Implicitization of curves and surfaces using predicted support

Ioannis Z. Emiris
University of Athens
Athens, Greece
emiris@di.uoa.gr

Tatjana Kalinka
University of Athens
Athens, Greece
tatjana.kalinka@gmail.com

Christos Konaxis
University of Athens
Athens, Greece
ckonaxis@di.uoa.gr

ABSTRACT

We reduce implicitization of rational parametric curves and (hyper)surfaces to linear algebra, by interpolating the coefficients of the implicit equation. For this, we may use any method for predicting the implicit support. We focus on methods that exploit input structure in the sense of sparse (or toric) elimination theory, namely by computing the Newton polytope of the implicit polynomial. We offer a public-domain implementation of our methods, and study their numerical stability and efficiency on several classes of plane curves and surfaces, and discuss how it can be used for approximate implicitization in the setting of sparse elimination.

Categories and Subject Descriptors

G.1 [**Mathematics of Computing**]: Numerical Analysis; I.1 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation; I.3.5 [**Computing Methodologies**]: Computer Graphics—*Computational Geometry and Object Modeling*

General Terms

Algorithms, Experimentation

Keywords

Implicitization, Sparse elimination, Newton polytope, Numerical linear algebra

1. INTRODUCTION

Implicitization is the problem of changing the representation of parametric objects to implicit (or Cartesian) form. This problem lies at the heart of several questions in computer-aided geometric design (CAGD) and geometric modeling, including intersection problems and membership queries. In several situations, it is important to have both representations available. Implicit representations encompass a larger class of shapes than parametric ones. Moreover, the class of implicit curves and surfaces is closed under certain operations such as offsetting, while its parametric counterpart is not. Implicitization is also of independent interest, since certain questions in areas as diverse as robotics or statistics, e.g. [4], reduce to deriving the implicit form.

Here we follow a classical symbolic-numeric method, which reduces implicitization to interpolating the coefficients of the defining equation. We implement interpolation by (numeric) linear algebra operations, following a symbolic phase of implicit support prediction, i.e. computing a (super)set of the monomials appearing in the implicit equation. Standard methods to interpolate the unknown coefficients by linear algebra, are divided in two main categories, dense and sparse methods. The former require a bound on the total degree of the target polynomial, whereas the latter require only a bound on the number of its terms, thus exploiting any sparseness of the target polynomial. Moreover, its performance depends on the actual number of non-zero terms in this polynomial. In fact, a priori knowledge of the support essentially answers the first task of such a method. For more information see [25].

Our first contribution is to exploit sparse (or toric) variable elimination theory to predict the implicit Newton polytope, i.e. the convex hull of the implicit support.

DEFINITION 1. *Given a polynomial $\sum_j c_{ij} t^{a_{ij}}$, its support is the set $A_i = \{a_{ij} \in \mathbb{N}^n : c_{ij} \neq 0\}$; its Newton polytope is the convex hull of the support.*

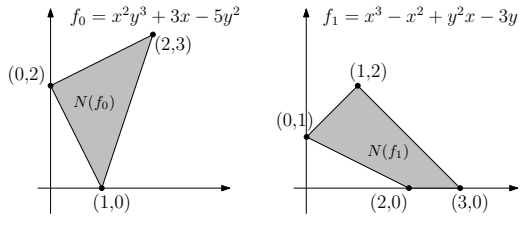


Figure 1: Example of Newton polygons

One reason for revisiting interpolation is the current increase of activity around various approaches capable of predicting the implicit support, e.g. [5, 6, 21, 22]. Although our team has been focussing on sparse elimination [9, 11, 12], the present work can use the implicit support predicted by any method.

SNC 2011, June 7-9, 2011, San Jose, California.

In fact, [22, 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 expect that our software will interface implicit support predictors, such as those developed in [5, 9, 11, 22], with linear algebra. We juxtapose the use of exact and numerical linear algebra, which eventually can rely on state-of-the-art libraries, such as Eigen or LAPACK, respectively.

We discuss approximate implicitization, which is of high practical importance in CAGD [8, 19], in the setting of sparse elimination. In our view, approximate implicitization is one of the main motivations for reducing implicitization to interpolation.

Our second contribution is to offer a public-domain Maple implementation. We study the numerical stability and efficiency of our algorithms on several classes of 2- and 3-d examples. The Maple code has been published on our Wiki webpage that contains support prediction and implicitization related experiments:

http://ergawiki.di.uoa.gr/index.php/Implicitization

One central question is how to evaluate the computed monomials to obtain a suitable matrix, when performing exact or numerical matrix operations. We compare results obtained by using random integers, random complex unitary numbers and complex roots of unity.

A *parametrization* of a geometric object of co-dimension one, in a space of dimension $n+1$, can be described by a set of parametric functions:

$$x_0 = f_0(t_1, \ldots, t_n), \ldots, x_n = f_n(t_1, \ldots, t_n),$$

where $t := (t_1, t_2, \ldots, t_n)$ is the vector of parameters and $f := (f_0, \ldots, f_n)$ is a vector of continuous functions, including polynomial, rational, and trigonometric functions, also called *coordinate functions*. These are defined on some product of intervals $\Omega := \Omega_1 \times \cdots \times \Omega_n,\ \Omega_i \subseteq \mathbb{R}^n$, of values of $t_1, \ldots, t_n$. Implicitization of planar curves and surfaces in 3-dimensional space corresponds to $n = 1$ and $n = 2$ respectively.

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 s.t. $p(f_0(t), \ldots, f_n(t)) = 0$, for all t in Ω. We restrict ourselves to the case when this is a principal ideal, and we wish to compute its defining polynomial

$$p(x_0, \ldots, x_n) = 0, \tag{1}$$

given its Newton polytope or a polytope that contains it. We can regard the variety in question 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

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

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

$$x_i = \frac{f_i(t)}{g_i(t)},\ i = 0, \ldots, n, \tag{2}$$

which can be represented as polynomials

$$F_i := x_i g_i(t) - f_i(t) \in (\mathbb{R}[x_i])[t],\ i = 0, \ldots, n, \tag{3}$$

where we have to take into account that the $g_i(t)$ cannot vanish.

Several algorithms exist for this problem, including methods based on resultants, Gröbner bases, moving surfaces, and residues. Our approach relies on any support prediction method, see sect. 3. We focus on sparse (or toric) elimination: In the case of curves, the implicit support is directly determined for generic parametric expressions with the same supports. In the case of (hyper)surfaces, the implicit support is provided by that of a sparse resultant.

The rest of the paper is structured as follows. Previous work is discussed in the next two sections. Sect. 2 discusses existing methods for interpolating the implicit polynomial by linear algebra. Sect. 3 discusses implicit support prediction and sparse elimination. Our algorithm is detailed in sect. 4, and its use in experiments is analyzed in sect. 5. We conclude with open questions in sect. 6, whereas the Appendix contains further experimental results.

2. EXISTING INTERPOLATION METHODS

This section examines how implicitization had been reduced to a linear algebra question. Let S be (a superset of) the support of the implicit polynomial $p(x_0, \ldots, x_n) = 0$ with unknown coefficients P, $|P| = |S|$. We sometimes refer to S as implicit support, with the understanding that, later, interpolation may set some of the corresponding coefficients to zero.

2.1 Exact implicitization

The most direct method to reduce implicitization to linear algebra is to construct a $|S| \times |S|$ matrix M, indexed by S (columns) and $|S|$ different values (rows) at which all monomials get evaluated. Then, P is in the kernel of M. This idea was used in [12, 17, 22].

In [21], 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 [2], 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 parameterizations, including polynomial, rational, and trigonometric representations. The resulting matrix has Henkel-like structure [16]. When it is computed over floating-point numbers, the resulting implicit polynomial does not necessarily have integer coefficients. This is the situation we face when using numerical methods, such as SVD, see subsec. 4.2. In [2], they discuss some extra processing to yield the integer relations among the coefficients; we also use a similar method.

2.2 Approximate implicitization

In practical applications of CAGD, precise implicitization often can be impossible or very expensive to obtain. Approximate implicitization over floating-point numbers appears to be an effective solution. There are direct [8, 24] and iterative techniques [1].

We describe the basic direct method [8]. Given a parametric (spline) curve or surface $x(t),\ t \in \Omega \subset \mathbb{R}^n$, the goal of [8] 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 $|\eta(t)| \leq \epsilon$. Now,

$$q(x(t)) = (Mp)^T \alpha(t),$$

where matrix M is built from monomials in x. It may be constructed as in our case, or it may contain a subset of the monomials comprising the exact implicit support. Here, p is the vector of implicit coefficients, hence $Mp = 0$ returns the exact solution. Otherwise, $\alpha(t)$ is the basis of the space of polynomials that describes $q(x(t))$, and is assumed to form a partition of unity: $\sum \alpha_i = 1$, and to be nonnegative over Ω:

$$\alpha_i \geq 0, \ \forall i, \forall t \in \Omega.$$

One may use the Bernstein-Bézier basis with respect to the interval Ω, in the case of curves, or a triangle which contains Ω, in the case of surfaces.

In [8, 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 found out that using unitary numbers leads to better numerical stability. Since both our and their methods rely on SVD, our experiments confirm their idea.

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. Existing approaches have used upper bounds on the total implicit degree, thus ignoring any sparseness structure.

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 integral offset of the implicit polytope:

Offset can be repeated, thus producing a list of implicit supports yielding implicit equations whose approximation error should be proportional to the number of peeling steps. Clearly, the computed implicit equation is of lower degree than the actual one. It is an open question to bound the difference of the corresponding zero sets.

3. SUPPORT PREDICTION

Most existing approaches employ total degree bounds on the implicit polynomial to compute a superset of the implicit support, e.g. [2]. This fails to take advantage of the sparseness of the input in order to accelerate computation, and to exploit sparseness in the implicit polynomial in the sense of Prop. 2. For this, computing the Newton polytope of a rational hypersurface was posed in [23] for generic Laurent polynomial parameterizations, in the framework of sparse elimination theory.

Algorithms based on tropical geometry have been offered in [7, 21, 22]. This method computes the abstract tropical variety of a hypersurface parameterized by generic Laurent polynomials in any number of variables, thus yielding its implicit support; it is implemented in `TrIm`. For non-generic parameterizations of rational curves, the implicit polygon is predicted. In higher dimensions, the following holds:

PROPOSITION 1. *[21, prop.5.3] Let $f_0, \ldots, f_n \in \mathbb{C}[t_1^{\pm 1}, \ldots, t_n^{\pm 1}]$ 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 which is constructed combinatorially from $P_0, \ldots, P_n$ as in [21, sec.5.1] contains a translate of the Newton polytope of g.*

The tropical approach was improved in [5, sec.5.4] to yield the precise implicit polytope in $\mathbb{R}^3$ for non-generic parameterizations of surfaces in 3-space.

In [6], they determine the Newton polygon of a curve parameterized by rational functions, without any genericity assumption. In a similar direction, an important connection with combinatorics was described in [13], 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 [12] 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 [18]. This approach was quite expensive and, hence, applicable only to small examples; it is refined and improved in this paper.

In [11], sparse elimination is applied to determine the vertex representation of the implicit Newton polygon of planar curves. The method uses mixed subdivisions of the input Newton polygons and regular triangulations of pointsets defined by the Cayley trick. It can be applied to polynomial and rational parameterizations, 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, this polygon is guaranteed to contain the Newton polygon of the implicit equation. The method can be seen as a special case of the following general approach based on sparse elimination.

3.1 Sparse elimination

Sparse 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.

PROPOSITION 2. *[12, sec.3] Consider polynomial system $F_0, \ldots, F_n \in K[t]$ as in (3), defining a hypersurface, and let A_i be the support of F_i. Then, the total degree of the implicit polynomial 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, $j \in \{0, \ldots, n\}$ is bounded by the mixed volume of the F_i, $i \neq j$, seen as polynomials in t; cf also [23, thm.2(2)].*

The classical results for the dense case follow as corollaries. Take a surface parameterized by polynomials of degree d, then the implicit polynomial is of degree d^2. For tensor parameterizations of bi-degree (d_1, d_2), the implicit degree is $2d_1d_2$.

Let $\{F_0, F_1, \ldots, F_n\}$ be a polynomial system where $F_i \in K[t_1, \ldots, t_n]$, with symbolic coefficients c_{ij}. Its *resultant* $\mathcal{R}$ is a polynomial in $\mathbb{Z}[c_{ij}]$, vanishing iff $F_0 = F_1 = \cdots = F_n = 0$ has a common root in a specific variety. This is the projective variety over the algebraic closure $\overline{K}$ of K, in the case of projective resultants, or the toric variety X defined by the supports of the F_i's in the case of sparse (or toric) resultants, s.t. it contains the topological torus as a dense subset: $(\overline{K}^*)^n \subset X$. 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 $\mathcal{R}$.

The *Minkowski sum* $A+B$ of convex polytopes $A, B \subset \mathbb{R}^n$ is the set $A+B = \{a+b \mid a \in A, b \in B\} \subset \mathbb{R}^n$. Given $n+1$ convex polytopes P_i of dimension n, a *fine* or *tight mixed subdivision* of $P = \sum_{i=0}^{n} P_i$, is a collection of n-dimensional convex polytopes σ, called (Minkowski) *cells*, s.t.: (1) They form a polyhedral complex that partitions P, and (2) Every cell σ is a Minkowski sum of subsets $\sigma_i \subset P_i$: $\sigma = \sigma_0 + \cdots + \sigma_n$, where $\dim(\sigma) = dim(\sigma_0) + \cdots + dim(\sigma_n) = n$.

A cell σ is called *i-mixed*, or *v_i-mixed*, if it is the Minkowski sum of n 1-dimensional segments $E_j \subset P_j$ and one vertex $v_i \in P_i : \sigma = E_0 + \cdots + v_i + \cdots + E_n$. A mixed subdivision is called *regular* if it is obtained as the projection of the lower hull of the Minkowski sum of lifted polytopes $\hat{P}_i := \{(p_i, \omega(p_i)) \mid p_i \in P_i\}$. If the lifting function ω is sufficiently generic, then the induced mixed subdivision is tight. We recall a surjection from the regular fine mixed subdivisions to the vertices of the resultant polytope:

THEOREM 3. *[20] Given a polynomial system and a regular fine mixed subdivision of the Minkowski sum of the Newton polytopes of the supports of the polynomials in the system, 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 $c \in \{-1, +1\}$.

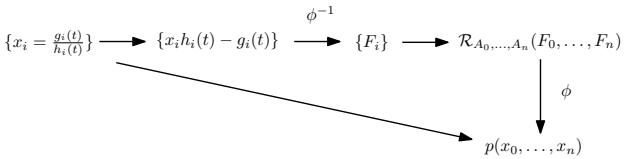
Computing all regular fine mixed subdivisions reduces, due to the so-called Cayley trick, to computing all regular triangulations of a point set of cardinality $|A_0|+\cdots+|A_n|$ in dimension $2n$. This is the Cayley embedding of the A_i's. The set of all regular triangulations corresponds to the vertices of the secondary polytope. We enumerate all regular triangulations of the Cayley embedding: it is equivalent to enumerating all regular fine mixed subdivisions of $A_0+\cdots+A_n$. Each such subdivision yields a vertex of $N(\mathcal{R})$.

We use sparse elimination to implicitize (hyper)surfaces, based on the implementation of [9]. This is an ongoing project that aims at optimizing the computation of $N(\mathcal{R})$ without enumerating all mixed subdivisions. Still, we sketch the theoretical procedure.

Let $A_i \subset \mathbb{Z}^n$, $i = 0, \ldots, n$, be the supports of the polynomials in (3), of cardinality m_i, and P_i, $i = 0, \ldots, n$ the corresponding Newton polytopes. For each A_i, we introduce symbolic coefficients c_{ij}, $j = 1, \ldots, m_i$, and define the polynomial F_i in the c_{ij}'s with the same support. Each c_{ij} corresponds to a (possibly constant) linear polynomial in x_i, with coefficients from f_i. The sparse resultant $\mathcal{R} = \mathcal{R}_{A_0,\ldots,A_n}(F_0, \ldots, F_n)$ of the F_i's with respect to t is well-defined, and is an irreducible integer polynomial in the c_{ij}'s. Consider an epimorphism of rings

$$\phi : K \to \mathbb{C}[x_i] : \; c_{ij} \mapsto \text{ linear polynomial in } x_i, \qquad (4)$$

yielding a specialization of the coefficients c_{ij} to the actual coefficients of (2). The specialized sparse resultant $\mathcal{R}' := \phi(\mathcal{R})$ is a polynomial in x_i, which coincides with the implicit equation, provided that $\mathcal{R}'$ does not vanish, a certain genericity condition is satisfied, and the parametrization is generically 1-1 [23, thm.2]. The following diagram illustrates these concepts:



If the latter condition fails, then $\mathcal{R}'$ is a power of the implicit equation [3], [23, thm.3]. When the genericity condition fails for a specialization of the c_{ij}'s, the support of the specialized resultant is a superset of the support of actual implicit polynomial modulo a translation, provided the sparse resultant does not vanish. This follows from the fact that the method computes the same implicit polytope as the tropical approach, whereas the latter is characterized in prop. 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, [20, sec.5].

4. IMPLICITIZATION ALGORITHM

The steps of our implicitization algorithm are given below, and apply to any support prediction method.
Input: Polynomial or rational parametrization $x_i = f_i(t_1, \ldots, t_n)$.
Output: Implicit polynomial $p(x_i)$ in the monomial basis in $\mathbb{N}^{n+1}$.
1. Support prediction determines (a superset of) the implicit polytope vertices.
2. Compute all lattice points $S \subseteq \mathbb{N}^{n+1}$ in the polytope.
3. Repeat $|S|$ times: Select value τ for t, evaluate $x_i(t)$, $i = 0, \ldots, n$, thus evaluating each monomial in S.
4. Given $|S| \times |S|$ matrix M, solve $M\vec{p} = 0$ for p; return the primitive part of polynomial $\vec{p}^{\top} S$.

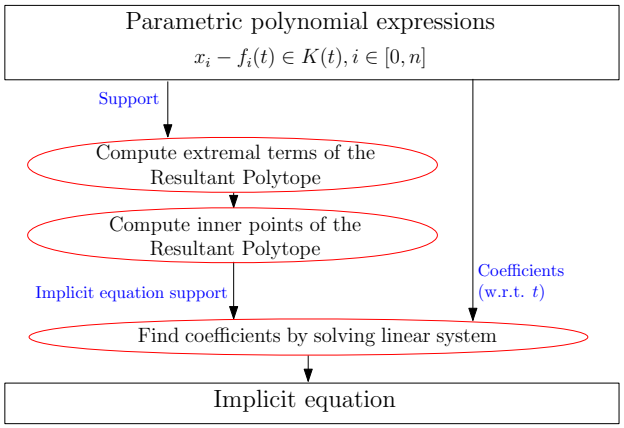


Figure 3: Implicitization with support prediction

4.1 Building the matrix

We focus on two support prediction methods. The first applies only to curves and is described in [sect. 3], [11]. The second is general and computes the support of the resultant of system (3).

In the case of parametric curves, given the predicted polygon, we compute the m lattice points a_i that it contains.

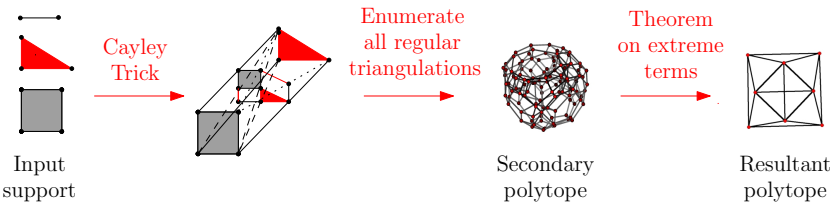


Figure 2: Computing the resultant polytope of 3 Newton polygons.

Each $a_i = (a_{i0}, a_{i1})$ is an exponent of a (potential) monomial of the implicit polynomial. In order to obtain the coefficients we evaluate $x_i(t),\ i = 0, 1$, at some $\tau_k,\ k = 1, \ldots, m$, and construct an $m \times m$ matrix M with rows indexed by $\tau_1, \ldots, \tau_m$ and columns by $a_1, \ldots, a_m$. Let $x(\tau_k)^{a_i}$ denote the evaluated ith monomial $x_0(\tau_k)^{a_{i0}} x_1(\tau_k)^{a_{i1}}$ at τ_k. Then

$$M = \begin{bmatrix} x(\tau_1)^{a_1} & \cdots & x(\tau_1)^{a_m} \\ \vdots & \cdots & \vdots \\ x(\tau_m)^{a_1} & \cdots & x(\tau_m)^{a_m} \end{bmatrix} \begin{matrix} \tau_1 \\ \vdots \\ \tau_m \end{matrix} \tag{5}$$

To apply the general support prediction [9], consider the polynomials $F_i,\ i = 0, \ldots, n$ in Equation (3), 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 $a_i = (a_{i1}, \ldots, a_{id}) \in V$ 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:

$$c := (c_1, \ldots, c_d) = \underbrace{(c_{01}, \ldots, c_{0d_0}}_{\text{from } F_0}, \underbrace{c_{11}, \ldots, c_{id_1}}_{\text{from } F_1}, \ldots\ldots, \underbrace{c_{n1}, \ldots, c_{nd_n})}_{\text{from } F_n}.$$

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 $\phi(S)$, i.e. we identify each $\phi(c)^{a_k}$ 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. $\phi(c)^{a_k} = \phi(c)^{a_l}$. By examination of the $\phi(c)^{a_k}$, we remove duplicates. In the following we assume that $\phi(S)$ has no multiple entries.

Matrix M is constructed similarly as before. Let $\phi(S) = \{a_1, \ldots, a_m\} \subset \mathbb{N}^d$ be the lattice points, which form a superset of the resultant support. Each column contains a product $\phi(c)^{a_k} = \phi(c_1^{a_{k1}} \cdots c_d^{a_{kd}})$ 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:

$$M = \begin{matrix} \\ \\ \end{matrix} \begin{matrix} (a_{11}, \ldots, a_{1d}) & \cdots & (a_{m1}, \ldots, a_{md}) \end{matrix} \\ \begin{bmatrix} \phi(c_1^{a_{11}} \cdots c_d^{a_{1d}})(\tau_1) & \cdots & \phi(c_1^{a_{m1}} \cdots c_d^{a_{md}})(\tau_1) \\ \vdots & \cdots & \vdots \\ \phi(c_1^{a_{11}} \cdots c_d^{a_{1d}})(\tau_m) & \cdots & \phi(c_1^{a_{m1}} \cdots c_d^{a_{md}})(\tau_m) \end{bmatrix} \begin{matrix} \tau_1 \\ \vdots \\ \tau_m \end{matrix} \tag{6}$$

Each $\phi(c)^{a_k} \in \phi(S)$ is a product of linear polynomials in x_i.

It appears to be usual case, that some $\phi(c)^{a_k}$ project into same polynomials in x_i's. This allows us to noticeably decrease size of the matrix M by removing the duplicates and using only $\phi(c)^{a_k}$ that have unique representations in x_i. After expanding and simplifying $\phi(c)^{a_k}$, we get a set of monomials in x_i's. Let m' be the number of integer points in the convex hull they define, thus yielding a superset of the implicit support. We form an $m' \times m'$ matrix M', whose kernel yields the implicit coefficients. The columns of M' are indexed by monomials in the x_i's, hence in $\mathbb{Z}^n$, and the matrix entries are evaluated monomials in the x_i's, while the entries of matrix M are evaluated polynomials in the x_i's.

Often, matrix M' is of larger size than matrix M, see tables 3, 5. This is not a paradox, since ϕ is *not* an orthogonal projection of the c_{ij}'s to a space of lower dimension. Applying ϕ to a monomial in the c_{ij}'s of total degree δ results to a product of linear polynomials in the x_i's, which are developed to yield all monomials of total degree $< \delta$. For example, consider the monomial $c_{01}^3 c_{12}^4$ and the mapping $c_{01} \mapsto x + 1, c_{12} \mapsto y$. Then, we obtain the monomials $x^3y^4, x^2y^4, xy^4, y^4$. If cancellations do not occur among the new monomials, the resulting matrix is larger.

Moreover, this new support may not be the set of vertices of a convex polytope, hence we must perform a convex hull computation and then compute the set of integer points it contains. However, both computations take place in a space of low dimension ($n \ll d$), while computing the integer points inside the polytope in $\mathbb{R}^d$ defined by the set of monomials in the c_{ij}'s can be a hard task.

4.2 Maple Implementation

Assume that matrix M has corank 1, i.e. $\text{rank}(M) = m-1$. Solving the linear system

$$M\vec{p} = \vec{0}, \tag{7}$$

yields the implicit coefficient p_i for each $\phi(c)^{a_k}$. The kernel is one-dimensional, hence some entry p_i is set to 1. We form the inner product of the vector of the monomials indexing the columns of M with $\vec{p}$, and then take the primitive part of the resulting polynomial to define the implicit equation.

When M is evaluated at random integer values or unitary complex numbers, we tried Maple function `Linear-`

`Solve`, from package `LinearAlgebra`, and `Linear` from package `SolveTools`. Equivalently, we can compute the nullspace null(M) of M using the command `NullSpace()` of the `LinearAlgebra` package. All of these functions return the same output in roughly the same runtime with command `LinearSolve` being slightly faster in larger examples. Exact Maple methods can treat indefinites usually encountered in parametric expressions (a, b, c in tables 2,4). For larger examples, we trade exactness for speed and apply Singular Value Decomposition (SVD) with command `SingularValues()`, thus computing

$$M\vec{p}^{\top} = (U\Sigma V^{\top})\vec{p}^{\top} = \vec{0}^{\top} \Leftrightarrow \Sigma\vec{v}^{\top} = \vec{0}^{\top},\ V\vec{v}^{\top} = \vec{p}^{\top}. \quad (8)$$

A basis of null(M) consists of the last columns of V corresponding to the zero singular values of M, because V is orthogonal. When corank(M) $= 1$, the last row of $V^{\top}$ corresponds to $\vec{p}$. The same derivation holds if M is rectangular, say $\mu \times m, \mu \geq m$. Then Σ is of the same dimensions, U is $\mu \times \mu$, and V is $m \times m$, where its last column is the sought vector.

A central part in our linear system construction is held by the evaluation of matrix M at convenient τ. Implementing our methods in Maple, we have tried integer and complex values. In the former case, we used random and mutually prime integers to achieve exactness. The chosen value is discarded if it makes some denominator vanish among the parametric expressions. We also tried complex values for τ: Given an $m \times m$ matrix, we used $2m$-th roots of unity, and random unitary complexes, i.e. with modulus equal to 1.

Table 1 shows representative timings about these options that we examined. In particular, SVD is about 10 times faster than exact linear algebra on significant inputs, as expected; see the last two columns of the table. We plan to use LinBox or Eigen as a faster alternative for our exact algorithm. In what concerns the various evaluation points, our experiments show that runtimes do not vary significantly in small examples but in larger ones the random integers and roots of unity evaluated as floats seem to give faster timings.

However, unitary complexes seem to offer the most stable results, from the numerical viewpoint. Random integers give matrices which are closer to having numerical corank 1, although this property is not reproducible in every experiment. The numerical stability of the result is measured by comparing ratios of singular values of matrix M; we employ the *condition number* $\kappa(M) = \sigma_1/\sigma_m$, and σ_1/σ_{m-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 repeat the computation. Another approach is to consider the row of $V^{\top}$ corresponding to the singular value σ_i of M that satisfies $\sigma_1/\sigma_i \gg \sigma_1/\sigma_{i-1}$, but experiments indicate that this does not improve neither the numerical stability or the correctness of the result. We plan to use LAPACK for further examination of numerical stability, while employing larger examples.

When using numerical methods the computed implicit equation is not a polynomial with integer coefficients. In [2], some post processing is done to discover the integer relations among the coefficients. Then, the polynomial is multiplied by an appropriate number to recover the implicit polynomial with integer coefficients. We utilize a similar method by converting the computed kernel-vector in real or complex space to a rational vector. In practice this is done by assigning a small value to the `Digits` environment variable of Maple, then setting all coefficients smaller than a certain threshold, defined by the problem's condition number, equal to zero. The result is not always correct, so its validity is checked by plugging in the implicit equation the parametric expressions and testing if the result is identically zero. The overall process is computationally hard and it can be avoided whenever an implicit equation with floating point coefficients is sufficient for a specific problem.

Table 1: Runtimes on Maple (seconds)

	SVD			NullSpace
curve	root of 1	unitary $\mathbb{C}$	rand.$\mathbb{Z}$	rand.$\mathbb{Z}$
Cardioid	0.356	0.289	0.076	0.132
Conchoid	0.12	0.092	0.048	0.084
Nephroid	0.012	1.15	0.012	2.3
Talbot's	0.132	0.084	0.108	1.562
Ranunculoid	83.749	-	121.084	8809.43

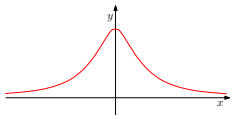
5. EXPERIMENTS

First, we describe some of the examples in details. Instead of x_0, x_1, x_2 we use x, y, z. In several cases, when the input is a parametric family we denote by $a, b, c \in \mathbb{R}^*$ the nonzero parameter. These are set equal to some nonzero constant when using numerical methods. For trigonometric parameterizations, we use the standard conversion to an algebraic form by

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

5.1 Curves

Witch of Agnesi. $x = at, y = a/(1+t^2)$.



The curve support prediction provides 3 implicit polygon's vertices: $(0,0),(2,1),(0,1)$. Their convex hull contains one lattice point. We construct a 4×4 matrix and by solving the corresponding system we find 3 nonzero coefficients; the implicit polynomial is $-a^3 + x^2y + a^2y$.

General support prediction method: The polynomials are: $c_{00} + c_{01}t = 0$, $c_{10} + c_{11}t^2 = 0$, with coefficients $c_{00} = -x, c_{01} = a, c_{10} = a-y, c_{11} = -y$, and 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 $\{c_{01}^2c_{10}, c_{00}^2c_{11}\}$, which are specialized to $\{a^2(a-y),\ (-x)^2(-y)\}$, that index the 2×2 matrix M. We solve $M\vec{p} = 0$ to find $\vec{p} = [p_0, p_1]$, yielding the implicit polynomial

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

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

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

Folium of Descartes.

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

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 $a^4x_1y_1$, ${x_1}^3a^3$, $a^3{y_1}^3$, ${x_1}^2a^2{y_1}^2$, ${x_1}^3{y_1}^3$.

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. Solving linear system gives us 3 nonzero coefficients, and we get implicit equation: $x^3+y^3-3axy=0$.

Nephroid. Rational representation:

$$x=\frac{6(1-t^2)(1+t^2)^2-4(1-t^2)^3}{(1+t^2)^3},\ y=\frac{32t^3}{(1+t^2)^3}$$

The curve support prediction method returns vertices $(0,0)$, $(6,0)$, $(0,6)$. Their convex hull contains 28 lattice points. Solving the linear system we find implicit polynomial $48x^2-12y^4-64-24x^2y^2+y^6-12x^4+3x^2y^4+x^6-60y^2+3x^4y^2$.

Conhoid. Rational representation:

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

The curve support prediction has to be applied with care, because of the special structure of the supports: one has to use the more robust statement of [10]. For the general support prediction, we have: $x-a-1+xt^2+at^2-t^2=0$, $y-2at-2t-2at^3+2t^3-yt^4=0$, with coefficients $c_{00}=x-a-1$, $c_{01}=x-a+1$, $c_{10}=y$, $c_{11}=-2a-2$, $c_{12}=-2a+2$, $c_{13}=-y$, and supports: $\{0,2\},\{0,1,3,4\}$. The predicted support has 6 lattice points: $(0,4,2,0,0,0)$, $(1,3,0,2,0,0)$, $(2,2,0,1,1,0)$, $(2,2,1,0,0,1)$, $(3,1,0,0,2,0)$, $(4,0,0,0,0,2)$, yielding the monomials ${c_{01}}^4{c_{10}}^2$, $c_{00}{c_{01}}^3{c_{11}}^2$, ${c_{00}}^2{c_{01}}^2c_{11}c_{12}$, ${c_{00}}^2{c_{01}}^2c_{10}c_{13}$, ${c_{00}}^3c_{01}{c_{12}}^2$, ${c_{00}}^4{c_{13}}^2$. Solving $M\vec{p}=0$ gives $\vec{p}=[1,1,-2,2,1,1]$, and implicit polynomial $16x^2a^2-16x^2-32x^3a-32y^2xa+16y^2x^2+16y^2a^2+16x^4$.

5.2 Surfaces

Infinite cylinder. The rational parameterization yields the following polynomial system: $1-xt^2-x-t^2$, $-y-yt^2+2t$, $-z+s$, with supports $\{(0,0),(2,0);(0,0),(1,0),(2,0);(0,0),(0,1)\}$. The support prediction gives 4 vertices of the resultant polytope: $(0,2,2,0,0,0,0,0)$, $(1,1,0,2,0,0,0)$, $(1,1,1,0,1,0,0)$, $(2,0,0,0,2,0,0)$.

Specialization ϕ maps the coefficients to polynomials in x,y,z: $\phi(c_{0j})=\{1-x,-1-x\}$, $\phi(c_{1j})=\{-y,2,-y\}$, $\phi(c_{2j})=\{-z,1\}$. Substituting c_{ij} in the predicted monomials by the corresponding univariate polynomials we obtain 9 lattice points: $(0,0,0)$, $(0,2,0)$, $(2,2,0)$, $(2,0,0)$, $(0,1,0)$, $(1,0,0)$, $(1,1,0)$, $(1,2,0)$, $(2,1,0)$. We get the canonical form of the implicit polynomial: $x^2+y^2=a^2$.

Infinite cone. In some cases of rational parametrizations our method results in a multiple of the actual implicit equation. For example, the infinite cone is a quadratic surface with canonical equation $z^2=\frac{x^2+y^2}{a^2}$, but our result has degree 4 (see Table 5). This is due to the fact that we do not compute the minimal variety when there are base points.

Sine surface. Rational representation:

$$x=\frac{2t}{1+t^2},\ y=\frac{2s}{1+s^2},\ z=\frac{2s+2t-2st^2-2ts^2}{1+s^2+t^2+s^2t^2}.$$

Applying support prediction we obtain 1027 lattice points in the resultant polytope. Removing duplicates leaves 87 distinct matrix entries. Solving the linear system, we obtain a kernel vector with 66 nonzero entries. Meanwhile, substituting c_{ij} by the corresponding univariate polynomials leads to a polytope with 125 lattice points; solving the linear system yields 7 nonzero kernel-vector entries. The implicit polynomial is $-2y^2z^2+4x^2y^2z^2-2x^2y^2-2x^2z^2+z^4+y^4+x^4$.

5.3 Results

See the tables in the Appendix for all results concentrated. All experiments where performed on a Celeron 1.60GHz, 1Gb memory, linux machine.

In particular, tables 2,3 contain information on our experiments for curves, including runtimes (given in sec.) on Maple v. 11. In table 3 columns are labeled as: degrees of parametric and implicit equations, number of monomials in the parametric equations, number of lattice points in the resultant polytope when applying the general support prediction method ("Lat.pts."), size of the matrix constructed when applying the curve support prediction [11] (field "Cur.matr."), time in seconds for the prediction routine and linear solving ("Cur.time"), matrix size for using the general support prediction ("Gen.matr."), runtime for the removal of duplicates and linear solving ("Gen.time"), and number of nonzero kernel-vector entries for the matrix based on the c_{ij}'s ("Gen.nonzero"), matrix size for the case when, after using general support prediction, we map the c_{ij} to univariate polynomials ("Map.matr."), runtime for mapping and linear solving ("Map.time"), number of monomials of the implicit curve ("Map.nonzero").

Table 4, 5 contain the information on our experiments for surfaces. In table 5 columns are labeled as: parametric and implicit degree, number of monomials in the parametric equations, number of lattice points in the resultant polytope ("Lat.pts."), matrix size with duplicates removed ("Gen.matr."), runtime for duplicate removal and solving ("Gen.time"), number of nonzero entries of the kernel vector ("Gen.nonzero"), matrix size for mapping c_{ij} to univariate polynomials ("Map.matr."), runtime for mapping and solving ("Map.time"), and number of monomials of the implicit surface ("Map.nonzero")

The tables contain blank entries (-) when the linear system appeared too large to try solving.

As expected (see table 3), the curve support prediction method shows far better results than the general support prediction method applied to curves. Comparing the two matrix constructions, based on the general support prediction, we conclude that, while in some cases mapping gives us a smaller matrix (e.g. Cardioid, Nephroid), in most cases duplicate removal proves to be more effective (e.g. Folium of Descartes, Sine surface). Another observation is that,

right now, a significant fraction of the complexity is due to the general support prediction method, which can readily be improved by current work in our team or by using other prediction methods such as tropical geometry.

6. CURRENT WORK

To improve numerical stability, we may use more evaluation points than monomials, thus forming a rectangular matrix to which we apply SVD, as explained following equation (8). Our current work examines whether some matrix structure can be revealed while exploiting the freedom in choosing appropriate τ values to evaluate the monomials. Our team strives to improve the support calculating algorithm [9] for arbitrary-dimensional hypersurfaces.

We consider specific challenges, such as the bicubic surface [14], of implicit degree 18, containing 1330 terms: the approach of [12] could not handle it because it generates 737129 regular triangulations (by TOPCOM) in a file of 383MB. Also, there is a surface challenge suggested by [15] and meant for a practical problem, with parametrization expressions of total degrees 9,9,6, respectively.

Our approach represents an implicit (hyper)surface by a kernel vector. It is challenging to devise suitable CAGD algorithms that exploit this representation, e.g. for surface-surface intersection, as in [8]. Another practical problem is to implicitize curve or surface splines defined by k segments or patches, respectively. Assuming the k parametric representations yield polynomials with (roughly) the same Newton polytopes, one could use the implicit polytope defined by any of these systems. 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.

It is possible to approximate k (pieces of) manifolds with a single implicit equation, by applying SVD on $[M_1 \cdots M_k]^\top$. We could also extend our approach to interpolating the implicit polynomial in other bases, such as Bernstein or Lagrange, by predicting the resultant support in these bases.

Acknowledgement. IZE and TK are partially supported by Marie-Curie Initial Training Network "SAGA" (ShApes, Geometry, Algebra), FP7-PEOPLE contract PITN-GA-2008-214584.

7. REFERENCES

[1] M. Aigner, A.Poteaux, B. Juttler. Approximate implicitization of space curves. *Symbolic & Numer. Comp*, Springer, 2011. To appear

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

[3] D. A. Cox, J. B. Little, and D. O'Shea. *Using Algebraic Geometry*, vol. 185 of GTM. Springer, 1998.

[4] M. Cueto, E. Tobis, and J. Yu. An implicitization challenge for binary factor analysis. *J. Symbolic Computation*, 45(12):1296–1315, 2010.

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

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

[7] A. Dickenstein, E. M. Feichtner, and B. Sturmfels. Tropical discriminants. J. AMS, 1111–1133, 2007.

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

[9] I. Z. Emiris, V. Fisikopoulos, and C. Konaxis. Regular triangularions and resultant polytopes. In *Proc. Europ. Workshop Comp. Geometry*, pp. 137–140, 2010.

[10] I. Z. Emiris, C. Konaxis, and L. Palios. Computing the Newton polygon of the implicit equation. Tech. Report 0811.0103v1, arXiv, 2007.

[11] I. Z. Emiris, C. Konaxis, and L. Palios. Computing the Newton polygon of the implicit equation. *Math. in Computer Science, Special Issue*, 4(1):25–44, 2010.

[12] I.Z. Emiris and I. S. Kotsireas. Implicit polynomial support optimized for sparseness. In *Proc. Conf. Comput. science Appl.*, pp. 397–406, Springer, 2003

[13] A. Esterov and A. Khovanski. Elimination theory and Newton polytopes. *ArXiv Math.*, Nov. 2006.

[14] L. Gonzalez-Vega. Implicitization of parametric curves and surfaces by using multidimensional Newton formulae. *J. Symbolic Comput.*, 23(2-3):137–151, 1997.

[15] R. Krasauskas. Personal communication, 2011.

[16] I. S. Kotsireas and E. Lau. Implicitization of polynomial curves. In *Proc. ASCM*, pp. 217–226, Beijing, 2003.

[17] A. Marco and J. Martinez. Implicitization of rational surfaces by means of polynomial interpolation. *CAGD*, 19:327–344, 2002.

[18] J. Rambau. TOPCOM: Triangulations of point configurations and oriented matroids. *Intern. Conf. Math. Software*, pp. 330–340. World Scientific, 2002.

[19] M. Shalaby and B. Jüttler. Approximate implicitization of space curves and of surfaces of revolution. *Algebraic Geometry & Geom. Modeling*, pp. 215–228. Springer, 2008.

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

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

[22] B. Sturmfels and J. Yu. Tropical implicitization and mixed fiber polytopes. In *Soft. for Algebraic Geom*, vol. 148 of *IMA* pp. 111–131, Springer, 2008.

[23] B. Sturmfels and J. T. Yu. Minimal polynomials and sparse resultants. *In Proc. Zero-dimensional schemes (Ravello, 1992)*, pp. 317–324. De Gruyter, 1994.

[24] E. Wurm, J. Thomassen, B. Juttler, T. Dokken. Comparative benchmarking of methods for approximate implicitization. In *Geom. Modeling & computing 2003*, pp. 537–548. 2004

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

Appendix

Table 2: Curves

Curve	Parametric form	Implicit polynomial
Cardioid	$a(2\cos(t)-\cos(2t));\ a(2\sin(t)-\sin(2t))$	$-3a^4-6a^2y^2+y^4+8a^3x-6a^2x^2+2x^2y^2+x^4$
Conchoid	$a+\cos(t);\ a\tan(t)+\sin(t)$	$a^2y^2-2axy^2+a^2x^2-x^2+x^2y^2-2ax^3+x^4$
Folium of Descartes	$\frac{3at}{1+t^3};\ \frac{3at^2}{1+t^3}$	$y^3-3xy+x^3$
Nephroid	$a(3\cos(t)-\cos(3t));\ a(3\sin(t)-\sin(3t))$	$-64-60y^2-12y^4+y^6+48x^2-$ $24x^2y^2+3x^2y^4-12x^4+3x^4y^2+x^6$
Ranunculoid	$6\cos(t)-\cos(6t);\ 6\sin(t)-\sin(6t)$	$-52521875-1286250x^2-1286250y^2-$ $-32025(x^2+y^2)^2+93312x^5-933120x^3y^2+466560xy^4-$ $-812(x^2+y^2)^3-21(x^2+y^2)^4-42(x^2+y^2)^5+(x^2+y^2)^6$
Talbot's curve	$\frac{(a^2+c^2\sin^2(t))\cos(t)}{a};$ $\frac{(a^2-2c^2+c^2\sin^2(t))\sin(t)}{b}$	$-16a^4c^8+32a^6c^6-16a^8c^4-8a^6b^2c^2y^2+$ $+32a^4b^2c^4y^2-8a^2b^2c^6y^2-a^4b^4y^4+10a^2b^4c^2y^4-$ $-b^4y^4c^4+b^6y^6+8a^8c^2x^2+8a^6c^4x^2-32a^4c^6x^2+$ $+16a^2c^8x^2-2a^6b^2x^2y^2+2a^4b^2c^2x^2y^2-$ $-20a^2b^2c^4x^2y^2++3a^2b^4x^2y^4-a^8x^4-$ $-8a^6c^2x^4+8a^4c^4x^4+3a^4b^2x^4y^2+a^6x^6$
Tricuspoid	$a(2\cos(t)+\cos(2t));\ a(2\sin(t)-\sin(2t))$	$-27a^4+18a^2y^2+y^4+24axy^2+$ $+18a^2x^2+2x^2y^2-8*ax^3+x^4$
Witch of Agnesi	$at;\ \frac{a}{1+t^2}$	$-a^3+a^2y+x^2y$

Table 3: Curves - results

Curve	Param. degree	Impl. degree	Param. m.nr.	Lat. pts.	Cur. matr.	Cur. time	Gen. matr.	Gen. time	Gen. nonzero	Map. matr.	Map. time	Map. nonzero
Cardioid	4,4	4	3,4	33	15	0.128	33	0.248	33	25	0.656	7
Conchoid	2,3	4	2,4	6	15	0.094	6	0.096	6	15	0.308	6
Folium of Descartes	3,3	3	3,3	10	10	0.092	5	0.032	3	11	0.144	3
Nephroid	4,4	6	4,5	454	28	0.256	426	-	-	49	5.452	10
Ranunculoid	12,12	12	7,12	-	91	8809.43	-	-	-	-	-	-
Talbot's curve	6,6	6	4,7	1600	28	109.342	421	-	-	-	-	23
Tricuspoid	4,4	4	3,4	33	15	0.216	33	0.236	33	25	0.540	8
Witch of Agnesi	1,2	3	2,2	2	4	0.016	2	0.044	2	4	0.048	3

Table 4: Surfaces

Surface	Parametric form	Implicit polynomial
Infinite cylinder	$a\cos(t);\ a\cos(t);\ s$	$-a^2+y^2+x^2$
Hyperbolic paraboloid	$ts;\ t;\ \frac{s^2}{b^2}-\frac{t^2}{a^2}$	$-x^2b^2+y^2a^2-za^2b^2$
Infinite cone	$at\cos(s);\ at\sin(s);\ t$	$x^2z^2+y^2z^2-a^2z^4$
Whitney umbrella	$ats;\ at;\ as^2$	y^2z-ax^2
Monkey saddle	$at;\ as;\ a(t^3-3ts^2)$	$-3xy^2-a^2z+x^3$
Handkerchief surface	$at;\ as;\ a(\frac{t^3}{3}+2(t^2-s^2)+ts^2)$	$-3a^2z+3xy^2+x^3+6x^2a-6ay^2$
Crossed surface	$at;\ as;\ at^2s^2$	$-a^3z+x^2y^2$
Quartoid	$t;\ s;\ -\frac{(t^2+s^2)^2}{a^3}$	$za^3+x^4+2x^2y^2+y^4$
Peano surface	$t;\ s;\ (2t^2-as)(as-t^2)$	$z+2x^4-3x^2ya+y^2a^2$
Bohemian dome	$\cos(t);\ \sin(t)+\cos(s);\ \sin(s)$	$2x^2y^2-2x^2z^2-4y^2+x^4+z^4+2y^2z^2+y^4$
Swallowtail surface	$a(ts^2+3s^4);\ a(-2ts-3s^3);\ at$	$-15axy^2z+3ay^4+y^2z^3-4xz^4+12ax^2z^2-9a^2x^3$
Sine surface	$\sin(t);\ \sin(s);\ \sin(t+s)$	$-2y^2z^2+4x^2y^2z^2-2x^2y^2-2x^2z^2+z^4+y^4+x^4$
Enneper's surface	$t-\frac{t^3}{3}+ts^2;$ $2-\frac{s^3}{3}+t^2s;$ t^2-s^2	$352836-78732x^2y^2z+749412z^2+101088z^3x^2y-303264x^2yz^2-$ $-25272x^2y^2z^3-62127z^5+75816x^2y^2z^2+314928x^2yz-$ $-4860x^4z^3-2916x^6+69984x^2y^3+23328y^4z^2-26244y^4z+$ $+72576yz^5+997272yz-669222y^2z-18144y^2z^5+209952y^3z-$ $-186624y^3z^2+2592x^2z^5-106920x^2z^3+34992x^4+5832x^4z^2+$ $+8748x^4y^2-34992x^4y-183708x^2y^2-268272z^4y-1122660z^2y+$ $+602640z^3y+67068z^4y^2-6912z^6y+653913z^2y^2+1728z^6y^2-$ $-228420z^3y^2+38880z^3y^3-4860z^3y^4-2304z^8-536544y^3+$ $+183708y^4-34992y^5+2916y^6-577368z-5616z^6-8748x^2y^4-$ $-34992x^2+2916x^2z^4+305451x^2z^2+7776z^7-314928x^2z+256z^9-$ $-524151z^3+916353y^2+263898z^4+174960x^2y-866052y-1728x^2z^6$

Table 5: Surfaces - results

Surface	Param. degree	Impl. degree	Param. m.nr.	Lat. pts.	Gen. matr.	Gen. time	Gen. nonzero	Map. matr	Map. time	Map. nonzero
Infinite cylinder	2,2,1	2	2,3,2	4	4	0.036	4	9	0.072	3
Hyperbolic paraboloid	1,1,2	2	2,2,3	3	3	0.028	3	7	0.064	3
Infinite cone	3,2,1	4	4,3,2	14	8	0.040	6	19	0.224	3
Whitney umbrella	2,1,2	3	2,2,2	2	2	0.024	2	4	0.056	2
Monkey saddle	1,1,3	3	2,2,3	3	3	0.028	3	8	0.064	3
Handkerchief surface	1,1,3	3	2,2,5	2	2	0.020	2	4	0.036	5
Crossed surface	1,1,4	4	2,2,2	5	5	0.032	5	10	0.072	2
Quartoid	1,1,4	4	2,2,4	4	4	0.012	4	16	0.140	4
Peano surface	1,1,4	4	2,2,4	4	4	0.020	4	10	0.080	4
Bohemian dome	2,4,2	4	2,6,3	142	58	2.764	-	125	83.362	7
Swallowtail surface	4,3,1	5	3,3,2	12	12	0.052	12	25	0.432	6
Sine surface	2,2,4	6	3,3,8	1027	87	9.244	66	125	107.102	7
Enneper's surface	3,3,2	9	4,3,3	439	258	74.304	258	106	33.562	57