Reliable Location with Respect to the Projection of a Smooth Space Curve*

Rémi Imbach Guillaume Moroz

Marc Pouget§

INRIA Nancy Grand Est,
LORIA laboratory, Nancy, France.
firstname.name@inria.fr

Abstract

Consider a plane curve $\mathcal{B}$ defined as the projection of the intersection of two surfaces in $\mathbb{R}^3$ or as the apparent contour of a surface. In general, $\mathcal{B}$ has node or cusp singular points and thus is a singular curve. Our main contribution is the computation of a data structure answering point location queries with respect to the subdivision of the plane induced by $\mathcal{B}$. This data structure is composed of an approximation of the space curve together with a topological representation of its projection $\mathcal{B}$. Since $\mathcal{B}$ is a singular curve, it is challenging to design a method based only on reliable numerical algorithms.

In recent work, the authors show how to describe the set of singularities of $\mathcal{B}$ as regular solutions of a so-called ball system suitable for a numerical subdivision solver. Here, the space curve is first enclosed in a set of boxes with a certified path-tracker to restrict the domain where the ball system is solved. Boxes around singular points are then computed such that the correct topology of the curve inside these boxes can be deduced from the intersections of the curve with their boundaries. The tracking of the space curve is then used to connect the smooth branches to the singular points. The subdivision of the plane induced by $\mathcal{B}$ is encoded as an extended planar combinatorial map allowing point location. We experimented with our method, and we show that our reliable numerical approach can handle classes of examples that symbolic methods cannot.

Keywords: Singular curve topology, point location algorithm, geometric approximation
AMS subject classifications: 65D99

*Submitted: November 9, 2017; Accepted: January 15, 2018.
§This research was supported by the ANR JCJC SingCAST (ANR-13-JS02-0006).

1 Introduction

Let $\mathcal{C}$ be a smooth space curve defined as the intersection of two surfaces $P = Q = 0$, with P, Q real smooth functions in x, y, z. We aim at computing, in a compact domain $\mathbf{B}_0 \subset \mathbb{R}^2$, the geometry and the topology of $\mathcal{B} = \pi_{(x,y)}(\mathcal{C})$ where $\pi_{(x,y)} : \mathbb{R}^3 \to \mathbb{R}^2$ denotes projection into the (x, y)-plane. In general, $\mathcal{B}$ is not smooth and has singular points, i.e. points where $\mathcal{B}$ has no well defined tangent direction. Generically, the only singular points of a projected curve are transversal crossings of two branches of the curve, called nodes. A special occurrence of our problem is the case where $Q = P_z$, the derivative of P with respect to z. $\mathcal{B}$ is then called the apparent contour of the surface $P = 0$, and generic singular points of $\mathcal{B}$ are nodes and cusps, i.e. projections of points where $\mathcal{C}$ has a vertical tangent. Figure 1 shows, for a torus $P = 0$, its intersection with the surface $P_z = 0$ in bold, and its apparent contour with cusp and node singularities.

By computing the geometry of $\mathcal{B}$ in $\mathbf{B}_0$, we mean being able to draw $\mathcal{B} \cap \mathbf{B}_0$ with an arbitrary precision. To reach this goal, we use a certified interval path tracker to compute a sequence of boxes (i.e. multi-dimensional extensions of intervals) of width as small as desired, such that each box intersects $\mathcal{C}$, and their union encloses $\mathcal{C}$. The projection of these boxes is thus a geometric approximation of the plane curve $\mathcal{B}$.

We want our topological encoding to be able to answer location queries. The curve $\mathcal{B}$ decomposes the box $\mathbf{B}_0$ into connected components of dimension two that we call faces. For a point p in $\mathbf{B}_0$ but not on $\mathcal{B}$, one wants to find the face to which p belongs. Such queries can be answered by considering a combinatorial encoding of the embedding of the projected curve, i.e. the subdivision of $\mathbf{B}_0$ induced by the curve $\mathcal{B}$. We use Combinatorial Maps (CMaps defined in Definition 1.2) and their extensions to the non-connected case, the eXtended Planar Maps (XPMaps defined in Definition 1.3), to encode embeddings of plane curves. In a first step, we focus on singularities and isolate the singular points of $\mathcal{B}$ in boxes of width as small as desired. In the second step we compute the topology in these isolating boxes. Finally, we use the tracking of the space curve to connect the singular points and construct the representation of the embedding.

The work presented here is a first step toward the computation of the topology with reliable approximated geometry of the apparent contour of a smooth manifold of $\mathbb{R}^n$. Such a manifold arises naturally in the design of parallel or cable mechanisms, and its apparent contour represents the boundary of the workspace of such a mechanism [22]. The encoding of the topology and the geometry we propose can thus be seen as a reliable tool to validate a robot configuration, to check if the clearance with respect to special configurations is large enough, or to check whether or not the robot passes through a singularity during a motion.

The paper is organized as follows. Section 2 describes how the curve $\mathcal{C}$ is enclosed by tracking. Section 3 reviews the encoding of singularities of $\mathcal{B}$ by the ball system and shows how the enclosure of $\mathcal{C}$ is used to restrict its solution domain. For an apparent contour, an algorithm is presented to determine the type, node or cusp, of a singularity isolated with the ball system. Section 4 is dedicated to the computation of the local topology at special points, i.e. singular points and x-extreme points. Section 5 explains the construction of the XPMap representing the embedding of the curve $\mathcal{B}$. Section 6 reports experiments on the implementation of our numerical approach. The remainder of Section 6 presents previous work, details our contributions, formally defines our geometric and topological representations, and reviews some basics of reliable numerical interval solvers.

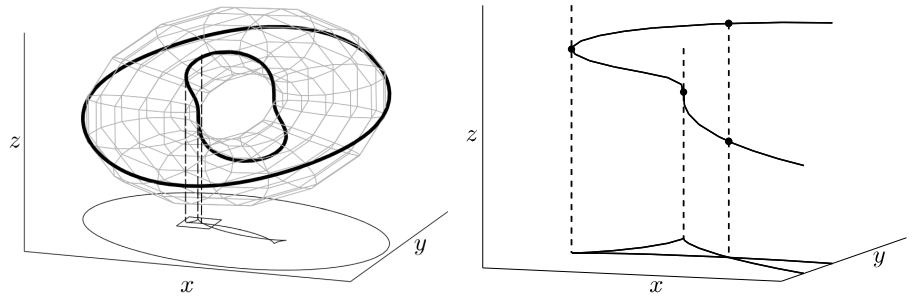


Figure 1: Left: a torus $P = 0$, the curve $P = P_z = 0$ in bold, its apparent contour, and a zoom zone. The vertical dashed lines show fibers over three singularities. Right: a zoom, with preimages near the projection of cusp and node singularities.

1.1 Previous work

State-of-the-art symbolic methods that compute the topology of plane real curves defined by polynomials are based on the Cylindrical Algebraic Decomposition, and use resultant and sub-resultant theory to isolate critical points [10, 24]. One advantage of these methods is that they can handle any type of singularity of the curve. The drawbacks are their high complexity as a function of the degree of the curve and the global aspect of the approach: computing the topology in the whole plane or in a small box have almost the same cost.

Numerical methods based on interval arithmetic are able to compute and certify the topology of a non-singular curve [15, 21, 27]. One advantage is the local aspect of the approach: the topology can be computed in a small box, and the smaller the box the faster the computation. However, they fail near any singular point of the curve. Isolating singularities of a planar curve $f(x, y) = 0$ with a numerical method is a challenge, since the set of singular points is described by the non-square system $f = \frac{\partial f}{\partial_x} = \frac{\partial f}{\partial_y} = 0$, and singularities are not necessarily regular solutions of this system. The latter system can be translated into a square system using combinations of its equations with first derivatives [6], and non-regular solutions can be handled through deflation systems [19, 26], but the resulting systems are usually still overdetermined.

We are not aware of a numerical algorithm that can certify in practice the computation of the topology of any singular curve, but several promising approaches have been presented. The subdivision approach presented in [3] is an extension of the singular case of the one in [27], relying on global non-adaptive separation bounds for algebraic systems. This approach can theoretically certify the topology of any singular curve, but due to the worst-case bounds, this algorithm cannot be practical. A numerical algebraic geometric approach is presented by [20] using irreducible decomposition, generic projection and plane sweep, deflation and homotopy to compute the topology of a singular curve in any codimension. Even though this work has been implemented in [2][1], certification of all the algorithm steps appears to be challenge. The numerical approach in [5], based on Bezoutian and eigenvalue computation, can handle singular curves, but even if multiprecision gives accurate results, no certification is provided.

[1] See also http://www.bertinireal.com/.

Instead of designing a general numerical method able to handle any singular curve, an alternative is to focus on restricted classes of singular curves. A natural example is when the plane curve to be studied is the projection of a smooth space curve living in a higher dimension. According to the classification of singularities of mappings (see [1, 8, 29] for example), it appears that the generic singularities of such a projected curve are only transversal intersections of two branches, and in the case of the apparent contour of a surface, ordinary cusps also occur. From an algorithmic point of view, the authors of [7] use these elements to derive an algorithm isolating the singularities arising in generic mappings from $\mathbb{R}^2$ to $\mathbb{R}^2$. Our problem of isolating the singularities of the projection of a generic algebraic space curve was investigated in [13]. The authors use resultant and sub-resultant theory to represent the singularities as the solutions of a regular bivariate system suited to a branch and bound solving approach. To overcome the drawbacks of resultant and sub-resultant, [12] contains studies of the geometric configurations of the space curve that induce singularities on the projected curve, and descriptions of them as the regular solutions of a four dimensional system.

1.2 Detailed overview

Our main contribution is the computation of a data structure allowing location of a point with respect to the projection of a smooth space curve. Our data structure is the union of a geometric approximation of the space curve and a topological representation of its projection.

For the geometric approximation of the space curve $\mathcal{C}$ and thus its projection $\mathcal{B}$, we compute a sequence of 3-dimensional boxes $(\boldsymbol{C}_i)_{i=1}^m$ enclosing all the connected components of $\mathcal{C}$, that is $\mathcal{C} \subset \bigcup_{i=1}^m \boldsymbol{C}_i$. We use a reliable numerical solver (Algorithm 1) to find intersections of the curve with the boundary of the input box and at least one point on each connected component of $\mathcal{C}$ using a critical point method. We then use these points as starting points for a tracking algorithm (Specified in Algorithm 2 and detailed in Appendix A) which is an adapted version of the one in [21]. Since we want to only use numerical algorithms, we must avoid some degenerate configurations for the systems of critical points and boundary points. Such assumptions, stated in Section 2.1, are satisfied for generic curves and projections.

For the topological representation, the basic numerical tools are the reliable numerical solver and tracker already used for the geometric approximation. It is worth noting that while the topology of the space curve $\mathcal{C}$ is directly given by the connected components of boxes enclosing it, the topology of the plane curve $\mathcal{B}$ has little to do with its enclosure by the projection of the box enclosure of $\mathcal{C}$. The general idea is first to isolate the singularities and critical points of the plane curve $\mathcal{B}$, then to refine the boxes around those special points until the topology inside these boxes is trivial. Finally, we use smooth path tracking to connect the singularities and the critical points. Since we use numerical algorithms, we require assumptions in Sections 3.1 and 4.2, that are generically satisfied.

The first step is to isolate the singularities of the plane curve $\mathcal{B}$. In previous work [12], we have shown how the singularities can be described as the regular solutions of a so-called ball system involving 4 equations in 4 unknowns. The ball system could be solved in $\mathbb{R}^4$ with a reliable numerical solver; however, in four dimensions, this global subdivision approach becomes costly. To overcome this issue, we use the box enclosure of the space curve to restrict the solution domain of the ball system.

The second step is to compute *witness* boxes for the singularities and critical points of $\mathcal{B}$, that is isolating boxes such that the topology of the curve inside these boxes can

be deduced from the intersections of the curve with their boundaries. Special care is devoted to the refinement of boxes such that the intersections of the curve with their boundaries do not eventually occur at the corners.

The last step uses the tracking of the space curve to connect its smooth branches to the witness boxes of the singular and critical points and thus compute a combinatorial map for each connected component. The construction of the extended planar map encoding the embedding of $\mathcal{B}$ is incremental on the connected components. The point location algorithm is based on a vertical ray shooting principle but eventually needs to compute intersections of the space curve $\mathcal{C}$ with planes.

We implemented the presented algorithms and tested them to compute the topology of apparent contours of algebraic surfaces of degrees up to 15. Our experiments show that our method can handle classes of examples symbolic methods cannot handle, and that multi-precision arithmetic is needed for such difficult examples. More specifically, for isolation of the singularities, the efficiency of our approach to restrict the solving domain of the ball system is demonstrated.

1.3 Notation and definitions

In this paper, real intervals are connected sets $[a, b]$ with $a, b \in \mathbb{R} \cup \{\pm\infty\}$ and $a \leq b$. Lowercase boldface letters denote real intervals and uppercase boldface letters denote boxes, that is, vectors of intervals. Let $\mathbf{x}$ be a real interval, let $l(\mathbf{x})$ denote its lower bound, $u(\mathbf{x})$ its upper bound and $\mathrm{w}(\mathbf{x})$ its width, defined as $u(\mathbf{x}) - l(\mathbf{x})$ if $\mathbf{x}$ is bounded and ∞ otherwise. If $\mathbf{x}$ is bounded, $\mathrm{m}(\mathbf{x})$ denotes its midpoint. If $\epsilon \in \mathbb{R}^+$, $\epsilon\mathbf{x}$ holds for $[\mathrm{m}(\mathbf{x}) - \epsilon\frac{\mathrm{w}(\mathbf{x})}{2}, \mathrm{m}(\mathbf{x}) + \epsilon\frac{\mathrm{w}(\mathbf{x})}{2}]$. Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a box, $\partial\mathbf{X}$ denotes the boundary of $\mathbf{X}$ and $i(\mathbf{X}) = \mathbf{X} \setminus \partial\mathbf{X}$ its interior, where $A \setminus B$ is the set of elements of A not in B. The width $\mathrm{w}(\mathbf{X})$ is defined as $\max_{1 \leq i \leq n} \mathrm{w}(\mathbf{x}_i)$ and the midpoint $\mathrm{m}(\mathbf{x})$ as $(\mathrm{m}(\mathbf{x}_1), \ldots, \mathrm{m}(\mathbf{x}_n))$ if $\mathbf{X}$ is bounded. If $\epsilon \in \mathbb{R}^+$, $\epsilon\mathbf{X}$ is defined to mean $(\epsilon\mathbf{x}_1, \ldots, \epsilon\mathbf{x}_n)$.

Uppercase letters denote sets of boxes. A *domain* of $\mathbb{R}^n$ is a set defined as $\mathbf{X} \setminus (\bigcup_{\mathbf{Y} \in \mathrm{Y}} i(\mathbf{Y}))$ where $\mathbf{X}$ is a box in $\mathbb{R}^n$ and Y a possibly empty set of boxes, of $\mathbb{R}^n$. If X is a domain of $\mathbb{R}^n$, ∂X denotes its boundary.

For a differentiable real function P of the variables $x_1, \ldots, x_n$, P_{x_i} denotes its partial derivative with respect to x_i, and $P_{x_i x_j}$ its derivative with respect to x_i and x_j. Let $P_1, \ldots, P_n$ be n real functions of $x_1, \ldots, x_n$, a solution x of $P_1 = \ldots = P_n = 0$ is *regular* if the Jacobian matrix $A = [(P_i)_{x_j}]$ evaluated at x has full rank; otherwise x is *singular*.

Cursive letters denote sets of points. We mostly work with points, boxes and curves in $\mathbb{R}^2$, $\mathbb{R}^3$ or $\mathbb{R}^4$. We use the following naming scheme: objects in $\mathbb{R}^2$ are named with the letter B, in $\mathbb{R}^3$ with the letter C and in $\mathbb{R}^4$ with the letter D.

A *graph* is a triplet $G = (V, E, I)$ where V, E are two finite sets whose elements are called *vertices* and *edges*, respectively, and $I : E \to V \times V$ is called the incidence relation of G. G is *directed* (resp. *non-directed*) if images of edges by I are seen as ordered sets (respectively non-ordered pairs).

1.4 Geometrical and topological representations

Our first goal is to compute a geometrical approximation of the projected curve $\mathcal{B}$ in a box $\mathbf{B}_0$. Let us formalize this notion as follows.

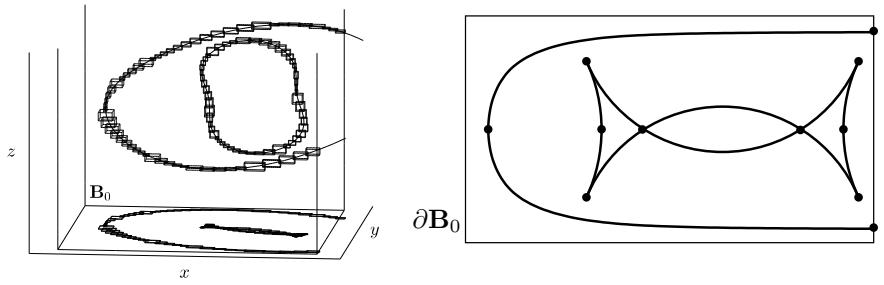
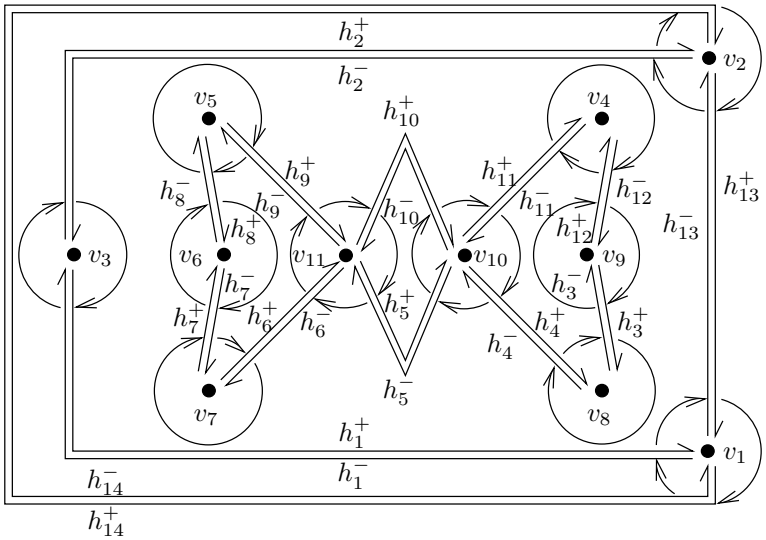


Figure 2: Left: a δ-approximation of $\mathcal{C}$ in $\mathbf{B}_0 \times \mathbb{R}$, and a δ-approximation of $\mathcal{B}$ in $\mathbf{B}_0$. Right: an embedding of the graph G for the apparent contour of a torus. Black circles represent the points associated with the vertices of G, and thick curves the curves associated with its edges. Thin lines represent the boundary of $\mathbf{B}_0$.



$H_0 = \emptyset$, $H_1 = \{h_1^+, h_1^-, h_2^+, h_2^-, h_{13}^+, h_{13}^-, h_{14}^+, h_{14}^-\}$, $H_2 = \{h_3^+, h_3^-, \ldots, h_{12}^+, h_{12}^-\}$
edges: (h_i^+, h_i^-) for $1 \leq i \leq 14$, vertices: $v_1 = (h_1^-, h_{14}^+, h_{13}^-), \ldots$
faces: $f_0 = \emptyset$, $f_1 = (h_{13}^+, h_{14}^+)$, $f_2 = (h_1^+, h_2^+, h_{14}^-)$, $f_3 = (h_1^+, h_{13}^-, h_2^-)$, $f_4 = (h_3^+, h_4^-, \ldots, h_{12}^-)$, $f_5 = (h_3^-, h_{12}^+, h_{11}^-, h_4^+)$, $f_6 = (h_5^+, h_{10}^-)$, $f_7 = (h_6^+, h_9^-, h_8^+, h_7^-)$.
ext: (H^1, f_1), (H^2, f_4) *cont*: (f_1, f_0), (f_4, f_3)

Figure 3: An XPMap encoding the topology of the apparent contour of the torus in $\mathbf{B}_0$. Each edge is represented by a pair of half-edges, which is a cycle of the permutation α. Curved arrows around vertices represent cycles of the permutation φ.

Definition 1.1 (δ-approximation) *Let $\mathcal{X}$ be a subset of $\mathbb{R}^n$. A sequence of boxes $(\mathbf{X}_i)_{i=1}^m$ is a δ-approximation of $\mathcal{X}$ if $\mathcal{X} \subset \bigcup_{i=1}^m \mathbf{X}_i$, and for all $1 \leq i \leq m$, $\mathrm{w}(\mathbf{X}_i) \leq \delta$ and $\mathcal{X} \cap \mathbf{X}_i \neq \emptyset$.*

We will consider in particular δ-approximations of $\mathcal{C}$ in $\boldsymbol{C}_0 = \mathbf{B}_0 \times \mathbb{R}$ or in a domain C_0. Obviously the projection of a δ-approximation of $\mathcal{C}$ in $\boldsymbol{C}_0$ is a δ-approximation of $\mathcal{B}$ in $\mathbf{B}_0$, in particular it yields a drawing of $\mathcal{B}$ with the guarantee that any point of the drawing is at most at distance δ from the original curve. The left part of Figure 2 represents a δ-approximation of the curve $\mathcal{C}$ in $\boldsymbol{C}_0$ and its projection, for the torus example.

We also want to encode the topology of the curve $\mathcal{B} \cap \mathbf{B}_0$ as an embedding of a graph G, to be able to answer location queries for points of $\mathbf{B}_0$. This is achieved by computing a set V of vertices representing points on the curve with at least one vertex per connected component, and such that $(\mathcal{B} \cap \mathbf{B}_0) \setminus V$ is a set of smooth curves identified as the set E of edges. The right part of Figure 2 shows an embedding of such a graph G when $\mathcal{B} \cap \mathbf{B}_0$ is the apparent contour of the torus. The embedding of G defined by $\mathcal{B} \cap \mathbf{B}_0$ is encoded by an extended planar map. More precisely, each connected component of $(\mathcal{B} \cap \mathbf{B}_0) \cup \partial\mathbf{B}_0$ is encoded by a combinatorial map, and the inclusion of a component in a face of another is encoded by the extended map. We thus recall the definition of these combinatorial structures.

Definition 1.2 ([18, 16]) *A Combinatorial Map (CMap) is a triplet (H, σ, α) where H is a set of half-edges, σ is a permutation on H and α an involution on H.*

An edge of G is associated with a cycle of α, and a vertex with a cycle of σ. Cycles of σ encode counter-clockwise orderings of outgoing half-edges around vertices. The cycles of the permutation $\varphi = \sigma^{-1} \circ \alpha$ describe the faces of the combinatorial map. All such face cycles are in counter-clockwise order, except for one which is called the *exterior face* of the combinatorial map.

When $(\mathcal{B} \cap \mathbf{B}_0) \cup \partial\mathbf{B}_0$ has several connected components, it remains to encode the containment relationship between the exterior face of each CMap within a non-exterior face of another CMap. In [16] a combinatorial structure, called eXtended Planar Map (XPMap) is proposed to represent such a relationship. Let $\sigma|_{H^i}$ (resp. $\alpha|_{H^i}$) denote the restriction of the permutation σ (resp. α) defined on H to elements of a subset $H^i \subseteq H$.

Definition 1.3 ([16]) *An eXtended Planar Map (XPMap) is a tuple $(H, H_0, \sigma, \alpha,$ ext, $cont)$ where $H = H^1 \cup \ldots \cup H^{n'}$ is the union of pairwise disjoint non empty sets of half-edges, H_0 is an empty set of half-edges representing the infinite face, $(H^i, \sigma|_{H^i}, \alpha|_{H^i})$ are CMaps for all $1 \leq i \leq n'$, ext is a relation that labels one face of each CMap $(H^i, \sigma|_{H^i}, \alpha|_{H^i})$ as the exterior face, and $cont$ is a relation that assigns each exterior face to one non-exterior face of a CMap $(H^i, \sigma|_{H^i}, \alpha|_{H^i})$ or to the infinite face.*

In our case, the infinite face is $\mathbb{R}^2$, Figure 3 shows an XPMap characterizing $(\mathcal{B} \cap \mathbf{B}_0) \cup \partial\mathbf{B}_0$ as an embedding of the apparent contour of the torus. The cycle $f_1 = (h_{13}^+, h_{14}^+)$ is the exterior face of the connected component of $(\mathcal{B} \cap \mathbf{B}_0) \cup \partial\mathbf{B}_0$ containing $\partial\mathbf{B}_0$. This exterior face f_1 is contained in the infinite face f_0 creating a hole in it. More gererally, any non-infinite face of an XPMap can be described by exactly one clockwise cycle of half-edges and possibly several inner counter-clockwise cycles of half-edges corresponding to the exterior face of the CMaps it contains.

1.5 Interval arithmetic tools

The certification of our algorithms is based on interval arithmetic (see [14, 23, 25, 28]), that is a way of computing with intervals (whose endpoints are floating numbers) instead of computing with floating numbers, while carefully handling rounding to overcome numerical approximations that naturally occur with floating number arithmetic.

1.5.1 Convergent interval functions

If F be a function, an inclusion function for F is a function defined on boxes such that the image of a box $\mathbf{X}$ is a box $\mathbf{Y} \supseteq \{F(x)|x \in \mathbf{X}\}$. As a consequence, if $0 \notin \mathbf{Y}$, one can certify that F does not vanish in $\mathbf{X}$. An inclusion function for F is convergent if for any decreasing sequence of boxes converging to a point $\{p\}$, the image of the sequence converges to $\{F(p)\}$.

In this article, we consider C^k-smooth functions, that is functions that are $k \geq 1$ times differentiable and with continuous partial differentials. In addition, we also assume that we are given convergent inclusion functions for such functions and all their differentials up to order k.

1.5.2 Criteria for existence and uniqueness of solutions

Let $P_1, \ldots, P_n$ be n functions C^1 in n unknowns and $S = \{P_1 = 0, \ldots, P_n = 0\}$ the associated system of equations. A box $\mathbf{X} \subset \mathbb{R}^n$ *isolates* a solution of S if there exists a unique $x \in \mathbf{X}$ such that $P_1(x) = \ldots = P_n(x) = 0$.

Several criteria can be found in the interval arithmetic literature that certify existence and uniqueness of a solution of S in a box, see for instance [14, 23, 25, 28]. Most of them are based on the Brouwer fixed point theorem, and use interval Newton operators that contract a box around a solution.

Letting F be the multivariate function with components $P_1, \ldots, P_n$ and $\mathbf{X}$ an interval of $\mathbb{R}^n$, interval Newton operators are of the form $N(\mathbf{X}) = y - \boldsymbol{V}$, where $y \in \mathbf{X}$, and $\boldsymbol{V}$ is a box containing the set $\{v \,|\, \exists x \in \mathbf{X}$ such that $J(x)v = F(y)\}$, where J is the Jacobian of F. Among other interval Newton operators is the Krawczyk operator [17, 23], that takes y as the midpoint of $\mathbf{X}$ and an approximate inverse of the Jacobian of F in y to determine the box $\boldsymbol{V}$. Let us denote by K_S the Krawczyk operator for the system S, and recall that $i(\mathbf{X})$ is the interior of $\mathbf{X}$. Namely,

$$K_S(\mathbf{X}) = y - \mathbf{Y}F(y) + \{I - \mathbf{Y}J(\mathbf{X})\}(\mathbf{X} - y),$$

where $\mathbf{Y}$ is a nonsingular real matrix approximating the inverse of the Jacobian matrix of F at $\mathrm{m}(\mathbf{X})$. The Krawczyk operator satisfies:

- $K_S(\mathbf{X}) \subset i(\mathbf{X}) \Rightarrow \mathbf{X}$ contains a unique solution of S and the sequence $\mathbf{X}^{(0)} = \mathbf{X}$, $\mathbf{X}^{(i+1)} = K_S(\mathbf{X}^{(i)})$ converges asymptotically quadratically to this solution.
- $K_S(\mathbf{X}) \cap \mathbf{X} = \emptyset \Rightarrow$ there is no solution of S in $\mathbf{X}$.

1.5.3 Reliable numerical isolation

Interval evaluation, the Krawczyk operator and bisection of boxes can be used together to design a simple reliable numerical method to isolate all solutions of S in a given initial box $\mathbf{X}_0$. Such methods are described for instance in [14, 23, 25, 28], and called interval branch and bound algorithms or subdivision methods. These algorithms successfully isolate all the solutions of S in a bounded domain, provided they are

Algorithm 1 `IsolateSols`$(S, \mathbf{X}_0, \delta)$

Input: A bounded box $\mathbf{X}_0$ of $\mathbb{R}^n$, a system S of n functions C^1 in n unknowns with only regular solutions in $\mathbf{X}_0$, the Krawczyk operator K_S, and a non-negative real number δ.

Output: Two sets $\mathrm{X}^{\mathrm{sol}}, \mathrm{X}^{\mathrm{ind}}$ of boxes such that:

- boxes of $\mathrm{X}^{\mathrm{sol}} \cup \mathrm{X}^{\mathrm{ind}}$ are pairwise disjoint,
- $x \in \mathbf{X}_0$ is a solution of $S \Rightarrow \exists \mathbf{X} \in \mathrm{X}^{\mathrm{sol}} \cup \mathrm{X}^{\mathrm{ind}}$ s.t. $x \in \mathbf{X}$,
- $\mathbf{X} \in \mathrm{X}^{\mathrm{sol}} \Rightarrow \mathbf{X} \subset i(\mathbf{X}_0)$ and $K_S(\mathbf{X}) \subset i(\mathbf{X})$,
- $\mathbf{X} \in \mathrm{X}^{\mathrm{ind}} \Rightarrow \mathbf{X} \cap \partial\mathbf{X}_0 \neq \emptyset$ and $K_S(\mathbf{X}) \subset i(\mathbf{X})$ and $\mathrm{w}(\mathbf{X}) < \delta$.

regular and in the interior of the domain. Note that in the case of a polynomial system it is possible to extend such branch and bound methods to unbounded initial domains; see [25, Section 5.6] or [28, Section 5.10].

In the following, we consider the procedure `IsolateSols` with the specifications given in Algorithm 1. We briefly explain the output of this algorithm, and refer to [11] for details. The algorithm does not identify solutions that are exactly on the boundary of the input domain $\mathbf{X}_0$, and uses δ-inflation. A consequence is that a solution on $\partial\mathbf{X}_0$ can only be isolated in a box containing a part of the boundary in its interior; the refinement of such a box is stopped due to the threshold on the width δ, and the box is output in $\mathrm{X}^{\mathrm{ind}}$. Similarly, for a regular solution not on, but near the boundary (even outside $\mathbf{X}_0$), that is at distance less than δ from $\partial\mathbf{X}_0$, the algorithm may return a box in $\mathrm{X}^{\mathrm{ind}}$. On the other hand, with the additional assumption that there is no solution on the boundary, setting $\delta = 0$ certifies the isolation of all solutions in the domain $\mathbf{X}_0$, that is, that $\mathrm{X}^{\mathrm{ind}}$ is empty.

2 Enclosing the Space Curve $\mathcal{C}$

In this section, we introduce Algorithm 3, computing a δ-approximation of $\mathcal{C}$. This goal is achieved by first finding at least one point on each connected component of $\mathcal{C}$, then using these points as initial points for a certified path tracker. Let us first introduce the notions of x- or y-critical points and boundary points.

Definition 2.1 *A point $p \in \mathcal{C}$ is x-critical if the x component of the tangent of $\mathcal{C}$ at p vanishes. The x-critical points of $\mathcal{C}$ are the solutions of the system $P = Q = R = 0$, where $R = P_y Q_z - P_z Q_y$. Similarly, $p \in \mathcal{C}$ is y-critical if it is the solution of the system $P = Q = R' = 0$ where $R' = P_x Q_z - P_z Q_x$.*

Definition 2.2 *If C is a box or a domain in $\mathbb{R}^3$, a point of $\mathcal{C} \cap \partial C$ is called a boundary point.*

Isolating boxes for boundary points and x-critical points of $\mathcal{C}$ are computed with the procedure `IsolateSols` defined in Algorithm 1. We introduce the notion of *implicit points* to manipulate a point known as the unique solution of a system S in a box $\boldsymbol{C} = \mathbf{x} \times \mathbf{y} \times \boldsymbol{z}$.

Definition 2.3 *Let P, Q define the curve $\mathcal{C}$, let $E \in \mathbb{R}[x, y, z]$, and let S be the system $P = Q = E = 0$. The ordered pair $(\boldsymbol{C}, S)$ is an implicit point of $\mathcal{C}$ if $K_S(\boldsymbol{C}) \subset i(\boldsymbol{C})$. If p is the unique solution of S in $\boldsymbol{C}$, we say that p is implicitly defined by $(\boldsymbol{C}, S)$.*

In the special case where $E = x - x_0$ with $x_0 \in \mathbb{R}$, we say that $(x_0 \times \mathbf{y} \times \boldsymbol{z}, S)$ is an implicit point of $\mathcal{C}$ if $K_{S_0}(\mathbf{y} \times \boldsymbol{z}) \subset i(\mathbf{y} \times \boldsymbol{z})$, where $S_0 = \{P(x_0, y, z) = Q(x_0, y, z) = 0\}$. Similarly, in the special case where $E = y - y_0$ with $y_0 \in \mathbb{R}$, we say that $(\mathbf{x} \times y_0 \times \boldsymbol{z}, S)$ is an implicit point of $\mathcal{C}$ if $K_{S_0}(\mathbf{x} \times \boldsymbol{z}) \subset i(\mathbf{x} \times \boldsymbol{z})$, where $S_0 = \{P(x, y_0, z) = Q(x, y_0, z) = 0\}$.

In Sec. 2.2, we characterize a connected component of $\mathcal{C} \cap \boldsymbol{C}_0$: it is diffeomorphic either to a circle and contains at least two x-critical points, or to $[0, 1]$ and its extremities are boundary points. We also show how to obtain these points as implicit points. In Sec. 2.3, we specify Algorithm 2, our certified path tracker, and in Sec. 2.4, we specify Algorithm 3 for computing a δ-approximation of $\mathcal{C}$. In Sec. 2.1, we state the assumptions allowing our approach to be correct and terminating.

2.1 Assumptions

Recall that $\mathbf{B}_0 = (\mathbf{x}_0, \mathbf{y}_0)$ and $\boldsymbol{C}_0 = \mathbf{B}_0 \times \mathbb{R}$. We assume the functions P and Q are C^3 functions[2] and that convergent inclusion functions are given for P, Q and all their derivatives up to order 3. We make the following assumptions on P, Q and $\mathcal{C}$:

(A_1) The curve $\mathcal{C}$ is smooth above the box $\mathbf{B}_0$.

(A_2) $\mathcal{C}$ is compact over $\mathbf{B}_0$ and $\boldsymbol{z}_0$ is a bounded interval such that $\mathcal{C} \subset \mathbf{B}_0 \times \boldsymbol{z}_0$.

(A_3) $P(x_*, y, z) = Q(x_*, y, z) = 0$ has finitely many regular solutions when $x_* = l(\mathbf{x}_0)$ or $u(\mathbf{x}_0)$, $y \in \mathbf{y}_0$ and $z \in \mathbb{R}$.

$P(x, y_*, z) = Q(x, y_*, z) = 0$ has finitely many regular solutions when $y_* = l(\mathbf{y}_0)$ or $u(\mathbf{y}_0)$, $x \in \mathbf{x}_0$ and $z \in \mathbb{R}$.

(A_4) $P = Q = R = 0$ has finitely many regular solutions in $\boldsymbol{C}_0$, and no solution in $\partial \boldsymbol{C}_0$.

(A_5) Over a point of $\partial \mathbf{B}_0$, $P = Q = 0$ has only one solution and no solutions above its corners.

2.2 Connected components of $\mathcal{C} \cap \boldsymbol{C}_0$ and initial boxes

The following proposition characterizes the topology of the connected components of $\mathcal{C} \cap \boldsymbol{C}_0$.

Proposition 2.1 *Assuming (A_1), the connected components of $\mathcal{C} \cap \boldsymbol{C}_0$ are smooth one dimensional manifolds possibly with boundary. In addition, assuming (A_2) and (A_3), any connected component $\mathcal{C}^k$ of $\mathcal{C} \cap \boldsymbol{C}_0$ satisfies at least one of the following statements:*

(a) $\mathcal{C}^k$ has exactly two boundary points,

(b) $\mathcal{C}^k$ has at least two x-critical points.

[2] In Algorithm 4, we need to isolate the solutions of $\{P = P_z = P_{zz} = 0\}$, the Jacobian of which involves derivatives of P of order 3.

Proof: The first part of the proposition is straightforward. One dimensional manifolds are diffeomorphic either to $]0,1[$, or to $]0,1]$, or to $[0,1]$, or to a circle. Let $\mathcal{C}^k$ be a connected component of $\mathcal{C}$. From assumption (A_2), it is compact, hence it is diffeomorphic either to $[0,1]$, or to a circle. Suppose first $\mathcal{C}^k$ has an intersection with $\partial \boldsymbol{C}_0$. From (A_3), this intersection is a point, hence $\mathcal{C}^k$ is diffeomorphic to $[0,1]$ and has a second intersection with $\partial \boldsymbol{C}_0$. Suppose now that $\mathcal{C}^k$ does not intersect $\partial \boldsymbol{C}_0$. Hence it is diffeomorphic to a circle and, since it is compact, it has minimum and maximum x-coordinates. Assertion (b) follows. □

As a direct consequence of Prop. 2.1, the following corollary gives a constructive characterization of a set containing at least one point on each connected component of $\mathcal{C} \cap \boldsymbol{C}_0$.

Corollary 2.1 *Consider the following systems of equations:*

(S_1) $P(l(\mathbf{x}_0), y, z) = Q(l(\mathbf{x}_0), y, z) = 0$, *for* $y \in \mathbf{y}_0$ *and* $z \in \mathbb{R}$.

(S_2) $P(u(\mathbf{x}_0), y, z) = Q(u(\mathbf{x}_0), y, z) = 0$, *for* $y \in \mathbf{y}_0$ *and* $z \in \mathbb{R}$.

(S_3) $P(x, l(\mathbf{y}_0), z) = Q(x, l(\mathbf{y}_0), z) = 0$, *for* $x \in \mathbf{x}_0$ *and* $z \in \mathbb{R}$.

(S_4) $P(x, u(\mathbf{y}_0), z) = Q(x, u(\mathbf{y}_0), z) = 0$, *for* $x \in \mathbf{x}_0$ *and* $z \in \mathbb{R}$.

(S_5) $P(x, y, z) = Q(x, y, z) = R(x, y, z) = 0$, *for* $(x, y, z) \in \boldsymbol{C}_0$.

Assuming $(A_3), (A_4)$, *the set of solutions of* $S_1, \ldots, S_4$ *is finite and is the set of boundary points of* $\mathcal{C} \cap \boldsymbol{C}_0$, *the set of solutions of* S_5 *is finite and is the set of* x*-critical points of* $\mathcal{C} \cap \boldsymbol{C}_0$. *Then assuming* (A_2), *the set of solutions of* $S_1, \ldots, S_5$ *is a finite set of points in* $\mathbf{B}_0 \times \boldsymbol{z}_0$ *containing at least one point on each connected component of* $\mathcal{C}$.

The solutions of S_5 are obtained by calling `IsolateSols(`S_5`,`$\mathbf{B}_0 \times \boldsymbol{z}_0$`,0)`; from assumption (A_4) this process terminates and the outputs $\mathrm{C}^{\mathrm{sol}}, \mathrm{C}^{\mathrm{ind}}$ are such that $\mathrm{C}^{\mathrm{ind}} = \emptyset$ (since S_5 has no solution on $\partial \boldsymbol{C}_0$) and $\mathrm{C}^{\mathrm{sol}}$ contains boxes isolating the solutions of S_5. For each solution $\boldsymbol{C} \in \mathrm{C}^{\mathrm{sol}}$, the implicit point $(\boldsymbol{C}, S_5)$ defines an x-critical point of $\mathcal{C}$.

The solutions of S_1 are obtained by calling `IsolateSols(`S_1`,`$\mathbf{y}_0 \times \boldsymbol{z}_0$`,0)`. From assumptions (A_3) and (A_5), S_1 has a finite number of solutions on $\mathbf{y}_0 \times \mathbb{R}$ that are regular and has no solutions on $\partial \mathbf{y}_0 \times \mathbb{R}$. Hence the procedure terminates, and the two obtained sets $\mathrm{B}^{\mathrm{sol}}, \mathrm{B}^{\mathrm{ind}}$ are such that $\mathrm{B}^{\mathrm{ind}} = \emptyset$ and $\mathrm{B}^{\mathrm{sol}}$ contains two dimensional boxes isolating the solutions of S_1. A solution $\mathbf{B} = (\mathbf{y}, \boldsymbol{z})$ in $\mathrm{B}^{\mathrm{sol}}$ defines an implicit point $(\boldsymbol{C}, \{P, Q, E\})$ where $\boldsymbol{C} = l(\mathbf{x}_0) \times \mathbf{y} \times \boldsymbol{z}$ and E is the polynomial $x - l(\mathbf{x}_0)$, as described in Definition 2.3. When needed, $\boldsymbol{C}$ is contracted with $l(\mathbf{x}_0) \times K_{S_1}(\pi_{(y,z)}(\boldsymbol{C}))$. The solutions of S_2, S_3 and S_4 are obtained similarly.

2.3 Certified numerical path-tracking

Our path-tracking procedure `Track` is specified in Algorithm 2, while its description is given in Appendix A. The tracker is used in Algorithm 3 to compute the connected components of $\mathcal{C} \cap \boldsymbol{C}_0$, and later in Section 5.1.2 to deduce the topology of $\mathcal{B}$.

2.4 Computing a δ-approximation of $\mathcal{C}$

In the following, we assume that P, Q satisfy assumptions (A_1) and (A_2) of Section 2.1: $\mathcal{C}$ is smooth and compact in $\boldsymbol{C}_0$. Based on the procedure `Track`, Algorithm 3 computes a δ-approximation of $\mathcal{C} \cap \boldsymbol{C}_0$.

Algorithm 2 `Track`$(< P, Q >, C_0, (\boldsymbol{C}^0, S^0), \{(\boldsymbol{C}^j, S^j)\}_j, \delta)$

Input: A system $P = Q = 0$ defining a smooth compact curve $\mathcal{C}$ in the domain C_0, an implicit point $(\boldsymbol{C}^0, S^0)$ of $\mathcal{C}$, a finite set $\{(\boldsymbol{C}^j, S^j)\}_j$ of implicit points of $\mathcal{C}$ containing (but not restricted to) the boundary points of $\mathcal{C}$ in C_0 and $(\boldsymbol{C}^0, S^0)$, $\delta > 0$.

Output: A δ-approximation of the connected component of $\mathcal{C} \cap C_0$ containing $(\boldsymbol{C}^0, S^0)$ and the set C^{on} of implicit points of $\{(\boldsymbol{C}^j, S^j)\}_j$ that are on the same connected component as $(\boldsymbol{C}^0, S^0)$.

Algorithm 3 Compute a δ-approximation of $\mathcal{C}$ in $\boldsymbol{C}_0$

Input: A system $P = Q = 0$ and an initial domain $\boldsymbol{C}_0 = \mathbf{B}_0 \times \mathbb{R}$ defining a smooth curve $\mathcal{C} \cap \boldsymbol{C}_0$, the set C^b of implicit points defining the boundary points of $\mathcal{C} \cap \boldsymbol{C}_0$, and the set C^x of implicit points defining the x-critical points of $\mathcal{C} \cap \boldsymbol{C}_0$, $\delta > 0$.

Output: A sequence of boxes $(\boldsymbol{C}_i)_{i=1}^m$ that is a δ-approximation of $\mathcal{C} \cap \boldsymbol{C}_0$.

1: Let $\mathcal{L}^b$ (resp. $\mathcal{L}^x$) be a list containing elements of C^b (resp. C^x)
2: $k \leftarrow 0$, $m_k \leftarrow 0$
3: **while** $\mathcal{L}^b \neq \emptyset$ **do**
4: $\quad k = k + 1$, $(\boldsymbol{C}, S) \leftarrow$ `pop_front`$(\mathcal{L}^b)$
5: $\quad ((\boldsymbol{C}_i)_{i=m_{k-1}+1}^{m_k}, \mathrm{C}^{\mathrm{on}}) \leftarrow$ `Track`$(< P, Q >, \boldsymbol{C}_0, (\boldsymbol{C}, S), \mathrm{C}^b \cup \mathrm{C}^x, \delta)$
6: $\quad$ remove $\mathrm{C}^{\mathrm{on}} \cap \mathrm{C}^b$ from $\mathcal{L}^b$ and $\mathrm{C}^{\mathrm{on}} \cap \mathrm{C}^x$ from $\mathcal{L}^x$
7: **while** $\mathcal{L}^x \neq \emptyset$ **do**
8: $\quad k = k + 1$, $(\boldsymbol{C}, S) \leftarrow$ `pop_front`$(\mathcal{L}^x)$
9: $\quad ((\boldsymbol{C}_i)_{i=m_{k-1}+1}^{m_k}, \mathrm{C}^{\mathrm{on}}) \leftarrow$ `Track`$(< P, Q >, \boldsymbol{C}_0, (\boldsymbol{C}, S), \mathrm{C}^b \cup \mathrm{C}^x, \delta)$
10: $\quad$ remove $\mathrm{C}^{\mathrm{on}} \cap \mathrm{C}^x$ from $\mathcal{L}^x$
11: Let $m = m_k$
12: **return** $(\boldsymbol{C}_i)_{i=1}^m$

Algorithm 3 first computes δ-approximations for every connected component of $\mathcal{C}$ that is diffeomorphic to $[0, 1]$. This is addressed by calling the procedure `Track` with boundary points as initial points. The boundary points are implicitly defined by elements of C^b. Thus letting $\mathcal{C}^k$ be a connected component containing a boundary point defined by $(\boldsymbol{C}, S) \in \mathrm{C}^b$, the call to the procedure `Track` in Step 5 terminates, returns a δ-approximation of $\mathcal{C}^k$ and identifies the two extremities and the x-critical points of $\mathcal{C}^k$.

When entering the **while** loop in Step 3 for the first time, the list $\mathcal{L}^b$ contains the two extremities of each connected component of $\mathcal{C} \cap \boldsymbol{C}_0$ that is diffeomorphic to $[0, 1]$. Each time the instruction in Step 6 is performed, the two extremities of the connected component that has been tracked are removed from $\mathcal{L}^b$, and the size of $\mathcal{L}^b$ decreases. When $\mathcal{L}^b$ is empty, each connected component that is diffeomorphic to $[0, 1]$ has been approximated, and the implicit points in $\mathcal{L}^x$ define the x-critical points that belong to the connected components that are diffeomorphic to circles.

The connected components of $\mathcal{C}$ that are diffeomorphic to a circle are approximated in the **while** loop beginning at Step 7 of Algorithm 3. Recall that each connected component of $\mathcal{C} \cap \boldsymbol{C}_0$ that is diffeomorphic to a circle contains at least two x-critical points. Each time the loop is performed, the size of $\mathcal{L}^x$ decreases, and when $\mathcal{L}^x$ is empty, each connected component of $\mathcal{C}$ has been approximated.

3 Isolating Singularities of $\mathcal{B}$

When $\mathcal{C}$ is defined by two analytic maps P, Q, [12] describes, under genericity conditions on P, Q, the type of singularities arising in the projection $\mathcal{B}$: they are only nodes (two branches of $\mathcal{C}$ induce a self intersection in $\mathcal{B}$), or cusps ($\mathcal{C}$ has a vertical tangent). [12] also introduces a system called *ball system* whose solutions are in one-to-one correspondence with the singularities of $\mathcal{B}$, and shows that the ball system only has regular solutions if and only if singularities of $\mathcal{B}$ are either nodes or ordinary cusps.

We first restate the assumptions and the main results of [12]. We then show how an enclosure of $\mathcal{C}$ helps to restrict the domain where the ball system is solved while ensuring that all cusps and nodes are obtained. Then, we present Algorithm 4, that decides, for a given solution of the ball system, if the corresponding singularity is an ordinary cusp or a node when $\mathcal{B}$ is an apparent contour. Within this section, Σ denotes the set of singular points of $\mathcal{B} \cap \mathbf{B}_0$.

3.1 Assumptions

Consider the following assumptions:

- (A_6) For any (α, β) in $\mathbf{B}_0$, the system $P(\alpha, \beta, z) = Q(\alpha, \beta, z) = 0$ has at most 2 real roots, counting multiplicities.
- (A_7) There are finitely many points (α, β) in $\mathbf{B}_0$ such that $P(\alpha, \beta, z) = Q(\alpha, \beta, z) = 0$ has 2 real roots, counting multiplicities.
- (A_8) The singularities of the curve $\mathcal{B}$ in $\mathbf{B}_0$ are either nodes or ordinary cusps.

Notice that the Thom Transversality Theorem implies that $(A_1) \ldots, (A_8)$ hold for generic smooth maps P, Q defining $\mathcal{C}$ (see [8, Theorem 3.9.7 and §4.7]).

3.2 Ball system

Following a geometric modelling, [12] defines a 4 dimensional system whose solutions map to the singularities of $\mathcal{B}$. In this modelling, two solutions (x, y, z_1) and (x, y, z_2) of $P = Q = 0$ (or $P = P_z = 0$) are mapped to the point (x, y, c, r_2) with $c = (z_1 + z_2)/2$ and $r_2 = (z_1 - z_2)^2$. Figure 4 illustrates this mapping for singularities of the apparent contour of a torus. We review the main results of [12].

Lemma 3.1 ([12, Lemma 4]) *Let P, Q be two analytic functions in x, y, z satisfying the Assumptions $(A_1), (A_2), (A_6)$ and (A_7), and let $\mathcal{S}$ be the set of solutions of the so-called* ball system*:*

$$\begin{cases} \dfrac{1}{2}(P(x,y,c+\sqrt{r_2}) + P(x,y,c-\sqrt{r_2})) = 0 \\ \dfrac{1}{2\sqrt{r_2}}(P(x,y,c+\sqrt{r_2}) - P(x,y,c-\sqrt{r_2})) = 0 \\ \dfrac{1}{2}(Q(x,y,c+\sqrt{r_2}) + Q(x,y,c-\sqrt{r_2})) = 0 \\ \dfrac{1}{2\sqrt{r_2}}(Q(x,y,c+\sqrt{r_2}) - Q(x,y,c-\sqrt{r_2})) = 0 \end{cases} \tag{1}$$

in $\mathbf{B}_0 \times \mathbb{R} \times \mathbb{R}^+$. Then $\pi'_{(x,y)}(\mathcal{S}) = \Sigma$, where $\pi'_{(x,y)}$ is the projection from $\mathbb{R}^4$ to the (x, y)-plane.

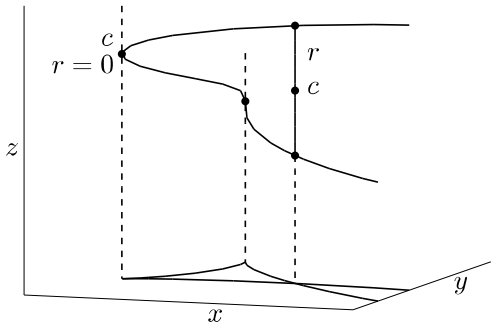


Figure 4: Singularities of the apparent contour $\mathcal{B}$ of the torus. For node and cusp singularities of $\mathcal{B}$, their preimages on the space curve $\mathcal{C}$ as well as corresponding centers c and radii $r_2 = r^2$ for the ball system are represented.

Lemma 3.2 ([12, Lemma 5]) *Under the Assumptions $(A_1), (A_2), (A_6)$ and (A_7), all the solutions of the ball system in $\mathbf{B}_0 \times \mathbb{R} \times \mathbb{R}^+$ are regular if and only if (A_8) is satisfied.*

Remark 3.1 *If P, Q are two C^3 functions then Lemmas 3.1 and 3.2 still hold.*

In addition, the solutions of the ball system with $r_2 = 0$ map to cusps in 2d and the curve in 3d has a vertical tangent (collinear to the z-axis); whereas the solutions with $r_2 \neq 0$ map to nodes.

3.3 Solution domain

In [12], the ball system is solved within the box $\mathbf{B}_0 \times \mathbb{R} \times \mathbb{R}^+$ with a subdivision solver. Using the δ-approximation $(\boldsymbol{C}_i)_{1 \leq i \leq m}$ of $\mathcal{C}$ computed by Algorithm 3, we propose to reduce significantly the search domain for the singularities. Indeed, given a singular point σ of $\mathcal{B}$, there exists $1 \leq i \leq m$ such that $\sigma \in \mathbf{B}_i$, where $\mathbf{B}_i = \pi_{(x,y)}(\boldsymbol{C}_i)$. Hence it is possible to isolate all singularities by solving the ball system within $\mathbf{B}_i \times \mathbb{R} \times \mathbb{R}^+$, for $1 \leq i \leq m$. In addition, Proposition 3.1 shows how to bound the solution domain in the c and r_2 components.

Proposition 3.1 *Let $(\boldsymbol{C}_i)_{1 \leq i \leq m}$ be a δ-approximation of $\mathcal{C}$. For $1 \leq i \leq m$, let $\boldsymbol{C}_i = (\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{z}_i)$, $\mathbf{B}_i = \pi_{(x,y)}(\boldsymbol{C}_i) = (\mathbf{x}_i, \mathbf{y}_i)$, and for $1 \leq i < j \leq m$, let $\mathbf{B}_{ij} = (\mathbf{x}_{ij}, \mathbf{y}_{ij}) = \mathbf{B}_i \cap \mathbf{B}_j$ and consider the sets:*

- $\boldsymbol{D}_i = (\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{z}_i, [0, (\frac{\mathrm{w}(\boldsymbol{z}_i)}{2})^2])$,
- $\boldsymbol{D}_{ij} = (\mathbf{x}_{ij}, \mathbf{y}_{ij}, \frac{(\boldsymbol{z}_i + \boldsymbol{z}_j)}{2}, (\frac{(\boldsymbol{z}_i - \boldsymbol{z}_j)}{2})^2)$

Then, under the Assumptions $(A_1), (A_2), (A_6)$ and (A_7), all solutions of the ball system lie in $(\bigcup_{1 \leq i \leq m} \boldsymbol{D}_i) \cup (\bigcup_{1 \leq i < j \leq m} \boldsymbol{D}_{ij})$.

Proof: Let $p = (x_p, y_p, c_p, r_p) \in \mathbf{B}_0 \times \mathbb{R} \times \mathbb{R}^+$ be a solution of the ball system and $\sigma = \pi'_{(x,y)}(p)$ the corresponding singularity in Σ. From Assumption (A_8), σ is either an ordinary cusp or a node.

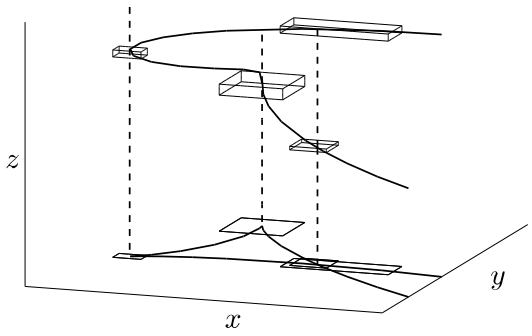


Figure 5: Some boxes and their projections containing singularities of $\mathcal{B}$. Cusps singularities are in boxes $\mathbf{B}_i$, nodes in boxes $\mathbf{B}_i \cap \mathbf{B}_j$.

Suppose first it is an ordinary cusp. Then $r_p = 0$, and σ is the projection of a single point $p' = (x_p, y_p, c_p)$ of $\mathcal{C}$. Hence there exists $1 \leq i \leq m$ such that $p' \in \boldsymbol{C}_i$. Thus, we have $c_p \in \boldsymbol{z}_i$ and $p \in \boldsymbol{D}_i$ (see Figure 5).

Suppose now σ is a node. Then $r_p > 0$, and σ is the projection of two points $p_- = (x_p, y_p, c_p - \sqrt{r_p})$ and $p_+ = (x_p, y_p, c_p + \sqrt{r_p})$ of $\mathcal{C}$. Hence there exist $1 \leq i \leq m$ and $1 \leq j \leq m$ such that $p_- \in \boldsymbol{C}_i$ and $p_+ \in \boldsymbol{C}_j$. If $i = j$, we have $c_p \in \boldsymbol{z}_i$ and $r_p \in [0, (\frac{\mathrm{w}(\boldsymbol{z}_i)}{2})^2]$, and finally $p \in \boldsymbol{D}_i$. If $i \neq j$ (this case is illustrated in Figure 5), c_p lies in $\frac{\boldsymbol{z}_i + \boldsymbol{z}_j}{2}$ that is the center of the two intervals $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, and r_p lies in $(\frac{\boldsymbol{z}_i - \boldsymbol{z}_j}{2})^2$, that is the square of the corresponding radius. □

3.4 Singularities of an apparent contour

For a generic 3-dimensional curve, the singularities of its projection $\mathcal{B}$ are only nodes. When the curve $\mathcal{B}$ is the apparent contour of a surface, its singularities generically also include cusps. We introduce Algorithm 4 to distinguish these two types of singularities.

Let $\boldsymbol{D} = (\mathbf{x}, \mathbf{y}, \boldsymbol{c}, \boldsymbol{r})$ be a box isolating a solution $p = (x, y, c, r) \in \boldsymbol{D}$ of the ball system and let $\sigma = \pi'_{(x,y)}(p)$ be the associated singularity of $\mathcal{B}$. If $0 \notin \boldsymbol{r}$ then $r \neq 0$ and σ is a node. Otherwise, σ can be either a cusp or a node. Recall that σ is an ordinary cusp of $\mathcal{B}$ only if it is the projection of a point of $\mathcal{C}_{P \cap P_z}$ that has a vertical tangent (collinear to the z-axis). In other words, if $\sigma = (\alpha, \beta)$ is a cusp, there exists a unique $\gamma \in \mathbb{R}$ such that $\sigma = \pi_{(x,y)}(\alpha, \beta, \gamma)$ and $P(\alpha, \beta, \gamma) = P_z(\alpha, \beta, \gamma) = P_{zz}(\alpha, \beta, \gamma) = 0$. According to assumption (A_6), γ is a triple root of $P(\alpha, \beta, z)$, that is, $P_{zzz}(\alpha, \beta, \gamma) \neq 0$. The regularity of the curve $\mathcal{C}_{P \cap P_z}$ thus implies that (α, β, γ) is a regular solution of the system $P = P_z = P_{zz} = 0$ (see [13, Lemma 10]). Noting that $K_{(P, P_z, P_{zz})}$ is the Krawczyk operator for the latter system, the test on Line 2 of Algorithm 4 is eventually true for a small enough box whose projection contains a cusp.

While refining the box $\boldsymbol{D}$, Algorithm 4 thus terminates for a box

$$\boldsymbol{D}_* = (\mathbf{x}_*, \mathbf{y}_*, \boldsymbol{c}_*, \boldsymbol{r}_*) \subset \boldsymbol{D},$$

containing p such that either $0 \notin \boldsymbol{r}_*$ and $\pi'_{(x,y)}(p)$ is a node, or

$$K_{(P, P_z, P_{zz})}((\mathbf{x}_*, \mathbf{y}_*, \boldsymbol{c}_*)) \subset i((\mathbf{x}_*, \mathbf{y}_*, \boldsymbol{c}_*)),$$

Algorithm 4 Singularity types for an apparent contour

Input: P in $\mathbb{Q}[x, y, z]$, a box $\boldsymbol{D} = (\mathbf{x}, \mathbf{y}, \boldsymbol{c}, \boldsymbol{r})$ s.t $K_b(\boldsymbol{D}) \subset i(\boldsymbol{D})$, where K_b is the Krawczyk operator for the ball system.
Output: The type of the singularity contained in $\pi'_{(x,y)}(\boldsymbol{D}) = (\mathbf{x}, \mathbf{y})$.
1: **while** $0 \in \boldsymbol{r}$ **do**
2: **if** $K_{(P,P_z,P_{zz})}((\mathbf{x}, \mathbf{y}, \boldsymbol{c})) \subset i((\mathbf{x}, \mathbf{y}, \boldsymbol{c}))$ **then return** cusp
3: $\boldsymbol{D} = K_b(\boldsymbol{D})$
4: **return** node

and $\pi'_{(x,y)}(p)$ is an ordinary cusp.

4 Topology at Special Points of $\mathcal{B}$

In this section, we compute *witness* boxes for the singularities and critical points of $\mathcal{B}$, that is, isolating boxes such that the topology of the curve inside these boxes can be deduced from the intersections of the curve with their boundaries. Special care is taken to refine the boxes such that the intersections of the curve with their boundaries do not eventually occur at the corners. The case of a node is detailed; the other cases are only sketched, since they are similar.

4.1 Boundary points

We define the boundary points of $\mathcal{C}$ as the set $\mathcal{C} \cap (\partial \mathbf{B}_0 \times \mathbb{R})$, and the boundary points of $\mathcal{B}$ as the set $\mathcal{B} \cap \partial \mathbf{B}_0$. With Assumptions $(A_3) - (A_5)$, the boundary points of $\mathcal{C}$ and $\mathcal{B}$ are in one-to-one correspondence, are neither x nor y-critical points of $\mathcal{C}$ nor of $\mathcal{B}$, and are not the corners of $\mathbf{B}_0$. The isolating boxes of boundary points of $\mathcal{C}$ can thus be refined until their projections onto $\partial \mathbf{B}_0$ are disjoint.

4.2 Preprocessing of nodes, cusps and x-extreme points

We add the following assumptions.

(A_9) The two points of $\mathcal{C}$ above a node of $\mathcal{B}$ are neither x-critical nor y-critical points of $\mathcal{C}$.

(A_{10}) $P = Q = R' = 0$ has finitely many regular solutions in $\boldsymbol{C}_0$ and none in $\partial \boldsymbol{C}_0$, and they are the y-critical points of the curve $\mathcal{C}$.

Note that a cusp of $\mathcal{B}$ corresponds to a point of $\mathcal{C}$ with a vertical tangent (collinear to the z-axis), that is both an x and y-critical point of $\mathcal{C}$. With all our assumptions, the x or y-critical points of $\mathcal{C}$ are in $i(\boldsymbol{C}_0)$, and

- x and y-critical points of $\mathcal{C}$ are in one-to-one correspondence with cusps of $\mathcal{B}$.
- x-critical points of $\mathcal{C}$ that are not y-critical are in one-to-one correspondence with smooth x-critical points of $\mathcal{B}$.
- y-critical points of $\mathcal{C}$ that are not x-critical are in one-to-one correspondence with smooth y-critical points of $\mathcal{B}$.

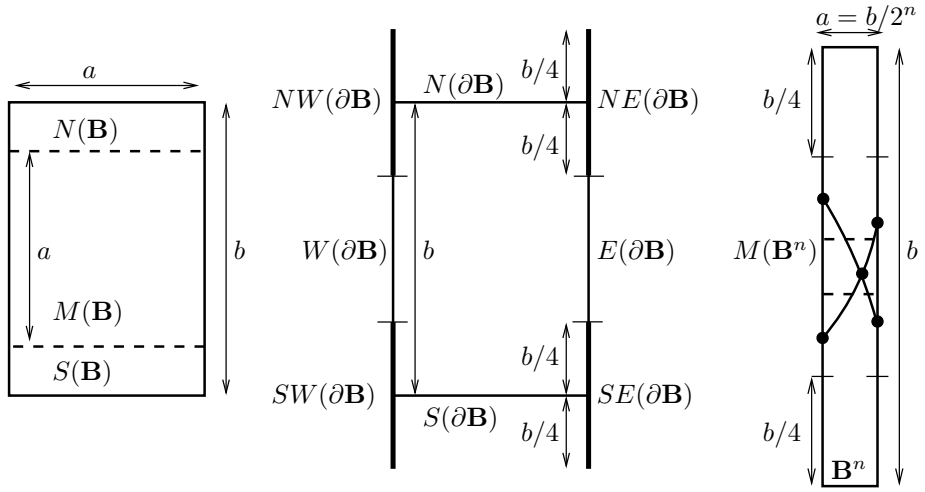


Figure 6: Left (resp. middle): the decomposition of $\mathbf{B}$ (resp. $\partial\mathbf{B}$). Right: a witness box $\mathbf{B}^n$ for a node.

The first step is to compute implicit points for y-critical points of $\mathcal{C}$, by solving the system $P = Q = R' = 0$ with `IsolateSols` on $\mathbf{B}_0 \times \mathbf{z}_0$. Assumption (A_{10}) ensures the termination of the process with input $\epsilon = 0$. Then, the 4-dimensional boxes isolating nodes and cusps (the solutions of the ball system) and the 3-dimensional boxes isolating x and y-critical points (the solutions of $P = Q = R = 0$ and $P = Q = R' = 0$) are refined until their projections in the (xy)-plane satisfy

- a box of a cusp overlaps exactly one x-critical and one y-critical point,
- a box of a node overlaps no x nor y-critical point,
- a box of an x-critical point that is not overlapping a cusp, does not overlap any y-critical point.

An x-critical point of $\mathcal{B}$ that is smooth, i.e. is not also y-critical, is called *x-extreme*. A point of $\mathcal{B}$ a points that is an x-extreme, a node or a cusp is called a *special* point. Similarly, as in [13], we define a *witness* box for a special point of the curve $\mathcal{B}$.

Definition 4.1 *A* witness *box for a special point (x-extreme, node or cusp) of the curve $\mathcal{B}$ is a box containing this point, such that the topology of the curve inside the box is the one of the graph connecting its center to the crossings of the curve on its boundary.*

This definition implies that a witness box has 4 crossings of the curve on its boundary for a node and 2 crossings for a cusp or an x-extreme point. The right part of Figure 6 shows a witness box for a node.

4.3 Topology at a node singularity

Algorithm 6 computes a witness box for a node. The idea is that if the curve $\mathcal{C}$ does not contain an x or y-critical point above an isolating box $\mathbf{B}$ of the node, $\mathbf{B}$ is a

Algorithm 5 `IsWitnessNodeBox(B)`

Input: A box $\mathbf{B}$ containing a unique node of $\mathcal{B}$, such that $\mathbf{B} \times \mathbb{R}$ does not contain any x or y-critical point of $\mathcal{C}$.

Output: **False** when the box is not witness. **True** when $\mathcal{B}$ intersects exactly twice $E(\partial\mathbf{B})$, twice $W(\partial\mathbf{B})$ and does not intersect $NW(\partial\mathbf{B}) \cup N(\partial\mathbf{B}) \cup NE(\partial\mathbf{B}) \cup SW(\partial\mathbf{B}) \cup S(\partial\mathbf{B}) \cup SE(\partial\mathbf{B})$.

1: Let $\mathbf{B} = (\mathbf{x}_B, \mathbf{y}_B)$
2: $(\mathrm{X}_1^{\mathrm{sol}}, \mathrm{X}_1^{\mathrm{ind}})$ = `IsolateSols(`$P(x = l(\mathbf{x}_B), y, z) = Q(x = l(\mathbf{x}_B), y, z) = 0, \mathbf{y}_B \times \boldsymbol{z}_0, \mathrm{w}(\mathbf{y}_B)/4$`)`
3: $(\mathrm{X}_2^{\mathrm{sol}}, \mathrm{X}_2^{\mathrm{ind}})$ = `IsolateSols(`$P(x = u(\mathbf{x}_B), y, z) = Q(x = u(\mathbf{x}_B), y, z) = 0, \mathbf{y}_B \times \boldsymbol{z}_0, \mathrm{w}(\mathbf{y}_B)/4$`)`
4: **if** $|\mathrm{X}_1^{\mathrm{ind}}| > 0$ **or** $|\mathrm{X}_2^{\mathrm{ind}}| > 0$ **then**
5: **return False**
6: $(\mathrm{X}_3^{\mathrm{sol}}, \mathrm{X}_3^{\mathrm{ind}})$ = `IsolateSols(`$P(x, y = l(\mathbf{y}_B), z) = Q(x, y = l(\mathbf{y}_B), z) = 0, \mathbf{x}_B \times \boldsymbol{z}_0, 0$`)`
7: $(\mathrm{X}_4^{\mathrm{sol}}, \mathrm{X}_4^{\mathrm{ind}})$ = `IsolateSols(`$P(x, y = u(\mathbf{y}_B), z) = Q(x, y = u(\mathbf{y}_B), z) = 0, \mathbf{x}_B \times \boldsymbol{z}_0, 0$`)`
8: **if not** $|\mathrm{X}_1^{\mathrm{sol}}| + |\mathrm{X}_2^{\mathrm{sol}}| + |\mathrm{X}_3^{\mathrm{sol}}| + |\mathrm{X}_4^{\mathrm{sol}}| = 4$ **then**
9: **return False**
10: **return True**

witness box when the curve $\mathcal{B}$ crosses its boundary 4 times, or equivalently the curve $\mathcal{C}$ crosses $\partial\mathbf{B} \times \mathbb{R}$ 4 times. To avoid the problem of crossings on the corners of $\mathbf{B}$, the refinement is performed such that its x-coordinate is exponentially smaller than its y-coordinate, so that $\mathcal{B}$ will eventually cross only the left and right sides of $\mathbf{B}$ and far from the corners (see the right part of Figure 6).

Let $\mathbf{B} = (\mathbf{x}_B, \mathbf{y}_B)$ be a 2-dimensional box in the (x, y)-plane and let $a = \mathrm{w}(\mathbf{x}_B)$ and $b = \mathrm{w}(\mathbf{y}_B)$. Assuming $a < b$, we decompose $\mathbf{B}$ into three boxes and $\partial\mathbf{B}$ into eight closed segments as illustrated in the left and middle parts of Figure 6, and we name the boxes and segments with respect to their cardinal directions around the center of $\mathbf{B}$. In particular, $M(\mathbf{B})$ is $(\mathbf{x}_B, \mathrm{m}(\mathbf{y}_B) + [-a/2, a/2])$, and $E(\partial\mathbf{B})$ and $W(\partial\mathbf{B})$ are the respective segments $(u(\mathbf{x}_B), \mathrm{m}(\mathbf{y}_B) + [-b/4, b/4])$ and $(l(\mathbf{x}_B), \mathrm{m}(\mathbf{y}_B) + [-b/4, b/4])$.

Proposition 4.1 *Let $\mathbf{B}$ be a box containing a unique node of $\mathcal{B}$ such that $\mathbf{B} \times \mathbb{R}$ does not contain any x or y-critical point of $\mathcal{C}$. Then*

(i) Algorithm 5 terminates,

*(ii) if Algorithm 5 returns **True** then $\mathbf{B}$ is a witness box for the node,*

*(iii) if $\mathcal{B}$ intersects exactly twice $E(\partial\mathbf{B})$, twice $W(\partial\mathbf{B})$ and does not intersect $NW(\partial\mathbf{B}) \cup N(\partial\mathbf{B}) \cup NE(\partial\mathbf{B}) \cup SW(\partial\mathbf{B}) \cup S(\partial\mathbf{B}) \cup SE(\partial\mathbf{B})$, then Algorithm 5 returns **True**.*

Proof: Since there is no x-critical point of $\mathcal{C}$ in $\mathbf{B} \times \mathbb{R}$, the system $P(x = l(\mathbf{x}_B), y, z) = Q(x = l(\mathbf{x}_B), y, z) = 0$ (resp. $P(x = u(\mathbf{x}_B), y, z) = Q(x = u(\mathbf{x}_B), y, z) = 0$) has only regular solutions on $\mathbf{y}_B \times \mathbb{R}$. In addition, since $\mathrm{w}(\mathbf{y}_B)/4 > 0$, the procedure `IsolateSols` called in Step 2 (resp. Step 3) of Algorithm 5 terminates even if solutions lie on $\partial\mathbf{y}_B \times \mathbb{R}$. Suppose now the sets of undetermined boxes $\mathrm{X}_1^{\mathrm{ind}}, \mathrm{X}_2^{\mathrm{ind}}$ are empty. Then $P = Q = 0$ has no solution above the corners of $\mathbf{B}$, and in particular the systems $P(x, y = l(\mathbf{y}_B), z) = Q(x, y = l(\mathbf{y}_B), z) = 0$ and $P(x, y = u(\mathbf{y}_B), z) = Q(x, y =$

Algorithm 6 Witness box for a node

Input: A box $\boldsymbol{D}$ containing a unique solution of the ball system that projects into a node σ and such that $\pi'_{(x,y)}(\boldsymbol{D}) \times \mathbb{R}$ does not contain any x or y-critical point of $\mathcal{C}$.

Output: A witness box for σ.

1: Let $\boldsymbol{D}' = \boldsymbol{D}$ and $\mathbf{B} = \pi'_{(x,y)}(\boldsymbol{D})$
2: Let $n = 0$, $a_n = \mathrm{w}(\mathbf{B})$, $b_n = \mathrm{w}(\mathbf{B})$ and $\mathbf{B}^n = \mathbf{B}$
3: **repeat**
4: $\quad n = n + 1$, $a_n = a_{n-1}/4$ and $b_n = b_{n-1}/2$
5: $\quad$ **repeat**
6: $\quad\quad \boldsymbol{D}' = K_b(\boldsymbol{D}')$, $\mathbf{B}' = \pi'_{(x,y)}(\boldsymbol{D}')$
7: $\quad\quad \mathbf{B}^n = \mathrm{m}(\mathbf{B}') + ([-a_n/2, a_n/2], [-b_n/2, b_n/2])$
8: $\quad$ **until** $\mathbf{B}' \subseteq M(\mathbf{B}^n)$
9: **until** $\mathbf{B}^n \subseteq \mathbf{B}$ **and** `IsWitnessNodeBox(`$\mathbf{B}^n$`)`
10: **return** $\mathbf{B}^n$

$u(\mathbf{y}_B), z) = 0$ have no solution on $\partial\mathbf{x}_B \times \mathbb{R}$. Hence the procedures `IsolateSols` called in Step 6 and 7 of Algorithm 5 terminate if and only if the two latter systems have only regular solutions on $\mathbf{x}_B \times \mathbb{R}$, and this is the case since $\mathcal{C}$ has no y-critical point over $\mathbf{B}$; (i) follows.

Suppose Algorithm 5 returns **True**. This implies that $\mathcal{B}$ crosses the boundary of $\mathbf{B}$ in exactly four distinct points. Since $\mathcal{C}$ has neither x nor y-critical point above $\mathbf{B}$, the two branches of $\mathcal{C}$, those projections pass through the node, are x and y-monotone above $\mathbf{B}$. The projections of these two branches thus cross the boundary of $\mathbf{B}$ in exactly four distinct points. In addition, there cannot be any other branch of $\mathcal{C}$, since it would either generate more crossings on the boundary or the existence of critical points in the box; thus (ii) follows.

Proof of (iii). If there exists a box $X \in \mathrm{X}_1^{\mathrm{ind}}$, it should include one of the points in $l(\mathbf{x}_B) \times \partial\mathbf{y}_B$ and its width should be less than $\mathrm{w}(\mathbf{y}_B)/4$. Since the curve $\mathcal{B}$ does not intersect $NW(\partial\mathbf{B}) \cup SW(\partial\mathbf{B})$ such a box cannot contain any solution; thus $\mathrm{X}_1^{\mathrm{ind}}$ is empty. Similarly, $\mathrm{X}_2^{\mathrm{ind}}$ is empty and the 2 solutions on $E(\partial\mathbf{B})$ (resp. $W(\partial\mathbf{B})$) are reported in $\mathrm{X}_1^{\mathrm{sol}}$ (resp. $\mathrm{X}_2^{\mathrm{sol}}$). In addition, the curve $\mathcal{B}$ intersects neither $S(\partial\mathbf{B})$ nor $N(\partial\mathbf{B})$, and $\mathrm{X}_3^{\mathrm{sol}}$ and $\mathrm{X}_4^{\mathrm{sol}}$ are empty. Hence the number of reported solutions is 4, and the algorithm returns **True**. □

Note that Algorithm 5 may return a false negative: a box that is a witness box may not be classified as such. This can happen when the curve crosses the box near its corners. The idea of Algorithm 6 is to refine the box of a node to avoid such a case, that is, such that property (iii) of Proposition 4.1 eventually holds. The sequence $\mathbf{B}^n$ of boxes constructed in Algorithm 6 is illustrated in the right part of Figure 6.

Proposition 4.2 *Algorithm 6 correctly computes a witness box.*

Proof: For n fixed, the boxes $M(\mathbf{B}^n)$ and $\mathbf{B}'$ have the same center, and the width of $M(\mathbf{B}^n)$ is a_n. During the repeat loop of line 5, the width of $\mathbf{B}'$ is strictly decreasing, and the condition $\mathbf{B}' \subseteq M(\mathbf{B}^n)$ of Line 8 will be true after a finite number of loops.

If $\mathbf{B}^n_\sigma = \sigma + [-a_n, a_n] \times [-2b_n, 2b_n]$, this is a strictly decreasing sequence with respect to inclusion, and for $m > n$, $\mathbf{B}^m \subset \mathbf{B}^n_\sigma$. Note also that $NW(\partial\mathbf{B}^n) \cup NE(\partial\mathbf{B}^n) \cup SW(\partial\mathbf{B}^n) \cup SE(\partial\mathbf{B}^n) \subset \mathbf{B}^n_\sigma$. The boxes $\mathbf{B}^n$ and $\mathbf{B}$ both contain the node σ in their interior, so the condition $\mathbf{B}^n \subseteq \mathbf{B}$ of Line 9 will be true for any n large enough.

It remains to show that for n large enough Algorithm 6 will report $\mathbf{B}^n$ as witness. We will show that $\mathbf{B}^n$ satisfies the sufficient condition (iii) of Proposition 4.1.

Since $\mathbf{B} \times \mathbb{R}$ does not contain any x and y-critical points of $\mathcal{C}$, for n large enough, one can assume that $\mathcal{C} \cap (\mathbf{B}_\sigma^n \times \mathbb{R})$ has two connected components $\mathcal{C}^1$ and $\mathcal{C}^2$ that are x and y-monotone. These two branches project onto two curves $\mathcal{B}^1 = \pi_{(x,y)}(\mathcal{C}^1)$ and $\mathcal{B}^2 = \pi_{(x,y)}(\mathcal{C}^2)$ that are x-monotone, cross at σ and have bounded slopes. On the other hand, the aspect ratio $\frac{b_n}{a_n} = 2^n \frac{b_0}{a_0}$ of the box $\mathbf{B}^n$ is increasing. For n large enough, $\mathcal{B}^1$ (resp. $\mathcal{B}^2$) thus crosses $E(\partial\mathbf{B})$ and $W(\partial\mathbf{B})$ exactly once, and there is no other intersection on $NW(\partial\mathbf{B}^n) \cup N(\partial\mathbf{B}^n) \cup NE(\partial\mathbf{B}^n) \cup SW(\partial\mathbf{B}^n) \cup S(\partial\mathbf{B}^n) \cup SE(\partial\mathbf{B}^n)$. Condition (iii) of Proposition 4.1 is thus satisfied, and the returned box $\mathbf{B}^n$ is a witness box. □

4.4 Topology at a cusp singularity

Algorithms 5 and 6 are easily adapted for a cusp. The input of Algorithm 6 should then be a box $\boldsymbol{D}$ containing a unique solution of the ball system that projects into a cusp σ and such that $\pi'_{(x,y)}(\boldsymbol{D}) \times \mathbb{R}$ does not contain any other x or y-critical point of $\mathcal{C}$ than the point projecting onto the cusp. The input of Algorithm 5 should then be a box $\mathbf{B}$ containing a unique cusp of $\mathcal{B}$, such that $\mathbf{B} \times \mathbb{R}$ does not contain any other x or y-critical point of $\mathcal{C}$ other than the point projecting onto the cusp. The test for the number of reported solutions at Line 8 should be with the value 2 instead of 4. The output will be true when $\mathcal{B}$ intersects $E(\partial\mathbf{B})$ exactly once, $W(\partial\mathbf{B})$ exactly once, and does not intersect $NW(\partial\mathbf{B}) \cup N(\partial\mathbf{B}) \cup NE(\partial\mathbf{B}) \cup SW(\partial\mathbf{B}) \cup S(\partial\mathbf{B}) \cup SE(\partial\mathbf{B})$.

For the proof of correctness, the same arguments hold when the limit of the tangent to $\mathcal{B}$ at the cusp is not collinear with the y-axis, since in this case the slope of the branch is bounded. When this limit is the y-axis, the same algorithm with the variables x and y swapped will behave as above. In other words, the box $\mathbf{B}^n$ is elongated in the x-direction instead of the y-direction, and intersections of the curve with its boundary will eventually appear on the north or south sides and far from the corners. The solution is thus to run the two algorithms in parallel, and stop as soon as one has identified a witness box.

4.5 Topology at an x-extreme point

For an x-extreme point, the method is similar to the one for a cusp with a limit of the tangents collinear with the y-axis. Indeed, the tangent to the curve at an x-extreme point is collinear with the y-axis, so that the curve is locally y-monotone. The box $\mathbf{B}^n$ is thus elongated in the x-direction, and intersections of the curve with its boundary will eventually appear on the north or south sides and far from the corners.

5 Global Topology of $\mathcal{B}$ as an Embedded Graph

In this section, the certified tracking of the curve $\mathcal{C} \cap \boldsymbol{C}_0$ together with the local topology at the special points of its projection $\mathcal{B} \cap \mathbf{B}_0$ are combined to compute the global topology of $\mathcal{B} \cap \mathbf{B}_0$. We use an XPMap to encode this topology and design a point location algorithm, that is, given $p \in \mathbf{B}_0 \setminus \mathcal{B}$, find the connected component of $\mathbf{B}_0 \setminus \mathcal{B}$ to which p belongs.

In Section 5.1, we compute a graph G such that $\mathcal{B} \cap \mathbf{B}_0$ is an embedding of G. The vertices of G are the special points computed in Section 4, and the edges are computed

by tracking between these points. In addition, to restrict the point location algorithm to the box $\mathbf{B}_0$, we consider the curve $\mathcal{B}_\partial = (\mathcal{B} \cap \mathbf{B}_0) \cup \partial\mathbf{B}_0$ as the embedding of a graph G_∂. The graph G_∂ and its embedding $\mathcal{B}_\partial$ have in general several connected components. Section 5.2 shows how to compute a CMap encoding the topology of one component. The only geometric task for this is to order branches of $\mathcal{B}$ around node singularities. Section 5.3 shows how to answer location queries with respect to one connected component of $\mathcal{B}_\partial$. In Section 5.4, the XPMap encoding the topology of $\mathcal{B}_\partial$ is constructed, and the point location algorithm is generalized to this structure.

5.1 Computing a graph of which $\mathcal{B} \cap \mathbf{B}_0$ is an embedding

Let B^x and B^n be the sets of witness boxes for the x-extreme points or cusps, and let the nodes of $\mathcal{B} \cap \mathbf{B}_0$ be computed in Section 4. We define V^x and V^n as the cylinders above witness boxes: $\mathrm{V}^x = \{\mathbf{B} \times \mathbb{R} \,|\, \mathbf{B} \in \mathrm{B}^x\}$ and $\mathrm{V}^n = \{\mathbf{B} \times \mathbb{R} \,|\, \mathbf{B} \in \mathrm{B}^n\}$. We note that C_0 (resp. B_0) is the domain in $\mathbb{R}^3$ (resp. $\mathbb{R}^2$) defined as $\boldsymbol{C}_0 \setminus \bigcup_{\boldsymbol{C} \in \mathrm{V}^x \cup \mathrm{V}^n} i(\boldsymbol{C})$ (resp. $\pi_{(x,y)}(C_0)$). Recall that C^b is the set of implicit points for the boundary points. We define V^b as $\{\boldsymbol{C} \,|\, (\boldsymbol{C}, S) \in \mathrm{C}^b\}$.

Since x-critical points of $\mathcal{C} \cap \boldsymbol{C}_0$ are isolated in boxes of V^x, the connected components of $\mathcal{C} \cap C_0$ are diffeomorphic to $[0, 1]$ and x-monotone. Their endpoints are $\mathcal{C} \cap \bigcup_{\boldsymbol{C} \in \mathrm{V}^x \cup \mathrm{V}^n \cup \mathrm{V}^b} \partial\boldsymbol{C}$[3]. Hence a point of the latter set is either the left-most or the right-most point of a connected component of $\mathcal{C} \cap C_0$, and we state the following definition.

Definition 5.1 *Let $\boldsymbol{C}$ be a box of $\mathrm{V}^x \cup \mathrm{V}^n$ or the box $\boldsymbol{C}_0$. We call a point c of $\mathcal{C} \cap \partial\boldsymbol{C}$ a connection of $\mathcal{C} \cap C_0$ in $\boldsymbol{C}$. If c is the left-most point of a connected component of $\mathcal{C} \cap C_0$, we say that c is an out-connection. Otherwise we call c an in-connection.*

Note that implicit points of C^b define connections of $\mathcal{C} \cap C_0$ in $\boldsymbol{C}_0$. During the computation of witness boxes as described in Section 4, the connections are also computed. We thus assume that for a box $\boldsymbol{C} \in \mathrm{V}^x \cup \mathrm{V}^n$, the connections of $\mathcal{C} \cap C_0$ in $\boldsymbol{C}$ are given by the set $\mathit{connect}(\boldsymbol{C}) = \{\ldots, (\boldsymbol{C}_i, S_i), \ldots\}$ of disjoint implicit points (i.e. $\boldsymbol{C}_i$ are pairwise disjoint). If $\boldsymbol{C} \in \mathrm{V}^b$, we let $\mathit{connect}(\boldsymbol{C}) = \{(\boldsymbol{C}, S)\}$ where S is the system defining the boundary containing $\boldsymbol{C}$ (see Corollary 2.1).

Let V be the set $\mathrm{V}^x \cup \mathrm{V}^n \cup \mathrm{V}^b$. Let E be the set of connected components of $\mathcal{C} \cap C_0$. We define the incidence relation $I : E \to V \times V$ such that for $e \in E$, $I(e) = (\boldsymbol{C}, \boldsymbol{C}')$ if and only if the extremities of e are an out-connection of $\mathcal{C} \cap C_0$ in $\boldsymbol{C}$ and an in-connection of $\mathcal{C} \cap C_0$ in $\boldsymbol{C}'$. By construction, the projections of edges do not cross, and the incidence of four edges at a node is correctly encoded. This yields the following proposition.

Proposition 5.1 *Let G be the graph (V, E, I). The curve $\mathcal{B} \cap \mathbf{B}_0$ is an embedding of G, seen as a non-directed graph.*

We denote by $v_1, v_2, \ldots$ the vertices of G (i.e. the boxes of $V = \mathrm{V}^x \cup \mathrm{V}^n \cup \mathrm{V}^b$) and by $e_1, e_2, \ldots$ the edges of G, (i.e. the connected components of $\mathcal{C} \cap C_0$). The graph G representing the apparent contour of the torus is shown in the left part of Figure 7.

We show in Section 5.1.1 how to distinguish in and out connections of $\mathcal{C} \cap C_0$ in boxes of V, and, in Section 5.1.2, how to compute the incidence relation I by

[3]Recall that boxes of V^b have exactly one coordinate reduced to one point; thus $\partial\boldsymbol{C} = \boldsymbol{C}$ if $\boldsymbol{C} \in \mathrm{V}^b$.

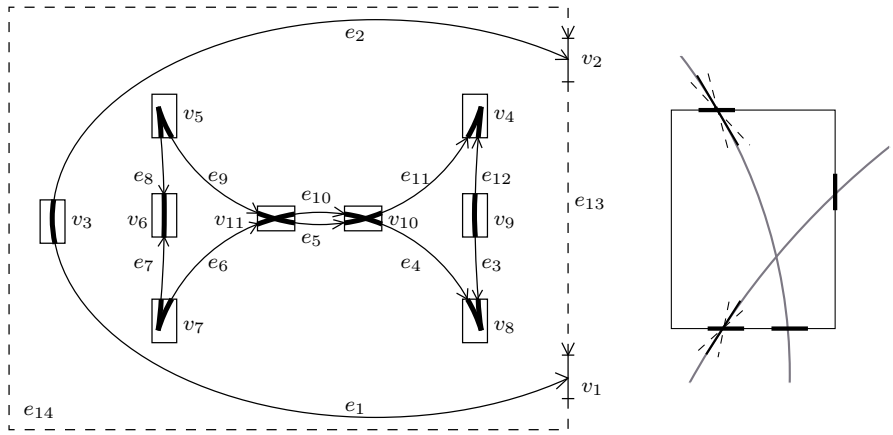


Figure 7: Left: The directed graph G_∂ for the apparent contour of the torus. Its edges are represented by thin arrows linking out-connections to in-connections of $\mathcal{C} \cap C_0$ in boxes of V. Thick lines represent the curve $\mathcal{B}$ in witness boxes. Dashed arrows represent the clockwise walk on $\partial\mathbf{B}_0$. Right: A witness box $\mathbf{B}$ for a node, and, in bold, the four boxes isolating the connections of $\mathcal{B}$ in $\mathbf{B}$.

computing δ-approximations of the connected components of $\mathcal{C} \cap C_0$. We consider G to be equipped with an application *approx* that maps each $e \in E$ to its δ-approximation. Furthermore, *approx* satisfies the three following properties:

($a1$) If e, e' are two edges of G, then $\mathit{approx}(e, \delta) = (\boldsymbol{C}_i^e)_{i=1}^{m_e}$ and $\mathit{approx}(e', \delta) = (\boldsymbol{C}_i^{e'})_{i=1}^{m_{e'}}$ satisfy $1.1\boldsymbol{C}_i^e \cap \boldsymbol{C}_{i'}^{e'} = \emptyset$ for all $1 \leq i \leq m_e$, for all $1 \leq i' \leq m_{e'}$.

($a2$) If e is an edge of G, $\boldsymbol{C}$ a box of V and $\mathit{approx}(e, \delta) = (\boldsymbol{C}_i^e)_{i=1}^{m_e}$, then $\pi_{(x,y)}(\boldsymbol{C}_i^e) \cap \pi_{(x,y)}(\boldsymbol{C}) \neq \emptyset$ if and only if $i = 1$ or $i = m_e$ and e has a connection in $\boldsymbol{C}$.

($a3$) If e is an edge of G and $\mathit{approx}(e, \delta) = (\boldsymbol{C}_i^e)_{i=1}^{m_e}$ then $\pi_{(x,y)}(\boldsymbol{C}_1^e), \pi_{(x,y)}(\boldsymbol{C}_m^e)$ contain no node, cusp or x-extreme point of $\mathcal{B}$.

In the following, we consider the faces of $\mathcal{B} \cap \mathbf{B}_0$ in $\mathbf{B}_0$ that are the connected components of $\mathbf{B}_0 \setminus \mathcal{B}$. It is thus convenient to represent the boundary of $\mathbf{B}_0$ explicitly in a graph $G_\partial = (V_\partial, E_\partial, I_\partial)$ of which the curve $\mathcal{B}_\partial = (\mathcal{B} \cap \mathbf{B}_0) \cup \partial\mathbf{B}_0$ is an embedding. G_∂ is defined as follows.

If V^b is empty, then $V_\partial = V \cup \{\mathbf{B}_\partial \times \mathbb{R}\}$ where $\mathbf{B}_\partial$ is a box reduced to the left bottom corner of $\mathbf{B}_0$ and $E_\partial = E \cup \{e_\partial\}$ where e_∂ is $\partial\mathbf{B}_0 \setminus \mathbf{B}_\partial$. Otherwise ($\mathrm{V}^b$ is not empty), $V_\partial = V$ and $E_\partial = E \cup E'$ where E' is the set of connected components of $\partial\mathbf{B}_0 \setminus \bigcup_{\boldsymbol{C} \in \mathrm{V}^b} i(\pi_{(x,y)}(\boldsymbol{C}))$. Then $I_\partial(e)$ is defined as $I(e)$ if $e \in E$. Otherwise $e \in E_\partial \setminus E$ and $I_\partial(e) = (v, v')$ if and only if the extremities of e are extremities of the segments $\pi_{(x,y)}(v)$ and $\pi_{(x,y)}(v')$, and the walk from v to v' around $i(\mathbf{B}_0)$ is a clockwise walk.

It is clear that the curve $\mathcal{B}_\partial$ is an embedding of G_∂. The left part of Figure 7 shows, in dashed arrows, the edges of $E_\partial \setminus E$.

5.1.1 In and out connections

We characterize in and out connections of $\mathcal{C} \cap C_0$ in a box of $\mathrm{V}^x \cup \mathrm{V}^n$ or in $\boldsymbol{C}_0$ using the direction of the tangent to the curve. For this, let $E^*(\partial\mathbf{B})$ (resp. $W^*(\partial\mathbf{B})$) denote

the set $(E(\partial\mathbf{B})\cup SE(\partial\mathbf{B})\cup NE(\partial\mathbf{B}))\cap\mathbf{B}$ (resp. $(W(\partial\mathbf{B})\cup SW(\partial\mathbf{B})\cup NW(\partial\mathbf{B}))\cap\mathbf{B}$). We first consider the case where $\boldsymbol{C}\in \mathrm{V}^x\cup\mathrm{V}^n$.

Proposition 5.2 *Let c be a connection of $\mathcal{C}\cap C_0$ in a box $\boldsymbol{C}\in\mathrm{V}^x\cup\mathrm{V}^n$, and $\mathbf{B}=\pi_{(x,y)}(\boldsymbol{C})$. Let $t=(t_x,t_y,t_z)$ be a tangent vector of $\mathcal{C}$ at c. Then*

(i) $t_x\neq 0$ *and* $t_y\neq 0$*;*

(ii) *c is an out-connection of $\mathcal{C}\cap C_0$ in $\boldsymbol{C}$ if and only if $\pi_{(x,y)}(c)\in E^*(\partial\mathbf{B})$ or $\pi_{(x,y)}(c)\in N(\partial\mathbf{B})$ and $t_xt_y>0$ or $\pi_{(x,y)}(c)\in S(\partial\mathbf{B})$ and $t_xt_y<0$;*

(iii) *c is an in-connection of $\mathcal{C}\cap C_0$ in $\boldsymbol{C}$ if and only if $\pi_{(x,y)}(c)\in W^*(\partial\mathbf{B})$ or $\pi_{(x,y)}(c)\in N(\partial\mathbf{B})$ and $t_xt_y<0$ or $\pi_{(x,y)}(c)\in S(\partial\mathbf{B})$ and $t_xt_y>0$.*

Proof: According to Section 4, the boxes in $\mathrm{V}^x\cup\mathrm{V}^n$ do not contain any x or y-critical points on their boundary, so property (i) holds. Claim (ii) (resp. (iii)) rephrases the conditions such that the tangent vector pointing out of the box is oriented to the left (resp. right) thus yielding an out-connection, (resp. in-connection). □

Proposition 5.2 is easily adapted for boundary points, that is, when c is a connection of $\mathcal{C}\cap C_0$ in $\boldsymbol{C}_0$. From assumptions $(A_4),(A_{10})$, $\partial\boldsymbol{C}_0$ contains neither x nor y-critical points, and (i) follows. Then one can easily show that properties (ii') and (iii'), obtained from (ii) and (iii) by swapping E and W, and N and S, hold, since in this case the tangent vector pointing inside the box is relevant.

Proposition 5.2 is used to decide if a connection c implicitly defined by the ordered pair $(\boldsymbol{C}_i,S_i)$ is an out or an in connection, as follows. $\boldsymbol{C}_i$ can be contracted with the Krawczyk operator K_{S_i} until $0\notin R(\boldsymbol{C}_i)$ and $0\notin R'(\boldsymbol{C}_i)$, where R and R' as in Definition 2.1 are the x and y component of a tangent vector. This process terminates from (i) of Proposition 5.2, and $(ii),(iii),(ii'),(iii')$ are used to finish the argument. The right part of Figure 7 illustrates this in the case of a node singularity. The cones containing the tangent vector $(R(\boldsymbol{C}_i),R'(\boldsymbol{C}_i))$ are drawn with dashed lines for two of the four boxes isolating the connections in its witness box.

5.1.2 Computing the incidence relation I and δ-approximations of edges

Let C^+ (resp. C^-) be the set of implicit points defining the out (resp. in) connections of $\mathcal{C}\cap C_0$ in boxes of $\mathrm{V}^x\cup\mathrm{V}^n$ and in $\boldsymbol{C}_0$. For each out-connection $(\boldsymbol{C}_i,S_i)\in\mathrm{C}^+$, the process `Track(`$<P,Q>$`,`$C_0$`,`$(\boldsymbol{C}_i,S_i)$`,`$\mathrm{C}^-\cup\{(\boldsymbol{C}_i,S_i)\}$`,`$\delta$`)` defined in Algorithm 2 is performed. This process terminates and returns a δ-approximation $(\boldsymbol{C}_i^e)_1^{m_e}$ of the connected component e of $\mathcal{C}\cap C_0$ having $(\boldsymbol{C}_i,S_i)$ as left-most point and the set C^{on} containing exactly the implicit point $(\boldsymbol{C}_j,S_j)$ defining the right-most point of e. This defines the edge $e\in E$ with $I(e)=(v,v')$, where v (resp. v') is the vertex of G having $(\boldsymbol{C}_i,S_i)$ (resp. $(\boldsymbol{C}_j,S_j)$) as one of its out (resp. in) connections. We let $\mathit{approx}(e,\delta)=(\boldsymbol{C}_i^e)_{i=1}^{m_e}$.

To ensure that the application *approx* satisfies the properties $(a1)$, $(a2)$ and $(a3)$ listed above, the tracking of edges is refined as follows. Each time a new approximation $\mathit{approx}(e,\delta)$ is computed, $(a2)$ is checked and a $\frac{\delta}{2}$-approximation of e is computed while it does not hold. This process terminates due to the properties of witness boxes. Then for each e' for which $\mathit{approx}(e',\delta)$ is already known, $(a1)$ is checked for e,e'. While it does not hold, $\frac{\delta}{2}$-approximations of e and e' are computed. This process terminates since e and e' are, by construction, non-intersecting smooth curves. Finally $(a3)$ is checked for e,e'. Let $\mathit{approx}(e,\delta)=(\boldsymbol{C}_i^e)_{i=1}^{m_e}$ and $\mathit{approx}(e',\delta)=(\boldsymbol{C}_i^{e'})_{i=1}^{m_{e'}}$. Since $(a2)$

Algorithm 7 CMap construction

Input: A graph $G^k = (V^k, E^k, I^k)$
Output: A CMap $(H^k, \sigma^k, \alpha^k)$ encoding the embedding $\mathcal{B}^k_\partial$ of G^k
1: Let $\mathcal{V}^k = \{i \,|\, v_i \in V^k\}$, $\mathcal{E}^k = \{i \,|\, e_i \in E^k\}$ and $(\mathcal{O}_i)_{i\in\mathcal{V}^k}$ be empty sets.
2: Let $H^k = \emptyset$, α^k be an involution on H^k and σ^k be a permutation on H^k
3: **for** $i \in \mathcal{E}^k$ **do**
4: Let $H^k = H^k \cup \{h_i^+, h_i^-\}$, $\alpha^k = \alpha^k \cup \{(h_i^+, h_i^-)\}$
5: Suppose $I^k(e_i) = (v_j, v_l)$ with $j, l \in \mathcal{V}^k$ and let $\mathcal{O}_j = \mathcal{O}_j \cup \{h_i^+\}$, $\mathcal{O}_l = \mathcal{O}_l \cup \{h_i^-\}$
6: **for** $i \in \mathcal{V}^k$ **do**
7: Order counter-clockwise the connections of $\mathcal{O}_i$ around $\partial(\pi_{(x,y)}(v_i))$
8: Let σ^k be the permutation on H^k which orbits are $\mathcal{O}_i$, for $i \in \mathcal{V}^k$.
9: **return** $(H^k, \sigma^k, \alpha^k)$

holds, the ball system can have solutions only in the 4-dimensional boxes constructed from $\boldsymbol{C}_1^e, \boldsymbol{C}_1^{e'}, \boldsymbol{C}_{m_e}^e, \boldsymbol{C}_{m_{e'}}^{e'}$ as in Proposition 3.1. Then, $\frac{\delta}{2}$-approximations of e and e' are computed while the ball system has a solution in one of the latter boxes. This process terminates since each singularity of $\mathcal{B}$ has a strictly positive distance from any point of the boundary of a witness box around it. To avoid x-extreme points, one has to check that boxes $\boldsymbol{C}_1^e$ and $\boldsymbol{C}_{m_e}^e$ do not contain solutions of the system (S_5) as defined in Corollary 2.1, and refine the boxes if needed.

5.2 Computing the CMap of one connected component

Let $\mathcal{B}^1_\partial, \ldots, \mathcal{B}^{n'}_\partial$ be the connected components of $\mathcal{B}_\partial$. Then G_∂ has n' connected components $G^1 = (V^1, E^1, I^1), \ldots, G^{n'} = (V^{n'}, E^{n'}, I^{n'})$ such that the vertices of G^i are the cusps, nodes, x-extreme and boundary points of $\mathcal{B}^i_\partial$, and the projections of the edges of G^i are the connected components of $\mathcal{B}^i_\partial \setminus i(B_0)$. We suppose that G^1 is the connected component containing the boundary points of $\mathcal{B}_\partial$. We aim here at computing, for each $1 \leq k \leq n'$, a CMap $(H^k, \sigma^k, \alpha^k)$ representing the embedding $\mathcal{B}^k_\partial$ of $G^k = (V^k, E^k, I^k)$. As emphasized in [18][§1.3.3], or in [16], computing the faces of an embedding reduces to ordering counter-clockwise the edges around each vertex. The CMap $(H^k, \sigma^k, \alpha^k)$ representing the embedding $\mathcal{B}^k_\partial$ of $G^k = (V^k, E^k, I^k)$ encodes this order in the permutation σ^k, and each face of $\mathcal{B}^k_\partial$ is an orbit of $\varphi^k = (\sigma^k)^{-1} \circ \alpha^k$ which is an ordered sequence of half-edges describing a counter-clockwise walk around it. A vertex of V^k corresponds either to a node, a cusp, a boundary point or a x-extreme point of $\mathcal{B}^k$. Around a cusp or an x-extreme point p, $\mathcal{B}^k \setminus p$ has only two branches; thus there is no need to order them.

Let p be a node of $\mathcal{B}^k$ and $\mathbf{B}$ its associated witness box. Since the connected components of $(\mathcal{B}^k \setminus p) \cap \mathbf{B}$ are non intersecting curves linking p to the projections of the connections of $\mathcal{C}$ in $\mathbf{B} \times \mathbb{R}$, a counter-clockwise ordering of the branches of $\mathcal{B}^k \setminus p$ around p is given by the counter-clockwise order of the connections of $\mathcal{B}^k$ in $\mathbf{B}$ (see the right part of Figure 7).

Now let p be a boundary point of $\mathcal{B}^k \cap \mathbf{B}_0$. A counter-clockwise ordering of the branches of $\mathcal{B}^k_\partial \setminus p$ around p can be directly deduced from the part of the boundary to which p belongs. Consider the left part of Figure 7, and let p be the boundary point of which v_1 is a witness box. Since p is on $E^*(\partial\mathbf{B})$, a counter-clockwise ordering is necessarily (e_1, e_{14}, e_{13}).

Algorithm 8 Point-face location on the boundary of a witness box

Input: The CMap $(H^k, \sigma^k, \alpha^k)$ of $\mathcal{B}^k_\partial$, a box $\boldsymbol{C} \in \mathrm{C}^x \cup \mathrm{C}^n$, and a point $p \in \partial\mathbf{B}$ where $\mathbf{B} = \pi_{(x,y)}(\boldsymbol{C})$.
Preconditions: p is not in $\mathcal{B}$
Output: The orbit of φ^k describing the face of $\mathcal{B}^k_\partial$ to which p belongs.
1: Let $v = (\ldots, h^*_i, \ldots)$ be the orbit of σ^k corresponding to $\boldsymbol{C}$
2: Let $\{\ldots, c_i = (\boldsymbol{C}_i, S_i), \ldots\}$ be the connections of $\mathcal{C}$ in $\boldsymbol{C}$ associated to v
3: Contract each c_i with $\boldsymbol{C}_i = K_{S_i}(\boldsymbol{C}_i)$ until their projections are disjoint and $p \notin \pi_{(x,y)}(\boldsymbol{C}_i)$
4: Order counter-clockwise connections c_i and p on $\partial\mathbf{B}$ and let $(c_1, p, c_2, \ldots)$ be such an ordering
5: Let f be the orbit of φ^k containing h^*_1
6: **return** f

Algorithm 7 performs the construction of the CMap of a connected component $\mathcal{B}^k_\partial$ of $\mathcal{B}_\partial$. In a first loop, pairs of half-edges are created from edges: to each edge $e_i \in E^k$ are associated two half-edges h^+_i and h^-_i, such that h^+_i is oriented as e_i, i.e. if $I^k(e_i) = (v_j, v_l)$, h^+_i leaves v_j and h^-_i leaves v_l. In other words, h^+_i represents a walk from left to right along $\pi_{(x,y)}(e_i)$. In addition, half-edges leaving a common vertex are collected. The second loop aims at ordering the half-edges around vertices to define faces.

5.3 Faces of a CMap

To define the relative positions of the CMaps encoding the different connected components and thus computing an XPMap, the exterior face of each CMap is first identified. Then, we propose a procedure to identify to which face of $\mathcal{B}^k$ a point $p \notin \mathcal{B}^k$ belongs.

5.3.1 Exterior face

Point-face location on the boundary of a witness box. As an intermediate step, Algorithm 8 identifies the face containing a point located on the boundary of a witness box. In the algorithm, h^* stands for a half-edge h^+ or h^-. In Step 3, the connections of $\mathcal{C}$ in $\boldsymbol{C}$ are contracted with the appropriated Krawczyk operator until their projections are pairwise disjoint and do not contain p. This step terminates, since $p \notin \mathcal{B}$. Then, it suffices to order counter-clockwise p and the projections of the connections of $\mathcal{C}$ in $\boldsymbol{C}$ on $\partial(\pi_{(x,y)}(\boldsymbol{C}))$ to conclude. The right (resp. left) part of Figure 8 illustrates this procedure when $\pi_{(x,y)}(\boldsymbol{C})$ is the witness box of a node (resp. cusp). The orbit of the node vertex is $(h^-_1, h^+_2, h^+_3, h^-_4)$ and $(c_1, p, c_2, \ldots)$ is a counter-clockwise order on the boundary of the box. The face containing p is given by the orbit of φ^k containing h^-_1.

Exterior face and leftmost box. We now explain Algorithm 9, that computes the orbit of φ^k describing the exterior face of $\mathcal{B}^k_\partial$. Note that by construction all faces but the exterior face are described by a counter-clockwise cycle of half-edges. The case where $k = 1$ is directly addressed, since $\mathcal{B}^1_\partial$ contains $\partial\mathbf{B}_0$. The exterior face of $\mathcal{B}^1_\partial$ is exactly $\mathbb{R}^2 \setminus \mathbf{B}_0$ and is described by the orbit of φ^1 containing the half-edges associated

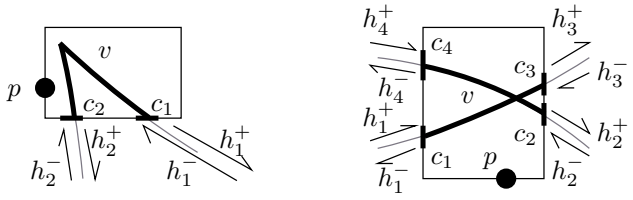


Figure 8: Finding the face containing a point p on the boundary of the witness box of a cusp (left) and of a node (right).

Algorithm 9 Exterior face

Input: The CMap $(H^k, \sigma^k, \alpha^k)$ of $\mathcal{B}^k_\partial$
Output: The orbit of φ^k describing the exterior face of $\mathcal{B}^k_\partial$ and a leftmost box $\boldsymbol{C}$ enclosing $\mathcal{B}^k_\partial$ if $k > 1$, $\boldsymbol{C} = \emptyset$ if $k = 1$.
1: **if** $k = 1$ **then**
2: Let e be an edge of $E_\partial \setminus E$ and h^+ be the half-edge associated with e
3: Let f be the orbit of φ^k containing h^+ and $\boldsymbol{C} = \emptyset$
4: **else**
5: Find a leftmost box $\boldsymbol{C}$ enclosing $\mathcal{B}^k_\partial$ and let p be any point on $W^*(\partial(\pi_{(x,y)}(\boldsymbol{C})))$
6: Let f be the result of Algorithm 8 with input $(H^k, \sigma^k, \alpha^k)$, $\boldsymbol{C}$ and p
7: **return** f, $\boldsymbol{C}$

with edges of $E_\partial \setminus E$ (see Steps 1, 2 and 3 of Algorithm 9). As an illustration, the exterior face of $\mathcal{B}^1_\partial$ on Figure 3 is the orbit (h^+_{13}, h^+_{14}).

Suppose now $k > 1$, and recall that $\mathcal{B}^k_\partial$ is an embedding of $G^k = (V^k, E^k, I^k)$. Recall also that for $\mathbf{B} = (\mathbf{x}, \mathbf{y})$, $W^*(\partial\mathbf{B}) = (l(\mathbf{x}), \mathbf{y})$. We state the following remark to identify the exterior face of $\mathcal{B}^k_\partial$.

Remark and Definition 5.1 *Let $k > 1$.*

(i) $\mathcal{B}^k_\partial$ has at least two cusps or x-critical points, that is $V^k \cap \mathrm{V}^x$ contains at least two elements.

We call a leftmost box enclosing $\mathcal{B}^k_\partial$ a box $\boldsymbol{C} = (\mathbf{x}, \mathbf{y}, \mathbb{R})$ of $V^k \cap \mathrm{V}^x$ minimizing $l(\mathbf{x})$ over all boxes of $V^k \cap \mathrm{V}^x$. Let $\boldsymbol{C}$ be one of the leftmost boxes enclosing $\mathcal{B}^k_\partial$, and $\mathbf{B} = \pi_{(x,y)}(\boldsymbol{C})$. Then

(ii) any point of $W^*(\partial\mathbf{B})$ lies in the exterior face of $\mathcal{B}^k_\partial$,

(iii) if $p \in W^*(\partial\mathbf{B})$ then $p \notin \mathcal{B}$.

Point (i) is a direct consequence of Proposition 2.1. In order to prove point (ii), consider a point $p \in W^*(\partial\mathbf{B})$, and suppose it does not belong to the exterior face of $\mathcal{B}^k_\partial$. Hence $\mathcal{B}^k_\partial$ has necessarily a cusp or x-critical point lying to the left of p. As a consequence, there is a box $\boldsymbol{C}' = (\mathbf{x}', \mathbf{y}', \mathbb{R})$ in $V^k \cap \mathrm{V}^x$ such that $l(\mathbf{x}') < p_x$, where p_x is the x-coordinate of p, and $\boldsymbol{C}$ is not a leftmost box enclosing $\mathcal{B}^k_\partial$. Consider now a point $p \in W^*(\partial\mathbf{B})$ such that $p \in \mathcal{B}$. Since $\boldsymbol{C}$ is the witness box, one has necessarily $p \in \mathcal{B}^k_\partial$ and a contradiction follows. Hence (iii) holds.

Algorithm 9 uses Remark 5.1 in Step 5. From point (iii), the call to the procedure described in Algorithm 8 terminates and returns the orbit of φ^k describing the exterior face of $\mathcal{B}^k_\partial$.

Algorithm 10 Point-face location in a CMap

Input: A point $p = (x_p, y_p) \in i(\mathbf{B}_0)$. The CMap $(H^k, \sigma^k, \alpha^k)$ encoding the embedding $\mathcal{B}^k_\partial$ of $G^k = (V^k, E^k, I^k)$. The exterior face f^{ext} of $(H^k, \sigma^k, \alpha^k)$. The boundary edges $E' = E_\partial \setminus E$ of G_∂.
Preconditions: $(c1)$ p is not in $\mathcal{B}^k_\partial$, $(c2)$ p is not in the projection of any box of V^k and $(c3)$ $\forall e_j \in E^k$, p is not in the projection of any box of $approx(e_j, \delta)$.
Output: The orbit of φ^k describing the face of $\mathcal{B}^k_\partial$ that contains p.

1: Let $S = \{P(x_p, y, z) = Q(x_p, y, z) = 0\}$, $\mathbf{B}^* = (\mathbf{y}^*, \boldsymbol{z}^*) = \emptyset$ and $f^* = \emptyset$
2: **for** $e_j \in E^k \setminus E'$ **do** //e_j is not a boundary edge
3: Let $\mathrm{X}^{\mathrm{sol}}$ be the result of Algorithm 11 with input e_j, p and S
4: **if** $\mathrm{X}^{\mathrm{sol}} \neq \emptyset$ **then**
5: Let $\mathbf{B}^*_j = (\mathbf{y}^*_j, \boldsymbol{z}^*_j)$ be the unique element in $\mathrm{X}^{\mathrm{sol}}$
6: **if** $\mathbf{B}^* = \emptyset$ **then** $\mathbf{B}^* = \mathbf{B}^*_j$
7: **else**
8: **while** $\mathbf{y}^*_j \cap \mathbf{y}^* \neq \emptyset$ **do** let $\mathbf{B}^*_j = K_S(\mathbf{B}^*_j)$ and $\mathbf{B}^* = K_S(\mathbf{B}^*)$
9: **if** $l(\mathbf{y}^*_j) < l(\mathbf{y}^*)$ **then** Let f^* be the orbit of φ^k containing h^-_j and let $\mathbf{B}^* = \mathbf{B}^*_j$
10: **for** $\boldsymbol{C} \in (\mathrm{V}^x \cup \mathrm{V}^n) \cap V^k$ where $\boldsymbol{C} = (\mathbf{x}, \mathbf{y}, \boldsymbol{z})$ **do**
11: **if** $x_p \in i(\mathbf{x})$ **and** $y_p < l(\mathbf{y}) < l(\mathbf{y}^*)$ **then** // when $\mathbf{B}^* = \emptyset$ we set $l(\mathbf{y}^*) = \infty$
12: Let $\mathbf{y}^* = \mathbf{y}$
13: Let f^* be the result of Algorithm 8 with input $(H^k, \sigma^k, \alpha^k)$, $\boldsymbol{C}$ and the point $(x_p, l(\mathbf{y}))$
14: **if** $f^* = \emptyset$ **then**
15: **if** $k = 1$ **then**
16: Let $e \in E'$ be the boundary edge above p and f^* be the orbit of φ^k containing h^-
17: **else** Let f^* be the exterior face f^{ext} of $(H^k, \sigma^k, \alpha^k)$ given by Algorithm 9
18: **return** f^*

5.3.2 Point-face location in a CMap

We now describe Algorithm 10 for the point-face location in a CMap. The preconditions $(c1), (c2), (c3)$ ensure its termination. In order to show the correctness of Algorithm 10, we consider the segment $\mathbf{B}^{seg} = x_p \times [y_p, u(\mathbf{y}_0)]$ and the set $\boldsymbol{C}^{seg} = \mathbf{B}^{seg} \times \mathbb{R}$, where $\mathbf{B}_0 = (\mathbf{x}_0, \mathbf{y}_0)$. The idea of the proposed approach is to find the closest intersection q of $\mathcal{B}^k_\partial$ with $\mathbf{B}^{seg}$. If q belongs to a smooth component $\pi_{(x,y)}(e_j)$ of $\mathcal{B}^k_\partial$ (i.e. a connected component of $\mathcal{B}^k_\partial \cap B_0$) the face is given by the orbit containing the half-edge h^-_j. If q is in the witness box $\boldsymbol{C} \in (\mathrm{V}^x \cup \mathrm{V}^n) \cap V^k$, we use Algorithm 8 to determine the face containing the intersection of $\mathbf{B}^{seg}$ with $S(\partial\pi_{(x,y)}(\boldsymbol{C}))$. Let us now give further details.

Proof:[Proof of correctness of Algorithm 10] Suppose first that $\boldsymbol{C}^{seg}$ does not intersect any box of $approx(e_i, \delta)$ where $e_i \in E^k$ nor any box of V^k. Then $\mathbf{B}^{seg}$ does not intersect $\mathcal{B}^k_\partial \setminus \partial\mathbf{B}_0$. If $k = 1$, i.e. $\mathcal{B}^k_\partial$ is the component of $\mathcal{B}_\partial$ containing $\partial\mathbf{B}_0$, we let p' be the point $(x_p, u(\mathbf{y}_0)) \in \partial\mathbf{B}_0$. Then p' lies on a boundary edge e_i of $E' = E_\partial \setminus E$ and the face of $\mathcal{B}^k_\partial$ containing p is described by the orbit of φ^k containing h^-_i. If $k > 1$, then p belongs to the exterior face of $\mathcal{B}^k_\partial$. The algorithm handles these cases in Steps 14 to 17.

Algorithm 11 Edge intersection

Input: An edge $e \in E^k$, a point $p = (x_p, y_p)$, the system $S = \{P(x_p, y, z) = Q(x_p, y, z) = 0\}$.
Preconditions: $(c1)$ p is not in $\mathcal{B}^k_\partial$ and $(c3)$ $\forall e_j \in E^k$, p is not in the projection of any box of $approx(e_j, \delta)$.
Output: A set $\mathrm{X}^{\mathrm{sol}}$ containing the unique intersection of e with $\boldsymbol{C}^{seg}$ if it exists, $\mathrm{X}^{\mathrm{sol}} = \emptyset$ otherwise.

1: Let $(\boldsymbol{C_i})_{i=1}^m = approx(e, \delta)$ and $\forall 1 \leq i \leq m$, $\boldsymbol{C}_i = (\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{z}_i)$ and $\mathbf{X}_i = (\mathbf{y}_i, \boldsymbol{z}_i)$
2: **for** $1 \leq i \leq m$ s.t. $x_p \in \mathbf{x}_i$ **and** $l(\mathbf{y}_i) > y_p$ **do** //$\boldsymbol{C}_i$ is located above p
3: Let $(\mathrm{X}^{\mathrm{sol}}, \mathrm{X}^{\mathrm{ind}}) = \texttt{IsolateSols(}S\texttt{,}1.1\mathbf{X}_i\texttt{,}\min(\frac{0.1\mathrm{w}(\mathbf{x}_i)}{2}, \frac{0.1\mathrm{w}(\mathbf{y}_i)}{2})\texttt{)}$
4: **if** $\mathrm{X}^{\mathrm{sol}} \neq \emptyset$ **then**
5: **break**
6: **return** $\mathrm{X}^{\mathrm{sol}}$

Consider now the case where $\boldsymbol{C}^{seg}$ does not intersect boxes of V^k but intersects some edges. The intersections of $\boldsymbol{C}^{seg}$ with the smooth and x-monotone components $e_j \in E^k \setminus E'$ are computed in the loop beginning in Step 2. Algorithm 11 computes, if it exists, the unique intersection $q' = (x_p, y_q, z_q)$ of e_j with $\boldsymbol{C}^{seg}$ and returns a non-empty set containing $\mathbf{B}^*_j \subset \boldsymbol{C}^{seg}$ with $(y_q, z_q) \in i(\mathbf{B}^*_j)$. $\mathbf{B}^*_j$ is made disjoint in the y-coordinate with the current intersection $\boldsymbol{B}^*$ in Step 8, and $\mathbf{B}^*_j$ is updated in Step 9 to contain the intersection of $\mathcal{C}$ with $\boldsymbol{C}^{seg}$ which projection is the closest above p. Due to the orientation of half-edges, the face below an edge e_j is described by the orbit of φ^k containing h_j^-. Thus f^* as updated in Step 9 is the face containing p.

Before describing the last case (i.e. when $\boldsymbol{C}^{seg}$ intersects some boxes of $(\mathrm{V}^x \cup \mathrm{V}^c \cup \mathrm{V}^b) \cap V^k$) we remark that $\boldsymbol{C}^{seg}$ can intersect only one boundary box in $\mathrm{V}^b \cap V^k$ corresponding to the implicit point $(\mathbf{B}, S_4)$ where S_4 is defined as in Corollary 2.1 and $\mathbf{B} = (\mathbf{x}, \boldsymbol{z})$ is such that $x_p \in \mathbf{x}$. Letting $e_j \in E^k$ be such that $(\mathbf{B}, S_4)$ is a connection of e_j and $approx(e_j, \delta) = (\boldsymbol{C}_i)_{i=1}^{m_j}$, it follows from property $(a2)$ of $approx$ that the boundary point implicitly defined by $(\mathbf{B}, S_4)$ is either in $\boldsymbol{C}_1$ or in $\boldsymbol{C}_{m_j}$. If the intersection $(x_p, u(\mathbf{y}_0))$ is in $\boldsymbol{C}_1$ or in $\boldsymbol{C}_{m_j}$, the appropriated face has been determined above. Otherwise, $(\mathbf{B}, S_4)$ can be refined with K_{S_4} until $(x_p, u(\mathbf{y}_0)) \notin \mathbf{B}$, and this case is handled in Steps 14 to 17 of the algorithm.

Suppose now that $\boldsymbol{C}^{seg}$ intersects some boxes of $(\mathrm{V}^x \cup \mathrm{V}^n) \cap V^k$. We can assume that there is a box $\boldsymbol{C} \in (\mathrm{V}^x \cup \mathrm{V}^n) \cap V^k$ with $l(\mathbf{y}) < l(\mathbf{y}^*)$, otherwise the output of the algorithm is determined by the loop of Step 2 or Steps 14 to 17. Such a box does not contain p from precondition $(c2)$. The loop beginning in Step 10 finds the box $(\mathbf{x}, \mathbf{y}, \boldsymbol{z}) \in (\mathrm{V}^x \cup \mathrm{V}^n) \cap V^k$ intersecting $\boldsymbol{C}^{seg}$ that minimizes $l(\mathbf{y})$, thus p lies in the same face than the point $(x_p, l(\mathbf{y}))$. Note that $(x_p, l(\mathbf{y})) \notin \mathcal{B}^k_\partial$, otherwise $(x_p, l(\mathbf{y}))$ is the projection of a connection of $\mathcal{C}$ in $\boldsymbol{C}$, and $\mathbf{B}^*$ found in the loop beginning at Step 2 necessarily satisfy $l(\mathbf{y}^*) \leq l(\mathbf{y})$. As a consequence, the input arguments given to Algorithm 8 in Step 13 satisfy its preconditions. □

Proof:[Proof of termination of Algorithm 10] It is shown below that if its input arguments satisfy preconditions $(c1)$ and $(c3)$, Algorithm 11 terminates. We have already shown that Algorithm 8 terminates due to condition $(c1)$. It only remains to prove the termination of the while loop in Step 8. The points of the curve represented by the boxes $\mathbf{B}^*$ and $\mathbf{B}^*_k$ belong to two different edges which are disjoint in projection on the (x, y)-plane, so after a finite number of contractions the y-coordinates will be disjoint intervals. □

Proof:[Proof of correctness and termination of Algorithm 11] Let $approx(e,\delta) = (\boldsymbol{C}_i)_{i=1}^m$ and $\boldsymbol{C}_i = (\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{z}_i)$ and $\mathbf{X}_i = (\mathbf{y}_i, \boldsymbol{z}_i)$. From properties $(a2)$ and $(a3)$ of *approx*, $e' = \mathcal{C} \cap \bigcup_{i=1}^m \boldsymbol{C}_i$ contains no x-extreme point of $\mathcal{C}$ and is a smooth x-monotone curve. Hence its intersections with $\boldsymbol{C}^{seg}$ are regular solutions of S. From property $(a1)$ of *approx*, for any $1 \leq i \leq m$, $1.1\boldsymbol{C}_i$ does not intersect any box of any other approximation. Hence for any $1 \leq i \leq m$, $1.1\mathbf{X}_i$ contains at most one solution of S. If $1.1\mathbf{X}_i$ contains a solution of S it is non-singular as a point of e'. As a first consequence of this, the call `IsolateSols(`S`,`$1.1\mathbf{X}_i$`,`$\min(\frac{0.1\mathrm{w}(\mathbf{x}_i)}{2}, \frac{0.1\mathrm{w}(\mathbf{y}_i)}{2})$`)` in Step 3 of Algorithm 11 terminates, and since any δ-approximation contains a finite number of boxes, Algorithm 11 terminates for any input.

Recall now that $e \subset e'$. Since both e and e' are smooth curves, a point of e' belongs to the same connected component of $\mathcal{C} \setminus \{c \in \mathcal{C} | \pi_{(x,y)}(c)$ is a singularity or a x-extreme point of $\mathcal{B}_\partial\}$. Hence for any $1 \leq i \leq m$, a solution of S in $1.1\mathbf{X}_i$ is a point of the edge represented by e. Reciprocally since $e \subset \bigcup_{i=1}^m \boldsymbol{C}_i$ ($approx(e,\delta)$ is a δ-approximation of e), there exists $1 \leq i \leq m$ such that $e \cap \boldsymbol{C}^{seg}$ is a solution of S in $\mathbf{X}_i$ satisfying $x_p \in \mathbf{x}_i$ and $l(\mathbf{y}_i) > y_p$ from preconditions $(c1)$ and $(c3)$ and $\mathrm{X}^{\mathrm{sol}}$ contains the corresponding solution. The correctness of Algorithm 11 follows. □

5.4 Embedding G_∂

In this part we show how to compute an XPMap $(H, H^0, \sigma, \alpha, ext, cont)$ representing the topology of $\mathcal{B}_\partial$, as defined in Section 1. It is shown above how to compute a CMap for each connected component $\mathcal{B}_\partial^k$ of $\mathcal{B}_\partial$, how to identify its exterior face and how to perform point-face location in $\mathcal{B}_\partial^k$. Thus computing the XPMap reduces to computing the relation *cont* that assigns each exterior face to a non-exterior face of another CMap or to the infinite face. We describe in Section 5.4.1 the construction of the relation *cont*. Point-face location in an XPMap is discussed in Section 5.4.2.

5.4.1 Constructing the XPMap

The relation *cont* that assigns the exterior face of a CMap to a non-exterior face of another CMap (or to the infinite face) can be computed iteratively thanks to the following remark, that can be proved similarly to Remark 5.1.

Remark 5.1 *Let $\mathcal{B}_\partial^k$ and $\mathcal{B}_\partial^{k'}$ with $k, k' > 1$ be two components of $\mathcal{B}_\partial$, let $\boldsymbol{C} = (\mathbf{x}, \mathbf{y}, \mathbb{R})$ be a leftmost box enclosing $\mathcal{B}_\partial^k$ and let $\boldsymbol{C}' = (\mathbf{x}', \mathbf{y}', \mathbb{R})$ be a leftmost box enclosing $\mathcal{B}_\partial^{k'}$. If $l(\mathbf{x}) \leq l(\mathbf{x}')$, then $\mathcal{B}^k$ lies in the exterior face of $\mathcal{B}^{k'}$.*

Suppose $\mathcal{B}_\partial^1, \ldots, \mathcal{B}_\partial^{n'}$ are indexed with respect to increasing $l(\mathbf{x}_k)$ where $\boldsymbol{C}^k = (\mathbf{x}_k, \mathbf{y}_k, \boldsymbol{z}_k)$ is a leftmost box enclosing $\mathcal{B}_\partial^k$. As a consequence of Remark 5.1, given k, k', k'' such that $1 \leq k'' < k < k' \leq n'$, the exterior face of $\mathcal{B}_\partial^k$ is not contained in any non-exterior face of $\mathcal{B}_\partial^{k'}$ and is contained in at least one non-exterior face of $\mathcal{B}_\partial^{k''}$ (since it is at least contained in a non-exterior face of $\mathcal{B}_\partial^1$). As a second consequence, if the exterior face f_k of $\mathcal{B}_\partial^k$ is contained in two non-exterior faces f''', f'' of respective components $\mathcal{B}_\partial^{k'''}, \mathcal{B}_\partial^{k''}$ with $1 \leq k''' < k''$, then the exterior face $f_{k''}$ of $\mathcal{B}_\partial^{k''}$ is contained in f''' and f_k is contained in the non-exterior face f'' of $\mathcal{B}_\partial^{k'''} \cup \mathcal{B}_\partial^{k''}$.

Algorithm 12 uses these arguments in an iterative process to construct the XPMap $(H, H^0, \sigma, \alpha, ext, cont)$ representing the topology of $\mathcal{B}_\partial$. Recall that $(H^0, \sigma^0, \alpha^0)$ is an empty CMap representing the infinite face $\mathbb{R}^2 \setminus \mathbf{B}_0$. We denote by f_0 the empty orbit of φ^0. The loop beginning in Step 2 computes CMaps $(H^k, \sigma^k, \alpha^k)$ for $\mathcal{B}^k$ for $1 \leq k \leq n'$,

Algorithm 12 XPMap construction

Input: The graph G_∂ and its connected components $G^k = (V^k, E^k, I^k)$ for $1 \leq k \leq n'$
Output: An XPMap $(H, H^0, \sigma, \alpha, ext, cont)$ representing the topology of $\mathcal{B}_\partial$

1: Let $H^0 = \emptyset$ and f_0 represent the infinite face
2: **for** $1 \leq k \leq n'$ **do**
3: Let $(H^k, \sigma^k, \alpha^k)$ be the result of Algorithm 7 with input G^k
4: Let $f_k, \boldsymbol{C}^k$ be the result of Algorithm 9 with input $(H^k, \sigma^k, \alpha^k)$
5: Let $H = H^1$, $\sigma = \sigma^1$, $\alpha = \alpha^1$, $ext = \{(H^1, f_1)\}$ and $cont = \{(f_0, f_1)\}$
6: Suppose $\boldsymbol{C}^k = (\mathbf{x}_k, \mathbf{y}_k, \boldsymbol{z}_k)$ for $1 < k \leq n'$
7: Re-index G^k, $(H^k, \sigma^k, \alpha^k)$, $f_k, \boldsymbol{C}^k$ wrt increasing $l(\mathbf{x}_k)$ for $1 < k \leq n'$
8: **for** $1 < k \leq n'$ **do**
9: Let p be any point on $W^*(\partial(\pi_{(x,y)}(\boldsymbol{C}^k)))$
10: Let $i = 0$ and $f = f_k$
11: **while** $f = f_{k-i}$ **do**
12: Let $i = i + 1$
13: Let f be the result of Algorithm 10 with input p, $(H^{k-i}, \sigma^{k-i}, \alpha^{k-i})$ and f_{k-i}
14: Let $H = H \cup H^k$, $\sigma = \sigma \cup \sigma^k$, $\alpha = \alpha \cup \alpha^k$, $ext = ext \cup \{(H^k, f_k)\}$, $cont = cont \cup \{f, f_k\}$
15: **return** $(H, H^0, \sigma, \alpha, ext, cont)$

together with their exterior faces f_k and the leftmost box enclosing $\mathcal{B}^k$. Since $\mathcal{B}^1_\partial$ is the connected component of $\mathcal{B}_\partial$ containing $\partial\mathbf{B}_0$, its exterior face is necessarily contained in the infinite face. In Step 5, an XPMap $(H, H^0, \sigma, \alpha, ext, cont)$ encoding the embedding of $\mathcal{B}^1$ is initialized with $H = H^1$, $\sigma = \sigma^1$, $\alpha = \alpha^1$, $ext = \{(H^1, f_1)\}$ and $cont = \{(f_0, f_1)\}$.

In Step 7, leftmost boxes $\boldsymbol{C}^k$ are sorted with respect to increasing $l(\mathbf{x}_k)$ and objects are re-indexed to apply the arguments detailed above. Let $k > 1$, and consider the k-th iteration of the **for** loop beginning in Step 8. Suppose the relation *cont* has been properly constructed for $\mathcal{B}^1_\partial, \ldots, \mathcal{B}^{k-1}_\partial$, i.e. $(H, H^0, \sigma, \alpha, ext, cont)$ encodes the faces of $\mathcal{B}^{<k}_\partial = \mathcal{B}^1_\partial \cup \ldots \cup \mathcal{B}^{k-1}_\partial$. One has to find the non-exterior face of $\mathcal{B}^{<k}_\partial$ containing the exterior face f_k of $\mathcal{B}^k_\partial$. One first finds a point p in f_k by taking any point on $W^*(\partial(\pi_{(x,y)}(\boldsymbol{C}^k)))$ as justified in Remark 5.1. Due to the property of a witness box, p lies in the same face of $\mathcal{B}^{<k}_\partial$ as $\mathcal{B}^k_\partial$. The **while** loop in Step 11 finds this face by calling Algorithm 10 with input p, $(H^{k-i}, \sigma^{k-i}, \alpha^{k-i})$ and f_{k-i} for increasing i. Remark 5.2 below shows that this input meets the preconditions of Algorithm 10 for any i and thus ensures the termination of these calls. An argument stated above shows that the first non-exterior face f obtained is the inner face of $\mathcal{B}^{<k}_\partial$ containing f_k, i.e. any other non-exterior face f' of $\mathcal{B}^{k'}_\partial$ containing f_k contains f. As a consequence, after Step 14, the relation *cont* is properly constructed for $\mathcal{B}^1_\partial, \ldots, \mathcal{B}^k_\partial$ and $(H, H^0, \sigma, \alpha, ext, cont)$ encodes the faces of $\mathcal{B}^{<k+1}_\partial$.

Remark 5.2 *Let $\boldsymbol{C}$ be a leftmost box enclosing $\mathcal{B}^k_\partial$, and p a point on $W^*(\partial(\pi_{(x,y)}(\boldsymbol{C})))$. From point (iii) of Remark 5.1, $p \notin \mathcal{B}$. Since witness boxes of V are pairwise disjoint, p is not in any box of $V^1_\partial \cup \ldots \cup V^{k-1}_\partial$. Then, $\forall e_i \in E^1_\partial \cup \ldots \cup E^{k-1}_\partial$, p is not in any box of approx(e_i, δ), as a consequence of the property (ii) of the application approx.*

5.4.2 Point-face location in a XPMap

We finally propose here a procedure that, given any point $p \in i(\mathbf{B}_0)$ and the XPMap encoding the embedding $\mathcal{B}_\partial$, finds the face of $\mathcal{B}_\partial$ to which p belongs. Such a procedure is easily obtained by rewriting Algorithm 10 while generalizing it for XPMaps.

We restate, in this general setting, the preconditions that ensure the modified algorithm will terminate: $(c1')$ p is not in $\mathcal{B}$, $(c2')$ p is not in the projection of any box of V_∂ and $(c3')$ $\forall e_k \in E$, p is not in the projection of any box of $approx(e_k, \delta)$.

Conditions $(c2')$ and $(c3')$ can be satisfied by refining boxes of $\mathrm{V}^x \cup \mathrm{V}^n$ and δ-approximations of elements of E, provided that $(c1')$ holds. Checking $(c1')$ with a fully numeric method is challenging since it goes back to the problem of deciding zero. In the special case where the coordinates of p are rational numbers and P, Q are polynomials, this can be addressed with symbolic computation, by computing $g(z) = gcd(P(x^p, y^p, z), Q(x^p, y^p, z))$. If $deg_z(g) \geq 1$ then $p \in \mathcal{B}$, in particular if $deg_z(g) = 2$ and g has two distinct roots, p is a node of $\mathcal{B}$ and if g has a double root, p is a cusp of $\mathcal{B}$. Otherwise $deg_z(g) = 0$ and $p \notin \mathcal{B}$.

6 Implementation and Results

We implemented the method presented in this paper and tested it to compute the topology of apparent contours of algebraic surfaces of degrees up to 15. We briefly describe this implementation, then we present the experiments we carried out and compare our approach with state-of-the-art methods. Let us recall the main steps that are performed to compute the topology of an apparent contour:

(1) Computing the set C^x of x-critical points by calling `IsolateSols` on S_5.

(2) Computing the set C^b of boundary points by calling `IsolateSols` on S_1, S_2, S_3, S_4.

(3) Computing a δ-approximation $(\boldsymbol{C}_i)_{i=1}^m$ of $\mathcal{C}$ with Algorithm 3.

(4) Computing the sets D^n of nodes and D^c of cusps. This is performed by solving the ball system S_b by calling `IsolateSols(`S_b`,.,0)` for each $\boldsymbol{D}_i$ and each $\boldsymbol{D}_{ij}$ as defined in Proposition 3.1. We denote by $m_\cap$ the number of non-empty $\boldsymbol{D}_{ij}$. Then nodes and cusps are distinguished with Algorithm 4.

(5) Computing the set C^y of y-critical points with `IsolateSols(`$\{P, Q, R'\}$`,`$\boldsymbol{C}_i$`,0)` for each $1 \leq i \leq m$, then separating the special points (*i.e* smooth critical points, nodes and cusps of $\mathcal{B}$) in the projection, as described in Section 4.2.

(6) Computing the sets B^x and B^n of witness boxes for x-critical (possibly cusp) points and nodes of $\mathcal{B}$.

(7) Computing the graph G_∂: the projections of connections of x-critical points and nodes of $\mathcal{B}$ are made disjoint, then a δ-approximation for each connected components of $\mathcal{C} \cap C_0$ is computed with calls to `Track` while ensuring that conditions $(a1)$, $(a2)$ and $(a3)$ described in Section 5.1 are satisfied. Let us call m' the sum of the number of boxes of each such δ-approximation.

(8) Embedding the graph with Algorithm 12.

Note that all these steps terminate when the assumptions $(A_1), \ldots, (A_{10})$ are satisfied. Moreover, the assumptions are generically satisfied and the algorithm terminates if and only if the assumptions are satisfied. Indeed, if (A_1), (A_2) is not satisfied then the tracking steps in (3) and (7) will not terminate. If (A_6) or (A_9) is not satisfied then

we have several points that project on the same point and step 6 will not terminate, since it cannot separate them. Finally, if (A_3), (A_4), (A_5), (A_7), (A_8) or (A_{10}) is not satisfied, then some of the steps using `IsolateSols` will not terminate, since they will be applied to a system with singular solutions.

6.1 Our implementation

The two cornerstones of our implementation are the procedure `IsolateSols` specified in Algorithm 1 and the procedure `Track` specified in Algorithm 3 and described in Appendix A.

Our implementation of the procedure `IsolateSols` with subdivision, centered-form interval evaluation and Krawczyk operator is exhaustively described in [11]. It uses adaptive multi-precision arithmetic. It is available in the mathematical software `SageMath`[4] as the package `subdivision_solver`[5]. Solving a system of polynomial equations in an unbounded box reduces to solving several transformed systems in bounded boxes (see. [25, Section 5.6] or [28, Section 5.10]). In particular, we don't need to know in advance the bounded interval $\boldsymbol{z}_0$ for the z-component. Solving a system on a box of the shape $\mathbf{x}\times\mathbf{y}\times\mathbb{R}$ reduces to solving two systems on the bounded boxes $\mathbf{x}\times\mathbf{y}\times[-1,1]$ and hence requires two calls to the solver provided by `subdivision_solver`. Notice that a δ-approximation of $\mathcal{C}\cap\boldsymbol{C}_0$ yields bounds for the z-coordinates of $\mathcal{C}\cap\boldsymbol{C}_0$. Once Step (3) is performed, we use these bounds to avoid the second call in each algorithm involving solving a system in a box with unbounded z-component.

The procedure `Track` is described in detail in Appendix A. In this description, we assume that computations are carried out with arbitrary precision. We implemented this procedure in `Python` within `SageMath` only for machine precision; our tracker stops when the width of the boxes of the enclosure reaches the machine precision. However, we never encountered this case in the experiments reported below. In Steps (3) and (7), the initial value $\delta = 1$ is used as input for `Track`. In step (7), the conditions $(a2)$ and $(a3)$ are enforced during the tracking process. The condition $(a1)$ is checked *a posteriori* on the δ-approximations. Each time the latter condition does not hold, *i.e.* each time two boxes $\boldsymbol{C}_i^e$ and $\boldsymbol{C}_j^{e'}$ of δ-approximations of edges e and e' have a non-empty intersection, the δ-approximations of e and e' are refined within $\boldsymbol{C}_i^e$ and $\boldsymbol{C}_j^{e'}$ while enforcing $(a2)$ and $(a3)$ until $(a1)$ holds.

The other algorithms used by our approach have been implemented in `Python` within `SageMath`.

6.2 Experimental data

The surfaces are defined by random dense polynomials P in $\mathbb{Z}[x,y,z]$ with odd total degrees from 5 to 15 and integer coefficients chosen uniformly in $[\![-2^8, 2^8]\!]$. We isolate the singularities and compute the topology of the apparent contour of the surface defined by P with two state-of-the-art methods and the approach described here. For each degree, five instances are considered, and we give averages of sequential times in seconds and standard deviations for each method.

[4] `http://www.sagemath.org/`
[5] `http://subdiv-solver.gforge.inria.fr`

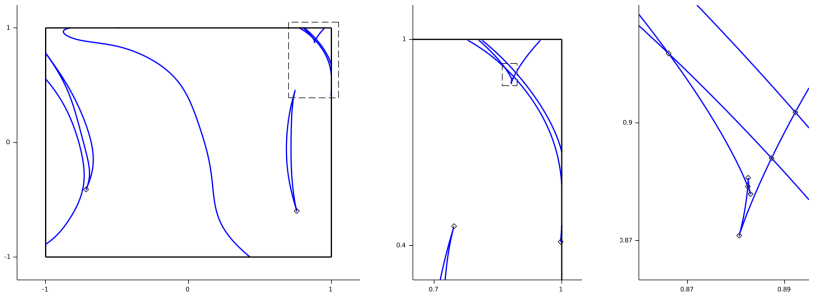


Figure 9: Left: the apparent contour of an algebraic surface of degree 13. Center (resp. Right): a detailed view of the same curve in the dashed box of the leftmost (resp. central) part. Cusps and nodes lying outside zooming areas are marked with squares.

6.3 State-of-the-art methods

In [13], the singularities of an apparent contour are characterized as the real solutions of a system of two polynomials that are coefficients of the sub-resultant chain. This system can be solved with `subdivision_solver` that has been designed as a solver dedicated to large dense polynomials. We use this approach to isolate the singularities in $\mathbf{B}_0 = [-1, 1] \times [-1, 1]$.

The package `Isotop`[6] for `Maple` computes the topology of a plane curve in $\mathbb{R}^2$, see [4]. Here, `Isotop` is used to compute the topology of the resultant of P and P_z with respect to z.

6.4 Results

We report results obtained with these three methods for isolating the singularities and computing the topology of the apparent contour of algebraic surfaces. We first give details for one of the surfaces of degree 13 we tested before giving synthetic data for surfaces of degrees from 5 to 15. Running times given below are sequential running times in seconds on an Intel(R) Core(TM) i5-3317U CPU @ 1.70GHz machine with Linux.

Details for a surface of degree 13. We detail the computation of the topology of the apparent contour of one of the surfaces of degree 13 we considered. Figure 9 displays its apparent contour and Table 1 details each step of the computation. Columns t give the running times. For Step (6), the column i gives the maximum number of times a box isolating a singularity or an x-critical point is contracted before it can be certified as a witness box. The symbol * signifies that multi-precision was required to address this step.

[6] `http://vegas.loria.fr/isotop/`

	Isolating singularities							
Step	(1)		(2)		(3)		(4)	
d	t	$\lvert C^x \rvert$	t	$\lvert C^b \rvert$	t	m, $m_\cap$	t	$\lvert D^n \rvert$, $\lvert D^c \rvert$
13	13.7	14	3.44	16	9.28	1012, 2478	7.87	4, 7

	Computing topology							
Step	(5)		(6)		(7)		(8)	
d	t	$\lvert C^y \rvert$	t	i	t	m'	t	n'
13	1.57	8	33*	5	23.4	1940	0.34	2

Table 1: Computation of the apparent contour of an algebraic surface of degree 13.

	Isolating singularities				Computing topology		
method	Sub-resultant approach		Our approach		`Isotop`	Our approach	
domain	$\mathbf{B}_0$		$\mathbf{B}_0$		$\mathbb{R}^2$	$\mathbf{B}_0$	
d	t $\pm$ σ	n	t $\pm$ σ	n	t $\pm$ σ	t $\pm$ σ	n
5	0.06 $\pm$ 0.04	0	1.60 $\pm$ 0.46	0	4.78 $\pm$ 0.26	3.17 $\pm$ 1.01	0
7	3.03 $\pm$ 3.30	0	3.73 $\pm$ 0.40	0	251 $\pm$ 16.1	8.13 $\pm$ 1.86	0
9	304 $\pm$ 478	1	10.2 $\pm$ 5.45	0	–	24.1 $\pm$ 16.2	0
11	>3600	4	36.0 $\pm$ 8.53	0	–	75.5 $\pm$ 13.5	0
13	–		43.5 $\pm$ 18.1	0	–	90.6 $\pm$ 37.9	1
15	–		97.9 $\pm$ 47.1	0	–	169 $\pm$ 71.0	0

Table 2: Sequential running times t in seconds (averaged over five runs), standard deviations σ, number n of runs requiring multi-precision for isolating the singularities and computing the topology of apparent contours of algebraic surfaces of degree d. $\mathbf{B}_0$ is $[-1,1] \times [-1,1]$ and $-$ means that the process has not been run.

Table 2. The first group of columns reports on the isolation of the singularities of the apparent contour in $\mathbf{B}_0$ with the approach using the sub-resultant system [13] and Steps (1) to (4) of our approach. The former approach suffers from the size in term of degree, number of monomials and bit-size of the coefficients of the equations of the sub-resultant system. For instance, the first equation of this system for the first polynomial of degree 9 we tested has degree 57, more than 1700 monomials and its coefficient bit-size is more than 130. Furthermore, the running times have a high standard deviation, and machine precision was not sufficient to carry out the computations for examples of high degree.

The group of columns "Computing topology" refers to the computation of the topology of the apparent contour, including the isolation of singularities. The column `Isotop` reports the running times of `Isotop` applied to the resultant of P and P_z with respect to z. As expected, the size of the resultant polynomial excludes this approach for surfaces of high degree. We tried `Isotop` for d up to 8. For $d = 8$, the running time was 1924 seconds.

In contrast, our method does not consider any resultant or sub-resultant polynomial; hence we deal with systems having almost the same degree and bit size as the input. On the other hand, we have up to four variables instead of two. The machine precision was sufficient to isolate the singularities of all examples, and only one example, the one detailed above, required use of more precision for the computation of the topology. Singularities are characterized here as the solutions of the ball system that involves four equations in four unknowns. The results in [12] already showed the advantage of our subdivision solver over the symbolic approach, but they also showed

the limitation of our approach when solving a four dimensional system on a large domain instead of a two dimensional one even with higher degree and bitsize. Here, the δ-approximation of $\mathcal{C}$ enables us to filter the domain where the ball system is solved, and the results of Table 2 show the efficiency of this strategy.

Table 3. This table details how the times reported in Table 2 are distributed among the main steps that our method performs.

	Isolating singularities				Computing topology				total
Step	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	
d	t	t	t	t	t	t	t	t	t
5	0.14	0.36	1.03	0.06	0.02	0.31	1.22	0.00	3.17
7	0.64	0.70	2.21	0.17	0.05	1.47	2.84	0.02	8.13
9	2.85	1.41	4.66	1.27	0.24	5.78	7.91	0.02	24.1
11	8.08	2.12	16.4	9.44	0.90	10.8	27.6	0.08	75.5
13	15.6	3.36	16.3	8.14	1.22	17.6*	28.1	0.06	90.6
15	30.5	4.87	29.1	33.3	2.12	24.7	45.2	0.00	169

Table 3: Distribution of times given in Table 2 between the main steps of our approach. (1) x-critical points; (2) boundary points; (3) δ-approximation of $\mathcal{C}$; (4) solving the ball system; (5) y-critical points; (6) witness boxes singularities; (7) connecting special points; (8) computing the XPMap; Symbol *: for one example, multi-precision was required.

For the computation of the singularities in Steps (1) to (4): Step (2) solves systems in 2 dimensions and is the least time consuming; Step (1) solves systems in 3 dimensions in the large domain $\mathbf{B}_0 \times \mathbb{R}$ which is more time consuming; Step (4) solves systems in 4 dimensions in $m + m_\cap$ small boxes. For the particular case of degree 13 detailed above, $m + m_\cap = 3490$. Comparing with the results of [12] that performed isolation on a unique large domain, the benefit the δ-approximation to reduce the solution domain is dramatic.

In Step (5), the y-critical points are found by solving a system in 3 dimensions in the m boxes of the δ-approximation of $\mathcal{C}$. Comparing the running times to address this step with the ones required for Step (1) illustrates again how using the δ-approximation of $\mathcal{C}$ speeds up the computation.

In Step (6), witness boxes for x-extreme points and singularities are computed. It is adressed by combining Algorithm 6, that contracts a box containing a node, and Algorithm 5, that checks if the resulting box is a witness box, as well as their equivalents for x-extreme points and cusps. In almost all cases we tried, Algorithm 5 succeeded with one iteration of Algorithm 6. The time for Step (6) approximately corresponds to $|\mathrm{C}^x|+|\mathrm{D}^n|$ times half the time for Step (2), except when i is high (see details given in Appendix B). It appeared in our experiments that most of the time spent in the procedure associated with Algorithm 5 is used to construct the systems defining the boundary points (that requires partial substitutions in large polynomials) rather than to solve them. For the example of degree 13 detailed above, approximately 95% of the time required by the latter procedure is spent defining these systems. For this example, multi-precision arithmetic was required to carry out Step (6). Determining a witness box for an x-extreme point for this particular example required 5 iterations in the equivalent of Algorithm 6 for x-extreme points before the equivalent of Algorithm 5 for x-extreme points certifies it to witness the x-extreme point. It should be considered as a proof of robustness of our implementation more than as a drawback

of the approach. Similarly to Step (2), Algorithm 5 involves isolating the intersections of $\mathcal{C}$ on the boundaries of a cylinder. Step (2) requires two calls to the numerical 0-dimensional solver whereas in Algorithm 5 we use the bounds for the z-component given by the δ-approximation of $\mathcal{C} \cap \boldsymbol{C}_0$. Hence Algorithm 5 requires roughly half the time of Step (2) to execute.

Among Steps (1) to (8), Step (7) is the most time consuming. It consists of two substeps. First, δ-approximations of connected components of $\mathcal{C} \cap C_0$ are computed while ensuring that conditions $(a2)$ and $(a3)$ hold. This results in approximations with smaller boxes than in Step (3); see the details in Appendix B. Then, the condition $(a1)$ is checked for each pair of δ-approximation. In most examples we tested, $(a1)$ was satisfied without refining the approximations. For the example of degree 13 detailed above, computing the approximations required 18.8s and checking $(a1)$ required 4.6s.

Most of the apparent contours we computed have only one connected component; this explains the running times for Step (8). See also column n' in the table in Appendix B.

7 Conclusion

In this article, we provided an algorithm computing the XPMap of the projection of a space curve. The space curve is defined by the intersection of two surfaces represented implicitly by convergent interval functions. Compared to the state of the art, our method is the first to solve this problem reliably for functions that are not polynomial. Moreover, in the polynomial case, our experiments show that our method handles polynomials of degree up to 15 that are not yet reachable by sub-resultant approaches. As future work, we plan to generalize our method to the projections of curves from $\mathbb{R}^n$ to $\mathbb{R}^2$. In this case, we would need to change the systems modelling the node and cusp singularities, but the tracking of the curve extends easily. We would also like to address the problem of computing the projection of 2-manifolds from $\mathbb{R}^4$ to $\mathbb{R}^3$.

References

[1] V. I. Arnold, A. Varchenko, and S. M. Gusein-Zade. *Singularities of Differentiable Maps: Volume I: The Classification of Critical Points Caustics and Wave Fronts*, volume 82. Springer Science & Business Media, 1988.

[2] Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler. Bertini: Software for numerical algebraic geometry. Available at bertini.nd.edu with permanent doi: dx.doi.org/10.7274/R0H41PB5, 2013.

[3] M. Burr, S. W. Choi, B. Galehouse, and C. K. Yap. Complete subdivision algorithms ii: Isotopic meshing of singular algebraic curves. *Journal of Symbolic Computation*, 47(2):131 – 152, 2012.

[4] J. Cheng, S. Lazard, L. Peñaranda, M. Pouget, F. Rouillier, and E. Tsigaridas. On the topology of real algebraic plane curves. *Mathematics in Computer Science*, 4:113–137, 2010.

[5] R. M. Corless, G. M. Diaz-Toca, M. Fioravanti, L. Gonzalez-Vega, I. F. Rua, and A. Shakoori. Computing the topology of a real algebraic plane curve whose defining equations are available only "by values". *Comput. Aided Geom. Design*, 30(7):675–706, 2013.

[6] J.P. Dedieu. *Points fixes, zéros et la méthode de Newton.* Mathématiques et Applications. Springer, 2006.

[7] N. Delanoue and S. Lagrange. A numerical approach to compute the topology of the apparent contour of a smooth mapping from R^2 to R^2. *Journal of Computational and Applied Mathematics*, 271:267–284, 2014.

[8] M. Demazure. *Bifurcations and catastrophes: geometry of solutions to nonlinear problems.* Universitext. Springer, Berlin, New York, 2000. École polytechnique.

[9] Dominique Faudot and Dominique Michelucci. A new robust algorithm to trace curves. *Reliable Computing*, 13(4):309–324, 2007.

[10] H. Hong. An efficient method for analyzing the topology of plane real algebraic curves. *Mathematics and Computers in Simulation*, 42(4–6):571–582, 1996.

[11] Rémi Imbach. A Subdivision Solver for Systems of Large Dense Polynomials. Technical Report 476, INRIA Nancy, March 2016.

[12] Rémi Imbach, Guillaume Moroz, and Marc Pouget. Numeric and certified isolation of the singularities of the projection of a smooth space curve. In *Proceedings of the 6th International Conferences on Mathematical Aspects of Computer and Information Sciences*, MACIS, 2015.

[13] Rémi Imbach, Guillaume Moroz, and Marc Pouget. A certified numerical algorithm for the topology of resultant and discriminant curves. *Journal of Symbolic Computation*, 80, Part 2:285 – 306, 2017.

[14] R. B. Kearfott. *Rigorous global search: continuous problems.* Nonconvex optimization and its applications. Kluwer Academic Publishers, Dordrecht, Boston, 1996.

[15] R.B. Kearfott and Z. Xing. An interval step control for continuation methods. *SIAM Journal on Numerical Analysis*, 31(3):892–914, 1994.

[16] Ullrich Köthe. Xpmaps and topological segmentation-a unified approach to finite topologies in the plane. In *International Conference on Discrete Geometry for Computer Imagery*, pages 22–33. Springer, 2002.

[17] R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing (Arch. Elektron. Rechnen)*, 4:187–201, 1969.

[18] Sergei K Lando and Alexander K Zvonkin. *Graphs on surfaces and their applications*, volume 141. Springer Science & Business Media, 2013.

[19] A. Leykin, J. Verschelde, and A. Zhao. Newton's method with deflation for isolated singularities of polynomial systems. *Theoretical Computer Science*, 359(13):111 – 122, 2006.

[20] Y. Lu, D. J. Bates, A. J. Sommese, and C. W. Wampler. Finding all real points of a complex curve. In *Algebra, geometry and their interactions*, volume 448 of *Contemp. Math.*, pages 183–205. Amer. Math. Soc., Providence, RI, 2007.

[21] B. Martin, A. Goldsztejn, L. Granvilliers, and C. Jermann. Certified parallelotope continuation for one-manifolds. *SIAM J. Numerical Analysis*, 51(6):3373–3401, 2013.

[22] Jean-Pierre Merlet. *Parallel Robots*, volume 74 of *Solid Mechanics and its Applications.* Kluwer Academic Publishers, Boston, 2000. INRIA.

[23] Ramon E Moore, R Baker Kearfott, and Michael J Cloud. *Introduction to interval analysis.* SIAM, 2009.

[24] B. Mourrain, S. Pion, S. Schmitt, J.-P. Técourt, E. P. Tsigaridas, and N. Wolpert. Algebraic issues in Computational Geometry. In J.-D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, Mathematics and Visualization, chapter 3, pages 117–155. Springer, 2006.

[25] A. Neumaier. *Interval methods for systems of equations.* Cambridge University Press, 1990.

[26] T. Ojika, S. Watanabe, and T. Mitsui. Deflation algorithm for the multiple roots of a system of nonlinear equations. *Journal of Mathematical Analysis and Applications*, 96(2):463–479, 1983.

[27] S. Plantinga and G. Vegter. Isotopic approximation of implicit curves and surfaces. In *SGP '04: Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 245–254, 2004.

[28] V. Stahl. *Interval Methods for Bounding the Range of Polynomials and Solving Systems of Nonlinear Equations.* PhD thesis, Johannes Kepler University, Linz, Austria, 1995.

[29] H. Whitney. On singularities of mappings of euclidean spaces. i. mappings of the plane into the plane. *Annals of Mathematics*, 62(3):pp. 374–410, 1955.

Algorithm 13 `isSolInSet(`$(\boldsymbol{C}, S)$`,`C`,`ϵ`)`

Input: An implicit point $(\boldsymbol{C}, S)$ defining a point p, a set C and a real number $\epsilon > 1$.
Output: **true** if $p \in C$, **false** if $p \notin \epsilon C$.

```
1: C' = εC
2: while not ( (C ⊂ i(C')) or (C ∩ C = ∅) ) do
3:     C = K_S(C)
4: if C ⊂ i(C') then return true
5: return false
```

A Implementation of Algorithm 2

Several certified numerical path-tracking algorithms can be found in the literature [9, 15, 21]. They approximate a smooth connected curve in a bounded box with a sequence of sets $(C_i)_{i=1}^{m_k}$. The sets C_i are in general boxes as in [9, 15]. It has recently been proposed to use *parallelotopes* instead of boxes [21]; a parallelotope C is the image by an affine transformation of a box $\boldsymbol{C}$. Aligning parallelotopes along the tangent to the curve yields a more efficient approximation.

For the sake of generality, we consider here approximating sets C_i that are either boxes or parallelotopes. If C is such a set, $\mathbf{h}(C)$ denotes its box hull, that is the smallest box containing C. The operators ∂, $i()$ are directly generalized for parallelotopes. Let C be a parallelotope and $\boldsymbol{C}, f$ be the box and the affine map such that $C = f(\boldsymbol{C})$. The width $\mathrm{w}(C)$ of C is the width of $\boldsymbol{C}$ and if $\epsilon \in \mathbb{R}$, the ϵ-inflation ϵC is defined as $f(\epsilon \boldsymbol{C})$.

Section A.1 introduces Algorithm 13 using ϵ-inflation to decide if an implicit point lies in a set C. Section A.2 recalls the parallelotope path-tracking algorithm of [21], and Section A.3 shows how it is adapted to meet the specifications of Algorithm 2.

A.1 Deciding if an implicit point lies in a set of the approximation

Let $(\boldsymbol{C}, S)$ define implicitly a point p. To check if p belongs to a set C, a naive approach consists of contracting $\boldsymbol{C}$ on p until $\boldsymbol{C} \subset i(C)$ or $\boldsymbol{C} \cap C = \emptyset$. If p lies on ∂C, this leads to a non-terminating process. To tackle this pitfall, we use the ϵ-inflation approach described in Algorithm 13, that takes as input C and a real number $\epsilon > 1$ and returns **true** if $p \in C$ and **false** if $p \notin \epsilon C$. Notice that when $p \in \epsilon C \setminus C$, it may return either **true** or **false**.

Proof:[Proof of termination and correctness of Algorithm 13] Assume $\epsilon > 1$ and let C' be ϵC. After applying Step 3 of Algorithm 13 a finite number of times, $\boldsymbol{C}$ is strictly included in a ball of diameter $\frac{\epsilon-1}{2}\mathrm{w}(C)$. Hence $\boldsymbol{C}$ can not intersect both $\partial C'$ and ∂C; thus either $\boldsymbol{C} \subset i(C')$ or $\boldsymbol{C} \cap C = \emptyset$, and the while loop of Algorithm 13 terminates, so does Algorithm 13. □

Remark A.1 *Let $(\boldsymbol{C}, S)$ be a point of $\mathcal{C}$, and C and $\epsilon > 1$ be such that both $\mathcal{C} \cap C$ and $\mathcal{C} \cap \epsilon C$ are diffeomorphic to $[0, 1]$. If `isSolInSet(`$(\boldsymbol{C}, S)$`,`C`,`ϵ`)` returns true, then $(\boldsymbol{C}, S)$ defines a point on $\mathcal{C}^k \cap i(\epsilon C)$, where $\mathcal{C}^k$ is such that $\mathcal{C} \cap C = \mathcal{C}^k \cap C$.*

Proof:[Proof of Remark A.1] Suppose `isSolInSet(`$(\boldsymbol{C}, S)$`,`C`,`ϵ`)` returns true, and $(\boldsymbol{C}, S)$ defines a point on a connected component $\mathcal{C}^{k'} \neq \mathcal{C}^k$ of $\mathcal{C}$ in ϵC. Then $\mathcal{C} \cap \epsilon C$ contains two connected components of $\mathcal{C}$, leading to a contradiction. □

A.2 The path-tracking algorithm of [21]

The authors of [21] propose a parameterized version of the Krawczyk operator to certify that in a given set, in practice a parallelotope, a curve is diffeomorphic to $[0,1]$. This operator is the cornerstone of an algorithm that uses a classical adaptive step-length control and constructs iteratively a certified approximation of a connected component of a curve within a bounded initial domain. We state here the specifications of this algorithm in our context of a curve of $\mathbb{R}^3$ defined by two polynomials P and Q.

Input: A system $P = Q = 0$ defining $\mathcal{C}$, an initial bounded box $\boldsymbol{C}_0$, an initial point $p_0 \in \boldsymbol{C}_0$, two strictly positive real numbers α, β, an integer $d \in \{-1, 1\}$.
Output: A flag in {**success**, **failure**} and a sequence C^{enc}. If the flag is **failure**, C^{enc} is empty. Otherwise, C^{enc} is a sequence of sets $(C_i)_{i=1}^m$ with $m > 1$ such that $\mathcal{C} \cap C_i$ is diffeomorphic to $[0,1]$ for each i, $C_i \subset i(\boldsymbol{C}_0)$ for $i \notin \{1, m\}$ and $\mathcal{C} \cap \bigcup_{i=1}^m C_i$ is diffeomorphic to a one-dimensional manifold. Furthermore, one has either $C_1 \cap C_m \neq \emptyset$ and in this case $\mathcal{C} \cap \bigcup_{i=1}^m C_i$ is diffeomorphic to a circle, or $C_1 \cap \partial \boldsymbol{C}_0 \neq \emptyset$ or $C_m \cap \partial \boldsymbol{C}_0 \neq \emptyset$ and in this case $\mathcal{C} \cap \bigcup_{i=1}^m C_i$ is diffeomorphic $[0,1]$.

The initial point p_0 is used to construct an initial parallelotope C_0, and this step succeeds if p_0 is close enough to $\mathcal{C}$. Otherwise the flag **failure** is returned. Then, d determines the direction in which $\mathcal{C}$ is followed. At each step, a new parallelotope C_i with a step-length γ is constructed, with $\alpha < \gamma < \beta$. Several properties are checked on C_i that guarantee the correctness of the algorithm, and γ is decreased until either $\gamma \leq \alpha$ or C_i satisfies the latter properties. When $\gamma \leq \alpha$, the algorithm stops and returns the flag **failure**. In case of success, γ is increased until it reaches β.

When $\alpha > 0$, the algorithm terminates. When $\alpha = 0$, it terminates with the flag **success**, provided that $\mathcal{C}$ is smooth in $\boldsymbol{C}_0$ and p_0 is sufficiently close to $\mathcal{C}$.

A.3 Meeting the specifications of Algorithm 2

We now show how to modify the algorithm described above to meet the specifications of the procedure `Track` described in Algorithm 2. These specifications are reproduced here for the sake of readability.

Input: A system $P = Q = 0$ defining a smooth curve $\mathcal{C}$, a domain C_0, an implicit point $(\boldsymbol{C}^0, S^0)$ of $\mathcal{C}$, a finite set $\{(\boldsymbol{C}^j, S^j)\}_j$ of implicit points containing the boundary points of $\mathcal{C}$ in C_0 and $(\boldsymbol{C}^0, S^0)$, $\delta > 0$.
Output: A δ-approximation of the connected component of $\mathcal{C} \cap C_0$ containing $(\boldsymbol{C}^0, S^0)$ and the set C^{on} of implicit points of $\{(\boldsymbol{C}^j, S^j)\}_j$ that are on the same connected component than $(\boldsymbol{C}^0, S^0)$.

First, the procedure `Track` takes as input a domain C_0 instead of a box $\boldsymbol{C}_0$; this leads in the algorithm to testing inclusion of parallelotopes within a domain rather than within a box. Recall that in our case the domain is possibly unbounded in z. We also fix α to 0. Provided (A_2) and (A_3) are satisfied ($\mathcal{C}$ is smooth and bounded above $\mathbf{B}_0$), this modified algorithm terminates with flag **success** if p_0 is sufficiently close to $\mathcal{C}$.

Then, the procedure `Track` takes as input a finite set $\{(\boldsymbol{C}^j, S^j)\}_j$ and returns a set C^{on} of implicit points of $\{(\boldsymbol{C}^j, S^j)\}_j$ that are on the approximated connected component. To achieve this goal, we modify the algorithm so it constructs a pair (C_i, ϵ_i) with $\epsilon_i > 1$ such that both $\mathcal{C} \cap C_i$ and $\mathcal{C} \cap \epsilon_i C_i$ are diffeomorphic to $[0,1]$. According to Remark A.1, if the procedure `isSolInSet(`$(\boldsymbol{C}^j, S^j)$`,`C_i`,`ϵ_i`)` returns **true**, it is a guarantee that $(\boldsymbol{C}^j, S^j)$ defines a point on $\mathcal{C}$ in $\epsilon_i C_i$. We also require that at

Algorithm 14 `TrackFromPoint(`$< P, Q >$`,`C_0`,`p_0`,`$\{(\boldsymbol{C}^j, S^j)\}_j$`,`$\delta$`)`

Input: A system $P = Q = 0$ defining $\mathcal{C}$, an initial domain C_0, an initial point $p_0 \in i(\boldsymbol{C_0})$, a finite set $\{(\boldsymbol{C}^j, S^j)\}_j$ of implicit points containing the boundary points of $\mathcal{C}$ in C_0, $\delta > 0$.

Output: A flag in {**success**, **failure**}, a sequence C^{enc} and a set C^{on}. If the flag is **failure**, C^{enc} and C^{on} are empty. Otherwise, C^{enc} is the sequence of sets $(C_i)_{i=1}^m$ with $m > 1$ such that $(\mathbf{h}(C_i))_{i=1}^m$ is a δ-approximation of a connected component $\mathcal{C}^k$ of $\mathcal{C} \cap C_0$ and C^{on} contains implicit points of $\{(\boldsymbol{C}^j, S^j)\}_j$ that are on $\mathcal{C}^k$.

most one point of $\{(\boldsymbol{C}^j, S^j)\}_j$ is in each C_i; this can be tested in the same way. The latter property guarantees in particular that each C_i intersecting the boundary of the domain contains no more than one boundary point and that the extremities of the approximated connected component of $\mathcal{C} \cap C_0$ are properly identified and reported in C^{on}.

Finally, the procedure `Track` takes as input $\delta > 0$ and returns a δ-approximation of a connected component. To achieve this, the condition $\mathrm{w}\,(\mathbf{h}(C_i)) < \delta$ is enforced at each iteration, and this replaces the condition $\gamma \leq \beta$. Then, when the algorithm described in Section A.2 with $d = 1$ as input terminates with the flag **success** and if it is detected that the tracked component is diffeomorphic to $[0, 1]$, another call with $d = -1$ allows approximation of the whole connected component.

These modifications give rise to the procedure `TrackFromPoint(,,,,)` specified in Algorithm 14. When assuming (A_2) and (A_3), this algorithm terminates with the flag **success** if p_0 is sufficiently close to $\mathcal{C}$.

In contrast to the latter procedure, `Track` takes as input an implicit point $(\boldsymbol{C}^0, S^0)$ on $\mathcal{C}$ rather than a point p_0 and guarantees that the returned δ-approximation is a δ-approximation of the connected component containing $(\boldsymbol{C}^0, S^0)$. One can use $p_0 = \mathrm{m}\big(\boldsymbol{C}^0\big)$ and call the procedure `TrackFromPoint(`$< P, Q >$`,`C_0`,`p_0`,`$\{(\boldsymbol{C}^j, S^j)\}_j$`,`$\delta$`)`. When the returned flag is **success**, the obtained δ-approximation is not necessarily the one of the connected component containing $(\boldsymbol{C}^0, S^0)$, in particular when two connected components are close to $\mathrm{m}\big(\boldsymbol{C}^0\big)$. On the other hand, since $(\boldsymbol{C}^0, S^0) \in \{(\boldsymbol{C}^j, S^j)\}_j$, the returned δ-approximation is the one of the connected component containing $(\boldsymbol{C}^0, S^0)$ if and only if $(\boldsymbol{C}^0, S^0) \in \mathrm{C}^{\mathrm{on}}$. If it is not the case, $\boldsymbol{C}^0$ is contracted with $K_{S^0}(\boldsymbol{C}^0)$ and `TrackFromPoint(`$< P, Q >$`,`C_0`,`$\mathrm{m}\big(\boldsymbol{C}^0\big)$`,`$\{(\boldsymbol{C}^j, S^j)\}_j$`,`$\delta$`)` is called again. This is performed until the returned flag is **success** and $(\boldsymbol{C}^0, S^0) \in \mathrm{C}^{\mathrm{on}}$. The termination of this recursion in ensured, since $\mathrm{m}\big(\boldsymbol{C}^0\big)$ becomes arbitrarily close to $\mathcal{C}$ and the initial set of the approximation will contain the implicit point $(\boldsymbol{C}^0, S^0)$.

Thus, after a finite number of iterations, `TrackFromPoint(,,,,)` returns the flag **success**, and $(\mathbf{h}(C_i))_{i=1}^m$ and C^{on} are a suitable output for `Track` with arguments $(< P, Q >, C_0, (\boldsymbol{C}^0, S^0), \{(\boldsymbol{C}^j, S^j)\}_j, \delta)$.

B Details for Results

	Isolating singularities							
Step	(1)		(2)		(3)		(4)	
d	t	$\lvert C^x\rvert$	t	$\lvert C^b\rvert$	t	m, $m_\cap$	t	$\lvert D^n\rvert$, $\lvert D^c\rvert$
5	0.13	4	0.36	4	0.59	89, 110	0.03	0, 1
5	0.14	1	0.36	4	0.99	139, 177	0.06	0, 1
5	0.09	0	0.36	8	0.67	45, 51	0.01	0, 0
5	0.21	2	0.36	8	1.57	236, 651	0.17	0, 1
5	0.15	4	0.36	8	1.31	171, 168	0.04	0, 0
7	0.51	1	0.70	8	2.63	404, 415	0.16	0, 0
7	0.76	5	0.73	4	2.40	349, 455	0.23	3, 2
7	0.49	0	0.69	2	2.63	492, 524	0.21	0, 0
7	0.65	6	0.70	6	1.71	248, 248	0.08	0, 0
7	0.75	7	0.70	4	1.66	243, 390	0.19	0, 3
9	2.63	7	1.24	4	2.61	379, 439	0.65	2, 2
9	2.28	5	1.24	6	2.84	397, 519	1.73	2, 1
9	2.05	4	1.24	4	1.48	210, 323	0.34	0, 3
9	5.11	13	1.26	10	10.9	1623, 2336	3.28	5, 5
9	2.21	2	2.05	8	5.41	453, 449	0.34	0, 0
11	10.4	11	2.21	18	18.4	2482, 3995	7.39	4, 4
11	8.50	12	2.07	8	12.5	1721, 2700	5.23	7, 7
11	6.17	8	2.08	6	17.0	2543, 11269	22.7	1, 6
11	7.12	6	2.11	10	11.1	1523, 2443	4.16	2, 4
11	8.17	5	2.11	8	22.8	3297, 3841	7.68	0, 2
13	10.3	3	3.38	6	9.91	1144, 1227	2.45	1, 1
13	13.7	14	3.44	16	9.28	1012, 2478	7.87	4, 7
13	13.7	7	3.25	12	6.40	706, 1257	2.89	1, 3
13	19.5	11	3.29	10	25.7	3159, 5570	15.3	5, 6
13	20.9	5	3.42	10	30.5	3647, 4713	12.1	3, 3
15	21.3	4	4.98	4	9.19	968, 1543	5.27	1, 3
15	28.3	8	4.89	12	41.7	4439, 7798	108	5, 3
15	39.2	10	4.80	10	18.4	1986, 3116	15.9	5, 5
15	32.7	10	4.83	10	37.0	3754, 6380	25.5	0, 3
15	30.9	10	4.82	14	39.2	4095, 8461	11.6	0, 0

	Computing topology							
Step	(5)		(6)		(7)		(8)	
d	t	$\lvert C^y\rvert$	t	i	t	m'	t	n'
5	0.02	3	0.57	1	0.99	197	0.00	1
5	0.02	4	0.14	1	1.28	257	0.00	1
5	0.00	2	0.00	0	0.50	109	0.00	1
5	0.02	3	0.29	1	1.99	353	0.00	1
5	0.02	2	0.58	1	1.35	255	0.00	1
7	0.04	4	0.30	1	2.52	419	0.00	1
7	0.09	6	2.53	1	4.71	712	0.06	2
7	0.03	0	0.00	0	2.46	490	0.00	1
7	0.02	3	1.84	1	1.92	315	0.00	1
7	0.06	3	2.71	3	2.59	432	0.05	2
9	0.28	6	7.89	6	5.88	839	0.00	1
9	0.15	5	3.92	1	4.55	685	0.00	1
9	0.25	6	2.23	1	2.23	358	0.10	2
9	0.38	11	13.0	6	21.8	2400	0.00	1
9	0.16	1	1.82	1	5.06	472	0.00	1
11	0.96	9	14.4	1	31.6	3041	0.00	1
11	0.82	15	18.1	1	27.8	2743	0.20	2
11	1.26	10	9.11	1	31.2	3375	0.19	2
11	0.84	8	7.71	1	17.5	2026	0.00	1
11	0.64	5	4.84	1	29.7	3392	0.00	1
13	0.42	3	6.05	1	12.1	1378	0.00	1
13	1.57	8	33.6*	5	23.4	1940	0.34	2
13	1.26	12	12.0	1	11.1	1236	0.00	1
13	1.64	9	24.2	1	47.8	3969	0.00	1
13	1.23	6	12.0	1	46.2	3945	0.00	1
15	1.06	5	11.3	1	12.2	1197	0.00	1
15	2.91	13	29.7	1	71.9	5161	0.00	1
15	2.86	8	36.2	2	35.0	2799	0.00	1
15	2.22	6	23.5	1	52.3	4019	0.00	1
15	1.55	5	22.5	1	54.5	4166	0.00	1