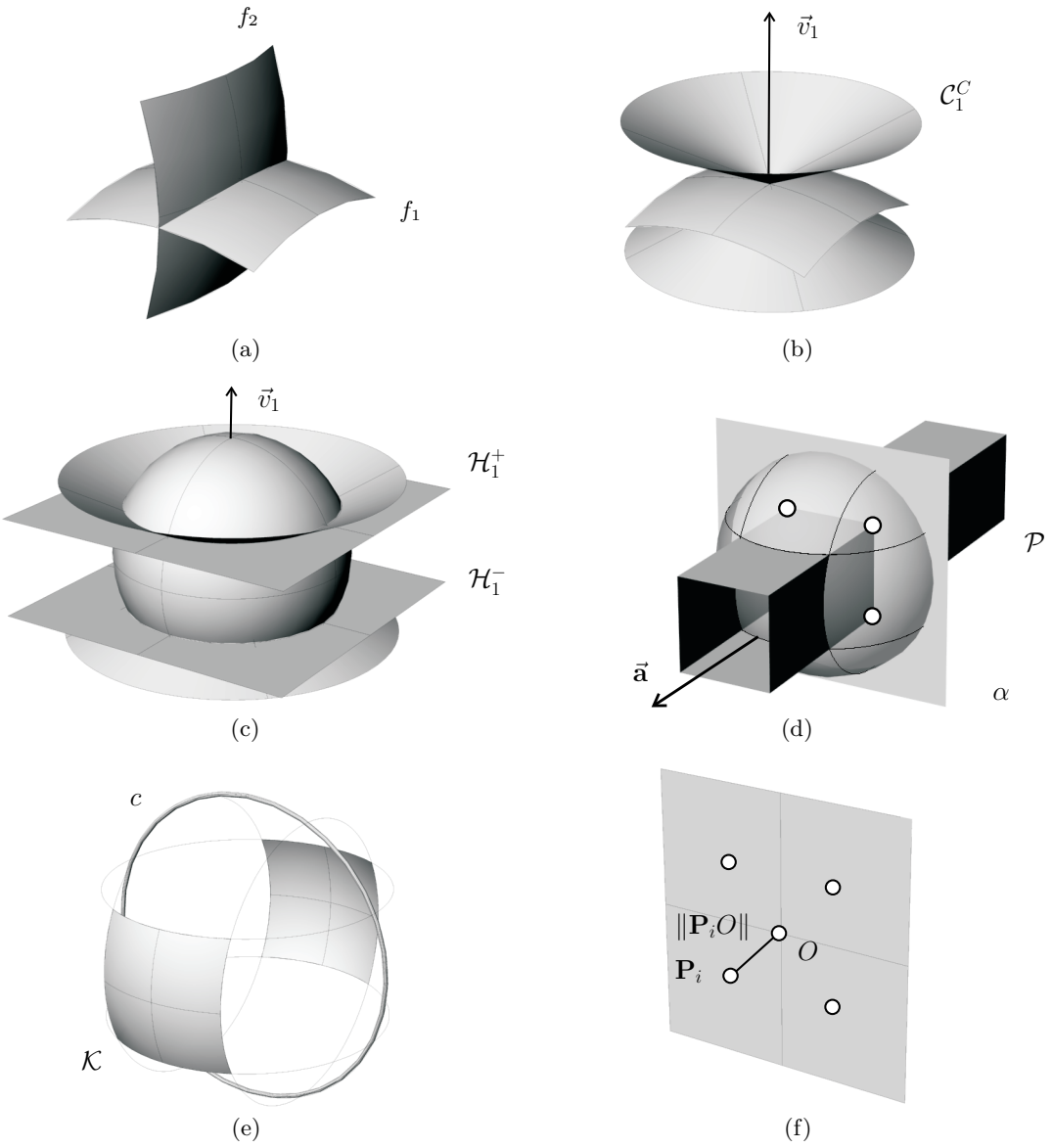


Figure 3: Algorithm 1, the no loop test (NLT): a) n hypersurfaces in $\mathbb{R}^{n+1}$, $n = 2$. b) the tangent bounding cone $\mathcal{C}_1^C$ of the horizontal surface and c) its bounding hyperplanes $\mathcal{H}_1^+$, $\mathcal{H}_1^-$ in $\mathbb{S}^2$. d) Prismatic subset $\mathcal{P}$ in $\mathbb{R}^3$ defined by the intersection of two pairs of bounding hyperplanes and plane α perpendicular to the direction of the prism. e) $\mathcal{K} \subseteq \mathbb{S}^2$ is the bounding set of the (normalized) tangent space of the solution variety and $c = \mathbb{S}^2 \cap \alpha$; if $c \cap \mathcal{K}$ is empty, the intersection curve has no closed loops. f) Instead of verifying this condition, the distances between the intersection points $\mathbf{P}_i = \mathcal{P} \cap \alpha$ and the center of Gaussian hypersphere O are measured. If $\|\mathbf{P}_i O\|_2 < 1$ for all i, the return value of NLT test is `TRUE`.

Let L_α be the linear space related to the hyperplane α ($\dim(L_\alpha) = n$). Then, linear subspaces $\tau_{\mathbf{y}}$ and L_α, of $\mathbb{R}^{n+1}$, are disjoint and hence $\dim(\tau_{\mathbf{y}}) \leq 1$ for all $\mathbf{y} \in \mathcal{I}$. If not, $\dim(\tau_{\mathbf{y}}) \geq 2$, there exist a non-trivial intersection ($\tau_{\mathbf{y}} \cap L_\alpha \neq \emptyset$) of both subspaces as well as its non-trivial image on the Gaussian hypersphere ($\tau_{\mathbf{y}}^G \cap \alpha \neq \emptyset$) which violates the assumption (22) 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

$$\tau_{\mathcal{I}}^G = \bigcup_{\mathbf{y} \in \mathcal{I}} \tau_{\mathbf{y}}^G \tag{23}$$

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

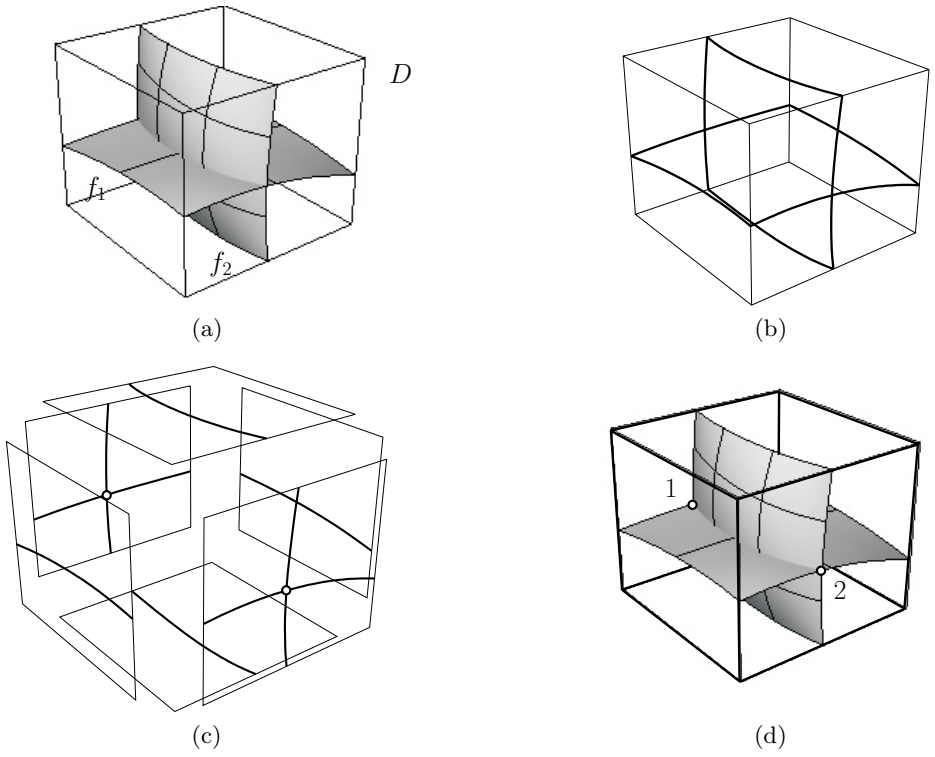


Figure 4: The single component test (SCT): two polynomial constraints in 3 variables: a) two hypersurfaces $f_1(\mathbf{x}) = 0$ and $f_2(\mathbf{x}) = 0$ over domain D, b) restriction of both constraints on the boundary of D, c) problem is reduced to solve $2n = 6$ boundary 2×2 systems, d) two intersection points guarantee a single component inside domain D.

REMARK 3.8. *Theorem 3.7 states that the dimension of the root set $\mathcal{I}$ is at most one, in general. Note that the zero-dimensional roots (points in $\mathbb{R}^{n+1}$) can occur only at the boundary of some hypersurface $f_i(\mathbf{x}) = 0$, $i = 1, \ldots, n$. If this point was interior for all $f_i(\mathbf{x})$, they would have a tangent hyperplane in common 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 (3). See Algorithm 1 and Fig. 3.

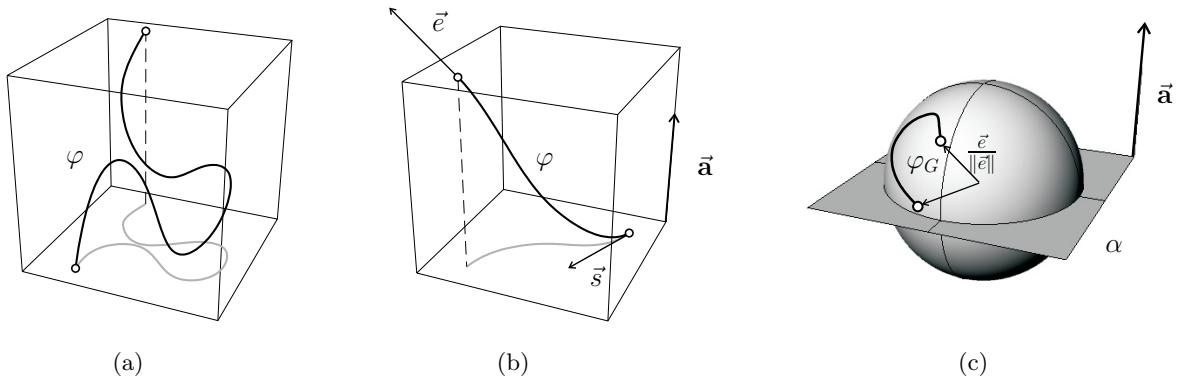
Some steps of the algorithm are now explained in some detail:

- In line 4, we construct a pair of hyperplanes that bound tangent hypercone $\mathcal{C}_i^C$ in $\mathbb{S}^n$ following [9]. These hyperplanes are symmetric with respect to the origin, see Fig. 3c), their normal vector $\vec{v}_i$ coincides with the axis vector of $\mathcal{C}_i^C$ and they intersect $\mathbb{S}^n$ in the same circles as $\mathcal{C}_i^C$.
- The mutual intersections of n pairs of bounding hyperplanes define a prismatic subset in $\mathbb{R}^{n+1}$, a *bounding hyperstrip* $\mathcal{P}$, which is unbounded in the direction perpendicular to all axis vectors $\vec{v}_i$, $i = 1, \ldots, n$; this direction $\vec{\mathbf{a}}$ is computed in line 6, see also Fig. 3d).
- In line 8, the intersection of the prismatic strip and a plane α is computed. There can be many planes that satisfy the condition in Theorem 3.7. However, we choose α to be perpendicular to $\vec{\mathbf{a}}$, the direction of the strip $\mathcal{P}$, in order to minimize the maximum of distances between $\mathbf{P}_i$ and the center of the Gaussian sphere.

In the surface-surface intersection literature, a similar idea of loop detection has been presented in [16], testing the mutual position of the bounding cones of isoparametric curves of the first surface, $Q(s,t)$, with the tangent cone of the second surface. If the bounding cone of s-curves of the first surface lies entirely inside the tangent cone of the second surface, the intersection curve has no loops. If not, the t-curves bounding cone of the first surface and the tangent cone of the second surface are tested and so on. The generalization of this method for a "no loop test" for intersection of n hypersurfaces in R^{n+1}, if at all possible, seems to be unsuitable since each isoparametric bounding cone would be tested with the tangent cones of the remaining hypersurfaces which would mean, in the worst case, n^2 tests.

3.2 Single Component Test

In order to robustly solve underconstrained polynomial system (3) 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, 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



(a) (b) (c)

Figure 5: Monotone solution curve is warrant via the NLT and SCT: A general single component solution curve with its top projection (gray) inside some domain in the solution space a), for which the NLT fails. b) A case with a successful NLT, where the solution curve φ is monotone with respect to some direction $\vec{\mathbf{a}}$, $\langle \varphi', \vec{\mathbf{a}} \rangle > 0$. c) Tangential image φ_G on the Gaussian hypersphere does not intersect the main circle defined by hyperplane α, for which $\vec{\mathbf{a}}$ is the normal vector. Compare with Fig. 3 d).

Algorithm 2 (see Fig. 4) { Single component test (SCT) in $\mathbb{R}^{n+1}$}

1: INPUT: NLT `TRUE`, coefficients of the system (3), domain $D = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \cdots \times [\alpha_{n+1}, \beta_{n+1}]$;
2: $S \leftarrow \emptyset$;
3: **for** $i = 1$ to $(n+1)$ **do**
4: substitute for x_i, $\{\alpha_i, \beta_i\}$, the boundary value of D in the i^{th} direction into system (3);
5: $S_i \leftarrow$ pairs of solutions of systems $\mathcal{F}(\mathbf{x})|_{x_i=\alpha_i} = 0$ and $\mathcal{F}(\mathbf{x})|_{x_i=\beta_i} = 0$;
6: $S = S \bigcup S_i$;
7: **end for**
8: OUTPUT: $|S|$

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 location 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 $\vec{\mathbf{a}} \in \mathbb{R}^{n+1}$) in which the inner product of the tangent of the intersection curve and $\vec{\mathbf{a}}$ have the same sign throughout. This vector $\vec{\mathbf{a}}$ is the normal vector of plane α from Theorem 3.7, constructed in line 6 of the NLT

Algorithm 3 { Subdivision SSI solver via NLT and SCT}

1: INPUT: two rational parametric surfaces $P(u,v)$, $(u,v) \in D_P \subseteq \mathbb{R}^2$ and $Q(s,t)$, $(s,t) \in D_Q \subseteq \mathbb{R}^2$, numerical accuracy ε_{num}, stepsize λ;
2: create an underconstrained polynomial system $\mathcal{F}(\mathbf{x}) = 0$ (see (2) and (3)) over domain $D = D_P \times D_Q$;
3: test the signs of the Bézier representations of each constraint and discard the no-root containing domain;
4: **if** maximal side of the domain $> \varepsilon_{num}$ **then**
5: **if** NLT($\mathcal{F}$)=`TRUE` **then**
6: switch(SCT($\mathcal{F}$))
7: case 0: discard D;
8: case 2: trace the solution curve with the stepsize λ;
9: case ≥ 4: subdivide $\mathcal{F}$ and go to line 3;
10: **else**
11: subdivide $\mathcal{F}$ and go to line 3;
12: **end if**
13: merge traced segments at point(s) on the common boundary;
14: **else**
15: return the center of the domain (numerical tolerance was reached);
16: **end if**
17: OUTPUT: set of polylines in 4-space that interpolate the solution curve as a pre-image of the intersection curve in the model space.

algorithm.

REMARK 3.9. *The distance of $\mathcal{K}$, the patch on the Gaussian hypersphere that bounds all feasible tangent vectors of the solution curve, from the hyperplane α correlates with the monotonicity of the solution curve, see Fig. 3 d), e). More precisely, the more remote $\mathcal{K}$ from α is on $\mathbb{S}^n$, the smaller the deviation between the normal vector of plane α, $\vec{\mathbf{a}}$, and the tangent vector of the solution curve. This distance can be determined by distance between the center of Gaussian*

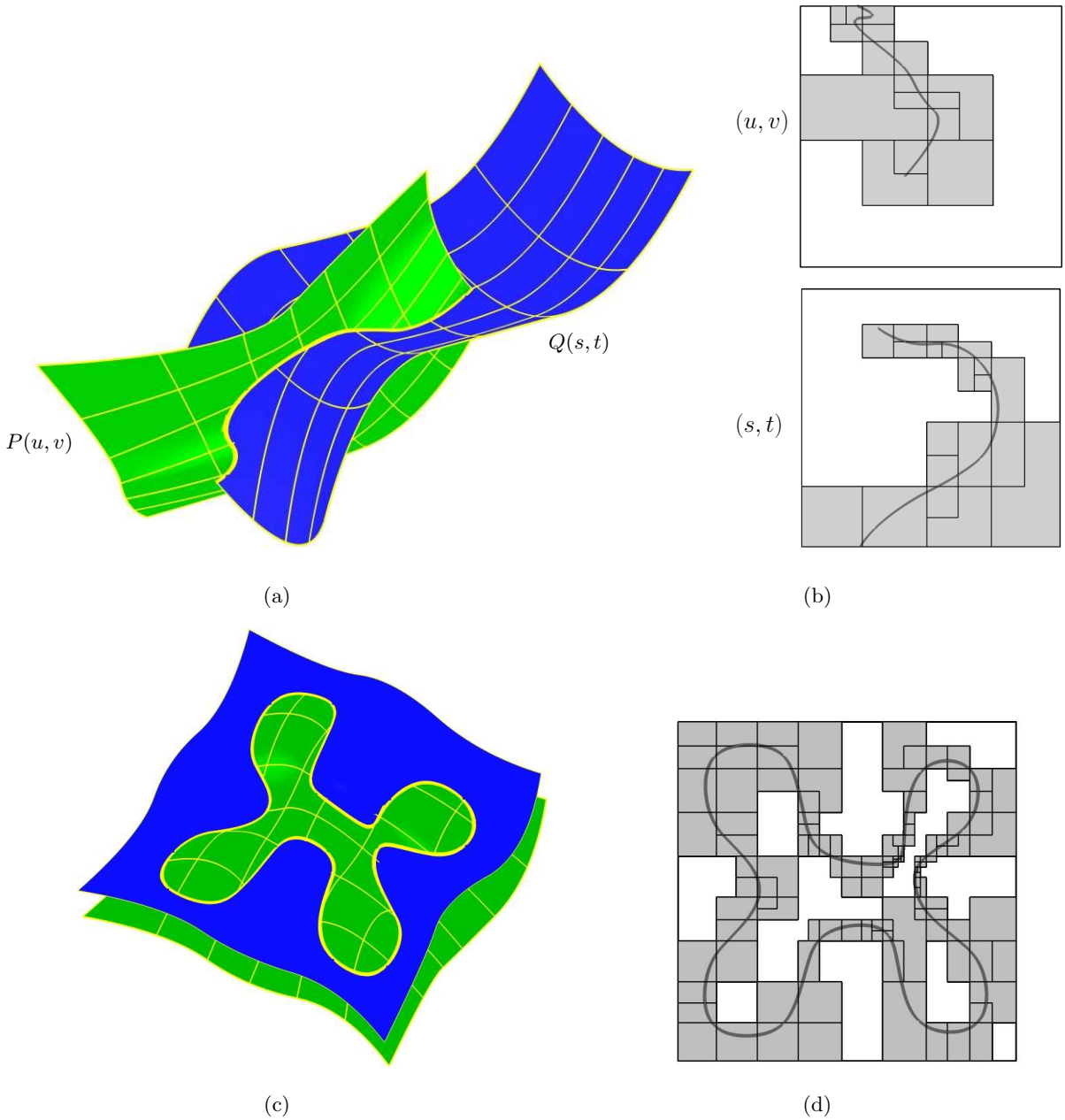
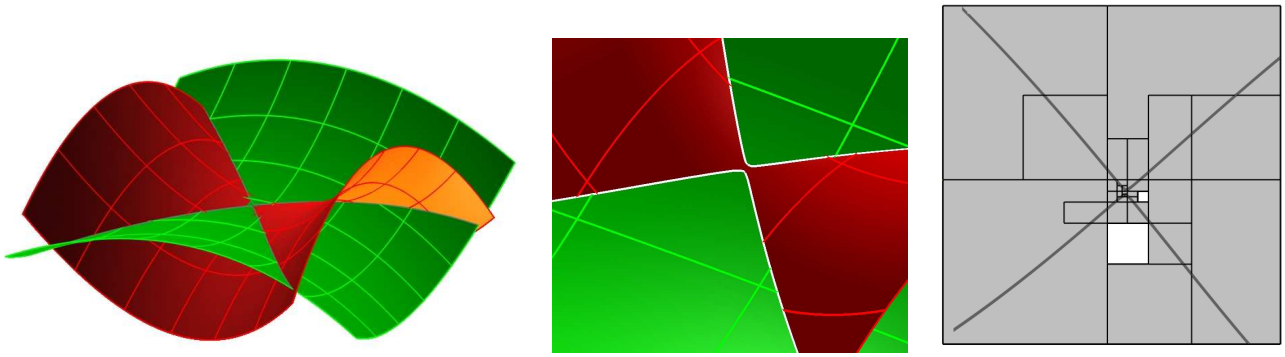


Figure 6: SSI computation via Algorithm 3: a) Two parametric surfaces and their open intersection curve (yellow). b) Preimages of the intersection curve in the parametric subspaces of both surfaces. Grey rectangular domains depict the subdivision scheme until both NLT and SCT are satisfied so that single segments can be traced. c) A closed loop intersection curve and d) its preimage in the (u,v) space of the upper (blue) surface.

hypersphere O and intersection points $\mathbf{P}_i$, see Fig. 3 f). If desired, the user may prescribe the distance between $\mathcal{K}$ and α by placing a tighter bound on $\|\mathbf{P}_iO\|_2$, 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 (segment of the) curve touches any 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.*

While the presented results hold for $\mathbb{R}^n$, as one possible application, we now formulate a subdivision-based SSI



(a) (b) (c)

Figure 7: a) Two hyperbolic paraboloids are intersected in a curve with an almost-singular point b). c) The (u, v) space of the green surface along with the preimage of the solution curve.

Table 2: Timings of presented SSI examples for the curve tracing parameter $\lambda = 0.01$, the numerical tolerance $\varepsilon_{num} = 10^{-6}$ and normalized parametric spaces of both surfaces.

SSI Example	Number of subdivisions	Number of traced segments	Total time min:sec.
Fig. 6 a)	311	24	0:00.4
Fig. 6 c)	4605	152	0:04.5
Fig. 7	321	37	0:01.2
Fig. 8 a)	522	33	0:00.8
Fig. 8 c)	586	50	0:00.7
Fig. 9 a)	19910	1001	1:13.6
Fig. 9 c)	1161	175	0:29:5

solver, see Algorithm 3, which exploits both above described tests.

4. EXAMPLES

In this section we present several examples of SSI computations. All examples were created using the GuIrit GUI user interface [4] of the Irit solid modeling system [5]. The SSI solver was implemented as a shared library extension (dll) in GuIrit.

All presented examples were tested on a PC with an Intel(R) Pentium(R) CPU (2.8GHz), 1 GB of RAM. Times might differ depending on the topological complexity of each specific example and on a numerical precision which is required in the curve tracing stage. With normalized surfaces to unit cube and curve tracing step $\lambda = 0.01$, meaning that two neighboring solution points in $\mathbb{R}^4$ are approximately within this distance, the computation timings varied, see Table 2, from a few seconds in the case of simple configuration 6 a) and b) to more than one minute for 9 a).

The intersection curve of two parametric surfaces $P(u, v)$ and $Q(s, t)$ and its preimages in (u, v) and (s, t) spaces are displayed at Fig. 7 b). Grey rectangles are projections of those domains in $\mathbb{R}^4$ which contain only one solution segment. Note that these segments start and end on the boundary of the domain in $\mathbb{R}^4$, but their projections onto $\mathbb{R}^2$ need not to inherit this property.

Due to the Remark 3.9, one can better safeguard the numerical tracing stage. Close by regions with complex topology, as in Fig. 9 b), where jumps from one segment of solution curve to another might occur, are isolated a priori.

Note that Algorithm 3 guarantees the correct topology of the solution curve in $\mathbb{R}^4$. Though, the topologies of the preimage curve in the (u, v) space and in model space may obviously differ, see Fig. 8. Hence, closed surfaces with cylindrical or toroidical topology require special attention in the interpretation of the result, since the solver reports the number of solution segments in $\mathbb{R}^4$.

5. CONCLUSION AND FUTURE WORK

Demonstrated on the SSI problem, in this work, we 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 numerical tolerance is reached, which causes slow and inaccurate results. Similarly, if the constraint hypersurfaces have tangent contact, a special treatment similar to the one presented in [14], is desired.

Other geometric problems whose solution is described by $n \times (n + 1)$ polynomial system, like medial axis computation, rounding of two surfaces or some kinematic problems,

[4] www.cs.technion.ac.il/~gershon/GuIrit
[5] www.cs.technion.ac.il/~irit

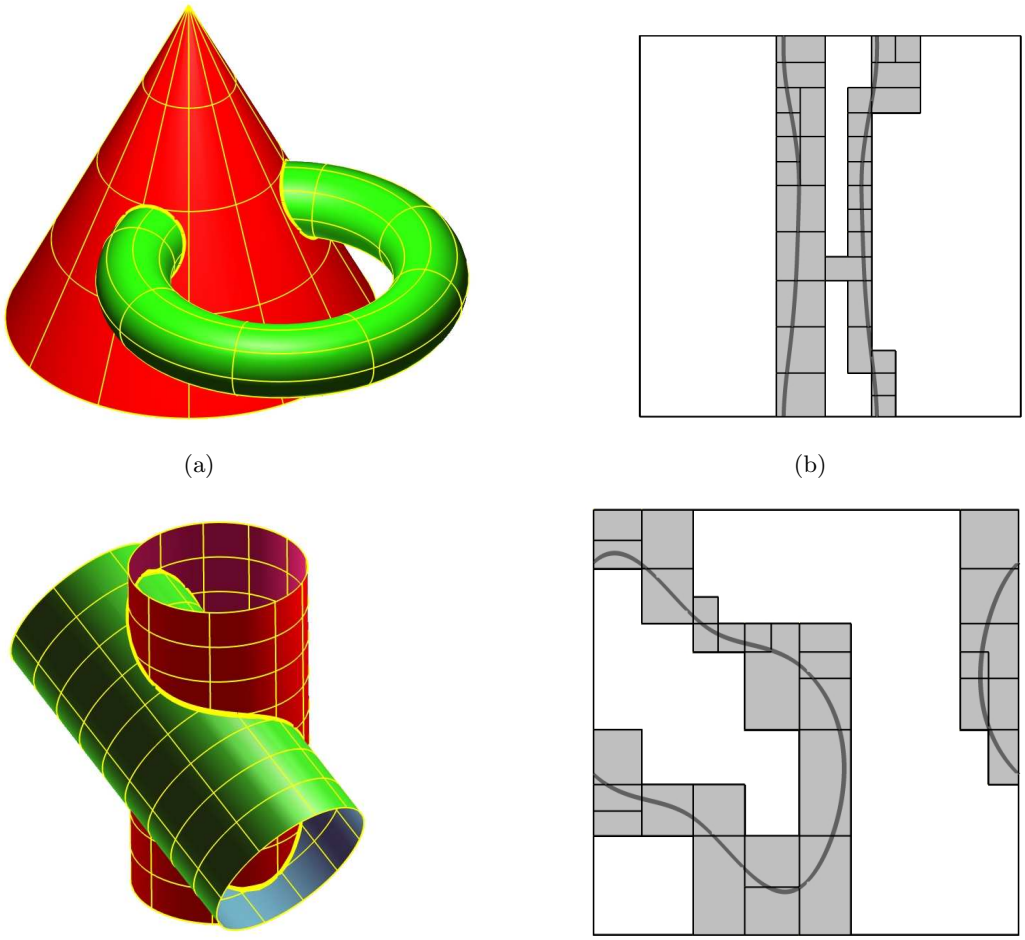


Figure 8: Inconsistent topology in model and parametric spaces. a) The intersection curve consists of two closed loops in model space, whereas b) in the parametric space of the torus, two open segments are obtained. c) Two cylinders are intersected in one closed loop. d) Yet, two segments are detected in parameter space.

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

[1] C. Bajaj, C. M. Hoffmannn, R. E. Lynch, and J. E. H. Hopcroft. Tracing surface intersections. *Computer Aided Geometric Design*, (5):285–307, 1988.

[2] G. Barequet and G. Elber. Optimal bounding cones of vectors in three and higher dimensions. *Information Processing Letters*, 93:83–89, 2005.

[3] R. E. Barnhill and S. N. Kersey. A marching method for parametric surface surface intersection. *Computer Aided Geometric Design*, (7):257–280, 1990.

[4] D. A. Cox, J. B. Little, and D. O'Shea. *Using algebraic geometry*. Springer, 2005.

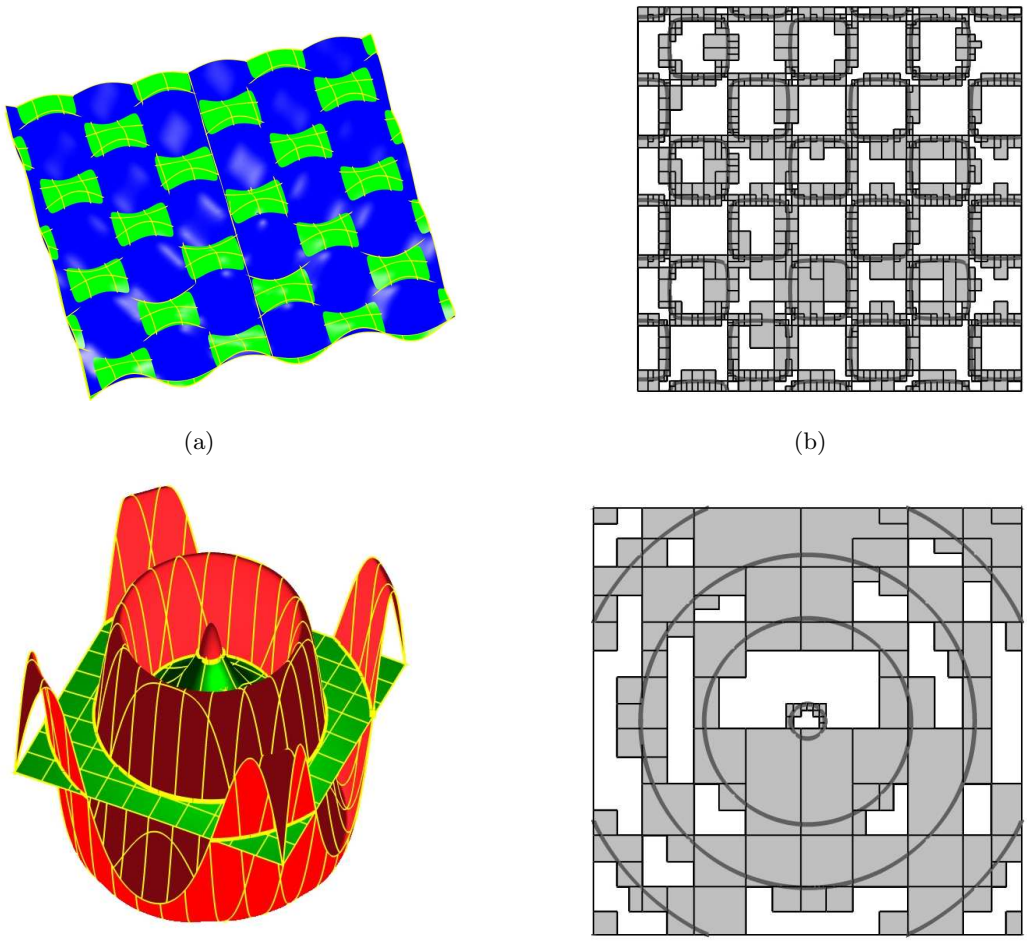
[5] T. Dokken, V. Skytt, and A. M. Ytrehus. Recursive subdivision and iteration in intersections. In T. Lyche and L. Schumaker, editors, *Mathematical methods in computer aided geometric design*, volume **5**, pages 207–214. 1989.

[6] G. Elber and T. Grandine. Efficient solution to systems of multivariate polynomials using expression trees. In *Tenth SIAM Conference on Geometric Design and Computing*, 2007.

[7] L. H. Figueirdo. Surface intersection using affine arithmetic. In *Proceedings of Graphics Interface '96*, pages 161–170, 1996.

[8] T. A. Grandine and F. W. Klein. New approach to the surface intersection problem. *Computer Aided Geometric Design*, (14):111 – 134, 1997.

[9] I. Hanniel and G. Elber. Subdivision termination criteria in subdivision multivariate solvers using dual



(a) (b) (c) (d)

Figure 9: a) Time demanding example, consisting of 28 intersecting segments b). c) An intersection curve (yellow) of two rational bicubic B-spline surfaces. d) Solution space of the green surface. These examples were provided courtesy of Tom Grandine, The Boeing Company.

hyperplanes representations. *Computer Aided Design*, (39):369–378, 2007.

[10] J. Hass, R. T. Farouki, C. Y. Han, X. Song, and T. W. Sederberg. Guaranteed consistency of surface intersections and trimmed surfaces using a coupled topology resolution and domain decomposition scheme. *Advances in Comput. Math.*, (27):1–26, 2007.

[11] J. Hoschek and D. Lasser. *Fundamentals of computer aided geometric design*. AK Peters, 1993.

[12] D. Manocha, A. Varshney, and H. Weber. Evaluating surface intersections in lower dimensions. In *Second International Conference on Curves and Surfaces*, 1993.

[13] B. Mourrain and J.-P. Pavone. Subdivision methods for solving polynomial equations. Technical Report 5658, INRIA Sophia Antipolis, 2005.

[14] H. Mukundan, K. H. Ko, T. Maekawa, T. Sakkalis, and N. M. Patrikalakis. Tracing surface intersections with validated ode system solver. In N. P. G. Elber and P. Brunet, editors, *ACM Symposium on Solid Modeling and applications*, 2004.

[15] M. Reuter, T. S. Mikkelsen, E. C. Sherbrooke, T. Maekawa, and N. M. Patrikalakis. Solving nonlinear polynomial systems in the barycentric bernstein basis. *Visual Computer*, (24):187–200, 2008.

[16] T. W. Sederberg and R. J. Meyers. Loop detection in surface patch intersection. *Computer Aided Geometric Design*, (5):161–171, 1988.

[17] R. Sharma and O. P. Sha. A tracing method for parametric bézier triangular surface plane intersection over triangular domain. *J. Indian Inst. Sci.*, (85):161–182, 2005.

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

[19] A. J. Sommese and C. W. Wampler. *The numerical solution of systems of polynomials arising in engineering and science*. World Scientific, 2005.