NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

PhD THESIS

Changing representation of curves and surfaces: exact and approximate methods

Tatjana A. Kalinka

ATHENS

MARCH 2013

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Εναλλαγές αναπαράστασης καμπυλών και επιφανειών: εγγυημένες και προσεγγιστικές μέθοδοι

Τατιάνα Α. Καλίνκα

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2013

PhD THESIS

Changing representation of curves and surfaces: exact and approximate methods

Tatjana A. Kalinka

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Εναλλαγές αναπαράστασης καμπυλών και επιφανειών: εγγυημένες και προσεγγιστικές μέθοδοι

Τατιάνα Α. Καλίνκα

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Ιωάννης Εμίρης, Καθηγητής ΕΚΠΑ

ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:

Ιωάννης Εμίρης, Καθηγητής ΕΚΠΑ

Θεοχάρης Θεοχάρης, Καθηγητής ΕΚΠΑ

Παναγιώτης Κακλής, Καθηγητής ΕΜΠ

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Ιωάννης Εμίρης,
Καθηγητής ΕΚΠΑ

Παναγιώτης Κακλής,
Καθηγητής ΕΜΠ

Θεοχάρης Θεοχάρης,
Καθηγητής ΕΚΠΑ

Παναγιώτης Σακκαλής,
Καθηγητής ΓΠΑ

Νικόλαος Μισυρλής,
Καθηγητής ΕΚΠΑ

Bernard Mourrain,
Διευθυντής ερευνών,
INRIA Sophia-Antipolis

Tor Dokken,
Διευθυντής ερευνών Ινστιτούτου
SINTEF (Oslo)

Ημερομηνία εξέτασης 15/03/2012

ABSTRACT

The main object of study in our dissertation is the representation change of the geometric objects from the parametric form to implicit. We compute the implicit equation interpolating the unknown coefficients of the implicit polynomial given a superset of its monomials. The latter is derived from the Newton polytope of the implicit equation obtained by the recently developed method for support prediction. The support prediction method we use relies on sparse (or toric) elimination: the implicit polytope is obtained from the Newton polytope of the sparse resultant of the system in parametrization, represented as polynomials. The monomials that correspond to the lattice points of the Newton polytope are suitably evaluated to build a numeric matrix, ideally of corank 1. Its kernel contains their coefficients in the implicit equation. We compute kernel of the matrix either symbolically, or numerically, applying singular value decomposition (SVD). We propose techniques for handling the case of the multidimensional kernel space, caused by the predicted support being a superset of the actual. This yields an efficient, output-sensitive algorithm for computing the implicit equation. We compare different approaches for constructing the matrix in Maple and SAGE software. In our experiments we have used classical algebraic curves and surfaces as well as NURBS. Our method can be applied to polynomial or rational parametrizations of planar curves or (hyper)surfaces of any dimension including cases of parameterizations with base points which raise important issues for other implicitization methods. The method has its limits: geometric objects have to be presented using monomial basis; in the case of trigonometric parametrizations they have to be convertible to rational functions. Moreover, the proposed technique can be applied for non-geometric problems such as the computation of the discriminant of a multivariate polynomial or the resultant of a system of multivariate polynomials.

SUBJECT AREA: Computational Algebra

KEYWORDS: implicitization, interpolation, Newton polytope, sparse resultant, linear algebra.

ΠΕΡΙΛΗΨΗ

Το κύριο αντικείμενο μελέτης στην παρούσα διατριβή είναι η αλλαγή αναπαράστασης γεωμετρικών αντικειμένων από παραμετρική σε αλγεβρική (ή πεπλεγμένη) μορφή. Υπολογίζουμε την αλγεβρική εξίσωση παρεμβάλλοντας τους άγνωστους συντελεστές του πολυωνύμου δεδομένου ενός υπερσυνόλου των μονωνύμων του. Το τελευταίο υπολογίζεται απο το Newton πολύτοπο της αλγεβρικής εξίσωσης που υπολογίζεται από μια πρόσφατη μέθοδο πρόβλεψης του συνόλου στήριξης της εξίσωσης. Η μέθοδος πρόβλεψης του συνόλου στήριξης βασίζεται στην αραιή (ή τορική) απαλοιφή: το πολύτοπο υπολογίζεται από το Newton πολύτοπο της αραιής απαλοίφουσας αν θεωρίσουμε την παραμετροποίηση ως πολυωνυμικό σύστημα. Στα μονώνυμα που αντιστοιχούν στα ακέραια σημεία του Newton πολυτόπου δίνονται τιμές ώστε να σχηματίσουν έναν αριθμητικό πίνακα. Ο πυρήνα του πίνακα αυτού, διάστασης 1 σε ιδανική περίπτωση, περιέχει τους συντελεστές των μονωνύμων στην αλγεβρική εξίσωση. Υπολογίζουμε τον πυρήνα του πίνακα είτε συμβολικά είτε αριθμητικά εφαρμόζοντας την μέθοδο του singular value decomposition (SVD). Προτείνουμε τεχνικές για να διαχειριστούμε την περίπτωση ενός πολυδιάστατου πυρήνα το οποίο εμφανίζεται όταν το προβλεπόμενο σύνολο στήριξης είναι ένα υπερσύνολο του πραγματικού. Αυτό δίνει έναν αποτελεσματικό ευαίσθητο-εξόδου αλγόριθμο υπολογισμού της αλγεβρικής εξίσωσης. Συγκρίνουμε διαφορετικές προσεγγίσεις κατασκευής του πίνακα μέσω των λογισμικών Maple και SAGE. Στα πειράματά μας χρησιμοποιήθηκαν ρητές καμπύλες και επιφάνειες καθώς και NURBS. Η μέθοδός μας μπορεί να εφαρμοστεί σε πολυώνυμα ή ρητές παραμετροποιήσεις επίπεδων καμπυλών ή (υπερ)επιφανειών οποιασδήποτε διάστασης συμπεριλαμβανομένων και των περιπτώσεων με παραμετροποίηση σε σημεία βάσης που εγείρουν σημαντικά ζητήματα για άλλες μεθόδους αλγεβρικοποίησης. Η μέθοδος έχει τον εξής περιορισμό: τα γεωμετρικά αντικείμενα πρέπει να αναπαριστώνται από βάσεις μονωνύμων που στην περίπτωση τριγωνομετρικών παραμετροποιήσεων θα πρέπει να μπορούν να μετασχηματιστούν σε ρητές συναρτήσεις. Επιπλέον η τεχνική που προτείνουμε μπορεί να εφαρμοστεί σε μη γεωμετρικά προβλήματα όπως ο υπολογισμός της διακρίνουσας ενός πολυωνύμου με πολλές μεταβλητές ή της απαλοίφουσας ενός συστήματος πολυωνύμων με πολλές μεταβλητές.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Υπολογιστική Αλγεβρα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: αλγεβρικοποίηση, παρεμβολή, Newton πολύτοπο, αραιή απαλοίφουσα, γραμμική άλγεβρα.

ACKNOWLEDGEMENTS

First of all I would like to express my gratitude to my supervisor, Ioannis Emiris, for generously accepting me as an Early Stage Researcher in the SAGA program, for all he has taught me, for his insights and support during my work on the thesis.

Second, I would like to thank Rimvydas Krasauskas for inviting me to participate in the SAGA program in the first place, for his encouragement and advices.

I would like to express my gratitude to Tor Dokken, coordinator of the SAGA program for his support and care during my secondment in SINTEF.

I would like to thank my co-authors Christos Konaxis, Thang Luu Ba and Oliver Barrowclough for productive collaboration and permission to include some co-authored material as part of my dissertation.

I am deeply grateful to all the organizers and fellows of the SAGA program for the opportunity to enter and the honor to be part of the scientific community. I would like give my warm thanks my colleagues from the ΕρΓΑ laboratory for their enthusiasm, encouragement and sharing their knowledge with me. Special thanks to Alicia Dickenstein and Ron Goldman, whom I consider myself fortunate to meet, and whose pieces of advise have been priceless.

Last but not least I would like to thank for their contribution all the members of my examination committee: Bernard Mourrain, Nikolaos Missirlis, Theoharis Theoharis, Panagiotis Kaklis, Panagiotis Sakkalis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we give an overview of the popular mathematical representations of the geometrical objects, discuss the methods for the change of representation and describe several classic approaches to implicitization. We concentrate our attention on the implicitization by interpolation and the related previous works since the subject of our thesis belongs to this category.

1.1 Representation of geometric objects

All the different areas of computing that involve geometry, and today that include Computer Graphics, Computer-Aided Geometric Design, Scientific Visualization, Computational Geometry, Robotics, Computer Vision and Image Processing, have specific requirements for the mathematical representations of the geometric objects, depending on the nature of the objects and that operations that are performed on them. When the object is represented by the Solid Modeling techniques, an emphasis is being put on the physical fidelity, while the Conformal Geometric Algebra provides a convenient way to manipulate symbolic representations of geometric objects independently of their coordinates.

In Computer-Aided Geometric Design (CAGD) there are several popular ways to represent curves and surfaces, each has its advantages. The Table 1.1 lists some of the most important requirements for the surface representation in CAGD. It is common for the model to be an approximation of some degree: complex shapes are usually approximated by simpler ones, for instance, by a polygon mesh, moreover, using floating point arithmetics also sets a limit on accuracy. However, it is essential for the model to be as accurate as possible for a number of applications, be it high quality rendering or robotics. From the designer point of view, probably the most important is that the representation provides an effective mechanism to determine the shape of the surface interactively, i.e. enables quick visualization and gives an intuitive understanding of the controls over the surface shape. In many applications it is necessary to ensure smoothness of the constructed object, continuity of the blends between the primitives. Continuity is defined for both the implicit and parametric representations, although the former have been shown to have advantages when performing blending operations.

Parametric representations satisfy most of these requirements, see Table 1.1; their properties have made them dominant in computer graphics and modelling.

With parametrization it is easy to obtain points on the geometric object needed for various visualization techniques such as tracing, rendering, or surface fitting. Currently Bézier and non-uniform rational B-spline (NURBS) parametrizations are the most popular form of

Table 1.1: Features of the popular representations of surfaces

Property	Polygon Mesh	Implicit Surface	Parametric Surface	Subdivision Surface
Accurate	No	Yes	Yes	Yes
Intuitive control	No	No	Yes	No
Continuity	No	Yes	Yes	Yes
Visualization	Yes	No	Yes	Yes
Intersection	No	Yes	No	No

representation used in CAGD systems. NURBS have become one of the industry standards for representing geometric information. Bézier or NURBS curves and surfaces usually are presented in the form of piecewise splines or surface patches, such representation allows to understand intuitively and easily control the shape of the object.

Subdivision surfaces can be considered a generalization of NURBS: they are generated by repeated refinement of control meshes based on a certain refinement scheme. Applying the recursive refinement infinitely many times produces a smooth surface. The difference of this way of representation from the NURBS is that it does not allow for intuitive shape editing; instead it makes possible to represent surfaces of arbitrary topology.

Implicit curves and surfaces have an essential disadvantage: it is difficult to calculate points on curves and surfaces in a predictable way. On the other hand, CAGD systems rely heavily on the ability to check quickly if a given point is inside or outside a given object. Queries of this nature, that enable many operations important in modelling, such as blending curves and surfaces or Boolean operations on solids, have simple solution for implicitly defined objects.

The polygon mesh data structure is the oldest method for representing surfaces in computer graphics and widely established standard in geometry processing. In recent years it have become increasingly popular: polygon meshes find use in many different areas wherever digitization of complex 3D objects is required. In CAGD triangle and polygon meshes present an alternative to NURBS because of their conceptual simplicity, that allows for flexible and highly efficient processing.

1.1.1 Parametric representation

Parametric form of the planar rational curve is defined by two functions

$$x = \frac{a(t)}{c(t)},\ y = \frac{b(t)}{c(t)} \tag{1.1}$$

and parametric surfaces are defined by three functions

$$x = \frac{a(s,t)}{d(s,t)},\ y = \frac{b(s,t)}{d(s,t)},\ z = \frac{c(s,t)}{d(s,t)}. \tag{1.2}$$

Generalizing, we define the rational parametrization as

$$x_0 = \frac{f_0(t_1,\dots,t_n)}{g_0(t_1,\dots,t_n)},\ x_1 = \frac{f_1(t_1,\dots,t_n)}{g_1(t_1,\dots,t_n)},\ \dots,\ x_n = \frac{f_n(t_1,\dots,t_n)}{g_n(t_1,\dots,t_n)}. \tag{1.3}$$

where f_i and g_i, $i = 0, \ldots, n$ are polynomials in $t_1, \ldots, t_n$.

Typically, the functions a, b, c, d are presented in a particular basis, for instance, in the Bernstein for rational Bézier curves or B-spline basis for NURBS. Most of the algebraic methods, including representation change methods, use polynomials in the power (monomial) basis: $x(t) = c_0 + c_1 t + \ldots + c_n t^n$. In order to apply the algorithms, polynomials can be converted from Bernstein basis to power basis, although some algebraic methods such as resultants can be formulated using the Bernstein basis [65].

A parametric curve can be viewed as a map from a line to a curve in the (x, y)-plane, and parametric surface as a map from a plane with points (s, t) to a surface in (x, y, z)-space.

While many curves and surfaces admit a global parametrization, others can only be locally and piecewise parametrized [66].

Development of the Bézier curves and surfaces, made independently by De Casteljau and Bézier, was the major breakthrough in CAGD.

A polynomial Bézier curve is given by

$$x(t) = \sum_{i=0}^{n} x_i(t) B_i^n(t) \tag{1.4}$$

where the $t \in [0, 1]$ and B_i^n are Bernstein polynomials, defined by

$$B_i^n(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1 - t)^{n-i} \tag{1.5}$$

where the binomial coefficients are given by

$$\begin{pmatrix} n \\ i \end{pmatrix} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \leq i \leq n; \\ 0 & \text{otherwise.} \end{cases}$$

Bézier curves have certain limitations: in order to generate a curve of complex shape, Bézier representation have to be of a high degree. However, B-spline, which in essence is a piecewise polynomial Bézier curve of guaranteed continuity between the pieces allows to model such curves. Bézier curve. Another important generalization is *rational* Bézier curve: it has weights introduced that adjust the influence each control point has on the curve.

Non-uniform rational basis spline (NURBS) are generalizations of both B-splines and rational Bézier curves and surfaces, the primary difference being the weighting of the control points which makes NURBS curves rational.

Parametrizations, as mentioned before, have many features that made them the preferred representations in CAGD systems: they provide the flexibility to design a large variety of shapes, can be evaluated reasonably fast by numerically stable and accurate algorithms, are invariant under affine as well as perspective transformations, ensure continuity of the represented objects. The only drawback is that changing the topology of a parametric surface and the spatial queries on parametric curves or surfaces are very expensive computationally.

1.1.2 Implicit representation

An implicit algebraic curve is given by an equation of the form

$$p(x, y) = \sum_{i,j} c_{i,j} x^i y^j = 0 \tag{1.6}$$

where p is a polynomial in x, y.

An implicit algebraic surface is given by an equation of the form

$$p(x, y, z) = \sum\nolimits_{i,j,k} c_{i,j,k} x^i y^j z^k = 0 \tag{1.7}$$

where p is a polynomial in x, y, and z.

Generalizing, for the parametrization in (1.3) we have implicit equation

$$p(x_0, \ \ldots, \ x_n) = 0 \tag{1.8}$$

where $p(x_0, \ldots, x_n)$ is a polynomial in $x_0, \ldots, x_n$.

While for any parametric curve given by (1.1) there exists an implicit representation (1.6), and, likewise for any parametric surface given by (1.2) there exists an corresponding implicit equation (1.7), opposite is not true. Therefore, the class of implicit geometric objects is larger that the one of the parametric geometric objects. Moreover, class of implicit curves and surfaces is closed under many geometric operations of interest, while the class of parametric representations is not. Some of these operations, in particular intersection, union and difference make implicit surfaces a valuable modelling tool. However, most natural application for the implicit equations is point membership classification, i.e., checking if a given point is inside, outside, or on the geometric object. Spatial queries simplify to function evaluations of p and checking the sign of the resulting value. On the other hand, generating sample points on an implicit surface is relatively difficult.

Recently implicit objects have gained an increasing importance in geometric modelling, visualization, animation, and computer graphics. Modern graphic hardware allows to explore some of their geometric properties which give them advantages over traditional modelling methods. Implicit surfaces can easily be blended into smooth, complex and intricate shapes. Despite the high computational cost, improvements in modern hardware allow visualization of the implicit surfaces in real time. Indeed, several methods for visual representation of the implicit surfaces, among them blob functions and moving least squares, have emerged [66].

Traditionally, Bernstein polynomials are used to define parametric objects. Recently in CAGD the implicit curves and surfaces defined by Bernstein form polynomials started to appear. Indeed, similar to the power form polynomials, a planar implicit curve can be defined as the zero set of a bivariate Bernstein form polynomial in $\mathbb{R}^2$, while an implicit surface is defined as the zero set of a trivariate Bernstein form polynomial in $\mathbb{R}^3$ [66]. However, in this thesis we restrict our attention to algebraic curves and surfaces defined by power (monomial) bases.

1.1.3 Polygon meshes

In contrast to the other popular in CAGD methods for surface representation, polygonal meshes do not provide accurate models. However, use of polygonal meshes for the representation of highly complex geometric objects remains a standard in most computer graphics applications.

In particular, triangle meshes are preferred due to their algorithmic simplicity, numerical robustness, and efficient display.

A triangle mesh is defined by a set of *vertices*

$$V = v_1, \ldots, v_V$$

and a set of triangular *faces* connecting them

$$F = f_1, \ldots, f_F,\ f_i \in V \times V \times V.$$

Sometimes *edges* are used to represent connectivity of a triangle mesh in terms

$$E = e_1, ..., e_E,\ e_i \in V \times V.$$

Relation between the numbers of vertices V, edges E, and faces F in a closed and connected (but otherwise unstructured) mesh is stated by the *Euler formula*:

$$V - E + F = 2(1 - g),$$

where g is the genus of the surface.

There are several data structures for representing polygonal meshes. For instance, the winged-edge data structure associates with each edge eight references: two vertices, two faces and four incident edges. Face-based data structures are especially convenient for subdivision. Here the basic structuring element is a face that contains pointers to its adjacent vertices and faces and for each adjacent face the index of the adjacent edge.

Approximation error of a triangle mesh is inversely proportional to its number of faces. Sufficient approximation is possible with just a moderate mesh complexity: the vertex density has to be locally adapted to the surface curvature. Flat areas can be sparsely sampled, while in curved regions the sampling density should be higher.

Popularity of both NURBS and polygon mesh representations in CAGD determines interest in the methods for representations change between these two forms [80]. Most of the commercial CAGD software packages have such operations implemented. A method for constructing interpolating or approximating implicit surfaces from the polygon mesh was proposed in [106].

1.1.4 Subdivision surfaces

First introduced in publications of Catmull and Clark [20] and by Doo and Sabin [45] in 1978, subdivision surfaces have become increasingly popular in recent years. Now they are widely used in many application areas, including computer graphics, solid modelling, computer game software, film animation, and others because they provide a simple and efficient tool for construction of smooth curves and surfaces.

As mentioned above, subdivision surfaces can be viewed as a generalization of B-spline surfaces since they are also controlled by a control mesh, but in contrast to piecewise polynomial representations like Bézier patches and NURBS, they can represent surfaces of arbitrary topology and are not restricted by geometric constraints. However the shape of the subdivision surface can not be controlled as intuitively and easily as with NURBS.

Subdivision surfaces are generated by repeated refinement of control meshes: after each refinement step the positions of the (old and new) vertices are adjusted based on a set of local averaging rules. In the limit this process results in a surface of provable smoothness [93].

Inherent hierarchical structure of subdivision surfaces allows for highly efficient algorithms (adaptive subdivision).

One of the main advantages of subdivision surfaces, as compared to B-spline and NURBS surfaces, is that the latter must be trimmed and pieced together in order to produce surfaces of general form. In contrast, subdivision surfaces are intrinsically capable of assuming general form. Other advantages of subdivision surfaces include scalability and good compatibility with application areas where meshes are used [5].

1.2 Change of representation

The modern CAGD and CAM systems operate with several different representations of geometric objects each of which is most suitable for some applications. Representation change is often required: from the cloud of sampled points a mesh is generated, which in turn after application of a subdivision algorithm gives a smooth subdivision surface. Representations change may also occur as a step in some process, like the polar/spherical coordinate representation in the algorithm for converting between parametric and implicit forms given in [113].

Parametric and implicit representations have complementary advantages with respect to certain geometric operations. Moreover, there are some applications where it is necessary that both representations are available. For example, finding surface-surface intersection is easy when one of the surfaces is given in implicit form, and the other in parametric form. Therefore one needs both the parametric representations and the implicit representations depending on different applications and it is important to design algorithms for efficient conversion from one to another. The process of converting parametric forms into implicit forms is named implicitization, and the converse process is parametrization.

Both implicitization and parametrization are classic topics in algebraic geometry and there are several different approaches to solve the implicitization and parametrization problems. For implicitization, the most popular methods are multidimensional resultants and Gröbner bases [101].

Parametrization presents a more difficult problem: through all the algebraic curves and surfaces have implicit representation, not every can be parametrized by rational functions [70]. Although the implicit representation for a geometric object is unique, there exist infinitely many different parametrizations of the same geometric object. The parametrization of lowest degree in this family, such for that to each point, except for possibly a finite number of points, there corresponds only one parameter value, is called proper parametrization. For curves there are effective algorithms for computing proper parametrization [95, 101, 103] and for turning an arbitrary parametrization into proper [3]. For surfaces the general parametrization algorithm is an open problem, although there are partial algorithms for finding parametric representations of particular classes of surfaces. The easiest class of surfaces to parametrize is those where one variable can be written as an explicit function of the other variables, i.e. in the case where $z = f(x, y)$ we can parametrize the surface as $x = u,\ y = v,\ z = f(u, v)$, with appropriate bounds for the parameters u and v. Into this category fall planes that can be then used for piecewise parametrization of cubes, tetrahedrons or other geometric objects popular in in practical applications. Several algorithms exist for implicitization and parametrization of quadratic and cubic surfaces [11, 23, 117, 118], there are parametrization methods for other special cases of surfaces, such as canal surfaces [92] tubular surfaces [100] or conchoid surfaces of spheres [91].

There are two different approaches to parametrization and implicitization problems: exact and approximate. All the above mentioned methods produce exact representations. Applying them in practice, one encounters several limitations: first, high computational cost when dealing with high degree curves and surfaces. In the case of parametrization, an exact rational parametrization can be obtained only for a limited class of implicit curves and surfaces. Finally, in practice the curve or surface is usually given by floating point coefficients with limited accuracy that makes recovering exact representation impossible.

Approximate methods can be used to generate local low degree approximations of high degree geometrical objects and rational approximations of not parametrizable surfaces. They are in general computationally less expensive than exact methods.

The first algorithms for the approximation of algebraic curves and surfaces by parametric curves and surfaces have been proposed in [116, 7]. In [7], algebraic surfaces are approximately parametrized by splines using a combination of symbolic and numerical techniques. E. Hartmann [68] introduced an algorithm for numerical parametrization that maps a parameter or a pair of parameters to a point on the curve or surface. In [120] they present a method for approximate rational parametrization of surfaces, based on numerical optimization techniques. The method computes patches of maximal size on the surfaces that satisfy certain quality constraints.

Approximate implicitization over floating-point numbers was first introduced by T. Dokken [43]. Today, there are direct [121] and iterative techniques [2]. We stop to discuss these methods in more detail later (1.3.4).

Next, we discuss several classical implicitization methods.

1.2.1 Implicitization by resultants

The first method, implicitization by resultants, involves the use of Elimination theory. The problem of deriving implicit equation from parametrization and can be reformulated as an elimination problem. Elimination theory studies methods and algorithms for eliminating variables from a set of polynomial equations.

Definition 1. Given a set of polynomials $f_0, \dots, f_n$, a *resultant* $\mathcal{R}$ is a polynomial expression in the coefficients of $f_0, \dots, f_n$ such that the vanishing of the resultant is a necessary and sufficient condition for $f_0 = f_1 = \dots = f_n = 0$ to have a common root.

Resultants play an important role in elimination theory providing a systematic approach for finding polynomials in an ideal that do not contain as many variables as generic elements of the ideal. First introduced in 19th century, resultants have been actively studied in the early 20th century [21, 111, 82]. The main idea behind the various formulations is to identify a set of n linearly independent polynomials that generate the ideal and that contain n terms. Then each term can be used as an unknown and the theory of linear system of equations can be applied.

Sylvester's resultant is a simple method for eliminating a variable from two algebraic equations. Given two polynomials $f(x) = a_n x^n + \ldots + a_0$ and $g(x) = b_m x^m + \ldots + b_0$ the Sylvester resultant is the determinant:

$$\mathcal{R} = \begin{vmatrix} a_n & a_{n-1} & \cdots & a_0 & 0 & \cdots & 0 \\ 0 & a_n & \cdots & a_1 & a_0 & \cdots & 0 \\ \vdots & & \ddots & & & \ddots & \\ 0 & \cdots & 0 & a_n & a_{n-1} & \cdots & a_0 \\ b_m & b_{m-1} & \cdots & b_0 & 0 & \cdots & 0 \\ 0 & b_m & \cdots & b_1 & b_0 & \cdots & 0 \\ \vdots & & \ddots & & & \ddots & \\ 0 & \cdots & 0 & b_m & b_{m-1} & \cdots & b_0 \end{vmatrix}$$

It can be shown that f and g have a common root if and only if the $(n+n) \times (m+n)$ determinant is zero [31].

We have presented the resultant as a tool for determining whether two polynomials have a common root. Now we show how to apply that tool to converting the parametric equation of a curve given by (1.1) into an implicit equation of the form (1.6).

We proceed by forming two auxiliary polynomials:

$$g(x,t) = c(t)x - a(t), \;\; h(y,t) = c(t)y - b(t)$$

Let us view $g(x,t)$ as a polynomial in t whose coefficients are linear in x, and $h(y,t)$ as a polynomial in t whose coefficients are linear in y. If we compute the resultant of $g(x,t)$ and $h(y,t)$, the result is a polynomial $p(x,y)$. Note that $g(x,t) = h(y,t) = 0$ only for values of x, y and t which satisfy the relationships $x = a(t)/c(t)$, $y = b(t)/c(t)$. Clearly, for these values of x, y and t, the resultant $p(x,y)$ must vanish. Conversely, any (x,y) pair for which $p(x,y) = 0$, causes the resultant of g and h to be zero. But, if the resultant is zero, then we know that there exists a value of t for which $g(x,t) = h(y,t) = 0$. In other words, all (x,y) for which $p(x,y) = 0$ lie on the parametric curve and therefore $p(x,y) = 0$ is the implicit equation of that curve.

Another popular resultant formulation for two univariate polynomials is Bézout resultant.

$$\mathcal{R} = \begin{vmatrix} c_{0,0} & \cdots & c_{0,n-1} \\ \vdots & & \vdots \\ c_{n-1,0} & \cdots & c_{n-1,n-1} \end{vmatrix}$$

where the $c_{i,j}$ are defined by the formula:

$$c_{i,j} = \sum_{\substack{k \leq min(i,j) \\ k+h=i+j+1}} (a_k b_h - a_h b_k)$$

and $b_{m+1} = \ldots = b_n = 0$.

It provides more compact matrix, compared with the Sylvester's, but has more complicated entries. In general, the Bézout resultant has dimension $n \times n$ while the Sylvester's resultant has dimension $(n+m) \times (n+m)$.

Parametric surfaces could be implicitized by forming the equations

$$f(x,s,t) = d(s,t)x - a(s,t), \;\; g(y,s,t) = d(s,t)y - b(s,t), \;\; h(z,s,t) = d(s,t)z - c(s,t)$$

and eliminating first t, then s. Alternatively, Dixon's resultant can be used for interpolation of surfaces, because it is formulated for the case of three polynomials in two variables.

Definition 2. Consider rational parametrization (1.3). Parameter values $(t_1, \ldots, t_n)$ for which $x_i = g_i = 0$ for all $i = 0, \ldots, n$ are called *base points* of the parametrization.

Applying resultants to compute implicit equation has certain drawbacks. First, resultants vanish in the presence of the base points. That is an important issue, since the geometric objects with base points are common. For instance, any parametrization of a rational surface whose algebraic degree is not a perfect square has base points.

Another issue is that the resulting expression often contains an extraneous factor. The separation of these extraneous factors can be a time consuming task that involves multivariate polynomial factorization.

1.2.2 Implicitization by Gröbner bases

Another method based on Elimination theory [31, Chapter 2, §8], the Gröbner bases algorithms provide an alternative for implicitizing parametric forms without the introduction of extraneous factors. Here we give a general outline of the method; there exist many performance improving specializations.

A Gröbner basis of an ideal I is a set of polynomials $g_1, ..., g_t$ such that the leading term of any polynomial in I is divisible by the leading term of at least one of the polynomials $g_1, ..., g_t$. A term order should be fixed: different term orders produce different Gröbner bases.

In order to use Gröbner basis method for implicitizing a rational parametric curve defined by (1.1) with $gcd(a, b, c) = 1$ (otherwise, the common factor can be removed), we define the ideal

$$I = \langle cx - a, cy - b \rangle \subset \mathbb{R}[x, y, t].$$

If $p(x, y) = 0$ is the implicit equation of (1.1), then $p \in I \cap \mathbb{R}[x, y]$. To guarantee that p appears in the Gröbner basis, we order the variables $t > x > y$, and then construct the Gröbner basis with the lexicographic ordering for the ideal I. Thus the Gröbner bases obtained will contain the curve's implicit form, i.e. an element which does not involve t, and an inversion, an element which is linear in t.

Similarly as with the resultant method, presence of the base points affects performance here. Moreover, computations get costly when applying the method to implicitize geometric objects of high degree or dimension. In practice, Gröbner bases calculations require more time and memory than resultant methods.

1.2.3 Implicitization by µ-bases

The concept of a μ-basis was first introduced in [33] to provide a compact representation for the implicit equation of a planar rational curve. Since then several generalizations to rational surfaces have appeared [25, 22].

The μ-basis of a rational curve consists of two polynomials $p(x, y, t)$ and $q(x, y, t)$ which are linear in x, y and have degree $\mu(\mu \leq [n/2])$ and $n - \mu$ in t respectively, where n is the degree of the rational curve.

The resultant of $p(x, y, t)$ and $q(x, y, t)$ with respect to t gives the implicit equation of the rational curve. Using a variant form of the Bézout resultant, the implicit equation of a rational curve can be written as the determinant of an $(n - \mu) \times (n - \mu)$ matrix. Efficient algorithms were also developed to compute the μ-basis of a rational curve [123, 24]

The μ-basis of a rational curve or surface can recover the parametric equation as well as derive the implicit equation. Thus it serves as a connection between the implicit form and the parametric form of a curve or surface.

The method of implicitizing by the μ-bases is efficient, generally working even if the base points are present. Overall it allows for the more compact matrix representations that the traditional resultant methods.

1.3 Implicitization by interpolation

In this section we discuss implicitization methods based on interpolation, that befall in two groups: the exact methods, that aim at computing the exact implicit equation by symbolic means and the approximate, that employ numerical solving. However, in both cases performance of the algorithm depends on the implicit degree bound, since it determines the interpolation space.

1.3.1 Bound on the implicit degree

Definition 3. Consider rational parametrization given by (1.3).

We call the maximum of the total degrees of the polynomials $(t_1, \ldots, t_n),\ i = 0, \ldots,\ n$ the *parametric degree*.

We call *implicit degree* the maximum degree of the implicit equation (1.8).

It is known from classical algebraic geometry that any degree n polynomial or rational parametric curve can be represented exactly using a degree n implicit equation [101]. However, in higher dimensions finding the implicit degree of the parametrization presents a more challenging problem.

The implicit degree of a surface can be thought of as the number of times that the surface is intersected by a generic straight line [69]. Consider a generic straight line as the intersection of two distinct planes in general position $a_1x + a_2y + a_3z + a_4 = 0$ and $b_1x + b_2y + b_3z + b_4 = 0$. The planes intersect the parametric surface (1.2) in curves

$$a_1a(s,t) + a_2b(s,t) + a_3c(s,t) + a_4d(s,t) = 0 \tag{1.9}$$

and

$$b_1a(s,t) + b_2b(s,t) + b_3c(s,t) + b_4d(s,t) = 0 \tag{1.10}$$

These curves are each degree n in s, t. By Bézout's theorem, these two curves intersect in n^2 points, which must also be the number of times that the straight line common to the two planes intersects the surface. Thus, the degree of the surface, and of its implicit equation, is n^2.

It is obvious, however, that there are plenty of surfaces with the implicit degrees that are not exact squares. The reason is that the base points of the parametrization account for some of the intersections, thus causing the degree decrease. If a base point exists, the intersection curve of any plane with the surface contains the base point, therefore the above two curves intersect at the base point and at $n^2 - 1$ other points. Since the base point does not map to a unique point on the surface, this does not represent a point at which the straight line intersects the surface, and the degree of the surface is $n^2 - 1$. Each additional simple base point diminishes the degree of the surface by one. If two general curves in the linear system are tangent at a base point, they intersect twice at the base point and the degree of the surface becomes $n^2 - 2$. If two general curves in the linear system have a double point in common, they intersect four times at that base point and the degree becomes $n^2 - 4$. Thus, a general degree formula is $n^2 - \rho$ where ρ is the total number of times that two general curves in the linear system intersect at base points. This also assumes that the surface has a one-to-one parametrization.

If the surface (1.2) is a tensor product surface, for instance, bicubic surface, of degree n_s and n_t in s and t respectively, its parametric degree is $n = n_s + n_t$. But there exist two base points at infinity, corresponding to $s = \infty$, $t = \infty$, counted at least n_s^2 and n_t^2 times respectively [28]. Thus the implicit degree of a bi-degree n_s, n_t parametrized rational surface is at most $(n_s + n_t)^2 - n_s^2 - n_t^2 = 2n_s n_t$. For example, degree of a bicubic surface is 18 [101].

The ability to establish a bound on the implicit degree is useful for several reasons. The complexity of many algebraic methods used for processing a geometric object depends on its implicit degree. Bounding the implicit degree is an important first step in the process of implicitization: this information may be used then either for creating specialized resultants or for determining an interpolation space to be used in order to obtain the implicit form.

The problem of bounding the implicit degree of an algebraic variety has been addressed in several studies. For instance, in [104] the authors analyze the implicit degree of algebraic curves and in [105] provide formulae for the partial degrees of the plane curves.

The degrees of the offset curves have been studied in [60] and [102].

An upper bound for the degree of a rational surface is given in [99]. In [29] a method based on the degree of rational maps and on the base points of the parametrization is presented. In [14] degree bound is studied as applied to implicitization of rational surfaces with the base points, here, as in [1], the degree of rational surfaces is computed by analyzing the base points and by means of syzygies.

The space of interpolation determined by the implicit degree can be further decreased if the partial degrees are known. Formulae that allow to compute partial degrees of a rational surface given by means of a proper parametrization was presented in [90]. For the case when the parametrization is not proper upper bounds on the partial degrees are given.

The bound on the degree of polygonal surface patches constructed from the triangular rational Bézier surfaces is analyzed in [119].

Bounds on the implicit degree as well as on partial degrees can be formulated in terms of Newton polytopes, see Proposition 6 [56].

Definition 4. Given a polynomial

$$\sum_{a \in A_i} c_{ia} t^a \in \mathbb{R}[t_1, \dots, t_n], \quad t^a = t_1^{a_1} \cdots t_n^{a_n}, a \in \mathbb{N}^n, c_{ia} \in \mathbb{R} - \{0\},$$

its *support* is the set $A_i = \{a \in \mathbb{N}^n : c_{ia} \neq 0\}$.

Its *Newton polytope* is the convex hull of the support.

Further the support and the Newton polytope of the implicit equation are referred as *implicit support* and *implicit polytope*, respectively.

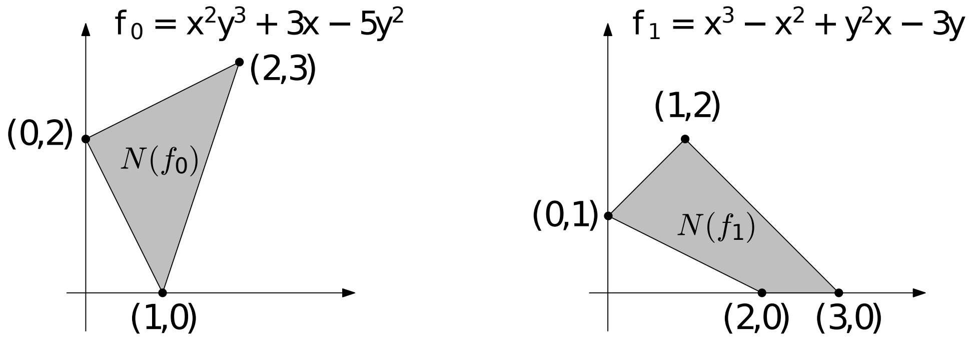

Figure 1.1: Newton polygons $N(f_i)$ of polynomials $f_i \in \mathbb{Z}[x, y]$.

1.3.2 Computing implicit polytope

One of the first publications [109] proposed to exploit sparseness in the implicit polynomial in context of computing the Newton polytope of a rational hypersurface for generic Laurent polynomial parametrizations.

Algorithms based on tropical geometry have been offered in [40, 108, 110]. This method computes the abstract tropical variety of a hypersurface parametrized by generic Laurent polynomials in any number of variables, thus yielding its implicit support; it is implemented in `TrIm`. For non-generic parametrizations of rational curves, the implicit polygon is predicted. In higher dimensions, the following holds:

Proposition 1. [108, prop.5.3] *Let $f_0, \dots, f_n \in \mathbb{C}[t_1^{\pm 1}, \dots, t_n^{\pm 1}]$ be any Laurent polynomials whose ideal of algebraic relations is principal, say $I = \langle g \rangle$, and $P_i \subset \mathbb{R}^n$ the Newton polytope of f_i. Then, the polytope constructed combinatorially from $P_0, \dots, P_n$ using tropical geometry contains a translate of the Newton polytope of g.*

The tropical approach was improved in [34] to yield the precise implicit polytope in $\mathbb{R}^3$ for generic parametrizations of surfaces in 3-space. In [71], they describe efficient algorithms implemented in the `GFan` library for the computation of Newton polytopes of specialized resultants, which may then be applied to predict the implicit polytope. Sparse elimination has been used for the same task [49],The latter, implemented in the `ResPol` software is faster on dimensions relevant here, namely for projected polytopes in up to 5 dimensions. In 2.2.3 we provide a detailed description of the software use for finding the Newton polytope.

The Newton polygon of a curve parametrized by rational functions, without any genericity assumption, is determined in [38]. In a similar direction, an important connection with combinatorics was described in [59], as they showed that the Newton polytope of the projection of a generic complete intersection is isomorphic to the mixed fiber polytope of the Newton polytopes associated to the input data.

In [56] a method relying on sparse elimination for computing a superset of the generic support from the resultant polytope is discussed, itself obtained as a (non orthogonal) projection of the secondary polytope. The latter was computed by calling `Topcom` [94]. This approach was quite expensive and, hence, applicable only to small examples; in our work we have used its refined and improved versions. For instance in [56] the benchmark example of bicubic surface is given as an open problem. In this thesis we present the problem being successfully resolved: we compute Newton polytope (Example 1) and implicit equation of the bicubic surface (Example 12).

In [55], sparse elimination is applied to determine the vertex representation of the implicit polygon of planar curves. The method relies on the study of the Newton polytope of a resultant. The tools used are mixed subdivisions of the input Newton polygons and regular triangulations of point sets defined by the Cayley trick. It can be applied to polynomial and rational parametrizations, where the latter may have the same or different denominators. The method offers a set of rules that, applied to the supports of sufficiently generic rational parametric curves, specify the 4, 5, or 6 vertices of the implicit polygon. In case of non-generic inputs, the predicted polygon is guaranteed to contain the Newton polygon of the implicit equation. The method can be seen as a special case of the general approach based on sparse elimination.

This algorithm (further referred as curves-only) has been used in our work to determine implicit polygone for the planar curves and has been integrated in our `Maple` code.

1.3.3 Methods for exact implicitization

When talking of the interpolation as applied to CAGD, the techniques such as spline interpolation, for generating parametric representation from a discrete set of points, are that first come to mind. In our case, interpolation is applied to the set of points generated from the parametric equations, thus definitely belonging to the curve or surface in question, in order to determine the coefficients of the implicit equation.

In general the implicitization algorithms that are based on interpolation have a wide range of applicability, work in the presence of base points and can compute the implicit equation both symbolically or numerically, depending on the interpolation method.

These are two main approaches, dense and sparse methods. The former require only a bound on the total degree of the target polynomial, whereas the latter require a bound on the number of its terms, thus exploiting any sparseness of the target polynomial. Apriori knowledge of the support helps significantly, by essentially answering the first step of sparse interpolation algorithms.

Most existing approaches employ total degree bounds on the implicit polynomial to compute a superset of the implicit support, e.g. [42, 30].

Successful applications of them can be found in [86] where an interpolating implicitization algorithm is presented. from the degrees of the parametrization allows to apply the interpolation methods for computing the implicit equation.

Let S be (a superset of) the support of the implicit polynomial $p(x_0, \dots, x_n) = 0$; $|S|$ bounds the support cardinality.

Sparse interpolation is the problem of interpolating a multivariate polynomial when information of its support is given [125]. This may simply be a bound on support cardinality, then sparse interpolation is achieved in $O(\sigma^3 \delta n \log n + |S|^3)$, where δ bounds the output degree per variable, σ is the actual support cardinality, and n the number of variables [10, 74]. A probabilistic approach runs in $O(\sigma^2 \delta n)$ [124] and requires as input only δ.

For the sparse interpolation of resultants, the quasi-Toeplitz structure of the matrix allows us to reduce complexity by one order of magnitude, when ignoring polylogarithmic factors, and arrive at a quadratic complexity in matrix size [19]. This was extended to the case of sparse resultant matrices [57].

Our matrices reveal what we call quasi-Vandermonde structure, since the matrix columns are indexed by monomials and the rows by values on which the monomials are evaluated.

This reduces matrix-vector multiplication to multipoint evaluation of a multivariate polynomial. It is unclear how to achieve this post-multiplication in time quasi-linear in the size of the polynomial support when the evaluation points are arbitrary, as in our case. Existing work achieves quasi-linear complexity for specific points [57, 88, 98, 114].

The most direct method to reduce implicitization to linear algebra is to construct a $|S| \times |S|$ matrix M, indexed by monomials with exponents in S (columns) and $|S|$ different values (rows) at which all monomials get evaluated. Then, vector p is in the kernel of M. This idea was used in [56, 87, 110]; it is the approach explored in our work, extended to an approximate implicitization as well. It is also the premise of [54] which extends results of this work to the case of high dimensional kernel space and applies these techniques to the problem of computing the discriminant of a multivariate polynomial.

In [108], they propose evaluation at unitary $\tau \in (\mathbb{C}^*)^n$, i.e., of modulus 1. This is one of the evaluation strategies examined below. Another approach was described in [30], based on integration of matrix $M = SS^T$, over each parameter $t_1, \ldots, t_n$. Then, p is in the kernel of M. In fact, the authors propose to consider successively larger supports in order to capture sparseness. This method covers a wide class of parametrizations, including polynomial, rational, and trigonometric representations, but the size of M is quite big and matrix entries take big values, so it is difficult to control its numeric corank. In some cases, its corank is ≥ 2. Thus, the accuracy, or quality, of the approximate implicit polynomial is unsatisfactory. The resulting matrix has Henkel-like structure [78]. When it is computed over floating-point numbers, the resulting implicit polynomial does not necessarily have integer coefficients. In [30], they discuss some post-processing to yield the integer relations among the coefficients, but only for small examples.

1.3.4 Approximate implicitization

Approximate implicitization over floating-point numbers was introduced by T. Dokken and coworkers in a series of papers. Today, there are direct [43] and iterative techniques [2]. In practical applications of CAGD, precise implicitization often can be impossible or very expensive to obtain. Moreover, exact implicit equations usually are of high degree and contain unwanted branches and singularities. Approximate implicitization over floating-point numbers appears to be an effective solution [43, 8]. We discuss approximate implicitization, in the setting of sparse elimination. Approximate implicitization is one of the main motivations for reducing implicitization to interpolation of the implicit coefficients.

We describe the basic direct method [43]: Given a parametric (spline) curve or surface $x(t),\ t \in \Omega \subset \mathbb{R}^n$, the goal is to find polynomial $q(x)$ such that $q(x(t) + \eta(t)g(t)) = 0$, where $g(t)$ is a continuous direction function with Euclidean norm $\|g(t)\| = 1$ and $\eta(t)$ a continuous error function with $|\eta(t)| \leq \varepsilon$. Now, $q(x(t)) = (Mp)^T \alpha(t)$, where matrix M is built from monomials in x. It may be constructed as we do, or it may contain a subset of the monomials of the implicit support. Moreover, p is the vector of implicit coefficients, hence $Mp = 0$ returns the exact solution, and $\alpha(t)$ is the basis of the space of polynomials which describes $q(x(t))$, and is assumed to form a partition of unity and to be nonnegative over Ω: $\sum_i \alpha_i = 1$, $\alpha_i \geq 0$, $\forall i, t \in \Omega$. One may use the Bernstein-Bézier basis with respect to Ω, in the case of curves, or a triangle which contains Ω, in the case of surfaces.

In [43, p.176] the authors propose to translate to the origin and scale the parametric object, so as to lie in $[-1, 1]^n$, in order to improve the numerical stability of the linear algebra operations. In our experiments, we have tried several ways of evaluation and found out that for numerical solving using unitary complex values leads to better numerical stability. Since both our and their methods rely on SVD, our experiments confirm their findings.

The idea of the above methods is to interpolate the coefficients using successively larger supports, starting with a quite small support and extending it so as to reach the exact one. All existing approaches, e.g. [30], have used upper bounds on the total implicit degree, thus ignoring any sparseness structure. This fails to take advantage of the sparseness of the input in order to accelerate computation. Our methods provide a formal manner to examine different supports, in addition to exploiting sparseness.

In the context of sparse elimination, the Newton polytope captures the notion of degree. Given an implicit polytope we can naturally define candidates of smaller support, the equivalent of lower degree in classical elimination, by an inner offset of the implicit polytope. The operation of scaling down the polytope can be repeated, thus producing a list of implicit supports yielding smaller implicit equations with larger approximation error. See Examples 9 and 10 for when the predicted implicit polytope is much larger than the true one.

If S is a superset of the implicit support, then the most direct method to reduce implicitization to linear algebra is to construct a $|S| \times |S|$ matrix M, indexed by monomials with exponents in S (columns) and $|S|$ different values (rows) at which all monomials get evaluated. Then the vector $\vec{p}$ of coefficients of the implicit equation is in the kernel of M. This idea was used in [53, 56, 87, 110]; it is also the starting point of our work. Here, moreover, we focus on sparse elimination, characterize the non-exactness of the kernel computation and offer ways to rectify it, and apply our software to computing discriminants.

A similar approach was based on integrating matrix $M = SS^\top$, over each parameter $t_1, \ldots, t_n$ [30]. Then $\vec{p}$ is in the kernel of M. In fact, the authors propose to consider successively larger supports in order to capture sparseness. This method covers polynomial, rational, and trigonometric parameterizations, but the matrix entries take big values (e.g. up to 10^{28}), so it is difficult to control its numeric corank, i.e. the dimension of its nullspace. Thus, the accuracy of the approximate implicit polynomial is unsatisfactory. When it is computed over floating-point numbers, the implicit polynomial does not necessarily have integer coefficients. The authors discuss post-processing to yield integer relations among the coefficients, but only in small examples.

1.4 Summary of the results

Let us summarize the main results of our work.

We take further the approach to implicitization by interpolation presented in [43]. The more efficient method for computing the implicit Newton polytope, proposed in [49], allows us to successfully apply the method to curves and surfaces of higher degree as well as hypersurfaces.

In the general case, the implicit Newton polytope is derived from the support of a symbolic sparse resultant related to the polynomials in the parametrization, whose Newton polytope is projected to the space of the x_i's [49]. The theoretical foundations of our approach are given in Theorem 4, Lemma 12 and Corollary 13. Having reduced implicitization to interpolation, we employ standard methods to determine the unknown coefficients by linear algebra.

We have explored several ways of constructing the interpolation matrix using different ways of evaluation. Our implicitization algorithm has proved to be an efficient, output-sensitive tool for finding exact implicit equation, able to compete successfully with such popular methods as Gröbner bases in the terms of computation time. Also, our method can be applied to parameterizations with base points which raise important issues for other implicitization methods.

In some instances the polynomial computed using our algorithm contains an extraneous factor. We have analyzed such cases the and propose techniques for handling them.

While approximate implicitization may be preferred for practical applications in CAGD systems, there are many areas outside geometry where exact implicitization may be applied. For instance, computing an exact implicit equation can be used to solve such problems as the computation of the discriminant of a multivariate polynomial or the resultant of a system of multivariate polynomials.

Computing the kernel of the interpolation matrix numerically allows us to find an approximate implicit equation. Our experiments with classic algebraic curves and surfaces, Bézier curves and NURBS curves and surface patches have allowed us to find better ways of evaluation when constructing the matrix in terms of accuracy and numerical stability. Our algorithm was implemented using `Maple` and `SAGE` mathematics software systems; the former being better suited for symbolic computations and the latter more suitable for numerical solving.

1.5 Thesis structure

In this chapter we have given a survey of the main representations of the geometric objects and the methods to change the representation. An emphasis was put on the parametric and implicit forms and the implicitization, since the latter is the main subject of our study.

The remainder of the thesis is organized as follows:

Chapter 2 offers a full description of our implicitization algorithm augmented by the predicted implicit polytope and the methods that compute the latter.

In the Section 2.2 we describe the method for computing the implicit polytope that are used in our work. We show the backgrounds of the method, as we give an overview of the sparse elimination theory. We demonstrate use of the latest implementation of the method in practice.

The Section 2.3 contains a detailed description of our implicitization algorithm. We discuss its complexity, different approaches to the construction of the matrix that allow us to exploit the particular support prediction method's specifics and the importance of choosing suitable values for matrix evaluation. Here we also study the cases when the resulting equation contains an extraneous factor and suggest possible solutions.

Chapter 3 is dedicated to practical implementation of our algorithm. We illustrate our results by examples of rational curves, surfaces and hypersurfaces, provide a comparison with the other prominent implicitization methods.

In the Section 3.1 we present our Maple implementation of the algorithm; give examples of exact and approximate implicitization of the algebraic curves and surfaces of power or Bernstein basis. Here we describe the measures we use to evaluate the accuracy of the approximate implicitization and compare our method with other popular implicitization methods

The Section 3.3 shows one of the non-geometric applications for our algorithm: computing discriminants.

In Chapter 4 we summarize the results of our study (Section 4.1) and discuss possible future directions for the research (Section 4.2).

Chapter 2

Implicitization with support prediction

The first part of the chapter is dedicated to the methods we employ to compute implicit polytope and present the software, implementation of the support predicting method. In the second part we introduce our implicitization algorithm, illustrating its application with the relevant examples.

2.1 Introduction

One reason for the interpolation of the implicit coefficients is the current increase of activity around various approaches capable of predicting the implicit support. Recent support prediction methods notably include tropical geometry methods, e.g. [34, 38, 71, 108, 110]; In our work we explored the methods based on sparse elimination theory [49, 55, 56, 53]. In the case of curves, the implicit support is directly determined for generic rational parametric expressions [55]. Theorem 4 settles the general case by showing that this approach yields a superset of the implicit support. We interface linear algebra interpolation with the implicit support predictors of [49, 55].

The present work can use the implicit support predicted by any method. In fact, [110, sec.4] states that ``*Knowing the Newton polytopes reduces computing the [implicit] equation to numerical linear algebra. The numerical mathematics of this problem is interesting and challenging [...]* "

We offer a publicly available Maple implementation and study its numerical stability and efficiency on several instances of curves and surfaces. One central question is how to evaluate the computed monomials to obtain a suitable matrix. We compare results obtained by using random integers, random complex unitary numbers and complex roots of unity. It appears that complex unitary numbers offer the best tradeoff of efficiency and accuracy for numerical computation, whereas random integers are preferred for exact kernel computation.

In the general case, the implicit support is provided by that of a symbolic sparse resultant related to the polynomials in (2.10), whose Newton polytope is projected to the space of the x_i's [49]. The theoretical foundations of our approach are given in Theorem 4, Lemma 12 and Corollary 13. Having reduced implicitization to interpolation, we employ standard methods to determine the unknown coefficients by linear algebra.

Our research, first presented in [51], centers around the implicitization problem: we explore possible applications of the recently developed method that allows to determine a superset of the implicit equation's monomials. Knowing the potential monomials allows us to compute the implicit equation following the classical method of interpolating the unknown coefficients. We have developed an implicitization algorithm that can be applied to curves, surfaces or hypersurfaces of any dimension of polynomial (preferably), rational or trigonometric representation. Our method works in the presence of base points, however it has its limits: geometric objects have to be of co-dimension one and presented using monomial basis. In the case of trigonometric parametrizations coordinate functions have to be convertible to rational functions [53]. The method can be applied to geometric objects of Bernstein basis, however such parametrizations have to be represented in monomial basis [52].

Implicitization has its applications outside the geometry, in areas as diverse as robotics or statistics. We demonstrate some of the non-geometrical applications of our implicitization method computing resultants and discriminants [54].

2.2 Support prediction

This section gives an overview of the methods for computing the implicit support, based on the sparse resultant, used in our work. However, tropical geometry has not been the focus of our work, indeed, developing our algorithm we have viewed support prediction as a black box. We have tried three different methods for computing Newton polytope of the implicit equation, making necessary adjustments to the implicitization algorithm, which can as easily exploit results of any other support predicting tool.

2.2.1 Sparse elimination theory

Sparse, or toric, elimination subsumes classical, or dense, elimination in the sense that, when Newton polytopes equal the corresponding simplices, the former bounds become those of the classical theory [56, sec.3], [109, thm.2(2)].

Consider the polynomial system $\bar{F}_0, \ldots, \bar{F}_n$ as in expression (2.10), defining a hypersurface, and let $A_i \subset \mathbb{Z}^n$ be the support of $\bar{F}_i$ and $P_i \subset \mathbb{R}^n$ the corresponding Newton polytope. The family $A_0, \ldots, A_n$ is *essential* if they jointly affinely span $\mathbb{Z}^n$ and every subset of cardinality $j, 1 \leq j < n$, spans a space of dimension $\geq j$. It is straightforward to check this property algorithmically and, if it does not hold, to find an essential subset. In the sequel, the input $A_0, \ldots, A_n \subset \mathbb{Z}^n$ is supposed to be essential.

For each $\bar{F}_i$, $i = 0, \ldots, n$, we define a polynomial $F_i \in K[t]$ with symbolic coefficients c_{ij} algebraically independent over $\mathbb{R}$, $K = \mathbb{C}(c_{ij})$, and the same support A_i, i.e. a generic polynomial with respect to A_i:

$$F_i = \sum_{j=1}^{|A_i|} c_{ij} t^{a_{ij}} \in K[t], \; a_{ij} \in A_i, \; i = 0, \ldots, n. \tag{2.1}$$

Obviously, each F_i has also the same Newton polytope P_i as $\bar{F}_i$.

When discussing implicitization methods we have given general definition of the resultant. Now we introduce our main tool, namely the sparse resultant of an overconstrained polynomial system.

Definition 5. The *sparse resultant* of polynomial system in expression (2.1) is an irreducible polynomial

$$\mathcal{R} \in \mathbb{Z}[c_{ij} : i = 0, \ldots, n, \; j = 1, \ldots, |A_i|],$$

defined up to sign, vanishing if and only if $F_0 = F_1 = \cdots = F_n = 0$ has a common root in the torus $(\mathbb{C}^*)^n$.

The Newton polytope $N(\mathcal{R})$ of the resultant polynomial is the *resultant polytope*. We call any monomial which corresponds to a vertex of $N(\mathcal{R})$ an *extreme term* of $\mathcal{R}$.

Definition 6. The *Minkowski sum* $A + B$ of convex polytopes $A, B \subset \mathbb{R}^n$ is the set $A + B = \{a + b \mid a \in A, b \in B\} \subset \mathbb{R}^n$. A *tight mixed subdivision* of $P = P_0 + \cdots + P_n$, is a collection of n-dimensional convex polytopes σ, called (Minkowski) *cells*.

They form a polyhedral complex that partitions P, and every cell σ is a Minkowski sum of subsets $\sigma_i \subset P_i$: $\sigma = \sigma_0 + \cdots + \sigma_n$, where $\dim(\sigma) = dim(\sigma_0) + \cdots + dim(\sigma_n) = n$.

Definition 7. A cell σ is called *v_i-mixed* if it is the Minkowski sum of n one-dimensional segments $E_j \subset P_j$ and one vertex $v_i \in P_i : \sigma = E_0 + \cdots + v_i + \cdots + E_n$.

Definition 8. A mixed subdivision is called *regular* if it is obtained as the projection of the lower hull of the Minkowski sum of lifted polytopes $\widehat{P}_i = \{(p_i, \omega(p_i)) \mid p_i \in P_i\}$.

If the lifting function ω is sufficiently generic, then the induced mixed subdivision is tight.

Definition 9. Let $K_1, K_2, ..., K_n$ be convex bodies (nonempty, compact, convex subsets) in $\mathbb{R}^n$.

The *mixed volume MV* is the unique symmetric function

$$|\lambda_1 K_1 + \cdots + \lambda_m K_m| = \sum_{1 \le i_1, \ldots, i_n \le m} \lambda_{i_1} \cdots \lambda_{i_n} VM(K_{i_1}, \ldots, K_{i_n})$$

for $m \in N, \lambda_1, \ldots, \lambda_m \geq 0$.

The mixed volume of n polytopes in $\mathbb{R}^n$ equals the sum of the volumes of all the mixed cells in a mixed subdivision of their Minkowski sum. We recall a surjection from the regular tight mixed subdivisions to the vertices of the resultant polytope:

Theorem 2. [107] *Given a polynomial system as in expression (2.1) and a regular tight mixed subdivision of the Minkowski sum $P = P_0 + \cdots + P_n$ of the Newton polytopes of the system polynomials, an extreme term of the resultant $\mathcal{R}$ equals*

$$c \cdot \prod_{i=0}^{n} \prod_{\sigma} c_{i\sigma_i}^{vol(\sigma)}$$

where $\sigma = \sigma_0 + \sigma_1 + \cdots + \sigma_n$ ranges over all σ_i-mixed cells, and $c \in \{-1, +1\}$.

Computing all regular tight mixed subdivisions reduces, due to the so-called Cayley trick, to computing all regular triangulations of a point set of cardinality $|A_0| + \cdots + |A_n|$ in dimension $2n$. Let the Cayley embedding of the A_i's be

$$A = \bigcup_{i=0}^{n} (A_i \times \{e_i\}) \subset \mathbb{Z}^{2n}, \; e_i \in \mathbb{N}^n,$$

where $e_0, \ldots, e_n$ form an affine basis of $\mathbb{R}^n$: e_0 is the zero vector,
$e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$, $i = 1, \ldots, n$.

Proposition 3. [Cayley trick] [63] *There exist bijections between: the regular tight mixed subdivisions, the tight mixed subdivisions, or the mixed subdivisions of the convex hull of $A_0 + \cdots + A_n$ and, respectively, the regular triangulations, the triangulations, or the polyhedral subdivisions of A.*

The set of all regular triangulations corresponds to the vertices of the secondary polytope $\Sigma(A)$ of A [63]. To compute the resultant polytope, one can enumerate all regular triangulations of A: it is equivalent to enumerating all regular tight mixed subdivisions of the convex hull of $A_0 + \cdots + A_n$. Each such subdivision yields a vertex of $N(\mathcal{R})$.

Thus the first support prediction algorithm, applicable both to curves and (hyper)surfaces [48] can be summarized as follows:

Input: Polynomial system $F_0, F_1, \ldots, F_n$.
Output: Resultant polytope $N(\mathcal{R})$ of $F_0, \ldots, F_n$.

- Step 1. Compute the supports $A_0, \ldots, A_n$ of the F_i's.
- Step 2. Compute the Cayley embedding of the A_i.
- Step 3. Enumerate all regular triangulations of the Cayley embedding: equivalent to enumerating all regular fine mixed subdivisions of $A_0 + \cdots + A_n$.
- Step 4. Each set of such subdivision yields a vertex of $N(\mathcal{R})$.

However, this method is inefficient even for medium sized inputs; the second support prediction algorithm we have used that computes the implicit polytope proved to be much more effective.

2.2.2 The implicit polytope

To predict the Newton polytope of the implicit equation, or implicit polytope we use the algorithm in [49] for the computation of resultant polytopes and their orthogonal projections. Note that the latter correspond to generic specializations of the resultant.

Given the supports $A_i,\ i = 0, \ldots, n$ of the polynomials in expression (2.1), the algorithm in [49] computes the resultant polytope $N(\mathcal{R})$ of their sparse resultant $\mathcal{R}$ without enumerating all mixed subdivisions of the convex hull of $A_0 + \cdots + A_n$. More precisely, it is an incremental algorithm to compute $N(\mathcal{R})$ by considering an equivalence relation on mixed subdivisions, where two subdivisions are equivalent iff they specify the same resultant vertex. The class representatives are vertices of the resultant polytope. The algorithm exactly computes vertex- and halfspace-representations of the resultant polytope or its projection. It avoids computing $\Sigma(A)$, but uses the above relationships to define an oracle producing resultant vertices in a given direction. It is output-sensitive as it computes one mixed subdivision per equivalence class, and is the fastest today in dimension up to 5; in higher dimensions it is competitive with the implementation of [71], relying on the GFan library. Moreover, there is an approximate variant that computes polytopes whose volume differs by $\leq 10\%$ from the true volume, with a speedup of up to 25 times.

Let us formalize the way that the polytope $N(\mathcal{R})$ is used in implicitization. Consider an epimorphism of rings

$$\varphi : K \to K' : \ c_{ij} \mapsto c'_{ij}, \tag{2.2}$$

yielding a generic specialization of the coefficients c_{ij} of the polynomial system in expression (2.1). We denote by $F_i' = \varphi(F_i),\ i = 0,\dots,n$, the images of F_i's under φ. Let $\mathcal{R} = \mathrm{Res}(F_0,\dots,F_n)$ be the resultant of polynomial system in (2.1) over K and $H = \mathrm{Res}(F_0',\dots,F_n')$ be the resultant of $F_0',\dots,F_n'$ over K'. Then, the specialized sparse resultant $\varphi(\mathcal{R})$ coincides (up to a scalar multiple from K') with the resultant H of the system of specialized polynomials provided that H does not vanish, a certain genericity condition is satisfied, and the parametrization is generically 1-1 [32],[109, thm.3]:

$$\varphi(\mathcal{R}) = c \cdot H,\ c \in K'. \tag{2.3}$$

If the latter condition fails, then $\varphi(\mathcal{R})$ is a power of H. When the genericity condition fails for a specialization of the c_{ij}'s, the support of the specialized resultant $\varphi(\mathcal{R})$ is a superset of the support of H modulo a translation, provided the sparse resultant does not vanish. This follows from the fact that the method computes the same polytope as the tropical approach, whereas the latter is characterized in Proposition 1. In particular, the resultant polytope is a Minkowski summand of the *fiber polytope* $\Sigma_\pi(\Delta, P)$, where polytope Δ is a product of simplices, each corresponding to a support A_i, $P = \sum_{i=0}^n P_i$, and π is a projection from Δ onto P. Then, $\Sigma(\Delta, P)$, is strongly isomorphic to the secondary polytope of the point set obtained by the Cayley embedding of the A_i's, [107, sec.5]. The algorithm in [49] provides the Newton polytope of $\varphi(\mathcal{R})$.

When specialization φ yields the coefficients of the polynomials in (2.10), i.e. $\varphi(F_i) = \bar{F}_i$, then $H = \mathrm{Res}(\bar{F}_0,\dots,\bar{F}_n) = p(x_0,\dots,x_n)$, where $p(x_0,\dots,x_n)$ is the implicit equation of the hypersurface defined by (2.10). Equation (2.3) reduces to

$$\varphi(\mathcal{R}) = c \cdot p(x_0,\dots,x_n),\ c \in \mathbb{C}[x_0,\dots,x_n], \tag{2.4}$$

hence [49] yields a superset of the vertices of the implicit polytope. The coefficients of the polynomials in (2.1), which define the projection φ, are those who are specialized to linear polynomials in the x_i's.

The above discussion is summarized in the following result, which offers the theoretical basis of our approach.

Theorem 4. *Given a parametric hypersurface, we formulate implicitization as an elimination problem, by defining the corresponding sparse resultant. The projection of the sparse resultant's Newton polytope contains a translate of the Newton polytope of the implicit equation.*

Let us now give two techniques for improving our approach. The following lemma is used at preprocessing before support prediction, since it reduces the size of the input supports.

Lemma 5. [71, lem.3.20] *If $a_{ij} \in A_i$ corresponds to a specialized coefficient of F_i, and lies in the convex hull of the other points in A_i corresponding to specialized coefficients, then removing a_{ij} from A_i does not change the Newton polytope of the specialized resultant.*

Furthermore, in order to eliminate some extraneous monomials predicted by our support prediction method, we may apply the following well-known degree bounds, generalized in the context of sparse elimination. For a proof, the reader may refer to [56].

Proposition 6. *The total degree of the implicit polynomial of the hypersurface corresponding to system (2.10) is bounded by $n!$ times the volume of the convex hull of $A_0 \cup \cdots \cup A_n$. The degree of the implicit polynomial in some x_j, $j \in \{0,\dots,n\}$ is bounded by the mixed volume of the $\bar{F}_i$, $i \neq j$, seen as polynomials in t.*

The classical results for the dense case follow as corollaries. Take a surface parametrized by polynomials of degree d, then the implicit polynomial is of degree d^2. For tensor parametrizations of bi-degree (d_1, d_2), the implicit degree is $2d_1d_2$. We use these bounds to reduce the predicted Newton polytope in certain cases, see also Corollary 14.

The resultant polytope $N(\mathcal{R})$ lies in $\mathbb{R}^{|A|}$ but we shall see that it is of lower dimension. Let us describe the hyperplanes in whose intersection lies $N(\mathcal{R})$. For this, let $\mathcal{A}$ be the $(2n+1) \times |A|$ matrix whose columns are the points in the A_i, where each $a \in A_i$ is followed by the i-th unit vector in $\mathbb{N}^{n+1}$.

Proposition 7. [63] *$N(\mathcal{R})$ is of dimension $|A| - 2n - 1$. The inner product of any coordinate vector of $N(\mathcal{R})$ with row i of $\mathcal{A}$ is: constant, for $i = 1, \ldots, n$, and equals the mixed volume of $F_0, \ldots, F_{j-1}, F_{j+1}, \ldots, F_n$, for $j = i - (n+1)$, $i = n+1, \ldots, 2n+1$.*

The last $n+1$ relations specify the fact that $\mathcal{R}$ is separately homogeneous in the coefficients of each F_i. The proposition implies that one obtains an isomorphic polytope when projecting $N(\mathcal{R})$ along $2n+1$ points in $\cup_i A_i$, which affinely span $\mathbb{R}^{2n}$; this is possible because of the assumption that $\{A_0, \ldots, A_n\}$ is an essential family. Having computed the projection, we obtain $N(\mathcal{R})$ by computing the missing coordinates as the solution of a linear system: we write the aforementioned inner products as $\mathcal{A}[X\,V]^T = C$, where C is a known matrix and $[X\,V]^T$ is a transposed $|A| \times u$ matrix, expressing the partition of the coordinates to unknown and known values, where u is the number of $N(\mathcal{R})$ vertices. If the first $2n+1$ columns of $\mathcal{A}$ correspond to specialized coefficients, $\mathcal{A} = [\mathcal{A}_1\,\mathcal{A}_2]$, where submatrix $\mathcal{A}_1$ is of dimension $2n+1$ and invertible, hence $X = \mathcal{A}_1^{-1}(C - \mathcal{A}_2 V)$.

2.2.3 Implementations of the method

In [49], they develop an incremental algorithm to compute the resultant polytope, or its orthogonal projection along a given direction. It is implemented in package `ResPol`[1].

The algorithm exactly computes vertex- and halfspace-representations of the target polytope and it is output-sensitive. It is very efficient for inputs relevant to implicitization: it computes the polytope of surface equations within 1 sec, assuming < 100 terms in the parametric polynomials, which includes all common instances in geometric modeling. This is the main tool for support prediction used in this work, thus we illustrate its use in implicitization.

`ResPol` takes as input a text file with 3 lines:

- Dimension of the input supports (in our case, number of parametric variables).
- Cardinality of each support | support points defining the projection (in our case, the exponents of monomials with coefficient x_i). The part of the line starting with | is optional; when absent the polytope is projected on the first coordinate per support.
- The supports of the polynomials defined by the parametric expressions.

Example 1. Consider the bicubic surface,

[1] `http://sourceforge.net/projects/respol`

$$\begin{aligned}
x_0 &= 3t_1(t_1-1)^2+(t_2-1)^3+3t_2,\\
x_1 &= 3t_2(t_2-1)^2+t_1^3+3t_1,\\
x_2 &= -3t_2(t_2^2-5t_2+5)t_1^3-3(t_2^3+6t_2^2-9t_2+1)t_1^2+\\
&\qquad t_1(6t_2^3+9t_2^2-18t_2+3)-3t_2(t_2-1).
\end{aligned} \tag{2.5}$$

and define the following in $(\mathbb{R}[x_i])[t_1,t_2]$:

$$\begin{aligned}
F_0 &= x_0-3t_1(t_1-1)^2-(t_2-1)^3-3t_2,\\
F_1 &= x_1-3t_2(t_2-1)^2-t_1^3-3t_1,\\
F_2 &= x_2+3t_2(t_2^2-5t_2+5)t_1^3+3(t_2^3+6t_2^2-9t_2+1)t_1^2-\\
&\qquad t_1(6t_2^3+9t_2^2-18t_2+3)+3t_2(t_2-1),
\end{aligned} \tag{2.6}$$

and prepare the input file for `ResPol`:
2
7 6 14
[[0, 0], [0, 1], [1, 0], [0, 2], [2, 0], [0, 3], [3, 0], [0, 0], [0, 1], [1, 0], [2, 0], [0, 3], [3, 0], [0, 0], [0, 1], [1, 0], [0, 2], [1, 1], [2, 0], [1, 2], [2, 1], [1, 3], [2, 2], [3, 1], [2, 3], [3, 2], [3, 3]]

The first line is the dimension of the Newton polytopes of the input polynomials, the second line the cardinalities of their supports, and the third contains the coordinates of the support points. Alternatively, the second line we could explicitly specify the support points that define the projection of $N(\mathcal{R})$, by their order in the set of the third line: 7 6 14 | 0 7 13.
Notice that these correspond to the support points that are exponents of the terms of F_0, F_1, F_2 whose coefficient contains the implicit variables x_0, x_1, x_2. It takes `ResPol` 0.1 sec. to output the implicit polytope's vertices $(0,0,0)$, $(18,0,0)$, $(0,18,0)$, $(0,0,9)$; this polytope contains 715 lattice points.

Example 2. Consider the rational parametric curve known as folium of Descartes:

$$x_0 = \frac{3t^2}{t^3+1},\ x_1 = \frac{3t}{t^3+1}. \tag{2.7}$$

It is represented by the following polynomials in $(\mathbb{R}[x_i])[t]$:

$$F_0 = -x_0+3t^2-x_0t^3,\ F_1 = -x_1+3t-x_1t^3$$

The input to `ResPol` is a file with 3 lines:

Figure 2.1: Newton polytope and supports of F_0 and F_1 in Example 2.

1
3 3 | 0 2 3 5
[[0], [2], [3], [0], [1], [3]]

The first line gives the dimension, the second the support cardinalities and, then, the indices (0,2,3,5) of the support points in the set of the third line, which are exponents of monomials whose coefficient contains an implicit variable x_0 or x_1. These define the projection space of $N(\mathcal{R})$. The third line gives the coordinates of the support points.

Given this input, `ResPol` outputs 7 vertices in 4-dimensional space: $(0,0,2,1)$, $(3,0,0,3)$, $(0,3,3,0)$, $(1,2,0,0)$, $(1,0,0,1)$, $(0,2,2,0)$, $(0,0,2,1)$. The first 2 coordinates of these vertices correspond to input coefficients containing x_0, whereas the other two, to coefficients containing x_1. The implicit vertices are 2-dimensional: their coordinate corresponding to x_0 is the sum of the first two coordinates of the predicted vertices, and their coordinate corresponding to x_1, is the sum of the last two: $(0,3),(3,3),(3,0),(1,1),(2,2)$. This is used as input to our implicitization code.

In practice, `ResPol` proves to be inefficient when the dimension of the projection space exceeds 8. For polynomial parameterizations, this dimension is equal to the number of parametric equations, but for rational parameterizations, is equal to the number of monomials in the denominators of the parametric equations. We can overcome this difficulty by introducing as many additional variables as the number of different denominators that appear in the parametric equations. This raises the input dimension which has lesser effect to `ResPol`'s efficiency. This demonstrated below.

Example 3 (Cont'd from Example 2)**.** We introduce new variable w expressing the common denominator t^3+1 and rewrite the system:

(2.7) as:

$$x_0 = \frac{3t^2}{w},\ x_1 = \frac{3t}{w},\ w = t^3+1.$$

This is represented by the following polynomials in t, w:

$$F_0 = -x_0 w + 3t^2,\ F_1 = -x_1 w + 3t,\ F_2 = 1 - w + t^3.$$

The Newton polygons of the F_i's are shown in Fig. 3. The input to `ResPol` contains 3 lines:

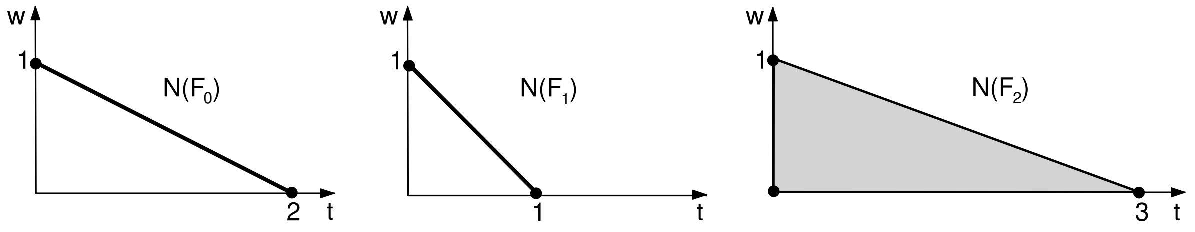

Figure 2.2: Newton polytopes of F_0, F_1, F_2 in Example 3.

2
2 2 3 | 0 2
$[[0,1],[2,0],[0,1],[1,0],[0,0],[0,1],[3,0]]$

`ResPol` gives the vertices $(0,3),(3,0),(3,3),(1,1)$ which do not require any transformation and are directly used in our implicitization routine.

2.3 Interpolation of the implicit equations

In this section we present our implicitization algorithm and give a detailed description of our interpolation process. We discuss selecting sample points best suitable for the given parametrization and solving method and propose possible solutions to the problem of multi-dimensional kernel space.

2.3.1 Problem of the implicitization

Given a set of parametric equations (1.3) the *implicitization problem* asks for the smallest algebraic variety containing the closure of the image of the parametric map $f : \mathbb{R}^n \to \mathbb{R}^{n+1} : t \mapsto f(t)$. This image is contained in the variety defined by the ideal of all polynomials p such that $p(f_0(t), \dots, f_n(t)) = 0$, for all t in Ω.

Our goal is computing its defining polynomial

$$p(x_0, \dots, x_n) = 0, \tag{2.8}$$

The variety in question can be regarded as the projection of the graph of map f to the last $n + 1$ coordinates. If f is polynomial, implicitization is reduced to eliminating t from the polynomial system

$$F_i := x_i - f_i(t) \in (\mathbb{R}[x_i])[t],\ i = 0, \dots, n,$$

seen as polynomials in t with coefficients which are functions of the x_i. This is also the case for rational parameterizations

$$x_i = f_i(t)/g_i(t),\ i = 0, \dots, n, \tag{2.9}$$

represented as polynomials in $(\mathbb{R}[x_0, \dots, x_n])[t, y]$:

$$\begin{aligned} F_i &:= x_i g_i(t) - f_i(t),\ i = 0, \dots, n, \\ F_{n+1} &:= 1 - y g_0(t) \cdots g_n(t), \end{aligned} \tag{2.10}$$

where y is a new variable and F_{i+1} assures that $g_i(t) \neq 0$.

In some cases of trigonometric parametrization, it can be converted to rational form by the standard half-angle transformation

$$\sin\theta = \frac{2\tan\theta/2}{1+\tan^2\theta/2},\ \cos\theta = \frac{1-\tan^2\theta/2}{1+\tan^2\theta/2},$$

where the parametric variable becomes $t = \tan\theta/2$.

2.3.2 Implicitization algorithm

The main steps of our algorithm are given below.

Input: Polynomial or rational parametrization $x_i = f_i(t_1, \dots, t_n)$.

Output: Implicit polynomial $p(x_i)$ in the monomial basis in $\mathbb{N}^{n+1}$.

- Step 1: Support prediction determines (a superset of) the implicit polytope vertices.
- Step 2: Compute all lattice points $S \subseteq \mathbb{N}^{n+1}$ in the polytope.
- Step 3: Repeat $\geq |S|$ times: Select value τ for t, evaluate $x_i(t)$, $i = 0, \dots, n$, thus evaluating each monomial with exponent in S. This yields a matrix M.

- Step 4: Given matrix M, solve $M\vec{p} = 0$.
- Step 5: Let $\mathrm{corank}(M) \geq 1$, then return the kernel vector $\vec{p}_i$ corresponding to a polynomial of the least total degree.
- Step 6: Return primitive part of polynomial corresponding to $\vec{p}^\top \cdot m$, where m is the set of monomials with exponent in S.

Comments:

1. Although in our work we use specific support prediction methods ([55, 49]), in principle any other technique for computing the implicit polytope can be applied as well.
2. For 2-D, 3-D and 4-D cases our `Maple` implementation includes routines that compute the lattice points in the implicit polytope; these utilize the `Maple` package convex [62]. Our `SAGE` implementation uses its build in functions for this task. For higher dimensions we have employed the software package `Normaliz2.7`.
3. We build a rectangular overconstrained matrix M when solving numerically in order to increase accuracy of solving.
4. An alternative approach for cases when $\mathrm{corank}(M) > 1$ is to factor the polynomial, corresponding to some $\vec{p}_i$ or computing polynomial GCD. This approach is preferable in the admittedly rare instances when all the potencial implicit polynomials are of the same total degree.

Algorithm 1: Sparse Implicitization

```
Input : Polynomial or rational parametrization $x_i = f_i(t),\ i = 0, \ldots, n$,
        Predicted implicit polytope $Q$, if $n \geq 2$
Output: Implicit polynomial $p(x_0, \ldots, x_n)$ in its monomial basis.
$\mathbb{N}^{n+1} \supseteq S \leftarrow$ lattice points in $Q$
foreach $s_i \in S$ do $m_i \leftarrow x^{s_i}$    // $x = (x_0, \ldots, x_n)$
$\vec{m} \leftarrow (m_1, \ldots, m_{|S|})$    // vector of monomials in $x$
Initialize $\mu \times |S|$ matrix $M$, $\mu \geq |S|$:
for $i \leftarrow 1$ to $\mu$ do
    select $\tau_i \in \mathbb{C}^{n+1}$
    for $j \leftarrow 1$ to $|S|$ do
        $M_{ij} \leftarrow m_j|_{t=\tau_i}$
$\{\vec{v}_1, \ldots, \vec{v}_k\} \leftarrow$ Basis of Nullspace$(M)$
if $k = 1$ then $p \leftarrow g_1$
else
    for $i \leftarrow 1$ to $k$ do $g_i \leftarrow$ primpart$(\vec{v}_i \cdot \vec{m})$ // inner product
    $p \leftarrow$ gcd$(g_1, \ldots, g_k)$
return $p$
```

While we exploit classical approach to interpolation, our contribution is, first, exploration of the overconstrained systems, and, second, handling of the cases when the kernel vector space is multidimensional.

2.3.3 Complexity

The complexity of the support prediction algorithm used in our work was given in [49, thm.10]:

Theorem 8. *Let Π denote some orthogonal projection of $N(\mathcal{R})$, $|\Pi|$ and $|\Pi^H|$ the number of vertices and facets of Π, $\mathcal{A}$ a pointset, $s_{\mathcal{A}}$ size of the largest triangulation of $\mathcal{A}$, s_Π that of Π.*

The time complexity of the algorithm for finding projections of resultant polytopes to compute $\Pi \subset \mathbb{R}^m$ is $O(m^5|\Pi|s_\Pi^2 + (|\Pi| + |\Pi^H|)n^5|\mathcal{A}|s_{\mathcal{A}}^2)$, which becomes $O^*(|\Pi|s_\Pi^2)$ when $|\Pi| \gg |\mathcal{A}|$.

Here and further $O^*(\cdot)$ denotes asymptotic bounds when ignoring polylogarithmic factors in the arguments.

The second part of the procedure is the computation of the lattice points contained in the predicted polytope. It is a NP-hard problem to detect a lattice point in a polytope when the dimension of the polytope is an input variable. When the dimension is fixed the algorithm in [9] counts the number of lattice points in a polytope within polynomial time in the size of the input. The software LattE [81] implements Barnivok's algorithm. The software package Normaliz [12] computes lattice points in polytopes, and is very fast in practice; this is the one interfaced to our software. Based on these algorithms, one can enumerate all lattice points in output-sensitive manner, i.e. in polynomial time in the output size, which of course can be exponential in the input size. The computation up to this point is essentially offline, because it does not require knowledge of the specific coefficients.

Suppose that, for the predicted support S, the exponent of every monomial in the i-th variable lies in $[0, \delta]$, for $i = 1, 2, \ldots, n$.

Proposition 9. [57, lem.4.3] *Consider a set S of monomials in n variables. Given n scalar values $p_1, p_2, \ldots, p_n$, the algorithm of [57] evaluates all the monomials of S at these values in $O^*(|S|n + n\sqrt{\delta})$ arithmetic operations and $O(|S|n)$ space.*

Now, we arrive at the complexity of constructing a $\mu \times |S|$ matrix M, with columns indexed by $|S|$ monomials and rows indexed by μ values.

Corollary 10. *Assume our algorithm builds a rectangular matrix $\mu \times |S|$, $\mu \geq |S|$. Then, all $\mu\,|S|$ entries are computed in $O^*(\mu\,|S|\,n)$ operations.*

Once the matrix is constructed, the kernel computation costs $O(|S|^{2.376})$ arithmetic operations on square matrices, which follows from the current record for matrix multiplication. An improved probabilistic bound is $O(\sigma^2\delta)$ following from Zippel's sparse interpolation algorithm, supposing the number of variables n is constant, where δ bounds the implicit degree per variable.

On $\mu \times |S|$ rectangular matrices, the numeric kernel computation has complexity $O(\mu\,|S|^2)$, by linear algebra. Thus we arrive at the following.

Theorem 11. *The overall complexity of our implicitization algorithm requires $O(\mu\,|S|^2)$ arithmetic operations.*

Proof. This is a higher complexity than the case of square matrices ($O(|S|^{2.376})$). The first phase of matrix construction with its cost $O^*(\mu\,|S|\,n)$ (Cor. 10) is clearly dominated. Hence the overall complexity of the algorithm $O(\mu\,|S|^2)$. □

The complexity of interpolating resultants is $O^*(|S|^2)$ where S is the set of lattice points in the predicted resultant support, because the dominating stage is a kernel computation for a structured matrix M. Using Weidemann's approach, the main oracle is post-multiplication of M by a vector, which amounts to evaluating a $(n+1)$-variate polynomial at *chosen* points, and this can be done in quasi-linear complexity in $|S|$ [114, 88]. For certain classes of polynomial systems, when one computes the resultant in one or more parameters, this may be competitive to current methods for resultant computation. The best such methods rely on developing the determinant of a resultant matrix in these parameters [18, 37]. The matrix dimension is in $O^*(t^n \deg \mathcal{R})$ [47], where $\deg \mathcal{R}$ is the total degree of $\mathcal{R}$ in all input coefficients, and t is the scaling factor relating the input Newton polytopes, which is bounded by the maximum degree of the input polynomials f_i in any variable. Then, developing *univariate* resultants has complexity in $O^*(t^{3.5n}(\deg \mathcal{R})^{3.5})$ [47, 57, 58]. Hence, our approach improves the complexity when the predicted support is small compared to t and $\deg \mathcal{R}$.

2.3.4 Building the matrix

We seek to construct the interpolation matrix M whose corank, or kernel space dimension, equals 1, i.e. its rank is 1 less than its column dimension. To cope with numerical issues, especially when computation is approximate, we construct a rectangular matrix M by choosing $\mu \geq |S|$ values of τ; this overconstrained system increases numerical stability.

In order to fully exploit the structure of the output provided by some of the support predicting software we have been approaching constructing of the matrix in a different ways. While the main and most often used matrix structure was the one indexed by the potential monomials, as given in the Algorithm 1 here we would like to outline also its alternative.

We have tried a support prediction method that applies only to curves [55] and two support prediction methods applicable to both planar curves and (hyper)surfaces: first, that computes resultant polytope [48], and second, improved, that computes implicit polytope [49]. While the curves-only and the second general methods provide us with a (super)set of the implicit vertices, which leads to straightforward matrix construction, output of the first general is slightly different. It returns a set of vertices of the polytope $N(\varphi(\mathcal{R}))$ of the specialized resultant $\varphi(\mathcal{R})$, where $\mathcal{R}$ is the resultant of the system of polynomials in (2.1). From here we can derive implicit polytope vertices and construct the matrix as usual, or we can built a matrix indexed by vertices of the resultant polytope.

Consider the polynomials $F_i,\ i = 0, \dots, n$ in Equation (2.10), with symbolic coefficients c_{ij}:

$$F_i = \sum_{j=1}^{d_i} c_{ij} t^{a_{ij}},$$

where d_i is the number of monomials in t with non-zero coefficient, i.e. the cardinality of support A_i. Given supports A_i, the output is the set V of vertices of the Newton polytope of the resultant of the F_i's. Each resultant vertex $a_i = (a_{i1}, \dots, a_{id}) \in V$ is a d-dimensional vector, where $d = \sum_{i=0}^{n} d_i$, and corresponds to an extreme resultant monomial c^{a_i}, where c is the d-dimensional vector of the symbolic coefficients of all F_i's:

$$\begin{aligned} c = &(c_1, \dots, c_d) = \\ &\underbrace{(c_{01}, \dots, c_{0d_0}}_{\text{from } F_0}, \underbrace{c_{11}, \dots, c_{id_1}}_{\text{from } F_1}, \dots\dots, \underbrace{c_{n1}, \dots, c_{nd_n})}_{\text{from } F_n}. \end{aligned}$$

Given the resultant polytope vertices, we compute the set S of all m lattice points in the resultant polytope. We apply specialization φ to the set of monomials in c_{ij}'s with exponents in S to obtain a set of polynomials (products of linear polynomials) in x_i's. We abuse notation and denote this set also by $\varphi(S)$, i.e. we identify each $\varphi(c)^{a_k}$ with its exponent a_k.

When we substitute the symbolic coefficients c_{ij} by the actual univariate polynomials in x_i, some of the vertices of the implicit polytope may map to identical expressions: $\exists a_k \neq a_l \in V$ s.t. $\varphi(c)^{a_k} = \varphi(c)^{a_l}$. By examination of the $\varphi(c)^{a_k}$, we remove duplicates. In the following we assume that $\varphi(S)$ has no multiple entries.

Matrix M is constructed in a following way. Let $\varphi(S) = \{a_1, \ldots, a_m\} \subset \mathbb{N}^d$ be the lattice points, which form a superset of the resultant support. Each column contains a product $\varphi(c)^{a_k} = \varphi(c_1^{a_{k1}} \cdots c_d^{a_{kd}})$ evaluated at various τ_k, for $k = 1, \ldots, m$. Thus, we define the following $m \times m$ matrix, with columns indexed by the a_k's:

$$M = \begin{array}{cccc} (a_{11}, \ldots, a_{1d}) & \ldots & (a_{m1}, \ldots, a_{md}) & \\ \left[\begin{array}{c} \varphi(c_1^{a_{11}} \cdots c_d^{a_{1d}})(\tau_1) \\ \vdots \\ \varphi(c_1^{a_{11}} \cdots c_d^{a_{1d}})(\tau_m) \end{array}\right. & \begin{array}{c} \cdots \\ \cdots \\ \cdots \end{array} & \left.\begin{array}{c} \varphi(c_1^{a_{m1}} \cdots c_d^{a_{md}})(\tau_1) \\ \vdots \\ \varphi(c_1^{a_{m1}} \cdots c_d^{a_{md}})(\tau_m) \end{array}\right] & \begin{array}{c} \tau_1 \\ \vdots \\ \tau_m \end{array} \end{array} \quad (2.11)$$

Each $\varphi(c)^{a_k} \in \varphi(S)$ is a product of linear polynomials in x_i. After expanding and simplifying, we get a set of monomials in x_i's.

Example 4. Let as illustrate matrix construction using simple classical curve "witch of Agnesi". It has rational parametric representation $x = at, y = a/(1 + t^2)$.

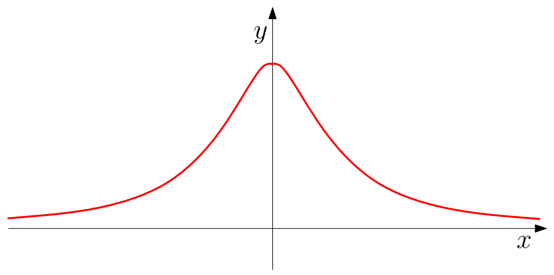

Figure 2.3: Algebraic curve "witch of Agnesi" (Example 4).

We apply general support prediction method [49], providing as input coefficients in respect to parameter t of the polynomials: $c_{00} + c_{01}t = 0$, $c_{10} + c_{11}t^2 = 0$, namely $c_{00} = -x, c_{01} = a, c_{10} = a - y, c_{11} = -y$, and corresponding supports $\{0, 1\}, \{0, 2\}$. The resultant polytope is the segment $[(0, 2, 1, 0)(2, 0, 0, 1)]$ which contains no internal points. These two vectors index the columns of M and correspond to the monomials $\{c_{01}^2 c_{10},\ c_{00}^2 c_{11}\}$, which are specialized to $\{a^2(a - y),\ (-x)^2(-y)\}$, that index the 2×2 matrix M. Thus,

$$M = \begin{bmatrix} a^3 - a^2 y_0 & -{x_0}^2 y_0 \\ a^3 - a^2 y_1 & -{x_1}^2 y_1 \end{bmatrix}.$$

where x_0, x_1, y_0, y_1 are values of $x(t), y(t)$ obtained by using random t. We solve $M \times \vec{p} = 0$ to find $\vec{p} = [p_0, p_1]$, yielding the implicit polynomial

$$p_0(a^3 - a^2 y) + p_1(-x^2 y),$$

hence $a^2 y + x^2 y - a^3$.

Alternatively, from the set $\{a^2(a - y),\ (-x)^2(-y)\}$ we can obtain the implicit polytope vertices, implicit support in x, y: $\{(0, 0), (0, 1), (2, 1)\}$. Its convex hull contains no internal points, hence M' is 3×3.

$$M' = \begin{bmatrix} c & y_0 & x_0^2y_0 \\ c & y_1 & x_1^2y_1 \\ c & y_2 & x_2^2y_2 \end{bmatrix}.$$

where $x_0, x_1, y_0, y_1, x_2, y_2$ are values of $x(t), y(t)$ obtained by random t and c is a constant.

Notice that the matrix, indexed by c_{ij}'s was smaller. Same is true for the larger examples, as illustrated below.

Example 5. Consider rational parametric curve "Folium of Descartes".

$$x = \frac{3at}{1+t^3},\ y = \frac{3at^2}{1+t^3}.$$

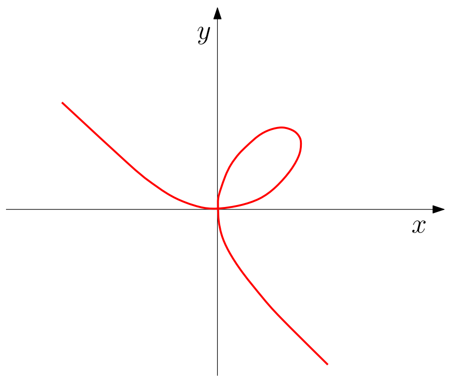

Figure 2.4: Algebraic curve "folium of Descartes".

The curve support prediction yields polygon vertices $(0,0)$, $(0,3)$, $(1,1)$, $(3,0)$, thus 10 lattice points totally. Solving the 10×10 matrix yields 3 nonzero coefficients, and implicit polynomial x^3+y^3-3axy.

General support prediction: The parametrization is represented by the polynomials $3at - x - xt^3$, $-yt^3-y+3at^2$ with supports $\{0,1,3\}, \{0,2,3\}$. Then, φ is defined as: $c_{00}=-x$, $c_{01}=3a$, $c_{02}=-x$, $c_{10}=-y$, $c_{11}=3a$, $c_{12}=-y$.

The method computes 6 vertices of the (symbolic) resultant polytope: $(0,0,3,3,0,0)$, $(0,2,1,1,2,0)$, $(0,3,0,1,0,2)$, $(2,0,1,0,3,0)$, $(2,1,0,0,1,2)$, $(3,0,0,0,0,3)$, which contains 4 inside points.

The straightforward approach would be to form a 10×10 matrix M. In this case, $\vec{p} = [1, p_1, p_1, -1-p_3-p_5, -p_2-p_4, p_1, p_2, p_3, p_4, p_5]$, where $p_i \in \mathbb{N}^*$. We can safely reduce matrix size by keeping only the distinct monomials in x, y, namely $a^4x_1y_1$, ${x_1}^3a^3$, $a^3{y_1}^3$, ${x_1}^2a^2{y_1}^2$, ${x_1}^3{y_1}^3$.

So we get a 5×5 matrix.

$$M = \begin{bmatrix} 81a^4x_1y_1 & -27{x_1}^3a^3 & -27a^3{y_1}^3 & 9{x_1}^2a^2{y_1}^2 & {x_1}^3{y_1}^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 81a^4x_5y_5 & -27{x_5}^3a^3 & -27a^3{y_5}^3 & 9{x_5}^2a^2{y_5}^2 & {x_5}^3{y_5}^3 \end{bmatrix}$$

where $x_1,\ldots,x_5, y_1,\ldots,y_5$ are values of x, y obtained by evaluating at (random) $t_1,\ldots,t_5$.

Solving $M\cdot\vec{p}=0$ gives 3 nonzero coefficients, and x^3+y^3-3axy.

Table 2.1: Comparing two matrix construction methods for the first general support prediction method.

Curve	Impl.	Resultant		Monomials	
	degree	matrix	nonzero	matrix	nonzero
Circle	2	4×4	4	9×9	3
Conchoid	4	6×6	6	15×15	6
Folium of Descartes	3	5×5	3	11×11	3
Witch of Agnesi	3	2×2	2	4×4	3
Surface					
Infinite cylinder	2	4×4	4	9×9	3
Hyperbolic paraboloid	2	3×3	3	7×7	3
Infinite cone	4	8×8	6	19×19	3
Whitney umbrella	3	2×2	2	4×4	2
Monkey saddle	3	3×3	3	8×6	3
Handkerchief surface	3	2×2	2	4×4	2
Crossed surface	4	5×5	5	10×10	5
Quartoid	4	4×4	4	16×16	4
Peano surface	4	4×4	4	10×10	4
Swallowtail surface	5	12×12	12	25×25	6
Sine surface	6	87×87	66	125×125	7

When substituting c_{ij} by the corresponding univariate polynomials we get 5 distinct monomials in x, y: x^3y^3, x^2y^2, xy, y^3, x^3. The convex hull of the supports of these monomials is defined by the vertices: $(3,0)$, $(0,3)$, $(1,1)$, $(3,3)$. We find 11 lattice points and build 11×11 matrix M.

Table 2.1 shows more examples comparing two matrix construction methods. However, the support prediction method in question proved to be time consuming and suitable only for very small examples. Its more effective alternative integrates projection from the resultant polytope to the implisit polytope, its output vertices of$N(\varphi(\mathcal{R}))$.

We compute all lattice points s_j contained in $N(\varphi(\mathcal{R}))\subset\mathbb{N}^{n+1}$ to obtain the set $S=\{s_1,\ldots,s_{|S|}\}$; each $s_j=(s_{j0},\ldots,s_{jn})$ is an exponent of a (potential) monomial $m_j=x^{s_j}=x_0^{s_{j0}}\cdots x_n^{s_{jn}}$ of the implicit polynomial, where x_i is given in (2.9). We evaluate $m_j, j=1,\ldots,|S|$ at some $\tau_k,\ k=1,\ldots,\mu,\ \mu\geq|S|$; we use $\mu>|S|$ evaluation points to improve the numerical stability of our algorithm. Let $m_j|_{t=\tau_k}=\prod_i\left(\frac{f_i(\tau_k)}{g_i(\tau_k)}\right)^{s_{ji}}$ denote the evaluated j-th monomial m_j at τ_k. Thus, we construct an $\mu\times|S|$ matrix M with rows indexed by $\tau_1,\ldots,\tau_\mu$ and columns by $m_1,\ldots,m_{|S|}$:

$$M=\begin{bmatrix} m_1|_{t=\tau_1} & \cdots & m_{|S|}|_{t=\tau_1}\\ \vdots & \cdots & \vdots\\ m_1|_{t=\tau_\mu} & \cdots & m_{|S|}|_{t=\tau_\mu}\end{bmatrix}$$

By construction of matrix M using sufficiently generic values τ, which thus correspond to well-distributed points on the parametric hypersurface, we have the following:

Lemma 12. *Any polynomial in the basis of monomials indexing M, with coefficient vector in the kernel of M, is a multiple of the implicit polynomial.*

Corollary 13. *Assume that the predicted polytope equals the actual one and that we construct a $\mu \times |S|$ matrix M, as above. Then, the vector of the implicit coefficients lies in the matrix kernel, hence rank$(M) < |S|$. If the points $x(\tau_i)$, $i = 1, \ldots, \mu$ are sufficiently generic, then M has corank 1, i.e. rank$(M) = |S| - 1$. Then, if we solve $Mp = 0$ for p, such that one of its entries is set to 1, this yields the coefficients of the implicit equation in a unique fashion.*

Note that the above result follows also from relation (2.4). In view of Lemma 12, we have the following corollary of Proposition 6.

Corollary 14. *Consider the irreducible factor of the polynomial obtained from any kernel vector of matrix M, that corresponds to the implicit equation. Its Newton polytope is the implicit polytope and it is bounded by the halfspaces defined in Proposition 6.*

As a consequence, we cannot directly apply the degree bounds on the predicted polytope since we do not know apriori its Minkowski summands; notice that the Newton polytopes of the factors are Minkowski summands of the implicit polytopes. However, we can do so if the intersection of the predicted polytope with the halfspaces defined by the degree bounds contains (a translate) of the implicit polytope.

2.3.5 Matrix evaluation

The goal of our work is developing of the implicitization tool suitable for a wide specter of parametric representations. However, different types of problems call for the different approaches; wherever we want to compute exact or approximate implicit equation adjustments are needed in order to insure better performance. In particular, result is affected by the way we pick values for evaluation when building a matrix.

First of all, we made sure to avoid values that make the denominators of the rational parametric expressions close to 0.

We have experimented with both integer and complex values. In the former case, we have used random integers (for parametrizations we discard values that makes some denominator vanish). We also tried complex values for τ: roots of unity, and random complex numbers of modulus equal to 1. It is important to pick sufficiently generic values τ, so that they correspond to well-distributed points on the curve/(hyper)surface. When we operate with Bézier curves or other geometric objects represented in Bernstein basis, it is essential to ensure that τ falls into the region of interest of the parametric function [52].

Exact solving

Our experiments of computing exact implicit equation (functions `LinearSolve`, `NullSpace` in `Maple`, `kernel` in `SAGE`) have lead us to conclusion that evaluation by random integers provides best results. Random rational values evaluated as floats may not be handled by the aforementioned solvers well, for the lost presicion is interpreted as linear system being overconstrained. Complex values used in evaluation introduce non-rational coefficients into the resulting expression. In comparison, the only downside of using integers is that building large matrizes their entries grow fast.

Comparing `Maple` solvers shows that `LinearSolve` is faster than `NullSpace` (see 2.2, 2.3).

Curve	Exact			#impl. monom.
	`NullSpace`	`LinearSolve`	matrix size	
Descartes' Folium	0.016	0.012	5×5	3
Tricuspoid	0.076	0.044	15×15	8
Talbot's curve	1.625	0.324	28×28	8
Nephroid	1.656	0.312	28×28	10
Fifth heart	5.3	0.104	33×33	43
Trifolium	19.7	0.26	45×45	37
Ranunculoid	8414.8	1.376	91×91	43

Table 2.2: Runtimes (sec), matrix size and number of terms in the resulting polynomial for curves.

Surface	Solving method		Matrix size	# implicit monomials
	`NullSpace`	`LinearSolve`		
Quartoid	0.06	0.036	16×16	4
Peano	0.028	0.024	10×10	4
Swallowtail	0.24	0.108	25×25	6
Sine	2224.5	1.164	125×125	7
Bohemian dome	2150.23.4	1.181	125×125	7
Enneper	310.14	0.766	103×103	23
Bicubic surface	$>$ 4hours	42.059	715×715	715

Table 2.3: Runtimes (sec), matrix size and number of terms in the resulting polynomial for for surfaces.

Numeric solving

In order to compute approximate implicit equation we apply singular value decomposition (SVD), known as an accurate and numerically stable method for computing an orthonormal basis of a range or a null space.

Definition 10. A matrix $M \in \mathbb{R}^{n\times m}$, $n,\ m \in \mathbb{Z}$ can be written as

$$M = (U\Sigma V^{\top}),$$

where $\Sigma \in \mathbb{R}^{n\times m}$ is diagonal, $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices. This factorization of M is called *singular value decomposition*.

The diagonal entries $\sigma_1, \dots, \sigma_r$ of matrix Σ are non-negative and in decreasing order $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r \geq 0$, where $r = min(m, n)$. They are known as *singular values*. The columns of the matrices U and V are called respectively *left* and *right singular vectors*. They are orthogonal and have unit length.

A basis of null(M) consists of the last rows of $V^{\top}$ corresponding to the zero singular values of M. When corank$(M) = 1$, the last row of $V^{\top}$ corresponds to $\vec{p}$.

When examining approximate methods we used the ratio of the last two singular values σ_m/σ_{m-1}, small ratio indicates that the matrix M is close to having corank 1. Also, we employ the *condition number* of the matrix $\kappa(M) = |\sigma_1/\sigma_{|S|}|$, to measure the sensitivity of the result to small changes in the input data. Large condition number signifies that even small changes in the input can result in a big change in the output, i.e. the matrix is ill-conditioned.

Table 2.4 shows representative timings about these options, which we examined with our implementation on Maple, optimized for the specific task. Our experiments show that runtimes do not vary significantly in small examples but in larger ones, the best results are given by random integers for the exact method, and unitary complexes and roots of unity evaluated as floats for the numeric method, with the former having a slightly better overall performance over the latter, both in terms of stability and speed.

Curve	implicit degree	lattice points	SVD (σ_m/σ_{m-1}) root of 1	unitary $\mathbb{C}$	rand.$\mathbb{Z}$
Folium	3	5	0.032 (10^{-12})	0.012 (10^{-12})	0.012 (10^{-13})
Conchoid	4	10	0.112 (10^{-14})	0.144 (10^{-14})	0.072 (10^{-13})
Bean curve	4	13	0.352 (10^{-11})	0.4 (10^{-14})	0.116 (10^{-13})
Tricuspoid	4	15	0.356 (10^{-14})	0.028 (10^{-14})	0.168 (10^{-14})
Cardioid	4	15	0.28 (10^{-14})	0.29 (10^{-14})	0.2 (10^{-12})
Nephroid	6	28	0.248 (10^{-15})	0.17 (10^{-34})	0.192 (10^{-21})
Talbot's curve	6	28	0.416 (10^{-14})	0.132 (10^{-17})	0.196 (10^{-16})
Trifolium	4	45	0.284 (10^{-34})	0.188 (10^{-77})	0.876 (10^{-20})
Fifth heart	8	33	0.224 (10^{-37})	0.124 (10^{-56})	0.63 (10^{-26})
Ranunculoid	12	91	3.764 (10^{-359})	2.224 (10^{-58})	79.853 (10^{-2})

Table 2.4: Comparison of matrix evaluation methods. Runtimes on Maple (sec), whereas the parenthesis contains $\sigma_{|S|}/\sigma_{|S|-1}$.

Geometric objects of Bernstein basis

Representation of the geometric objects which uses the Bernstein polynomials as basis is a popular industrial standard. Usually it is impossible to apply symbolic-computation-based implicitization for such parametrizations, however CAGD implementations of Bézier curves and NURBS patches represented using floating point arithmetic present interesting examples for numerical solving. With assistance from O. Barrowclough, SINTEF, we have tested our method against industrial examples of Bernstein basis geometric objects.

Our experiments show that the most important is to ensure that evaluation values belong to the region of interest of the parametrization. We have successfully tried evaluation by rational numbers, random or uniform as well as complex numbers. It appears that evaluating by rational values provides best results in the terms of speed and approximation quality. In fact, uniform values of specific distribution, allow to minimize approximation error. In particular, following [8] we have tried evaluation by Chebyshev nodes in the interval $[0,1]$,

$$\tau = \frac{1}{2} + \frac{1}{2}\cos\left(\frac{2i-1}{2n}\pi\right),\ i = 1,\ \dots\ , n.$$

2.4 Multidimensional kernel space

In general, by using correctly selected evaluation values τ we ensure that the matrix M has corank $= 1$.

Degree	Matrix size	Random rational		Chebyshev points		Roots of 1	
		runtime	cond.num.	runtime	cond.num.	runtime	cond.num.
3	7×7	0.04	10^4	0.04	10^4	0.04	10^4
3	10×10	0.16	10^5	0.19	10^4	0.48	10^{10}
4	15×15	0.11	10^8	0.1	10^5	0.22	10^8
7	36×36	0.43	10^4	0.41	10^4	0.93	10^9

Table 2.5: Comparison of matrix evaluation methods for rational Bézier curves. Runtimes on Maple (sec).

Degree	Matrix size	Random rational		Chebyshev points		Roots of 1	
		runtime	cond.num.	runtime	cond.num.	runtime	cond.num.
3	6×6	0.08	10	0.06	10	0.07	10
6	22×22	0.35	10^2	0.35	10^3	0.84	10^3
9	129×129	6.93	10^6	6.28	10^{15}	14.84	10^{14}
14	139×139	8.65	10^4	9.34	10^{13}	21.58	10^9

Table 2.6: Comparison of matrix evaluation methods for NURBS patches. Runtimes on Maple (sec).

However, for some inputs we obtain a matrix of corank > 1 when the predicted polytope Q is significantly larger than the actual one. We formalize this concept in Thm. 15 and its corollaries. It can be explained by the nature of our method: we rely on a *generic* resultant to express the implicit equation, whose symbolic coefficients are then specialized to the actual coefficients of the parametric equations. If this specialization is not generic, then the resulting implicit equation divides the specialized resultant.

We address such cases by computing the gcd of 2 or more polynomials g_i obtained from kernel vectors. There exist many algorithms for the exact [96, 97] or approximate gcd of multivariate polynomials. The first approximate approach, given polynomials f, g and error tolerance $\varepsilon > 0$, computes the maximum degree gcd of polynomials $\hat{f}, \hat{g}$ where $|f - \hat{f}|, |g - \hat{g}| < \varepsilon$ [50]. The second minimizes ε such that $\hat{f}, \hat{g}$ have gcd of at least a given degree r [75]. Another approach is to consider a pair of univariate polynomials of degree n and let d, e be non-negative integers. It is shown in [115] that allowing perturbations of (f_0, f_1) by addition of a pair (u, v) of polynomials of degrees at most e then the problem of computing $\gcd(f + u, g + v)$ of degree $\geq d$, has at most one solution, and if one exists, it can be computed in polynomial time. There exist similar techniques for several univariate polynomials [46]. Our software uses Maple's command `gcd` for exact, and package ApaTools[2] for approximate gcd computations.

Example 6 (Cont'd from Example 2)**.** The method in [55] yields 3 implicit polytope vertices: $(1,1)$, $(0,3)$, $(3,0)$. This polygon contains 5 lattice points which yield the potential implicit monomials y^3, xy, xy^2, x^2y, x^3 indexing the columns of matrix M in this order. The kernel of M is spanned by vector $[1, -3, 0, 0, 1]$; the implicit equation is $x^3 - 3xy + y^3$.

If we change the parametrization, substituting t by t^2, we obtain

$$x_0 = \frac{3t^4}{t^6 + 1}, x_1 = \frac{3t^2}{t^6 + 1},$$

[2]http://neiu.edu/∼zzeng/apatools.htm [122]

then the algorithm in [55] predicts an implicit polytope with vertices: $(2,2),(0,6),(6,0)$, containing 12 lattice points. We build a matrix M of size $\mu \times 12\,(\mu \geq 12)$; of corank 5. The polynomials corresponding to its kernel vectors are: $g_1 = x^2y^4 - 3x^3y^2 + x^5y$, $g_2 = -y^6 + 6xy^4 - 9x^2y^2 + x^6$, $g_3 = xy^5 - 3x^2y^3 + x^4y^2$, $g_4 = xy^4 - 3x^2y^2 + x^4y$, $g_5 = y^6 - 3xy^4 + x^3y^3$. Their gcd is the implicit equation.

Example 7 (Unit Sphere)**.** Consider its parameterization:

$$x_0 = \frac{2s}{1+t^2+s^2},\ x_1 = \frac{2st}{1+t^2+s^2},\ x_2 = \frac{-1-t^2+s^2}{1+t^2+s^2}.$$

`ResPol` predicts an implicit polytope with vertices: $(0,0,0)$, $(0,0,2)$, $(0,0,4)$, $(0,2,0)$, $(0,4,0)$, $(4,0,0)$. Implicit equation of the sphere being quadratic, here implicit polytope $P \subset Q$, where Q is predicted polytope. which contains the actual implicit polytope. It contains 35 lattice points. We build M of size $\mu \times 35\,(\mu \geq 35)$ of corank 10. The polynomials corresponding to the kernel vectors are:

$g_1 = -y^2 + y^2z^2 + y^4 + x^2y^2$,
$g_2 = -z^2 + z^4 + y^2z^2 + x^2z^2$,
$g_3 = -1 + z^2 + x^2 + y^2$,
$g_4 = -x + xz^2 + xy^2 + x^3$,
$g_5 = -yz + yz^3 + y^3z + x^2yz$,
$g_6 = -y + yz^2 + y^3 + x^2y$,
$g_7 = -xz + xz^3 + xy^2z + x^3z$,
$g_8 = -z + z^3 + y^2z + x^2z$,
$g_9 = -xy + xyz^2 + xy^3 + x^3y$,
$g_{10} = -1 + 2z^2 - z^4 + 2y^2 - 2y^2z^2 - y^4 + x^4$.

Computing the gcd of two randomly chosen polynomials yields, either the actual implicit equation $p = -1 + z^2 + x^2 + y^2$, or a multiple of p of degree 3.

Let us have a closer look at the numeric solving in the case of $\dim(\text{kernel}(M)) = 10$. Applying SVD in we obtain approximate results, i.e. polynomials with non-integer coefficients. Computing the kernel of M approximately yields polynomials with real coefficients.

The approximate gcd of the first two is:

$-0.9999998548199414 + 0.9999999857259533x^2 + 1.000000000052092y^2 + 1.000000000000000z^2$,
which is accurate to 7 decimal digits.

The following theorem establishes the relation between the dimension of the kernel space of M and the accuracy of the predicted support. It remains valid even in the presence of base points. In fact, it also accounts for them since then, Q is expected to be much smaller of P.

Theorem 15. *Let $P = N(p)$ be the Newton polytope of the implicit equation, and Q the predicted polytope. Then, assuming M has been built using sufficiently generic evaluation points, the dimension of its kernel space equals $\#\{m \in \mathbb{Z}^n : m + P \subseteq Q\} = \#\{m \in \mathbb{Z}^n : N(x^m \cdot p) \subseteq Q\}$.*

Proof. By Lem. 12, the kernel space of M contains the coefficient vectors $\vec{c}$ of all polynomials of the form fp, where $N(fp) \subset Q$, or, equivalently, $N(f) + N(p) \subset Q$.

Now, assume that there are r elements $a_1, \ldots, a_r \in \mathbb{Z}^n$ such that $N(x^{a_i} \cdot p) \subseteq Q$ and let $g_i = x^{a_i}p$, $i = 1, \ldots, r$. Then the coefficient vector $\vec{c}_i$ of g_i lies in the kernel space of M

because g_i vanishes on all evaluation points $m_i(\tau_i),\ i = 1, \dots, k$ used for constructing M, since p vanishes on these points. Moreover, the vectors $\vec{c_i}$ in the set $\{\vec{c_1}, \dots, \vec{c_r}\}$ are linearly independent. Obviously, every coefficient vector $\vec{c}$ of a polynomial of the form fp, where $N(fp) \subset Q$, can be written as a linear combination of the vectors $\vec{c_i}$, hence corank$(M) = r$. $\square$

Let the P, Q be as in Thm. 15 and assume $Q \supseteq P + R$, where R contains r lattice points and is maximal wrt the previous inclusion, i.e. if $R' \supsetneq R$, then $Q \subsetneq P + R'$; R can be a point.

Corollary 16. *Consider the set of polynomials as an $\mathbb{R}$-vector space in the monomial basis and let I be the $\mathbb{R}$-vector space generated by all polynomials of the form $pf \in \mathbb{R}[x_0, \dots, x_n]$, such that $NP(f) \subseteq R$. Assuming generic values for τ's, then corank$(M) = \dim_{\mathbb{R}}(I)$.*

Proof. I is generated, as an $\mathbb{R}$-vector space, by polynomials $x^{m_i}p, i = 1, \dots, r$, where $m_i \in \mathbb{Z}^n$ are lattice points in R and $\dim_{\mathbb{R}}(I) = \#\{m \in \mathbb{Z}^n : NP(x^m \cdot p) \subseteq Q\}$. Therefore, corank$(M) = \dim_{\mathbb{R}}(I)$. $\square$

Let $A_i,\ i = 0, \dots, n+1$ be the supports of the polynomials F_i and consider the generic polynomials

Corollary 17. *Let M be the matrix from Alg. 1, built with sufficiently generic evaluation points, and suppose the specialization of the polynomials in* (2.1) *to the parametric equations is sufficiently generic. Let $v_1, \dots, v_k$ be a basis of the kernel of M and $g_1, \dots, g_k$ be the corresponding polynomials (Step 4 of Alg. 1). Then the gcd of $g_1, \dots, g_k$ equals the implicit equation.*

Proof. This Corollary is deduced straight forward from Cor. 16. $\square$

Some examples where M is of corank > 1 are shown in the following tables; Table 2.7, contains parametric and implicit representations of the classic algebraic curves and surfaces. Table 2.8 shows: the number of the lattice points of the actual implicit polytope, the degree and the number of monomials in the implicit equation, the number of its lattice points of the predicted implicit polytope, the corank of matrix M, and the number of polynomials g_i of a certain degree (in parenthesis) obtained from the kernel vectors. It is obvious, that as the degree and the number of polynomials g_i of that degree, grows large, then more gcd operations are required to obtain the implicit equation or its multiple of lower degree.

Table 2.9 shows distribution of the polynomials corresponding to the kernel vectors in terms of degree and number of monomials; number of instances, when the gcd of two polynomials corresponding to kernel vector entries chosen randomly has certain degree and probability of the gcd being actual implicit equation; number of instances and a probability of the gcd of the result of the previous operation and a randomly chosen polynomial.

Consider tables 2.8 and 2.9. We see that almost in all the cases actual implicit expression is present among the polynomials, corresponding to the kernel vectors. It is the polynomial of the least degree. Notable exception here is the hypercone, where for all the 84 kernel vectors polynomials have same degree. Such distribution calls for the following improvement of our algorithm: computing gcd of the two polynomials of the least degree. In the case of the numeric solving we propose filtering out terms with the coefficients that are close to zero, sorting polynomials by degree then computing approximate gcd.

However, this measure is inefficient if the problem we are solving has similar characteristics to that of the hypercone. In such a case our proposed solution would be to take a smaller

Table 2.7: Parametric and implicit equations with matrix M of corank>1.

Geometric object	Parametric equations	Implicit equation
Trifolium curve	$(-(-1+t^2)^2(1-14t^2+t^4)/(1+t^2)^4)$; $2t(-1+t^2)(1-14t^2+t^4)/(1+t^2)^4$	$y^4-3xy^2+2x^2y^2+x^3+x^4$
Cayley sextic	$(4(1-t^2)^6-3(1-t^2)^4(1+t^2)^2)/(1+t^2)^6$; $(8(1-t^2)^5t-2(1-t^2)^3t(1+t^2)^2)/(1+t^2)^6$	$4(x^2+y^2-x)^3-27(x^2+y^2)^2$
Sphere	$2s/(1+t^2+s^2)$; $2st/(1+t^2+s^2)$; $(-1-t^2+s^2)/(1+t^2+s^2)$	$x^2+y^2+z^2-1$
Double sphere	$(2(1-t^2))s/((1+t^2)(1+s^2))$; $2t(1-s^2)/((1+t^2)(1+s^2))$; $(1-s^2)/(1+s^2)$	$x^2+y^2+z^2-1$
Eight surface	$(4(1-t^2))s(1-s^2)/((1+t^2)(1+s^2)^2)$; $2t(1-6s^2+s^4)/((1+t^2)(1+s^2)^2)$; $2s/(1+s^2)$	$x^2+y^2-4z^2+4z^4$
Hypercone	$r(1-t^2)(1-s^2)/((1+t^2)(1+s^2))$; $2r(1-t^2)s/((1+t^2)(1+s^2))$; $2rt/(1+t^2)$; r	$x^2+y^2+z^2-w^2$

"offset" of the predicted Newton polytope. Size of the "offset" is desided by the rule of the binary searchwe scale down by two, if there is no solutions, enlarge the "offset" by 0.5, and so on.

As an illustration let us apply the "offset" taking to the curve ``trifolium",

Example 8 (Trifolium)**.** Trigonometric representation of the ``trifolium" curve is:

$$x_0 = -a\cos(t)\cos(3t),\ x_1 = -a\sin(t)\cos(3t)$$

. Rational parametric representation:

$$x_0 = -\frac{(-1+t^2)^2(1-14t^2+t^4)}{(1+t^2)^4},\ x_1 = \frac{2t(-1+t^2)(1-14t^2+t^4)}{(1+t^2)^4}.$$

Implicit equation: $x^3+x^4-3xy^2+2x^2y^2+y^4=0$

Predicted support gives us Newton polygone Q with vertices $(0,0)$, $(0,8)$, $(8,0)$, while for the Newton polygone P of actual implicit equation the vertices would be $(1,2)$, $(0,3)$, $(0,4)$, $(4,0)$. See Figure 8.

If we proceed with the algorithm using Q for building matrix M, we need to evaluate 45 potential monomials (number of the lattice points contained in Q). Kernel space of M is 15-dimensional.

However, if we scale down the Q by two, new polygone Q' with vertices $(0,0)$, $(0,4)$, $(4,0)$ has 15 lattice points and solving produces 1-dimensional kernel vector space.

Example 9 (Hypercone)**.** We illustrate our method on a hypersurface of dimension 4, whose trigonometric representation is $x_0 = t_1\cos t_2\cos t_3,\ x_1 = t_1\cos t_2\sin t_3,\ x_2 = t_1\sin t_2,\ x_3 = t_1$. Its rational parametric representation is:

$$x_0 = \frac{t_3(1-t_1^2)(1-t_2^2)}{(1+t_1^2)(1+t_2^2)},\ x_1 = \frac{2t_3(1-t_1^2)t_2}{(1+t_1^2)(1+t_2^2)},\ x_2 = \frac{2t_3t_1}{1+t_1^2},\ x_3 = t_3. \quad (2.12)$$

Table 2.8: The table shows the number of the lattice points of the actual implicit polytope, the degree and number of monomials of the implicit equation, the number of the lattice points of the predicted implicit polytope, and the corank of M.

Geometric object	Implicit			Predicted		
	lattice points	degree	mono-mials	lattice points	corank of M	# g_i's of (degree)
Trifolium curve	8	4	5	43	15	1(4),2(5),3(6),4(7),5(8)
Cayley sextic	19	6	11	89	28	1(6),2(7),3(8),4(9),5(10),6(11),7(12)
Sphere	10	2	4	35	10	1(2),3(3),6(4)
Double sphere	10	2	4	125	45	3(4),4(5),9(6),11(7),18(8)
Eight surface	10	4	4	171	62	1(4),3(5),5(6),5(7),6(8),6(9),6(10), 6(11),6(12),6(13),6(14),4(15),2(16)
Hypercone	10	2	4	165	84	84(8)

This is an example where the resultant of the system (2.10) is a multiple of the implicit equation, hence it defines a variety strictly containing the image of the parametrization.

The ResPol predicts 4 implicit vertices: $(8,0,0,0),(0,8,0,0),(0,0,8,0),(0,0,0,8)$. Proposition 6, applied to the polynomials defined by the parametric expressions in (2.12), gives the following degree bounds: total degree ≤ 24, $\deg_{x_0} \leq 4$, $\deg_{x_1} \leq 4$, $\deg_{x_2} \leq 8$, $\deg_{x_3} \leq 16$ which improve the predicted support.

The initial predicted polytope contains 165 lattice points while the improved one contains 125. The corresponding interpolation matrices have corank 84 and 45 and it takes 485sec and 5.45sec, respectively, to compute their nullspace using LinearSolve and random integers. The polynomials obtained from the nullvectors have total degree 8 and 7, respectively, being multiples of the true implicit equation of the hypercone.

We apply scaling by one half to the initial predicted implicit polytope and try the new one with vertices $(4,0,0,0)$, $(0,4,0,0)$, $(0,0,4,0)$, $(0,0,0,4)$. This gives a matrix M of corank 10; all calculations now take 0.264sec. Repeating the procedure we build a matrix whose corank equals 1 for the polytope offset with vertices $(2,0,0,0)$, $(0,2,0,0)$, $(0,0,2,0)$, $(0,0,0,2)$. With this input data all calculations take 0.044sec. Thus we obtain the implicit equation $x_0^2+x_1^2+x_2^2-x_3^2$.

Example 10. We examine the hypersurface described in [79, 4.2].

$$\begin{aligned}
x_0 &= \frac{4}{-2t_2^2-1+2t_3t_2^2+2t_3+2t_3t_1^2},\\
x_1 &= \frac{-2t_2^2+1+2t_3t_2^2-2t_3+2t_3t_1^2}{-2t_2^2-1+2t_3t_2^2+2t_3+2t_3t_1^2},\\
x_2 &= \frac{4t_2(-1+t_3)}{-2t_2^2-1+2t_3t_2^2+2t_3+2t_3t_1^2},\\
x_3 &= \frac{4t_3t_1}{-2t_2^2-1+2t_3t_2^2+2t_3+2t_3t_1^2}.
\end{aligned} \tag{2.13}$$

To facilitate the computation of the predicted support, we express the common denominator

Table 2.9: Results for gcd of 2 and 3 polynomials corresponding to kernel vector entries chosen randomly.

Geometric object	kernel pols		Degree	gcd once		gcd twice	
	# monomials	# pols		Instances	Probability	Instances	Probability
Trifolium curve	5	1	4	54	0.51	993	0.73
	5,7	2	5	30		306	
	5,7,9	3	6	15		63	
	5,7,9,11	4	7	6		3	
	5,7,9,11,13	5	8				
Cayley sextic	11	1	6	153	0.4	5778	0.59
	11,14	2	7	100		2625	
	11,14,17	3	8	62		1029	
	11,14,17,20	4	9	36		324	
	11,14,17,20,23	5	10	19		69	
	11,14,17,20,23,26	6	11	8		3	
	11,14,17,20,23,26,29	7	12				
Sphere	4	1	2	30	0.67	333	0.93
	4	3	3	15		27	
	4,7	6	4				
Double sphere			2	218	0.22	17652	0.41
			3	259		13590	
	4,6	3	4	242		7770	
	4,6	4	5	146		2835	
	4,6,7,10	9	6	95		693	
	4,6,7,8,10	11	7	30		30	
	4,6,7,8,10,11,15	18	8				
Eight surface	4	1	4	233	0.12	26166	0.23
	4	3	5	304		24987	
	4	5	6	287		19821	
	4	5	7	252		15084	
	4,7	6	8	225		11076	
	4,7	6	9	189		7608	
	4,7	6	10	161		5208	
	4,7	6	11	125		2496	
	4,7	6	12	73		819	
	4,7	6	13	27		162	
	4,6,7	6	14	13		33	
	4,7	4	15	2			
	4,7	2	16				
Hypercone			2	307	0.09	81768	0.29
			3	648		98997	
			4	901		68304	
			5	757		28890	
			6	603		7419	
			7	270		474	
	4,7,11,16	84	8				

introducing a new variable w:

$$\begin{aligned} F_0 &= 4 - x_0 w, \\ F_1 &= -2t_2^2 + 1 + 2t_3 t_2^2 - 2t_3 + 2t_3 t_1^2 - x_1 w, \\ F_2 &= 4t_2(-1 + t_3) - x_2 w, \; F_3 = 4t_3 t_1 - x_3 w, \\ F_4 &= -2t_2^2 - 1 + 2t_3 t_2^2 + 2t_3 + 2t_3 t_1^2 - w. \end{aligned} \tag{2.14}$$

We compute the Newton polytope of this system's sparse resultant wrt s, t, u, w, projected to the space of x_0, x_1, x_2, x_3, and obtain the predicted implicit polytope with vertices $(0,2,0,4)$, $(6,0,0,0)$, $(2,4,0,0)$, $(0,0,6,0)$, $(0,0,0,6)$, $(2,0,0,0)$, $(0,0,4,0)$, $(0,0,0,4)$, it contains 144 lattice points. Proposition 6, when applied to the polynomials defined from the parametric expressions in (2.13), gives the following degree bounds: total degree ≤ 8, $\deg_{x_i} \leq 8, \forall i = 0,1,2,3$, which do not improve the predicted support. The matrix M has corank 8. Choosing an arbitrary nullvector, the corresponding polynomial is a multiple of the actual implicit equation.

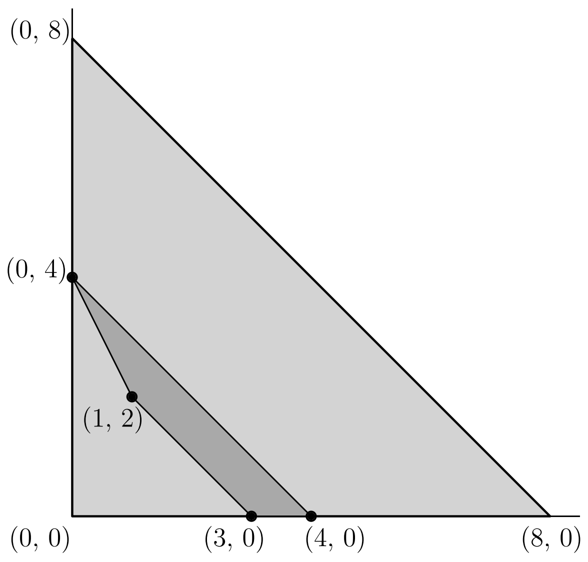

Figure 2.5: Actual and predicted support for the algebraic curve "trifolium" (Example 8).

We try taking an ``offset" of the predicted polytope. When scaled by 0.5, the system has no solution. Scaling by 0.75, we get a polytope containing 48 lattice points and obtain a polynomial of total degree 4. Factorizing it we get the implicit polynomial of degree 3

$$2x_1+2x_2^2+2x_1^2-2x_1x_3^2-2+2x_3^2-x_0x_2^2-2x_1^3+(1/2)x_0^2x_1+(1/2)x_0^2+x_0x_3^2-2x_1x_2^2,$$

equivalent to the one in [79, sec.4.2]. In Maple 13, our software with input the initial predicted polytope takes 2.2 sec, while with the scaled down polytope by 0.75 takes 0.348 sec.

Chapter 3

Implementation of the implicitization method

In this chapter we give a detailed report on the implementation of our method in Maple illustrating it with the examples. We report on a comparison of our implementation against popular implicitization methods, such as μ-bases and Gröbner bases. Then we show our method used to compute A-discriminants: one of its possible applications outside the geometry. We finish with the conclusions and some ideas for the future studies.

3.1 Exact and approximate solving

In this section we discuss the actual symbolic and numeric computations once the problem has been reduced to a question in linear algebra. We start with the software for the matrix operations, then present several detailed examples. Our algorithms are implemented in Maple and SAGE.

3.1.1 Maple

Our algorithm is implemented in Maple 13[1], based on the software for computing implicit polytopes [49], available as a C++ implementation[2]. The main functions are `imgen` (general implicitization, applicable for 2D, 3D and 4D geometrical objects,) and `imcurve` (for curves only, support prediction is part of the routine).

The functions take the following arguments:

- list of parametric expressions;

- only `imgen`: vertices of the predicted support;

- solving method parameter: ``l'' for `LinearSolve`, ``n'' for `Nullspace`, ``s'' for SVD;

- evaluation parameter: ``int'' for integers, ``unc'' for random complex numbers modulo 1, ``ruf'' for roots of unity evaluated as floating point numbers;

- ratio between number of rows and columns of the matrix (allows to construct over-constrained systems improving accuracy of numeric solving).

[1]http://ergawiki.di.uoa.gr/index.php/Implicitization
[2]http://sourceforge.net/projects/respol/files/

For exact kernel computation, we use function `LinearSolve()` from package `LinearAlgebra`, or function `Linear()` from package `SolveTools`. Equivalently, we may compute null(M) using the command `NullSpace()` of `LinearAlgebra`. SVD is implemented with command `SingularValues()`.

We have also implemented numerical versions of our algorithm in Matlab. The numerical stability of matrix M is measured by comparing ratios of singular values of M. We employ the *condition number* $\kappa(M) = |\sigma_1/\sigma_{|S|}|$, as well as ratio $|\sigma_1/\sigma_{|S|-1}|$, where σ_1 is the maximum singular value. By comparing these two numbers, we decide whether the matrix is of numerical corank 1, otherwise we instantiate a new matrix using new values.

All experiments, unless otherwise stated, were performed on a Celeron 1.6 GHz linux machine with 1 GB of memory.

In this set of experiments we show the size $\mu \times |S|$, $\mu > |S|$ of matrices used in numerical computation; the corresponding matrices for exact computation are $|S| \times |S|$.

A first conclusion is that SVD is expectedly faster than exact linear algebra, in most experiments. The best timings for the latter are obtained using function `LinearSolve` which sometimes outperforms SVD. This is partially due to the larger size of (rectangular) matrices used in SVD. A second observation is that our approximate methods gave very satisfactory results with respect to the accuracy of the computed implicit equation, see Section 3.1.4. Overall, our results are encouraging and indicate that our algorithms are worth applying to implicitization. However, as the matrix size grows, our current implementations show their limitations.

3.1.2 Examples

In order to give a better understanding of our implicitization method demonstrate the results of its application on a series of curves and surfaces, including the benchmark example of Bicubic surface.

Curve	SVD			#impl.
	time	accuracy (a)	matrix size	monom.
Descartes' Folium	0.012	$1.29 \cdot 10^{-12}$	10×5	3
Tricuspoid	0.028	$6.05 \cdot 10^{-6}$	30×15	8
Talbot's curve	0.132	$8.06 \cdot 10^{-16}$	56×28	8
Nephroid	0.17	$2.31 \cdot 10^{-21}$	56×28	10
Fifth heart	0.124	$1.09 \cdot 10^{-5}$	66×33	43
Trifolium	0.188	$6.37 \cdot 10^{-37}$	90×45	37
Ranunculoid	2.224	$7.71 \cdot 10^{-6}$	182×91	43

Table 3.1: Runtimes (sec) and accuracy of approximation for curves.

Example 11 (Folium of Descartes)**.** Let us consider the following curve:

$$x_0 = 3t^2/(t^3+1),\ x_1 = 3t/(t^3+1).$$

The algorithm in [55] yields 3 implicit polytope vertices: $(1,1)$, $(0,3)$, $(3,0)$. This polygon contains 5 lattice points which yield the potential implicit monomials $x_1^3, x_0x_1, x_0x_1^2, x_0^2x_1, x_0^3$ indexing the columns of matrix M in this order. To fill the rows of matrix M, we plug in to each monomial

Surface	SVD		matrix	# implicit
	time	accuracy (a)	size	monomials
Quartoid	0.036	$9.72 \cdot 10^{-14}$	16×16	4
Peano	0.024	$3.05 \cdot 10^{-14}$	10×10	4
Swallowtail	0.096	$1.52 \cdot 10^{-11}$	25×25	6
Sine	0.3	$1.03 \cdot 10^{-5}$	125×125	7
Bohemian dome	0.292	$1.68 \cdot 10^{-5}$	125×125	7
Enneper	0.42	$8.51 \cdot 10^{-9}$	103×103	23
Bicubic surface	74.63	$5.69 \cdot 10^{-5}$	715×715	715

Table 3.2: Runtimes (sec) and accuracy of approximation for surfaces.

the parametric expressions and evaluate using 5 random integer τ's: $19, 17, 10, 6, 16$. Then,

$$M = \begin{bmatrix} \frac{1270238787}{322828856000} & \frac{61731}{47059600} & \frac{66854673}{322828856000} & \frac{3518667}{322828856000} & \frac{185193}{322828856000} \\ \frac{24137569}{4394826072} & \frac{4913}{2683044} & \frac{1419857}{4394826072} & \frac{83521}{4394826072} & \frac{4913}{4394826072} \\ \frac{27000000}{1003003001} & \frac{9000}{1002001} & \frac{2700000}{1003003001} & \frac{270000}{1003003001} & \frac{27000}{1003003001} \\ \frac{1259712}{10218313} & \frac{1944}{47089} & \frac{209952}{10218313} & \frac{34992}{10218313} & \frac{5832}{10218313} \\ \frac{452984832}{68769820673} & \frac{36864}{16785409} & \frac{28311552}{68769820673} & \frac{1769472}{68769820673} & \frac{110592}{68769820673} \end{bmatrix}$$

The nullvector is $[1, -3, 0, 0, 1]$: its 3 nonzero entries correspond to monomials $x_1^3, x_0 x_1, x_0^3$, i.e. the actual monomials of the implicit equation. The latter turns out to be $x_0^3 - 3x_0x_1 + x_1^3$, which equals the true implicit equation of the curve.

Example 12 (Bicubic surface)**.** We consider the benchmark challenge of the bicubic surface [67]:

$$x_0 = 3t_1(t_1 - 1)^2 + (t_2 - 1)^3 + 3t_2, \; x_1 = 3t_2(t_2 - 1)^2 + t_1^3 + 3t_1,$$

$$x_2 = -3t_2(t_2^2 - 5t_2 + 5)t_1^3 - 3(t_2^3 + 6t_2^2 - 9t_2 + 1)t_1^2 + t_1(6t_2^3 + 9t_2^2 - 18t_2 + 3) - 3t_2(t_2 - 1).$$

The implicit degree in x_0, x_1 is 18, and 9 in x_2. The approach of [56] could not handle it because it generates 737129 regular triangulations (by TOPCOM) in a file of 383MB; our method computes the optimal support. The implicit polytope has vertices $(0,0,0), (18,0,0), (0,18,0), (0,0,9)$, and 715 lattice points. The nullvector of matrix M, computed in 42sec, contains 715 non-zero entries which correspond precisely to the actual implicit support.

Our last example concerns resultant computation. The support prediction software actually computes a resultant support so its straightforward application is to reduce resultant computation to interpolation; this is also the premise of [35, 112]. The main difference with interpolating the implicit equation is the absence of a parametric form of the resultant. But, this is provided by the parametrization of the resultant hypersurface, known as Horn-Kapranov parametrization [77], illustrated below.

Example 13. Let $f_0 = a_2x^2 + a_1x + a_0$, $f_1 = b_1x^2 + b_0$, with supports $A_0 = \{2, 1, 0\}, A_1 = \{1, 0\}$. Their (Sylvester) resultant is a polynomial in a_2, a_1, a_0, b_1, b_0. The algorithm in [49] computes its Newton polytope with vertices $(0,2,0,1,1)$, $(0,0,2,2,0)$, $(2,0,0,0,2)$; it contains 4 points, corresponding to 4 potential monomials $a_1^2 b_1 b_0,\; a_0^2 b_1^2,\; a_2 a_0 b_1 b_0,\; a_2^2 b_0^2$. The Horn-Kapranov parametrization of the resultant yields: $a_2 = (2t_1 + t_2)t_3^2 t_4,\; a_1 = (-2t_1 - 2t_2)t_3 t_4,\; a_0 = t_2 t_4,\; b_1 = -t_1 t_3^2 t_5,\; b_0 = t_1 t_5$, where the t_i's are parameters. We substitute these expressions to the predicted monomials, evaluate at 4 sufficiently random t_i's, and obtain a matrix whose kernel vector $(1, 1, -2, 1)$ yields $\mathcal{R} = a_1^2 b_1 b_0 + a_0^2 b_1^2 - 2a_2 a_0 b_1 b_0 + a_2^2 b_0^2$.

The complexity of interpolating resultants is $O^*(|S|^2)$ where S is the set of lattice points in the predicted resultant support, because the dominating stage is a kernel computation for a structured matrix M. Using Weidemann's approach, the main oracle is post-multiplication of M by a vector, which amounts to evaluating a $(n+1)$-variate polynomial at *chosen* points, and this can be done in quasi-linear complexity in $|S|$ [114, 88]. For certain classes of polynomial systems, when one computes the resultant in one or more parameters, this may be competitive to current methods for resultant computation. The best such methods rely on developing the determinant of a resultant matrix in these parameters [18, 37]. The matrix dimension is in $O^*(t^n \deg \mathcal{R})$ [47], where $\deg \mathcal{R}$ is the total degree of $\mathcal{R}$ in all input coefficients, and t is the scaling factor relating the input Newton polytopes, which is bounded by the maximum degree of the input polynomials f_i in any variable. Then, developing *univariate* resultants has complexity in $O^*(t^{3.5n}(\deg \mathcal{R})^{3.5})$ [47, 57]. Hence, our approach improves the complexity when the predicted support is small compared to t and $\deg \mathcal{R}$.

3.1.3 Curves and surfaces of Bernstein basis

Our approach to implicitization relies on the support prediction method that operates in the monomial basis. However, parametric representations of Bernstein basis, such as NURBS curves and surfaces are widely used in CAD systems. Being interested in exploring possibilities for practical applications of our method, we have run series of experiments curves and surfaces of Bernstein basis. Results are presented in Table 3.3.

We restricted our experiments to rational and non-rational Bézier curves defined in 1.4 and NURBS patches. However it is possible to apply the same method to planar splines or surface patches, i.e. manifolds given by k parametric pieces. The resulting single implicit equation will approximate the piecewise parametrization.

Two aspects that characterize application of our algorithm to NURBS curves and surfaces

- Parametrization has to be converted to monomial basis.
- When building matrix M we have to ensure that the evaluation points lie within the region of interest of the parametrization.

In our experiments we have used evaluation by rational numbers, random or uniform, and complex roots of unity. Note, that that in case of trigonometric parametrizations the former led to a loss of numerical stability. Here, random rational numbers in $[0, 1]$ provide the fastest results, which makes them our preferred evaluators. In Table 3.3 when not otherwise stated we use them.

We have tried also evaluation with Chebyshev nodes in $[0, 1]$.

When solving numerically the latter allows to minimize the approximation error [8]. Using complex roots of unity for evaluation appears to be the slowest mode; besides it introduces complex coefficients into the resulting approximate implicit equation.

In the course of conversion from the Bernstein to monomial basis precision loss may occur. Table 3.3 contains three such cases. The NURBS curves have been kindly provided by the authors of [121] as a part of the package with industrial examples. NURBS curves are defined by their order, and a knot vector, a set of weighted control points. The latter usually are given in floating point numbers. When the parametrization represented in monomial bases is evaluated, inaccuracy leads to the construction of a system that, being set to solving by exact method (such as `LinearSolve`) has no solution: the kernel vector contains only zeros. In order to remedy that, we apply filtering to the resulting "raw" parametric equations:

- If possible, we try to represent float coefficients as quotients of integers.
- The rest of the instances occur by removing all monomials (usually of high degree) with coefficients that are close to zero, whose absolute value is $< 10^{-2}$.

The latter have smaller predicted implicit polytope than the "raw" instance.

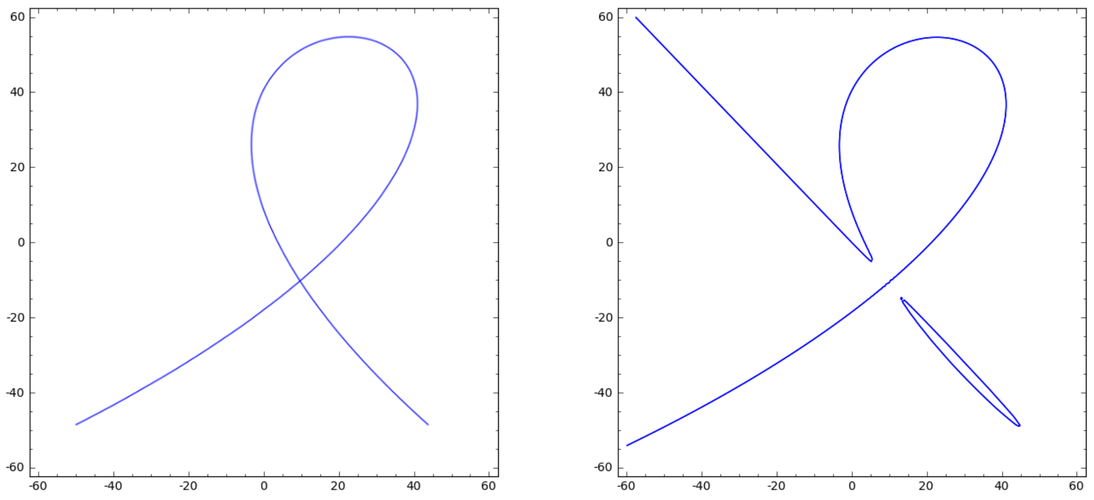

Figure 3.1: Bézier curve: parametric form (left), exact implicit representation (right).

If an exact solution is impossible to attain, we apply numerical solving to compute an approximate implicit equation.

Table 2.6 shows the results of our experiments with industrial examples presented in [121], which are NURBS patches with floating point coefficients.

In the Table 2.6 there are some cases presentedrational "float" values (Simplesweep) where `LinearSolve` does not produce a non-zero kernel; this example being obtained from an industrial source. In other cases, rounding yields an implicit equation of smaller degree than the actual one. This happens with Nested nodal and Simplesweep, which have implicit degree equal to 6. We conclude that, for NURBS patches with floating point coefficients, approximate implicitization is more reliable than exact implicitization.

3.1.4 Accuracy of the approximate implicitization

There are several possible sources for the errors when solving numerically. Errors can be introduced by using not sufficiently generic values for matrix evaluation. Processing parametrizations where coefficients are given in floating point numbers may result in mathematical truncation and rounding errors. Therefore, it is important to estimate the numeric accuracy, or quality, of the solution. In our work we have applied several measures to quantify the accuracy of approximate implicitization.

As mentioned before, we use the matrix condition number and the ratio between the two smallest singular values to evaluate the error in the coefficient vector computed by SVD.

We employ two measures that provide a lower bound on the accuracy of approximate implicitization:

Table 3.3: Bézier and NURBS curves, results of implicitization. Runtimes are given in seconds.

Curve	Degree	Exact solving	SVD	SVD Chebyshev nodes	SVD roots of unity
Bézier curve	4	0.16	0.12	0.13	0.21
Bézier curve	5	0.29	0.16	0.19	0.33
Bézier curve	6	0.47	0.16	0.21	0.53
Bézier curve	8	5.84	0.46	0.49	1.1
NURBS curve	3	-	0.05	0.05	0.09
NURBS curve	4	-	0.09	0.1	0.22
NURBS curve	7	-	0.43	0.36	0.93

Experiments have been performed on an Intel©Core2 Duo CPU, 2.20GHz, 3Gb memory, `SAGE 5.4`.

(a) Coefficient difference: measured as the Euclidean norm of the difference of the two coefficient vectors V_{exact}, V_{app}, obtained from exact and approximate implicitization, after padding with zero the entries of each vector which do not appear in the other.

(b) Evaluation norm: measured by considering the maximum norm of the approximate implicit equation when evaluated at a set of sampled points on the given parametric object. This is a lower bound on how far from zero can such a value be.

When using numerical methods, the computed implicit equation is not a polynomial with rational coefficients, hence we need to convert the computed real or complex kernel vector to a rational vector. This is achieved by setting all coefficients smaller than a certain threshold, defined by the problem's condition number, equal to zero. The result is not always equal to the exact implicit equation, so its accuracy is quantified by two measures discussed later. The overall process is computationally rather costly; it can be avoided whenever an implicit equation with floating point coefficients is sufficient for a specific application.

We can actually improve the accuracy of approximation if we disregard all real or complex entries of the coefficient vector with norm close to zero. This simple filtering, applied with a threshold of 10^{-6}, improves the accuracy under measure (a) by up to one order of magnitude. All results shown in the tables concerning approximate implicitization make use of this filtering.

The approximate implicit equation in all experiments below is obtained using the command `SingularValues()`, where the matrix is instantiated by unitary complex values τ, whereas the exact one is obtained using command `NullSpace()` using random integers. We used several parametric curves and surfaces in our experiments. The runtimes of approximate and exact methods, and the accuracy of approximation using measure (a) above, are shown in Table 3.1 and Table 3.2. These results confirm that SVD can give very good approximations of the actual implicit equation on most inputs.

One of the main difficulties of approximating the implicit equation is to build the matrix M so that its numeric corank is 1. Our experiments indicate, expectedly, that if the entries of M take big absolute values, then computations with M are less stable. We improve stability by avoiding values that make the denominators of the parametric polynomials evaluate close to 0. These values are singular points so we choose a box containing each such point and remove them when we pick different values. Moreover, we add more rows to M.

We compare the runtimes for exact and approximate methods, and the accuracy of the latter using both measures: in Table 3.4. Both measures give overall very encouraging results.

Surface	Max norm of approximate implicit polynomial
Bohemian dome	$7.21668 \cdot 10^{-10}$
Quartoid	$7.44845 \cdot 10^{-16}$
Sine	$1.25549 \cdot 10^{-5}$
Swallowtail	$1.98798 \cdot 10^{-10}$

Table 3.4: Accuracy of approximation under measure (b) over 100 sampled points

One of the main difficulties of approximating the implicit equation is to build the matrix M so that its numeric corank is 1. Our experiments indicate that if the entries of M take big absolute values, then computations with M are less stable. We improve stability by avoiding values that make the denominators of the parametric polynomials evaluate close to 0. These values are singular points so we choose a box containing each such point and remove them when we pick different values. Moreover, we construct rectangular matrix M adding more rows. Experiments show that performing SVD on such an over-constrained system allows to increase accuracy of the approximate implicitization.

Hausdorff distance

There is a relation between the measures of the approximate implicitization's accuracy presented above and the Hausdorff distance for measuring the distance between two hypersurfaces. Here we discusse the latter presenting the outcome of our joint work with the Thang Luu Ba.

Definition 11. Let the distance of a point $x \in \mathbb{R}^{n+1}$ to a set $V \subset \mathbb{R}^{n+1}$ be $D(x, S) := \inf_{y \in S} D(x, y)$, where D denotes the metric distance in $\mathbb{R}^{n+1}$. The *Hausdorff distance* of sets V_1, V_2 is

$$d_H(V_1, V_2) := \max\{\sup_{x \in V_1} D(x, V_2), \sup_{x \in V_2} D(x, V_1)\}.$$

If V_1, V_2 are algebraic hypersurfaces and compact, this becomes

$$d_H(V_1, V_2) = \max\{\max_{P \in V_1} \min_{Q \in V_2} D(P, Q), \max_{Q \in V_2} \min_{P \in V_1} D(P, Q)\}.$$

Suppose V_1, V_2 are parametrized by $f(t) := (f_0(t), \ldots, f_n(t))$ and $g(u) := (g_0(u), \ldots, g_n(u))$, where $t := (t_1, \ldots, t_n), u := (u_1, \ldots, u_n) \in \Omega$, then the Hausdorff distance is

$$d_H(V_1, V_2) = \max\{\max_{t \in \Omega} \min_{u \in \Omega} \sqrt{S(t, u)}, \min_{t \in \Omega} \max_{u \in \Omega} \sqrt{S(t, u)}\},$$

where $S(t, u)$ is the inner product $(f(t) - g(u)) \cdot (f(t) - g(u))$. In this case, the computation reduces to solving a nonlinear system, which is quite hard [89]. In the case of curves, when the nearest points are both inner points, the system becomes $S'_t(t, u) = S'_u(t, u) = 0$.

In [26, 27], the authors gave effective algorithms to compute the Hausdorff distance for B-spline curves and Bézier curves. However, there is no effective algorithm for computing Hausdorff distance in general.

Suppose V_1, V_2 are two polynomials representing an implicit equation. $V_i := \{x \in \mathbb{R}^{n+1} : p_i(x) = 0, p_i \in \mathbb{R}[x], i = 1, 2\}$. The computation of $d_H(V_1, V_2)$ appears to be difficult problem. even if the V_i are implicit equations of curves.

In [73], the author bounds the Hausdorff distance of spline curves by cM, where c is constant and M the maximum of absolute coefficients of the spline functions difference $p_1(x_1, x_2) - p_2(x_1, x_2)$.

There is no general connection between Hausdorff distance and the coefficients of $p(x_1, x_2)$, though we expect that, if the latter are sufficiently small, then $d_H(C_1, C_2)$ is small.

Let V_1 be parametrized by $(f_0(t), \ldots, f_n(t))$, $t \in \Omega$, and $V_2 = \{x \in \mathbb{R}^{n+1} : p(x) = 0,\ p \in \mathbb{R}[x]\}$. Then $d_H(V_1, V_2)$ is related to $p(f_0(t), \ldots, f_n(t))$ by the Łojasiewicz inequality [72]. The Łojasiewicz inequality asserts $\exists\, c, \alpha > 0 : d(x, V_2)^{\alpha} \leq c\,|p(x)|$, for every x in a compact domain. If $p(x) \in \mathbb{R}[x]$ is an approximate implicit equation of V_1,

$$d_H(V_1, V_2)^{\alpha} \leq c \max |p(f_0(t), \ldots, f_n(t))|, t \in \Omega.$$

In [73], if C_1 is a B-spline and $p(x)$ is a spline bivariate function, there is a bound of c supposing C_1 is close enough to C_2.

Our second measure (b) for evaluating the quality of approximate implicitization has followed this approach. Evaluating the exact maximal of $|p(f_0(t), \ldots, f_n(t))|, t \in \Omega$ is complicated so we approximate the distance by choosing a large sample for the parameters t and find $\max |f(t)|$.

The Fréchet distance is also a fundamental tool to compute the distance between two parametric curves. Given two parametric curves C_1 and C_2 in parametrized form $f, g : [0, 1] \to \mathbb{R}^2$, it is

$$d_F(C_1, C_2) := \inf_{\rho, \sigma} \max_{t \in [0,1]} D(f(\rho(t)), g(\sigma(t))),$$

where $\rho, \sigma : [0, 1] \to [0, 1]$ range over all continuous and non-decreasing functions (reparametrizations) with $\rho(0) = \sigma(0) = 0$ and $\rho(1) = \sigma(1) = 1$. Obviously, $d_H(C_1, C_2) \leq d_F(C_1, C_2)$, but the ratio d_H/d_F is not bounded. It is possible for two curves to have small Hausdorff but large Fréchet distance. In [4], the authors showed that, for closed convex curves, the Hausdorff equals the Fréchet distance, while the latter is $\leq \kappa + 1$ times the Hausdorff distance for κ-bounded curves. In [13], the authors studied the Fréchet distance for simple polygons but it seems likely that the Fréchet distance between general surfaces is not computable.

Example 14. Consider a parametric curve C_1:

$$x(t) = 3t + 1, y(t) = t^4 + 2t^3 - t + 1, t \in [0, 1].$$

The implicit equation of this curve is : $103 - 81y - 13x - 12x^2 + 2x^3 + x^4$. Its approximate implicit equation is: $p(x, y) := 103.000000000009 - 80.999999999993y - 13.000000000138x - 11.9999999999934x^2 + 1.99999999999878x^3 + x^4$.

Let C_2 be the curve with implicit equation $p(x, y)$. The accuracy of approximation under measures (a) and (b) is $\approx 2.075627 \cdot 10^{-11}$ and $\approx 9.192 \cdot 10^{-10}$, respectively. The reparametrizations of C_1, C_2 are, respectively,

$f(t) = (t, \frac{103 - 13t - 12t^2 + 2t^3 + t^4}{81})$,

$g(u) = (u, \frac{103.000000000009 - 13.0000000000138u - 11.9999999999934u^2 + 1.99999999999878u^3 + u^4}{80.99999999993})$

We compute $d_H(C_1, C_2)$ by evaluating $S(t, u)$ where $(t, u) \in [1, 4] \times [1, 4]$ and obtain $d_H(C_1, C_2) \leq 9.39 \cdot 10^{-14}$.

Example 15. Consider a parametric curve C_1:

$$x(t) = \frac{t+2}{t+1},\ y(t) = t^4 - 2t^2 + 2t + 1,\ t \in [0,1].$$

The implicit equation is: $-5 + y - 2x - 4xy + 14x^2 + 6x^2y - 10x^3 - 4x^3y + 2x^4 + x^4y$.

Its approximate equation is:
$p(x,y) := -5.0000000307301 + 0.999999998287437y - 1.99999993893286x - 3.9999999956547xy + 13.9999999566394x^2 + 5.99999999627234x^2y - 9.99999998715587x^3 - 3.99999999892032x^3y + 1.99999999866029x^4 + x^4y$.

Let C_2 be the curve with implicit equation $p(x,y)$. The accuracy of approximation under measures (a) and (b) is: $\approx 8.24533 \cdot 10^{-8}$ and $\approx 1.3511 \cdot 10^{-9}$, respectively. The reparametrizations of C_1, C_2 are, respectively

$f(t) = (t, \frac{-5-2t+14t^2-10t^3+2t^4}{-1+4t-6t^2+4t^3-t^4});$
$(u) = (u, \frac{-5.00000003073011-1.99999993893286u+13.9999999566394u^2-9.99999998715587u^3+1.99999999866029u^4}{-0.999999998287437+3.99999999565477u-5.99999999627234u^2+3.99999999892032u^3-u^4}).$

We compute the Hausdorff distance $d_H(C_1, C_2)$ by evaluating the inner product $(f(t) - g(u)) \cdot (f(t) - g(u))$, $(t,u) \in [\frac{3}{2}, 2]^2$. Thus we obtain $d_H(C_1, C_2) \leq 1.5853 \cdot 10^{-11}$.

Example 16. Consider a cylinder surface

$$S_1 :\ x = t + 1,\ y = t^3 + 3t^2 + 5t + 3,\ z = s$$

where $\Omega := (t,s) \in [0,1] \times [0,1]$.

Its implicit equation is $x^3 + 2x - y$ and its approximate implicit equation (S_2) is $4.18399 \cdot 10^{-8} - y + 2x + x^3$.

The accuracy of approximation under measures (a) and (b) is $\approx 4.18399 \cdot 10^{-8}$.

The Hausdorff distance $d_H(S_1, S_2)$ equals the Hausdorff distance of the two parametrized planar curves $C_1 : (t, t^3 + 2t)$ and $C_2 : (u, u^3 + 2u + 4.183988732 \cdot 10^{-8})$. Finding maximal absolute values of the polynomial

$$S(t,u) = (t-u)^2 + (-t^3 - 2t + u^3 + 2u - 4.18399 \cdot 10^{-8})^2, \quad (t,u) \in [1,2] \times [1,2],$$

we obtain $d_H(S_1, S_2) \leq 1.87114 \cdot 10^{-8}$.

Example 17. Consider Peano's surface

$$S_1 : x = t,\ y = s + 1,\ z = -s^2 + s(3t^2 - 2) - 1 + 3t^2 - 2t^4$$

where $\Omega := (t,s) \in [0,1] \times [0,1]$.

Its implicit equation is $z + y^2 - 3x^2y + 2x^4$ and its approximate implicit equation (S_2) is $-3.725155334 \cdot 10^{-8} + z - 4.414233699 \cdot 10^{-10}y + y^2 - 2.879451183 \cdot 10^{-10}x + 1.830328567 \cdot 10^{-12}xy + 2.350959794 \cdot 10^{-11}x^2 - 3x^2y + 5.645612348 \cdot 10^{-14}x^3 + 2x^4$.

The accuracy of approximation under measures (a) and (b) are $\approx 3.725528887 \cdot 10^{-8}$ and $\approx 2.983 \cdot 10^{-9}$, respectively. The reparametrizations of S_1 and S_2 are:
$f(t,s) = (t, s, -s^2 + 3t^2s - 2t^4)$, and $g(u,v) = (u, v, -3.725155334 \cdot 10^{-8} + 4.414233699 \cdot 10^{-10}v - v^2 + 2.879451183 \cdot 10^{-10}u - 1.830328567 \cdot 10^{-12}uv - 2.350959794 \cdot 10^{-11}u^2 + 3u^2v - 5.645612348 \cdot 10^{-14}u^3 - 2v^4)$.

To compute the Hausdorff distance $d_H(S_1, S_2)$, we need to evaluate the inner product

$$S(t, s; u, v) = (f(t, s) - g(u, v)) \cdot (f(t, s) - g(u, v)),\ (t, s), (u, v) \in [0, 1] \times [1, 2].$$

We know that $d_H(S_1, S_2) \leq \max_{(t,s)\in[0,1]\times[1,2]} \sqrt{S(t, s; t, s)}$, thus we maximize $S(t, s)$ and get $d_H(S_1, S_2) \leq 3.813440008 \cdot 10^{-8}$.

Lack of the efficient algorithm for computing Hausdorff distance makes difficult it's use in practice as a tool for measuring accuracy of the approximate implicitization. However as demonstrated by the examples, measure (a), i. e. Euclidean norm, is close enough to the Hausdorff measure to be used instead.

3.2 Comparison to other methods

We report on a comparison of our implementation against existing implicitization software, namely implementation of the μ-bases implicitization method [33], applicable for rational curves [16], and Maple function `Implicitize()`, which is based on integration of matrix M over each parameter, see [30].

Table 3.5 summarizes the total time to implicitize a curve, given its parametrization. We used the same algebraic curves as in other tables, grouped by degree; for each degree, the table shows the average runtime. In our experiments, μ-bases yield the fastest runtimes, whereas `Implicitize()` is the slowest of the three when run in exact mode or when the parametrization is rational.

However, μ-bases rely on exact computation over rational numbers, and an approximate computation would not offer good accuracy. Our algorithm removes this limitation and offers high-quality approximations.

Curve	degree	`Implicitize` exact	`Implicitize` numeric	Our software	μ-bases
Trisectrix of Maclaurin	3	1.92	0.064	0.02	0.016
Folium of Descartes	3	9.3	0.08	0.012	0.024
Tricuspoid	4	1.92	0.064	0.044	0.016
Bean	4	129.7	0.12	0.036	0.028
Talbot's	6	18.98	0.252	0.324	0.072
Fifth heart	8	799.74	0.44	0.104	0.08
Ranunculoid	12	>3000	1.64	1.376	0.3

Table 3.5: Comparing runtimes (sec) of: Maple function `Implicitize` (exact and numeric), our method (`LinearSolve`, random integers), and μ-bases.

Also we compare our method with the Gröbner bases implicitization.

Example 18. Consider the parametrized Enneper's surface:

$$f_1 = t_1/3 - (1/9)t_1^3 + t_1 t_2^2/3,\ f_2 = -t_2/3 + (1/9)t_2^3 - t_1^2 t_2/3,\ f_3 = t_1^2/3 - t_2^2/3.$$

We compute a Gröbner basis G of ideal $I = \langle x - f_1, y - f_2, z - f_3\rangle$ with respect to the lexicographic ordering $t_1 > t_2 > x > y > z$. The unique polynomial depending only on x, y, z is the implicit equation:

$$128z^7 - 27y^6 + 702x^2y^2z^3 - 9x^4z - 9y^4z - 48x^2z^3 - 64z^5 + 432x^2z^5 + 240x^2z^4 + 135y^4z^3 + 432y^2z^5 - 240y^2z^4 + 27x^6 + 81x^2y^4 - 162y^4z^2 + 144x^2z^6 - 144y^2z^6 - 81x^4y^2 + 135x^4z^3 + 162x^4z^2 - 64z^9 + 18x^2zy^2 - 48z^3y^2$$

We compared implicitization based on Gröbner bases implemented in Maple with our software using `LinearSolve` and random integers, see Table 3.6. The results show that for low degree curves (≤ 6) or surfaces (≤ 4), Gröbner bases outperform our software. The situation is reversed for higher degree. In particular, the bicubic surface takes under 40 sec with our method, it is infeasible using Gröbner bases on Maple, and takes 313 sec on Mathematica 8.0.

Curve / Surface	degree	Gröbner basis	Our software
Double sphere	2	0.112	1.860
Moebius strip	3	6.184	9.520
Bohemian dome	4	0.776	1.181
Eight surface	4	0.196	3.668
Swallowtail surface	5	0.192	0.108
Sine surface	6	1.240	1.164
Enneper's surface	9	0.668	0.776
Bicubic surface	18	$>$4 hours	42.059
Trifolium	4	0.032	0.26
Talbot's curve	6	0.104	0.324
Ranunculoid	12	7.341	1.376

Table 3.6: Comparing runtimes (sec) of Gröbner bases method implemented in Maple and our method (`LinearSolve`, random integers).

We compare our `Maple` implementation against `Maple`'s native function *Implicitize()* which employs integration of matrix M over each parameter [30], and implicitization using Gröbner bases in `Maple` . Results are shown in Table 3.7.

The input consists of a family of classical algebraic surfaces, the so called Plücker's conoid (Fig. 3.2): $x_0 = t,\ x_1 = s,\ x_2 = \frac{Re((t+I\cdot s)^a)}{|(t+I\cdot s)^a|}$. The surfaces have a base point at $t = s = 0$. By choosing appropriate values of parameter $a = 2b$ we obtain rational parameterizations of the surfaces with desired total degree. While implicitizing of the Plücker's conoid is a trivial problem, this family of surfaces provides a good illustration of the algorithm performance in relation to degree.

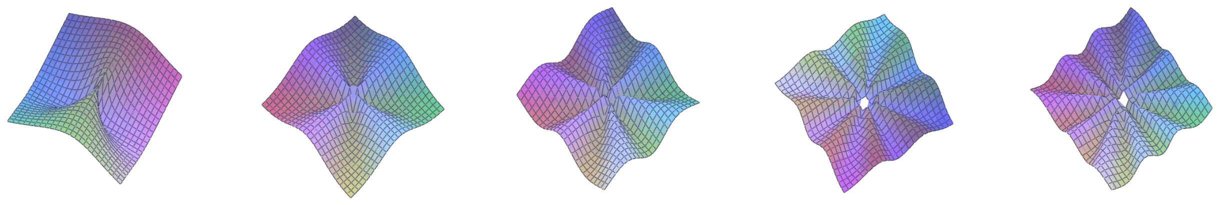

Figure 3.2: Plücker's conoid surfaces used in Table 3.7. Degree determines the number of the folds.

Table 3.7: Comparison of our method (exact and numerical) to `Maple`'s function *Implicitize()* and Gröbner bases.

Deg.	Exact	Gröbner	Numerical			Maple *implicitize()(numerical)*		
	runtime	runtime	runtime	accuracy (a)	accuracy (b)	runtime	accuracy (a)	accuracy (b)
3	0.016	0.031	0.031	10^{-15}	$9.07 \cdot 10^{-10}$	46.07	10^{-15}	$1.98 \cdot 10^{-9}$
5	0.016	0.046	0.032	10^{-10}	$3.57 \cdot 10^{-8}$	85.43	$3.67 \cdot 10^{-7}$	$6.83 \cdot 10^{-6}$
7	0.031	0.078	0.046	10^{-11}	$9.97 \cdot 10^{-8}$	359.49	$9.06 \cdot 10^{-7}$	$2.94 \cdot 10^{-4}$
9	0.046	0.078	0.063	10^{-10}	$1.35 \cdot 10^{-7}$	695.65	$2.86 \cdot 10^{-6}$	$7.55 \cdot 10^{-3}$
11	0.078	0.141	0.078	10^{-11}	$1.07 \cdot 10^{-6}$	> 2000	-	-

3.3 Computing discriminants

While our method shows the best performance when applied to computing symbolically exact implicit equations, it is true that in the modern CAGD systems orientation to numerical solving and approximations is preferred. However, besides geometry, there apparently exist many areas where exact implicitization is in demand.

In this section we apply our method to computing the discriminant of a multivariate polynomial, which characterizes the existence of multiple roots. It subsumes the resultant of an overconstrained system.

Discriminants are fundamental tools in several geometric applications, since they characterize the locus of discrete changes of a system; they also find their application in parametrization. The vanishing of the discriminant partitions coefficient space to cells of values for which the underlying polynomial has a fixed number of real roots. For mechanical and robotics systems expressed by polynomials, the discriminant variety partitions configuration space to instances that are connected by continuous movement without singularities, e.g. [61].

It is well known fact that the condition for a univariate quadratic polynomial

$$f = at^2 + bt + c$$

to have a double root is that its discriminant $D(f) = b^2 - 4ac$ vanishes.

A univariate cubic polynomial has a double root if and only if its discriminant vanishes:

$$D(c_0 + c_1 t + c_2 t^2 + c_3 t^3) = c_1^2 c_2^2 - 4c_1^3 c_3 - 4c_0 c_2^3 - 27c_0^2 c_3^2 + 18c_0 c_1 c_2 c_3.$$

More generally, consider a polynomial $f(t_1, \dots, t_n)$ in n variables. A multiple root of f is a point where f vanishes together with all its first derivatives $\partial f / \partial t_i$.

Definition 12. The discriminant $D(f)$ is a polynomial function in the coefficients of f, which vanishes whenever f has such a multiple root.

It can be shown that $D(f)$ exists and is unique (up to sign) if we require it to be irreducible and to have relatively prime integer coefficients.

We are interested in discriminants of polynomials with fixed support: given a set of m

lattice points $A \subset \mathbb{Z}^n$, let

$$F_A = \sum_{a \in A} c_a t^a$$

denote the generic polynomial in variables $t_1, \dots, t_n$ with exponents in A.

Definition 13. [64] *A-discriminant* is an irreducible polynomial $D_A = D_A(c)$ with integer coefficients in the vector of coefficients $c = (c_a : a \in A)$, defined up to sign, which vanishes for each choice of c for which F_A and all $\partial F_A / \partial t_i$ have a common root in $(\mathbb{C}\backslash\{0\})^n$.

Here, we consider roots with nonzero coordinates so as to be able to ignore trivial multiple roots. A-discriminants describe the singularities of a class of functions, called A-hypergeometric functions, which are solutions of certain linear partial differential equations. The A-discriminant is an affine invariant, in the sense that any configuration of points affinely isomorphic to A has the same discriminant.

A-discriminants include as special cases several fundamental algebraic objects.

If $A = \{(0,0), (1,0), \dots, (m,0),\ (0,1), (1,1), \dots, (n,1)\} \subset \mathbb{Z}^2$, then we can write F_A as $f(t_1) + t_2 g(t_1)$. Its A-discriminant is the resultant of f and g: It vanishes whenever f and g have a common root. More generally, the resultant of $f_0, \dots, f_k$ in k variables is the A-discriminant of an auxiliary polynomial

$$f_0(t_1, \dots, t_k) + \sum_{i=1}^{k} y_i f_i(t_1, \dots, t_k)$$

.

Another important example occurs when F_A consists of n^2 monomials $x_i y_j, i, j = 1, \dots, n$, i.e a bilinear form $F_A = \sum c_{ij} x_i y_j$ then its A-discriminant is the determinant of the matrix (c_{ij}). Moreover, D_A is a factor of the unmixed A-resultant of F_A and $\partial F_A / \partial t_i, i = 1, \dots, n$.

The extraneous factors in this resultant are powers of face discriminants, i.e. discriminants associated to the sub-configurations of A consisting of all points in a face of its convex hull.

Computing A-discriminants may be reduced to implicitization. Given A, we form the $(n+1) \times m, m > n+1$ integer matrix (also called A by abuse of notation) whose first row consists of ones, and whose columns are given by the points $(1, a)$ for all $a \in A$.

Let $B = (b_{ij}) \in \mathbb{Z}^{n \times (m-n-1)}$ be a matrix whose column vectors are a basis of the integer kernel of matrix A. Then B is of full rank.

We assume that its maximal minors have unit gcd (i.e. the rows generate $\mathbb{Z}^{m-n-1}$). Since the first row of A equals $(1, \dots, 1)$, the column vectors of B add up to 0.

The A-discriminant D_A is A-homogeneous, i.e it is quasi- homogenous relative to the weight defined by any vector in the row span of A. Let $d = m - n - 1$. The Horn-Kapranov parametrization [64, 76], is defined as:

$$x_j = \prod_{i=1}^{m} (b_{i1} y_1 + \cdots + b_{id} y_d)^{b_{ij}},\ j = 1, 2, \dots, d, \tag{3.1}$$

where $y_i, i = 1, \dots, d$ are parameters.

The implicit equation of its image is a polynomial Δ_B in $x = (x_1, \dots, x_d)$ such that the A-discriminant $D_A(x)$ is the product of $\Delta_B(x)$ and a monomial. Put it differently, Δ_A is the dehomogenized version of D_A. This reduces the computation of D_A to implicitizing the parametric hypersurface (3.1).

The complexity of our method, due to the nature of the support prediction approach we use to determine the space of interpolation, depends on the number of lattice points in the predicted polytope. The latter equals the Newton polytope of the discriminant or a superset, which seems to be not much larger than the Newton polytope itself, in practice. Hence, our method is output sensitive since it depends on the size of the target polynomial.

To illustrate our method, we focus on discriminants with with codimension 2 and 3 respectively (i.e. $n = m + 3$ and $n = m + 4$ respectively) [15, 36, 41], although our algorithm may compute discriminants for any d. In particular, we implicitize the parametric curve and surface given respectively by

$$x_j = \prod_{i=1}^{m} (b_{i1} + b_{i2}s)^{b_{ij}},\ j = 1, 2,$$

and

$$x_j = \prod_{i=1}^{m} (b_{i1} + b_{i2}s + b_{i3}t)^{b_{ij}}, j = 1, 2, 3.$$

In the following, we denote by l_i the inner product of the i-th row of B and the parameter vector $(1, s)$ or $(1, s, t)$, i.e. $l_i = b_{i1} + b_{i2}s$ or $l_i = b_{i1} + b_{i2}s + b_{i3}t$.

Let us present some examples of applying our method to computing discriminants of a polynomial in several variables.

Discriminant of codimension 2

Example 19. Consider a generic polynomial of two variables of degree 3,

$$F_A(t_1, t_2) = c_1t_1 + c_2t_2 + c_3t_1t_2 + c_4t_1^2 + c_5t_1^3$$

where $A = \{[1, 0], [0, 1], [1, 1], [2, 0], [3, 0]\} \subset \mathbb{Z}^2$.

We build the matrix A:

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

then the matrix B is as follows:

$$B = \begin{pmatrix} -1 & -1 \\ 1 & 2 \\ -1 & -2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Let $l_1 = -1 - s, l_2 = 1 + 2s, l_3 = -1 - 2s, l_4 = 1, l_5 = s$, then we have the parametrization

$$f_1 = \frac{l_2l_4}{l_1l_3} = \frac{1 + 2s}{(-1 - 2s)(-1 - s)}, f_2 = \frac{l_2^2l_5}{l_1l_3^2} = \frac{(1 + 2s)^2s}{(-1 - s)(-1 - 2s)^2}$$

Support prediction yields 4 Newton polygon vertices: $[0,0]$, $[2,0]$, $[3,0]$, $[3,2]$. The Newton polygon has 7 lattice points. Applying our method, we obtain implicit equation $\Delta_B(x,y) = x - y - 1$.

We perform substitution following

$$\Delta_B(f_1, f_2) = D_A\left(1, 1, 1, \frac{l_2 l_4}{l_1 l_3}, \frac{l_2^2 l_5}{l_1 l_3^2}\right),$$

which gives us A-discriminant of F_A: $D_A(c) = c_2 c_3 c_4 - c_2^2 c_5 - c_1 c_3^2$

Example 20. Consider a generic polynomial of three variables of degree 3,

$$F_A(t_1, t_2) = c_1 t_1 t_2 + c_2 t_1 t_3 + c_3 t_2 t_3 + c_4 t_1^2 + c_5 t_2^3 + c_6 t_3^3$$

where $A = \{[1,1,0], [1,0,1], [0,1,1], [2,0,0], [0,3,0], [0,0,3]\} \subset \mathbb{Z}^3$.

The matrix A is as follows

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 & 3 & 0 \\ 0 & 1 & 1 & 0 & 0 & 3 \end{pmatrix},$$

then matrix B is

$$B = \begin{pmatrix} 3 & -1 \\ -3 & -1 \\ 0 & 1 \\ 0 & 1 \\ -1 & 0 \\ 1 & 0 \end{pmatrix}$$

By defining $l_1 = 3 - s, l_2 = -3 - s, l_3 = s, l_4 = s, l_5 = -1, l_6 = 1$, we have the parametrization

$$f_1 = \frac{l_1^3 l_6}{l_2^3 l_5} = \frac{(3-s)^3}{(3+s)^3}, f_2 = \frac{l_3 l_4}{l_1 l_2} = \frac{s^2}{(3-s)(3+s)}$$

Computed Newton polygon has 12 lattice points and the implicit equation obtained by our method is $\Delta_B(x,y) = 1 - 2x - 36xy - 96xy^2 - 64xy^3 + x^2$, thus the A-discriminant of F_A is $D_A(c) = c_2^6 c_5^2 - 2c_1^3 c_6 c_2^3 c_5 - 36 c_1^2 c_6 c_3 c_4 c_2^2 c_5 - 96 c_1 c_6 c_3^2 c_4^2 c_2 c_5 - 64 c_6 c_5 c_3^3 c_4^3 + c_1^6 c_6^2$.

Let us try to compute this example approximately (matrix is evaluated by random unitary complex points). We obtain:

$1 - 2x - 36.0001xy - 96.0001xy^2 - 64xy^3 + x^2 + 1.3921 \cdot 10^{-21} I + (-2.1482 \cdot 10^{-16} + 3.2297 \cdot 10^{-15} I)y + (2.3068 \cdot 10^{-16} - 2.8561 \cdot 10^{-15} I)y^2 + (-4.8344 \cdot 10^{-17} + 1.9862 \cdot 10^{-15} I)y^3 - 5.3777 \cdot 10^{-19} Ix - 6.4659 \cdot 10^{-15} Ixy + -1.1053 \cdot 10^{-13} Ixy^2 - 2.1119 \cdot 10^{-13} Ixy^3 + 1.6829 \cdot 10^{-19} Ix^2 + (-2.1857 \cdot 10^{-16} + 3.2281 \cdot 10^{-15} I)x^2 y + (2.0665 \cdot 10^{-16} - 2.8528 \cdot 10^{-15} I)x^2 y^2 + (-1.7033 \cdot 10^{-16} + 1.8923 \cdot 10^{-15} I)x^2 y^3$.

If we filter out coefficients whose absolute value is smaller than 10^{-13}, the result is the approximate implicit polynomial

$$\Delta_B(x,y) = 1 - 2x - 36.0001xy - 96.0001xy^2 - 64xy^3 + x^2$$

and the approximate A-discriminant is
$D_A(c) = c_2^6 c_5^2 - 2c_1^3 c_6 c_2^3 c_5 - 36.0001 c_1^2 c_6 c_3 c_4 c_2^2 c_5 - 96.0001 c_1 c_6 c_3^2 c_4^2 c_2 c_5 - 64 c_6 c_5 c_3^3 c_4^3 + c_1^6 c_6^2$.

Note that the approximation is quite close, indeed, it is accurate up to 3 decimal digits.

Discriminant of codimension 3

Example 21. [36] Consider the matrix

$$B = \begin{pmatrix} 1 & -1 & 0 \\ 1 & -1 & 1 \\ 1 & -1 & 0 \\ -1 & 2 & 0 \\ -1 & 1 & -2 \\ -1 & 0 & 1 \end{pmatrix}$$

We have

$$f_1 = \frac{(1-s)^2(1-s+t)}{(-1+2s)(-1+s-2t)(-1+t)},$$
$$f_2 = \frac{(-1+2s)^2(-1+s-2t)}{(1-s)^2(1-s+t)},$$
$$f_3 = \frac{(1-s+t)(-1+t)}{(-1+s-2t)^2}$$

For this parametrization our support predicting software `ResPol` gives Newton polytope

$$[0,6,7],[6,0,0],[0,6,0],[0,0,7],[0,0,0],[6,6,4],[6,0,4],[6,6,0],[0,6,7]$$

with 308 inside points.

Computing the implicit equation we get kernel space of dimension 8; the polynomial of the least degree actual implicit equation:
$\Delta_B(x,y,z) = 16x^5y^5z^3 + 80x^4y^4z^3 - 8x^4y^4z^2 + 500x^4y^3z^2 + 3125x^4y^2z^2 + 160x^3y^3z^3 - 32x^3y^3z^2 + x^3y^3z + 1000x^3y^2z^2 - 225x^3y^2z + 160x^2y^2z^3 - 48x^2y^2z^2 + 3x^2y^2z + 500x^2yz^2 - 225x^2yz + 27x^2y + 80xyz^3 - 32xyz^2 + 3xyz + 16z^3 - 8z^2 + z.$

After substitution according to $l_1 = 1-s, l_2 = 1-s+t, l_3 = 1-s, l_4 = -1+2s, l_5 = -1+s-2t, l_6 = -1+t$ and

$$\Delta_B(f_1,f_2,f_3) = D\left(1,1,1,\frac{l_1^2 l_2}{l_4 l_5 l_6},\frac{l_4^2 l_5}{l_1 l_2 l_3},\frac{l_2 l_6}{l_5^2}\right),$$

the A-discriminant is as follows $D_A(c) = c_5^4c_6^3c_3^5 - 8c_2c_5^2c_6^4c_3^5 + 16c_2^2c_6^5c_3^5 + 3c_5^4c_6^2c_3^4c_1c_4 - 32c_2c_5^2c_6^3c_3^4c_1c_4 + 80c_2^2c_6^4c_3^4c_1c_4 + 27c_5^5c_3^4c_1^3 - 225c_2c_5^3c_6c_3^4c_1^3 + 500c_2^2c_5c_6^2c_3^4c_1^3 + 3c_5^4c_6c_3^3c_1^2c_4^2 - 48c_2c_5^2c_6^2c_3^3c_1^2c_4^2 + 160c_2^2c_6^3c_3^3c_1^2c_4^2 - 225c_2c_5^3c_3^3c_1^4c_4 + 1000c_2^2c_5c_6c_3^3c_1^4c_4 + c_5^4c_3^2c_1^3c_4^3 - 32c_2c_5^2c_6c_3^2c_1^3c_4^3 + 160c_2^2c_6^2c_3^2c_1^3c_4^3 + 3125c_2^3c_3^3c_1^6 + 500c_2^2c_5c_3^2c_1^5c_4^2 - 8c_2c_5^2c_3c_1^4c_4^4 + 80c_2^2c_6c_3c_1^4c_4^4 + 16c_2^2c_1^5c_4^5.$

Example 22. Consider the matrix

$$B = \begin{pmatrix} 3 & 0 & 0 \\ -1 & -1 & -1 \\ -1 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & 2 & 1 \\ -1 & 1 & -1 \end{pmatrix}$$

Let define

$$l_1 = 3, l_2 = -1-s-t, l_3 = -1-s, l_4 = -s+t, l_5 = 2s+t, l_6 = -1+s-t,$$

then we have the following parametrization

$$f_1 = \frac{l_1^3}{l_2 l_3 l_6} = \frac{27}{(-1+s-t)(-1-s-t)(-1-s)},$$
$$f_2 = \frac{l_5^2 l_6}{l_2 l_3 l_4} = \frac{(2s+t)^2(-1+s-t)}{(-1-s-t)(-1-s)(-s+t)},$$
$$f_3 = \frac{l_4 l_5}{l_2 l_6} = \frac{(-s+t)(2s+t)}{(-1+s-t)(-1-s-t)}$$

`ResPol` gives Newton polytope:

$[6,4,3],[6,0,0],[0,6,0],[0,0,9],[0,0,0],[4,6,5],[6,0,3],[6,4,0],[0,6,9],[4,6,0],[6,4,3]$.

Applying our interpolation method we compute kernel vector space of dimension 6. It appears, implicit polynomial of this parametrization, found as described above, has degree 10 and the A-discriminant is: $\Delta_B(x,y,z) = -14348907y^3 + 314928y^2z^8 + 43046721y^3z - 239112xy^4z^5 + 451980xy^4z^4 + 731916xy^4z^3 - 1023516xy^4z^2 + 393660xy^4z + 62208x^2yz^5 + 93312x^2yz^4 + 23328xy^2z^7 + 7912566xy^2z^5 + 98415xy^2z^4 - 13994613xy^2z^3 + 27103491xy^2z^2 + 1102248xyz^6 + 17537553xyz^4 + 1417176xyz^5 + 414072xy^3z^6 - 125388xy^3z^5 - 1062882xy^3z^4 + 5334093xy^3z^3 - 1200663xy^3z^2 + 3011499xy^3z - 729x^2y^4z^4 + 2187x^2y^4z^3 - 2187x^2y^4z^2 + 729x^2y^4z + 25272x^2y^3z^5 - 6804x^2y^3z^4 - 657666x^2y^3z^3 + 19683x^2y^3z^2 + 104976x^2y^3z + 432x^2y^2z^6 - 864x^2y^2z^5 + 1368576x^2y^2z^4 + 1465776x^2y^2z^3 + 2511405x^2y^2z^2 + 432x^3y^3z^4 - 864x^3y^3z^3 - 1512x^3y^3z^2 + 1944x^3y^3z + 66816x^3y^2z^3 + 86400x^3y^2z^2 + 1024x^3yz^4 + 1024x^4y^2z^2 + 314928y^5z^5 + 944784y^5z^3 - 944784y^5z^4 - 314928y^5z^2 + 944784y^4z^6 + 5196312y^4z^4 - 1889568y^4z^5 + 12754584y^4z^2 - 12754584y^4z^3 - 4251528y^4z - 944784y^3z^6 + 944784y^3z^7 - 25509168y^3z^4 + 12754584y^3z^5 - 43046721y^3z^2 + 27103491y^3z^3 - 12754584y^2z^5 + 12754584y^2z^6 + 43046721y^2z^2 - 86093442y^2z^3 + 43046721y^2z^4 + 4251528yz^7 + 43046721yz^5 - 43046721yz^4 - 729x^3y^3 - 59049x^2y^3 + 14348907z^6 - 1594323xy^3$.

Example 23. [15] We compute discriminant of $F_A = \sum_{a\in A} c_a t^a$ where

$$A = \{[0,2,0],[0,0,6],[0,1,2],[1,2,0],[1,1,3],[1,2,2],[1,1,2]\} \subset \mathbb{Z}^3.$$

We build a matrix A

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 2 & 0 & 1 & 2 & 1 & 2 & 1 \\ 0 & 6 & 2 & 0 & 3 & 2 & 2 \end{pmatrix}$$

An integer matrix B which forms a basis of kernel of A is

$$B = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ -1 & -1 & -2 \\ 0 & 2 & 1 \\ 2 & 0 & 0 \\ -1 & -1 & -1 \\ -1 & -1 & 0 \end{pmatrix}$$

Let $l_1 = 1+t, l_2 = s+t, l_3 = -1-s-2t, l_4 = 2s+t, l_5 = 2, l_6 = -1-s-t, l_7 = -1-s$, then we have the following parametrization:

$$f_1 = \frac{l_1 l_5^2}{l_3 l_6 l_7} = \frac{4(1+t)}{(-1-s-2t)(-1-s-t)(-1-s)},$$
$$f_2 = \frac{l_2 l_4^2}{l_3 l_6 l_7} = \frac{(s+t)(2s+t)^2}{(-1-s-2t)(-1-s-t)(-1-s)},$$
$$f_3 = \frac{l_1 l_2 l_4}{l_3^2 l_6} = \frac{(1+t)(s+t)(2s+t)}{(-1-s-2t)^2(-1-s-t)}$$

For this parametrization, computing Newton Polytope vertices results in

$$[0,3,9],[9,0,0],[0,9,0],[0,0,9],[0,0,0],[9,0,3],[0,9,3],[3,0,9],[0,3,9].$$

Kernel space that we compute has dimension 20; we obtain the implicit equation Δ_B of degree 9:

$\Delta_B(x,y,z) = 512xyz^3 - 576xyz^5 - 1024xy^2z^2 + 3712xy^2z^3 + 320xy^2z^4 - 1664xy^3z^2 + 320xy^3z^3 - 64xy^4z^2 - 608x^2yz^3 + 368x^2yz^4 - 960x^2yz^5 + 1824x^2y^2z^2 + 880x^2y^2z^3 + 1088x^2y^2z^4 - 64x^2y^2z^5 - 1296x^2y^3z + 64x^2y^3z^2 - 64x^2y^3z^3 + 64x^2y^3z^4 - 16x^2y^4z + 144x^3yz^3 - 640x^3yz^4 - 128x^3yz^5 + 108x^3y^2z + 60x^3y^2z^2 + 784x^3y^2z^3 + 128x^3y^2z^4 - 16x^3y^3z^2 + 16x^3y^3z^3 - 16x^4yz^3 + 64x^4yz^4 - 27x^4y^2z + 128x^4y^2z^2 + 32x^4y^2z^3 + 16x^5yz^3 + 2048y^2z^3 - 144x^2z^5 + 192x^3z^5 - 216x^3y^3 - 64x^4z^5$

and the following A-discriminant:

$D_A(c) = 1024c_2^5c_4^6c_1^3c_7^2 + 4608c_1^5c_5^4c_2^2c_3c_6c_7^3 - 1476c_1^3c_5^4c_2c_3^4c_6^2c_7^2 + 15104c_1^3c_5^4c_2^2c_4^2c_3^3c_6c_7 - 20224c_1^4c_5^4c_2^2c_4c_3^2c_6c_7^2 + 108c_1c_5^4c_3^7c_6^3c_7 + 2508c_1^2c_5^4c_2c_4c_3^5c_6^2c_7 - 4960c_1c_5^2c_2^2c_4^3c_3^5c_6^2c_7 + 54016c_1^3c_5^2c_2^3c_4^3c_3^2c_6c_7^2 + 14528c_1^2c_5^2c_2^2c_4^2c_3^4c_6^2c_7^2 - 576c_5^2c_2c_4^2c_3^7c_6^3c_7 + 768c_1c_5^2c_2c_4c_3^6c_6^3c_7^2 - 82944c_1^4c_5^2c_2^3c_4^2c_3c_6c_7^3 + 480c_1^3c_5^2c_2^2c_4c_3^3c_6^2c_7^3 - 10800c_1^4c_5^2c_2^2c_3^2c_6^2c_7^4 + 64c_1^2c_5^2c_2c_3^5c_6^3c_7^3 - 11648c_1^2c_5^2c_2^3c_4^4c_3^3c_6c_7 - 1408c_1^4c_5^6c_2c_3^3c_6c_7 + 46656c_1^5c_2^3c_6^2c_7^6 - 768c_1^4c_5^6c_2^2c_4^2c_3^2 - 13824c_2^4c_4^3c_1^4c_6c_7^4 + 3072c_1^5c_5^4c_2^3c_4^2c_7^2 - 256c_1^2c_5^2c_2^4c_4^6c_3^2 + 3072c_1^4c_5^2c_2^4c_4^4c_7^2 - 768c_1^3c_5^4c_2^3c_4^4c_3^2 + 1024c_1^6c_5^6c_2^2c_7^2 - 256c_1^5c_5^8c_2c_3^2 - 1024c_2^5c_4^7c_1^2c_3c_7 - 8704c_2^2c_4^2c_1c_3^5c_6^3c_7^3 - 1024c_2c_4c_3^7c_6^4c_7^3 + 79488c_2^3c_4^2c_1^3c_3^2c_6^2c_7^4 + 23040c_2^2c_4c_1^2c_3^4c_6^3c_7^4 + 1024c_2c_1c_3^6c_6^4c_7^4 - 108864c_2^3c_4c_1^4c_3c_6^2c_7^5 - 13824c_2^2c_1^3c_3^3c_6^3c_7^5 - 4288c_2^3c_4^4c_1c_3^4c_6^2c_7^2 - 2272c_1^2c_5^4c_2^2c_4^3c_3^4c_6 - 108c_5^4c_4c_3^8c_6^3 - 128c_1c_5^2c_2^3c_4^5c_3^4c_6 - 16c_5^2c_2^2c_4^4c_3^6c_6^2 + 41472c_1^5c_5^2c_2^3c_4c_6c_7^4 - 936c_1c_5^4c_2c_4^2c_3^6c_6^2 + 27c_1^2c_5^6c_3^6c_6^2 + 1312c_1^3c_5^6c_2c_4c_3^4c_6 - 12928c_2^3c_4^3c_1^2c_3^3c_6^2c_7^3 - 512c_2^2c_4^3c_3^6c_6^3c_7^2 - 3072c_1^3c_5^2c_2^4c_4^5c_3c_7 - 512c_2^4c_4^6c_1c_3^3c_6c_7 - 64c_2^3c_4^5c_3^5c_6^2c_7 - 8704c_2^4c_4^5c_1^2c_3^2c_6c_7^2 + 23040c_2^4c_4^4c_1^3c_3c_6c_7^3 - 1024c_1^5c_5^6c_2^2c_4c_3c_7 - 3072c_1^4c_5^4c_2^3c_4^3c_3c_7.$

3.4 Conclusions

Our implicitization algorithm was first implemented in `Maple` mathematics software system, since it was well suited for symbolic computations and we had an emphasis on exact solving. Later, when we started experimenting with the approximate implicitization and the need for a numerical solving have arisen we have employed `SAGE`.

Experiments with the curves and surfaces of Bernstein bases have shown that while it is always possible to compute an approximate implicit equation, the necessity of conversion to the monomial basis leads to the loss of accuracy. We have employed several methods to evaluate the accuracy of the approximation; some rely on knowing the exact implicit equation and thus, while more reliable, not universally applicable.

We have compared the implementation of our algorithm with other methods, implemented in the same software system, finding, that for the high dimensions, starting from surfaces, and for the large degrees it shows better results than many others. For instance, our method have proved to be competitive to the popular Gröbner bases method. However, when comparing the methods specified for curves only we found out that the μ-bases method implemented in `Maple` is to be faster than our `imcurve`.

Exploiting the fact that our implicitization tool is well suited for exact solving we have applied it to compute determinants and resultants. Analyzing complexity of the algorithms we conclude that in the case when the corresponding parametrization provides comparatively small Newton polytope using our algorithm is preferable.

Chapter 4

Conclusions and further remarks

In this chapter we give a summary of the results of our work and discuss the further research directions.

4.1 Results

We have developed an algorithm for computing implicit equations that combines linear algebra with the promising support prediction methods. The method applies to polynomial, rational and trigonometric parametrizations of classical algebraic equations of curves and (hyper)surfaces. The method is quite efficient; also it works for in the presence of the base points.

Our implicitization algorithm is closely related to the interpolation-based method presented in [43]. More efficient method for computing the implicit Newton polytope [49], used to determine the interpolation space, enabled us to implicitize curves and surfaces of high degree as well as hypersurfaces, among them the bicubic surface proposed in [43] as an open problem. Section 2.2 provides a detailed description of the support prediction method.

Our main contributions are the following:

- We have studied the cases where the resulting polynomial contains extraneous factors and have proved the Theorem 15 that shows the dependence of the interpolation matrix corank on the predicted Newton polytope. Hence one of the possible ways we offer that allow to minimize an extraneous factor: reducing the predicted Newton polytope. The results of our study of the multidimensional kernel space cases and related issues have first appeared in [54]; in this thesis they are described in Section 2.4.
- The series of experiments where we compute the kernel vector of the interpolation matrix by symbolic methods show that for sufficiently generic parametrizations our algorithm provides exact implicit equations. Being an effective tool for exact implicitization our method can be used also for solving non-geometrical problems such as computing of discriminant of a multivariate polynomial, as demonstrated in Section 3.3 or resultant of a system of multivariate polynomials (see Example 13).
- Computing the kernel vector of the interpolation matrix numerically, by applying singular value decomposition, results in an approximate implicit equation. We have experimented with curves and surfaces in monomial and Bernstein bases with an emphasis on the approximation accuracy, computational time and the resulting polynomial degree

seeking to find an optimal correlation of the three. We have discussed specifics of the implicitization of the curves and surfaces in Bernstein bases by our method in Sections 2.3.5 and 3.1.3.

- We have implemented our algorithm in `Maple` and `SAGE` mathematics software systems. Detailed description of our implementation was given in Section 3.1. The implementation features two main functions: `imgen` and `imcurve`, the former takes as an input parametric equations and the vertices of the predicted implicit polytope while the latter, customized for implicitization of curves, takes as an input only the parametrization. The reason we have chosen `Maple` is its particular suitability of for symbolic computations. The `SAGE` mathematics software system was used mainly for numerical computations, such as implicitization of curves and surfaces in Bernstein basis.

Our implicitization method (see Algorithm 1) takes as an input polynomial or rational parametrization $x_i = f_i(t_1, \dots, t_n)$, $i = 0, \dots, n$ and the implicit polytope (or a superset of it). The output is the implicit polynomial $p(x_i)$ (in certain cases a multiple of $p(x_i)$). The algorithm operates in the following steps:

1. Compute all lattice points $S \subseteq \mathbb{N}^{n+1}$ in the polytope.
2. Repeat $\mu \geq |S|$ times: Select value τ for t, evaluate $x_i(\tau)$, $i = 0, \dots, n$, then evaluate each monomial in S.
3. Construct the $\mu \times |S|$ matrix M with rows indexed by τ's and columns indexed by m_j's

$$M = \begin{bmatrix} m_1|_{t=\tau_1} & \cdots & m_{|S|}|_{t=\tau_1} \\ \vdots & \cdots & \vdots \\ m_1|_{t=\tau_\mu} & \cdots & m_{|S|}|_{t=\tau_\mu} \end{bmatrix}$$

4. Compute vector p in the kernel of M and return the primitive part of polynomial $p^\top S$.

It is important to choose sufficiently generic values τ when building the matrix M so that its rows are linearly independent. In case of the rational parametrization the τ values where the determinant vanishes should be avoided. We have addressed the issues related to selecting evaluation values in [2.3.5]. Our experiments have shown that random integers in range $-2|S| \dots 2|S|$ provide the best results in terms of solving time and numerical stability for implicitization of classical algebraic curves and surfaces. When the parametrization is defined in Bernstein basis it is important when selecting τ values to ensure that they lie within the region of interest of the parametrization.Random rational values and Chebyshev nodes have provided satisfying results here.

We have addressed the problem of measuring the quality of the approximation when the interpolation is done by numerical means in [3.1.4]. The condition number of the matrix M allows us to estimate corank of the matrix. When the exact implicit equation is available, the natural approach to evaluation of the accuracy is computing the distance between the two coefficient vectors.

In Section 3.2 we have demonstrated how our implicitization method performs in comparison with the others. Our algorithm has proved to be an efficient tool for finding exact implicit equation, able to compete successfully with such popular methods as Gröbner bases in the terms of computation time. An accuracy of the approximate implicit equations obtained by our method appears to be better that that of the `Maple`'s native function *Implicitize()*. In general, our method performs comparatively well for large degree and high dimension surfaces and hypersurfaces. Besides, our method can be successfully applied to parameterizations with base points which raise important issues for other implicitization methods.

We have evaluated the complexity of our algorithm in [2.3.3], proving the Theorem 11: the overall complexity of our implicitization algorithm requires $O(\mu\,|S|^2)$ arithmetic operations.

4.2 Future work

Here we us list several open problems that are out the scope of our main work yet present interesting challenges.

Implicitizing space curves. Our implicitization algorithm was intended and applied in this work to implicitize curves and surfaces of codimension 1 only. That does not mean that it is inapplicable otherwise; indeed, provided sufficiently generic parametrization of the space curve and the implicit support we can successfully perform interpolation. The dimension of the resulting kernel vector space corresponds to the number of equations constituting the implicit representation.

However, the support prediction method we use in our work cannot be applied to space curves. Hence the need for alternative methods to compute a bound on the implicit degree.

Matrix-based representation. The output of our algorithm is an implicit equation, however, for practical purposes such representation is often redundant.

When the curves and surfaces are of high degree the polynomials produced to compute intersection points and curves become difficult to operate and require an enormous storage space due to the large number and big size of the coefficients. An alternative is the so-called "polynomial algebra by values" approach where the huge polynomial equations coming from Elimination Theory are replaced by big structured and sparse numerical matrices. First the technique was introduced in [83] where a matrix is used for representation of the surface to find the intersection between a curve and a surface. Intersection problems between parametric and implicit curves have been studied in [84, 85]: instead of the polynomial derived from the matrix by computing its determinant a matrix pencil whose generalized eigenvalues are the roots of the determinant is used.

Resent publications prove that the matrix-based representation can be effectively used for solving such problems as membership, intersection, inversion, computation of singularities, determining the topology [39]. New approaches applicable to planar and space curves [16] and surfaces [17] have been proposed; here μ-basis of a parametrization is used to construct the matrix-based representation.

It is possible that our algorithm can be adapted in a way that while bypassing actual equation construction effectively provides an answer if the point belongs to the the geometric object. Our method represents an implicit (hyper)surface by a kernel vector. It is challenging to devise suitable CAGD algorithms that exploit this representation, for instance to compute surface-surface intersection, as in [43, 44].

Evaluation by integers. As our experiments show, evaluation by the random integers appears to be optimal for exact implicitization of classical algebraic curves and surfaces. Studying of the possible ways to reduce growth of the matrix entries when the degrees get higher while maintaining evaluation values sufficiently general so that all the rows of the constructed matrix remain linearly independent presents an interesting challenge.

Newton polytope in Bernstein basis. Our algorithm shows the best performance when applied to exact implicitization. However, it can be used for approximate implicitization of geometric objects represented in NURBS form after converting them to the power basis. The need for the conversion presents a major drawback: in the NURBS format curves and surfaces are usually given by floating point coefficients and recalculating the parametrization in power basis furthers the precision loss. As a result, approximate implicit representations of the NURBS curves and surfaces obtained by interpolation are often inaccurate.

Currently our method is restricted by the support prediction operating solely on the power basis. Hence another direction for improvement: our approach could be extended to interpolating the implicit polynomial in other bases, such as Bernstein or Lagrange. In fact, while the algorithm for computing the resultant polytope to Bernstein basis has yet to appeared, a method that allows to express the information necessary for dealing with the position and intersection problems by formulation of the Bézout matrix in Lagrange basis have been presented in [6].

Approximate implicit equation of lower degree. We would like to note that in this work when we refer to an implicit equation as approximate that meant it was computed by numerical means. In general the bounds on the implicit degree in such a case are determined by the predicted polytope, i.e. are the same as is for computing an exact implicit equation. Consequently, the approximate equations here have the same degree as the corresponding exact, if such exist, unless stated otherwise. However, one of the attractive features of the approximate implicitization is the possibility to obtain lower degree equation.

Our method can be employed in computing approximate implicit equation of the reduced degree: we use successively larger subsets of the predicted polytope, until the obtaining satisfyingly accurate approximation or reaching the bound set by the implicit polytope. Such low degree approximation and the means to ensure its optimal accuracy can be a subject for further study.

Implicitizing piecewise parametrization. In our work we have addressed implicitization of the Bézier curves and surface patches, however the fact is that in practice geometric objects represented in Bernstein basis usually are piecewise parametrized.

We consider applying our method to implicitize curve or surface splines defined by k segments or patches, respectively. Assuming the k parametric representations yield polynomials with the same Newton polytopes, one could use the implicit polytope defined by any of these systems, the corresponding k implicit polytopes would be the same. Then, we can form a single matrix M and evaluate it over points spanning all k segments or patches, thus expecting a single (approximate) implicit polynomial. Similarly, it is possible to approximate k manifolds with a single implicit equation, by applying SVD on $[M_1 \cdots M_k]^\top$.

Alternative numerical methods. In order to obtain an approximate solution we have employed the singular value decomposition; it would be interesting to compare the performance of different numerical methods when applied to interpolation. We consider, for instance, the method of least squares.

Index

Bibliography

[1] W. A. Adkins, J. W. Hoffman, and H. H. Wang. Equations of parametric surfaces with base points via syzygies. *J. Symb. Comput.*, 39(1):73--101, Jan. 2005.

[2] M. Aigner, A. Poteaux, and B. Jüttler. Approximate implicitization of space curves. In U. Langer and P. Paule, editors, *Symbolic and Numeric Computation*. Springer, Vienna, 2012.

[3] C. Alonso, J. Gutierrez, and T. Recio. Reconsidering algorithms for real parametric curves. *Appl. Algebra Eng. Commun. Comput.*, 6(6):345--352, 1995.

[4] H. Alt, C. Knauer, and C. Wenk. Comparison of Distance measures for planar curves. *Algorithmica*, 38:45--58, 2004.

[5] L.-E. Andersson and N. F. Stewart. *Introduction to the Mathematics of Subdivision Surfaces*. SIAM, 2010.

[6] D. A. Aruliah, R. M. Corless, L. González-Vega, and A. Shakoori. Geometric applications of the bézout matrix in the lagrange basis. In *Proceedings of the 2007 international workshop on Symbolic-numeric computation*, SNC '07, pages 55--64, New York, NY, USA, 2007. ACM.

[7] C. L. Bajaj and G. Xu. Spline approximations of real algebraic surfaces. *J. Symb. Comput.*, 23(2/3):315--333, 1997.

[8] O. J. D. Barrowclough and T. Dokken. Approximate implicitization of triangular Bézier surfaces. In *Proceedings of the 26th Spring Conference on Computer Graphics*, SCCG '10, pages 133--140, New York, NY, USA, 2010.

[9] A. Barvinok and J. Pommersheim. An algorithmic theory of lattice points in polyhedra. *Complexity*, 38:91--147, 1999.

[10] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. pages 301--309. ACM Press, New York, 1988.

[11] T. G. Berry and R. R. Patterson. Implicitization and parametrization of nonsingular cubic surfaces. *Computer Aided Geometric Design*, 18(8):723--738, 2001.

[12] W. Bruns, B. Ichim, and C. Söger. Normaliz. algorithms for rational cones and affine monoids. Available from http://www.math.uos.de/normaliz.

[13] K. Buchin, M. Buchin, and C. Wenk. Computing the Fréchet distance between simple polygons. *Comput. Geom.*, 41(1-2):2--20, 2008.

[14] L. Busé, D. A. Cox, and C. D'Andrea. Implicitization of surfaces in the projective space in the presence of base points. *Journal of Algebra and Its Applications*, 2(2):189--214, 2003.

[15] L. Busé, A. Dickenstein, and I. Z. Emiris. Discriminant with codimension 3, 2002. Manuscript, INRIA Sophia-Antipolis.

[16] L. Busé and T. Luu Ba. Matrix-based implicit representations of algebraic curves and applications. *Computer Aided Geometric Design*, 27(9):681--699, 2010.

[17] L. Busé and T. Luu Ba. The surface/surface intersection problem by means of matrix based representations. *Computer Aided Geometric Design*, 29(8):579--598, 2012.

[18] J. F. Canny and I. Z. Emiris. A subdivision-based algorithm for the sparse resultant. *J. ACM*, 47(3):417--451, May 2000.

[19] J. F. Canny, E. Kaltofen, and Y. Lakshman. Solving systems of non-linear polynomial equations faster. pages 121--128, 1989.

[20] E. Catmull and J. Clark. Seminal graphics. chapter Recursively generated B-spline surfaces on arbitrary topological meshes, pages 183--188. ACM, New York, NY, USA, 1998.

[21] A. Cayley. On the theory of elimination. *Dublin Math. J.*, II:116--120, 1848.

[22] F. Chen, D. A. Cox, and Y. Liu. The μ-basis and implicitization of a rational parametric surface. *J. Symb. Comput.*, 39(6):689--706, June 2005.

[23] F. Chen, L. Shen, and J. Deng. Implicitization and parametrization of quadratic and cubic surfaces by μ-bases. *Computing*, 79(2):131--142, Apr. 2007.

[24] F. Chen and W. Wang. The μ-basis of a planar rational curve—properties and computation. *Graphical Models*, 64(6):368--381, 2002.

[25] F. Chen and W. Wang. Revisiting the μ-basis of a rational ruled surface. *J. Symb. Comput.*, 36(5):699--716, Nov. 2003.

[26] X.-D. Chen, L. Chen, Y. Wang, G. Xu, J.-H. Yong, and J.-C. Paul. Computing the minimum distance between two Bézier curves. *J. Computational Applied Math*, 229:294--301, 2009.

[27] X.-D. Chen, W. Ma, G. Xu, and J.-C. Paul. Computing the Hausdorff distance between two B-spline curves. *Computer-Aided Design*, 42:1197--1206, 2010.

[28] E.-W. Chionh. *Base points, resultants, and the implicit representation of rational surfaces*. PhD thesis, Waterloo, Ont., Canada, Canada, 1990. UMI Order No.

[29] E.-W. Chionh and R. N. Goldman. Degree, multiplicity, and inversion formulas for rational surfaces using u-resultants. *Computer Aided Geometric Design*, 9(2):93--108, 1992.

[30] R. M. Corless, M. Giesbrecht, I. S. Kotsireas, and S. M. Watt. Numerical implicitization of parametric hypersurfaces with linear algebra. In *Proc. AISC*, pages 174--183, 2000.

[31] D. A. Cox, J. B. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer, NY, 2nd edition, 1996.

[32] D. A. Cox, J. B. Little, and D. O'Shea. *Using Algebraic Geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer-Verlag, NY, 1998.

[33] D. A. Cox, T. W. Sederberg, and F. Chen. The moving line ideal basis of planar rational curves. *Comput. Aided Geom. Design*, 15 (8):803--827, 1998.

[34] M. A. Cueto. *Tropical Implicitization*. PhD thesis, Dept Mathematics, UC Berkeley, 2010.

[35] M. A. Cueto and A. Dickenstein. Some results on inhomogeneous discriminants. In *Proc. XVI Latin Amer. Algebra Colloq., Bibl. Rev. Mat. Iberoamericana*, pages 41--62, 2007. arXiv:math/0610031v2 [math.AG].

[36] M. A. Cueto and A. Dickenstein. Some results on inhomogeneous discriminants. In *Proc. XVI Latin Amer. Algebra Colloq., Bibl. Rev. Mat. Iberoamericana*, pages 41--62, 2007. arXiv:math/0610031v2 [math.AG].

[37] C. D'Andrea. Macaulay-style formulas for the sparse resultant. *Trans. of the AMS*, 354:2595--2629, 2002.

[38] C. D'Andrea and M. Sombra. The Newton polygon of a rational plane curve. *Math. in Computer Science*, 4(1):3--24, 2010.

[39] G. M. Diaz-Toca, M. Fioravanti, L. González-Vega, and A. Shakoori. Using implicit equations of parametric curves and surfaces without computing them: Polynomial algebra by values. *Computer Aided Geometric Design*, 30(1):116--139, 2013.

[40] A. Dickenstein, E. M. Feichtner, and B. Sturmfels. Tropical discriminants. *J. AMS*, pages 1111--1133, 2007.

[41] A. Dickenstein and B. Sturmfels. Elimination theory in codimension 2. *J . Symbolic Computation*, 34:119--135, 2002.

[42] T. Dokken. Approximate implicitization. In *Mathematical Methods for Curves and Surfaces*, pages 81--102. Vanderbilt Univ., Nashville, USA, 2001.

[43] T. Dokken and J. B. Thomassen. Overview of approximate implicitization. *Topics in algebraic geometry and geometric modeling*, 334:169--184, 2003.

[44] T. Dokken and J. B. Thomassen. Weak approximate implicitization. In *Proc. IEEE Intern. Conf. Shape Modeling Appl.*, page 31, 2006.

[45] D. Doo and M. Sabin. Behaviour of recursive subdivision surfaces near extraordinary points. *Computer-Aided Design*, 10:356--360, 1978.

[46] M. Elkadi, A. Galligo, and T. Luu Ba. Approximate gcd of several univariate polynomials with small degree pertubations. *J. Symbolic Comput*, 47(4):410--421, 2012.

[47] I. Z. Emiris. On the complexity of sparse elimination. *J. Complexity*, 12:134--166, 1996.

[48] I. Z. Emiris, V. Fisikopoulos, and C. Konaxis. Regular triangularions and resultant polytopes. In *Proc. European Workshop Comp. Geometry*, pages 137--140, 2010.

[49] I. Z. Emiris, V. Fisikopoulos, C. Konaxis, and L. Peñaranda. An output-sensitive algorithm for computing projections of resultant polytopes. In *Proceedings of the 2012 symposuim on Computational Geometry*, SoCG '12, pages 179--188, New York, NY, USA, 2012. ACM. Final version to appear in IJCGA.

[50] I. Z. Emiris, A. Galligo, and H. Lombardi. Certified approximate univariate GCDs. *J. Pure & Applied Algebra, Special Issue on Algorithms for Algebra*, 117 & 118:229--251, May 1997.

[51] I. Z. Emiris, T. Kalinka, and C. Konaxis. Implicitization of curves and surfaces using predicted support. In *Electr. Proc. Inter. Works. Symbolic-Numeric Computation*, San Jose, Calif., 2011.

[52] I. Z. Emiris, T. Kalinka, and C. Konaxis. Sparse implicitization via interpolation. To appear in SAGA Volume (Springer), 2013.

[53] I. Z. Emiris, T. Kalinka, C. Konaxis, and T. Luu Ba. Implicitization of curves and (hyper)surfaces using predicted support. *Theoretical Computer Science*, 2012.

[54] I. Z. Emiris, T. Kalinka, C. Konaxis, and T. Luu Ba. Sparse implicitization by interpolation: Characterizing non-exactness and an application to computing discriminants. *Computer-Aided Design*, 45(2):252--261, 2013. Special Issue Conference on SPM.

[55] I. Z. Emiris, C. Konaxis, and L. Palios. Computing the Newton polygon of the implicit equation. *Mathematics in Computer Science, Special Issue on Computational Geometry and Computer-Aided Design*, 4(1):25--44, 2010.

[56] I. Z. Emiris and I. S. Kotsireas. Implicit polynomial support optimized for sparseness. In *Proc. Intern. Conf. Computational science appl.: Part III*, pages 397--406, Berlin, 2003. Springer.

[57] I. Z. Emiris and V. Y. Pan. Symbolic and numeric methods for exploiting structure in constructing resultant matrices. 33:393--413, 2002.

[58] I. Z. Emiris and V. Y. Pan. Improved algorithms for computing determinants and resultants. *J. Complexity, Special Issue*, 21:43--71, 2005. Special Issue on FOCM-02.

[59] A. Esterov and A. Khovanskiĭ. Elimination theory and newton polytopes. arXiv:0611107[math], 2006.

[60] R. T. Farouki and C. A. Neff. Algebraic properties of plane offset curves. *Computer Aided Geometric Design*, 7(1-4):101--127, 1990.

[61] J.-C. Faugère, G. Moroz, F. Rouillier, and M. Safey El Din. Classification of the perspective-three-point problem, discriminant variety and real solving polynomial systems of inequalities. In *Proc. ACM ISSAC*, pages 79--86, 2008.

[62] M. Franz. Convex: a maple package for convex geometry, version 1.1.3. Available at: http://www-math.uwo.ca.

[63] I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, Resultants and Multidimensional Determinants*. Birkhäuser, Boston, 1994.

[64] I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, resultants & multidimensional determinants*. Birkhauser, 2008.

[65] R. N. Goldman, T. W. Sederberg, and D. C. Anderson. Vector elimination: A technique for the implicitization, inversion, and intersection of planar parametric rational polynomial curves. *Computer Aided Geometric Design*, 1(4):327--356, 1984.

[66] A. Gomes, I. Voiculescu, J. Jorge, B. Wyvill, and C. Galbraith. *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[67] L. González-Vega. Implicitization of parametric curves and surfaces by using multidimensional Newton formulae. *J. Symbolic Comput.*, 23(2-3):137--151, 1997. Parametric algebraic curves and applications (Albuquerque, NM, 1995).

[68] E. Hartmann. Numerical parameterization of curves and surfaces. *Computer Aided Geometric Design*, 17(3):251--266, 2000.

[69] R. Hartshorne. *Algebraic geometry*. Graduate texts in mathematics. Springer, New York, 1977.

[70] C. M. Hoffmann. Conversion methods between parametric and implicit curves and surfaces. Technical Report CSD-TR-975, Purdue University, CS Dept., West Lafayette, IN 47907, USA, April 1990. 64 pages.

[71] A. N. Jensen and J. Yu. Computing tropical resultants. arXiv:1109.2368v1[math.AG], 2011.

[72] S. Ji, J. Kollár, and B. Shiffman. A global Łojasiewicz inequality for algebraic varieties. *Trans. Am. Math. Soc.*, 329(2):813--818, 1992.

[73] B. Jüttler. Bounding the Hausdorff distance of implicitly defined and/or parametric curves. In T. Lyche and L. Schumaker, editors, *Math. Methods in CAGD*, pages 1--10. 2000.

[74] E. Kaltofen and Y. Lakshman. Improved sparse multivariate polynomial interpolation algorithms. In P. Gianni, editor, *Proc. ACM Intern. Symp. on Symbolic & Algebraic Comput. 1988*, volume 358 of *Lect. Notes in Comp. Science*, pages 467--474. Springer-Verlag, 1989.

[75] E. Kaltofen, Z. Yang, and L. Zhi. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In *Proc. ISSAC'06*, pages 169--177, Genova, Italy, 2006.

[76] M. M. Kapranov. A characterization of a-discriminant hypersurfaces in term of the gauss map. *Math. Ann*, 290:277--285, 1991.

[77] M. M. Kapranov. A characterization of A-discriminantal hypersurfaces in terms of the logarithmic Gauss map. *Mathematische Annalen*, 290:277--285, 1991.

[78] I. S. Kotsireas and E. S. C. Lau. Implicitization of polynomial curves. In *Proc. ASCM*, pages 217--226, Beijing, 2003.

[79] R. Krasauskas and C. Mäurer. Studying cyclides with Laguerre geometry. *Comput. Aided Geom. Des.*, 17(2):101--126, 2000.

[80] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 313--324, New York, NY, USA, 1996. ACM.

[81] J. A. D. Loera, D. Haws, R. Hemmecke, P. Huggins, J. Tauzer, and R. Yoshida. A user's guide for latte v1.1. Software package LattE is available at http://www.math.ucdavis.edu/˜latte/, 2003.

[82] F. S. Macaulay. Some formulae in elimination. *Proc. London Math. Soc.*, 1(33):3--27, 1902.

[83] D. Manocha and J. F. Canny. A new approach for surface intersection. In *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, SMA '91, pages 209--219, New York, NY, USA, 1991. ACM.

[84] D. Manocha and J. Demmel. Algorithms for intersecting parametric and algebraic curves i: simple intersections. *ACM Trans. Graph.*, 13(1):73--100, 1994.

[85] D. Manocha and J. Demmel. Algorithms for intersecting parametric and algebraic curves ii: Multiple intersections. *CVGIP: Graphical Model and Image Processing*, 57(2):81--100, 1995.

[86] A. Marco and J. J. Martínez. Using polynomial interpolation for implicitizing algebraic curves. *Computer Aided Geometric Design*, 18(4):309--319, 2001.

[87] A. Marco and J. J. Martínez. Implicitization of rational surfaces by means of polynomial interpolation. *CAGD*, 19:327--344, 2002.

[88] V. Y. Pan. Simple multivariate polynomial multiplication. *J. Symb. Comp.*, 18:183--186, 1994.

[89] N. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer-Verlag, New York, 2001.

[90] S. Pérez-Díaz and J. R. Sendra. Partial degree formulae for rational algebraic surfaces. In *ISSAC*, pages 301--308, 2005.

[91] M. Peternell, D. Gruber, and J. Sendra. Conchoid surfaces of spheres. *Computer Aided Geometric Design*, 30(1):35--44, 2013.

[92] M. Peternell and H. Pottmann. Computing rational parametrizations of canal surfaces. *Journal of Symbolic Computation*, 23:255--266, 1997.

[93] J. Peters and U. Reif. *Subdivision Surfaces*, volume 3 of *Geometry and Computing*. Springer-Verlag, New York, 2008.

[94] J. Rambau. TOPCOM: Triangulations of point configurations and oriented matroids. In A. M. Cohen, X.-S. Gao, and N. Takayama, editors, *Intern. Conf. Math. Software*, pages 330--340. World Scientific, 2002.

[95] T. Recio and J. R. Sendra. Real reparametrizations of real curves. *J. Symb. Comput.*, 23(2/3):241--254, 1997.

[96] M. Sanuki and T. Sasaki. Computing approximate gcds in iii-conditioned cases. In *Proc. SNC 2007*, pages 170--179, ACM, Ontario, Canada, 2007.

[97] T. Sasaki and M. Suzuki. Three new algorithms for multivariate polynomial gcd. *J. Symbolic computation*, 13:395--411, 1992.

[98] T. Sauer. Lagrange interpolation on subgrids of tensor product grids. *Math. of Comput.*, 73:181--190, January 2004.

[99] J. Schicho. A Degree Bound for the Parameterization of a Rational Surface. Technical Report 97-24, RISC Report Series, University of Linz, Austria, October 1997.

[100] J. Schicho. Rational parametrization of real algebraic surfaces. In *ISSAC*, pages 302--308, 1998.

[101] T. W. Sederberg and J. Zheng. *Handbook of Computer Aided Geometric Design*, chapter Algebraic Methods for Computer Aided Geometric Design, pages 363--387. Elsevier, 2002.

[102] F. S. Segundo and J. R. Sendra. Degree formulae for offset curves. *J. Pure Appl. Algebra*, 195(3):301--335, Feb. 2005.

[103] J. R. Sendra and F. Winkler. Algorithms for rational real algebraic curves. *Fund. Inform*, pages 211--228, 1999.

[104] J. R. Sendra and F. Winkler. Computation of the degree of rational maps between curves. In *Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, ISSAC '01, pages 317--322, New York, NY, USA, 2001. ACM.

[105] J. R. Sendra and F. Winkler. Tracing index of rational curve parametrizations. *Comput. Aided Geom. Des.*, 18(8):771--795, Oct. 2001.

[106] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph.*, 23(3):896--904, Aug. 2004.

[107] B. Sturmfels. On the Newton polytope of the resultant. *J. Algebraic Combin.*, 3:207--236, 1994.

[108] B. Sturmfels, J. Tevelev, and J. Yu. The Newton polytope of the implicit equation. *Moscow Math. J.*, 7(2), 2007.

[109] B. Sturmfels and J. Yu. Minimal polynomials and sparse resultants. In F. Orecchia and L. Chiantini, editors, *Proc. Zero-dimensional schemes (Ravello, 1992)*, pages 317--324. De Gruyter, 1994.

[110] B. Sturmfels and J. Yu. Tropical implicitization and mixed fiber polytopes. In *Software for Algebraic Geometry*, volume 148 of *IMA Volumes in Math. & its Applic.*, pages 111--131. Springer, New York, 2008.

[111] J. Sylvester. On a Theory of the Syzygetic Relations of Two Rational Integral Functions, Comprising an Application to the Theory of Sturm's Functions, and that of the Greatest Algebraical Common Measure. *Philosophical Transactions of the Royal Society of London*, 143, 1853.

[112] S. Tanabe. On Horn-Kapranov uniformisation of the discriminantal loci. *Adv. Studies Pure Math.*, 46:223--249, 2007.

[113] C. Ünsalan and A. Erçil. Conversions between parametric and implicit forms using polar/spherical coordinate representations. *Computer Vision and Image Understanding*, 81(1):1--25, 2001.

[114] J. van der Hoeven and E. Schost. Multi-point evaluation in higher dimensions. Technical Report 00477658, HAL, 2010.

[115] J. von zur Gathen, M. Mignotte, and I. E. Shparlinski. Approximate polynomial gcd: small degree and small height pertubations. *J . Symbolic Computation*, 45(8):879--886, 2010.

[116] W. N. Waggenspack Jr. and D. C. Anderson. Piecewise parametric approximations for algebraic curves. *Computer Aided Geometric Design*, 6(1):33--53, 1989.

[117] W. Wang. *Handbook of Computer Aided Geometric Design*, chapter Modeling and processing with quadric surfaces., pages 777--795. Elsevier, 2002.

[118] X. Wang, F. Chen, and J. Deng. Implicitization and parametrization of quadratic surfaces with one simple base point. In *Proceedings of the twenty-first international symposium on Symbolic and algebraic computation*, ISSAC '08, pages 31--38, New York, NY, USA, 2008. ACM.

[119] J. Warren. A bound on the implicit degree of polygonal bézier surfaces. In *Mathematics Department, City University of Hong*, pages 513--525, 1990.

[120] E. Wurm, B. Jüttler, and M.-S. Kim. Approximate rational parameterization of implicitly defined surfaces. In *IMA Conference on the Mathematics of Surfaces*, pages 434--447, 2005.

[121] E. Wurm, J. B. Thomassen, B. Jüttler, and T. Dokken. Comparative benchmarking of methods for approximate implicitization. In M. Neamtu and M. Lucian, editors, *Geometric Modeling and Computing, Seattle 2003*, pages 537--548. Nashboro Press, 2004.

[122] Z. Zeng. Apatools: a software toolbox for approximate polynomial algebra. *ACM Comm. Comput. Algebra*, 42:177--179, 2009.

[123] J. Zheng and T. W. Sederberg. A direct approach to computing the μ-basis of planar rational curves. *J. Symbolic Comput*, 31(5):619--629, 2001.

[124] R. Zippel. Interpolating polynomials from their values. 9:375--403, 1990.

[125] R. Zippel. *Effective Polynomial Computation*. Kluwer Academic Publishers, Boston, 1993.