Sparse implicitization by interpolation: Characterizing non-exactness and an application to computing discriminants

Ioannis Z. Emiris [1] Tatjana Kalinka [1] Christos Konaxis [2] Thang Luu Ba [1,3]

Abstract

We revisit implicitization by interpolation in order to examine its properties in the context of sparse elimination theory. Based on the computation of a superset of the implicit support, implicitization is reduced to computing the nullspace of a numeric matrix. The approach is applicable to polynomial and rational parameterizations of curves and (hyper)surfaces of any dimension, including the case of parameterizations with base points. Our support prediction is based on sparse (or toric) resultants, in order to exploit the sparsity of the input and the output. Our method may yield a multiple of the implicit equation: we characterize and quantify this situation by relating the nullspace dimension to the predicted support. In this case, we obtain more than one multiples of the implicit equation; the latter can be obtained via polynomial gcd. All of the above techniques extend to the case of approximate computation, thus yielding a method of sparse approximate implicitization, which is important in tackling larger problems. We discuss our publicly available Maple implementation through several examples. For a novel application, we focus on computing the discriminant of a multivariate polynomial, which characterizes the existence of multiple roots and generalizes the resultant of a polynomial system. This yields an efficient, output-sensitive algorithm for computing the discriminant polynomial.

Key words: geometric representation, implicitization, linear algebra, sparse polynomial, discriminant

1. Introduction

Implicitization is the process of changing the representation of a geometric object from parametric to algebraic, or implicit. It is a fundamental operation with several applications in computer-aided design (CAD) and geometric modeling. There have been numerous approaches for implicitization, including resultants, Groebner bases, and moving lines and surfaces. In this paper, we restrict attention to hypersurfaces: Our approach is based on interpolating the unknown coefficients of the implicit polynomial given a superset of its monomials. The latter is computed by means of sparse (or toric) resultant theory, so as to exploit the input and output sparseness. Here is the main notion that formalizes sparseness (see Fig. 1).

Definition 1 *Given a polynomial* $f = \sum_a c_a t^a \in \mathbb{R}[t_1, \ldots, t_n]$, $t^a = t_1^{a_1} \cdots t_n^{a_n}, a \in \mathbb{N}^n, c_a \in \mathbb{R}$, *its support is the set* $\{a \in \mathbb{N}^n : c_a \neq 0\}$; *its Newton polytope* $N(f)$ *is the convex hull of its support. All concepts extend to the case of Laurent polynomials, i.e. with integer exponent vectors* $a \in \mathbb{Z}^n$.

We call the support and the Newton polytope of the implicit equation, *implicit support* and *implicit polytope*, respectively. Its vertices are called *implicit vertices*. The implicit polytope is computed from the Newton polytope of the sparse (or toric) resultant, or *resultant polytope*, of polynomials defined by the parametric equations. Under certain generically assumptions, the implicit polytope coincides with a projection of the resultant polytope, see Sect. 2. In general, the implicit polytope is contained in the projected resultant polytope, in other words, a superset of the implicit support is given by the lattice points contained in the projected resultant polytope. A superset of the implicit support can also be obtained by other methods, see Sect. 1.1; the rest of our approach does not depend on the method used to compute this support.

The predicted support is used to build a numerical matrix whose kernel is, ideally, 1-dimensional, thus yielding (up to a nonzero scalar multiple) the coefficients corresponding to the predicted implicit support. This is a standard case of *sparse interpolation* of the polynomial from its values. When dealing with hypersurfaces of high dimension, or when the support contains a large number of lattice points, then exact solving is expensive. Since the kernel can be computed numerically, our approach also yields an

Email addresses: `emiris@di.uoa.gr` (Ioannis Z. Emiris), `kalinkat@di.uoa.gr` (Tatjana Kalinka), `ckonaxis@acmac.uoc.gr` (Christos Konaxis), `thanglb@di.uoa.gr` (Thang Luu Ba).
[1] Dept Informatics and Telecoms, University of Athens, Greece
[2] Archimedes Center for Modeling, Analysis and Computation (ACMAC), University of Crete, Heraklio, Greece
[3] Dept Mathematics, Hanoi National Univ. of Education, Vietnam

approximate sparse implicitization method.

Our method of sparse implicitization was sketched in [11], where we presented an algorithm and some preliminary results on its implementation. Its main drawback is that the kernel of the matrix may be of high dimension. In this paper, we address this situation by presenting techniques that alleviate this phenomenon. More formally, we relate it to the geometry of the predicted support, which is a superset of the true implicit support. Another reason for obtaining a high-dimensional kernel is that the numeric evaluation of the support monomials may not be sufficiently generic. We study a method to obtain the true implicit polynomial by taking the greatest common divisor (gcd) of the polynomials corresponding to at least two and at most all of the kernel vectors, or via multivariate polynomial factoring.

Furthermore, we present our publicly available Maple implementation by offering several examples. We also explain how it depends on other software, most notably the software computing the resultant polytope.

Our main motivation is in changing the representation of geometric (hyper)surfaces given parametrically by polynomial, rational, or trigonometric parameterizations. Our method automatically handles the case of base points, so the user does not need to examine whether the given parameterization induces base points or not. Here, we extend our method to a more general geometric problem, namely to computing the discriminant of a multivariate polynomial, which is an important question with several geometric applications. The vanishing of the discriminant characterizes the existence of multiple roots in the given polynomial. This can be hard since explicit formulas only exist for low-degree univariate polynomials. In general, one can reduce discriminant computation to computing the resultant of a system comprised of the polynomial and its partial derivatives, but this is inefficient. Instead, we reduce discriminant computation to sparse implicitization, thus obtaining an output-sensitive algorithm, whose complexity depends on the size of the discriminant's Newton polytope. Moreover, this technique can be used to compute discriminants of well-constrained systems we well as resultants because they can be viewed as a special case of discriminants.

The paper is organized as follows: Sect. 1.1 overviews previous work and Sect. 2 describes our approach to predicting the implicit support while exploiting sparseness. Sect. 3 presents our implicitization algorithm based on computing a matrix kernel, either exactly or approximately, and focuses on the case of high dimensional kernels. Our Maple implementation is described in Sect. 4, whereas Sect. 5 applies our method to computing discriminants. We conclude with future work. Appendix A contains omitted results from examples in Sect. 5, while further experimental results are in Appendix B.

1.1. *Previous work*

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 $\boldsymbol{p}$ of coefficients of the implicit equation is in the kernel of M. This idea was used in [11,13,19,23]; it is also the starting point of this paper.

Our method of sparse implicitization was sketched in [11], where the overall algorithm was presented together with some results on its preliminary implementation, including the case of approximate sparse implicitization. The emphasis of that work was on sampling and oversampling the parametric object so as to create a numerically stable matrix, and examined evaluating the monomials on random integers, random complex numbers of modulus 1, and complex roots of unity. That paper also proposed ways to obtain a smaller implicit polytope by downscaling the original polytope when the corresponding kernel dimension was higher than one.

A similar approach was based on integrating matrix $M = SS^\top$, over each parameter $t_1, \dots, t_n$ [3]. Then $\boldsymbol{p}$ 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. They discuss post-processing to yield integer relations among the coefficients, but only in small examples.

Approximate implicitization over floating-point numbers was introduced in a series of papers. Today, there are direct [7,25] and iterative techniques [1]. An idea used in approximate implicitization is to use successively larger supports, starting with a quite small set and extending it so as to reach the exact implicit support. Existing approaches have used upper bounds on the total implicit degree, thus ignoring any sparseness structure. Our methods provide a formal manner to examine different supports, in addition to exploiting sparseness, based on the implicit polytope. When the kernel dimension is higher than one, one may downscale the polytope so as to obtain a smaller implicit support.

Sparse interpolation is the problem of interpolating a multivariate polynomial when information of its support is given [27, ch.14]. This may simply be a bound $\sigma = |S|$ on support cardinality; then complexity is $O(m^3 \delta n \log n + \sigma^3)$, where δ bounds the output degree per variable, m is the actual support cardinality, and n the number of variables. A probabilistic approach in $O(m^2 \delta n)$ requires as input only δ.

2. Implicitization by support prediction

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

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

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 three dimensional space corresponds to $n = 1$ and $n = 2$ respectively. We assume that, in the case of trigonometric functions, they may be converted to rational functions 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 $t = \tan\theta/2$. On parameterizations depending on both θ and its trigonometric function, we may approximate the latter by a constant number of terms in their series expansion.

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 $p(x_0, \ldots, x_n)$ such that $p(f_0(t), \ldots, f_n(t)) = 0$, for all t in Ω. We restrict ourselves to the case when this is a principal ideal, and we wish to compute its unique defining polynomial

$$p(x_0, \ldots, x_n) = 0, \qquad (1)$$

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

$$x_i = f_i(t)/g_i(t),\ i = 0, \ldots, n, \qquad (2)$$

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

$$\begin{aligned} F_i &:= x_i g_i(t) - f_i(t),\ i = 0, \ldots, n, &(3)\\ F_{n+1} &:= 1 - y g_0(t) \cdots g_n(t), \end{aligned}$$

where y is a new variable and F_{i+1} assures that all $g_i(t) \neq 0$. If one omits F_{n+1}, the generator of the corresponding (principal) ideal would be a multiple of the implicit equation. Then the extraneous factor corresponds to the g_i. Eliminating t, y may be done by taking the *resultant* of the polynomials in (3).

Let $A_i \subset \mathbb{Z}^n$, $i = 0, \ldots, n+1$ be the supports of the polynomials F_i and consider the generic polynomials

$$F'_0, \ldots, F'_n, F'_{n+1} \qquad (4)$$

with the same supports A_i and symbolic coefficients c_{ij}.

Definition 2 *Their sparse resultant $Res(F'_0, \ldots, F'_{n+1})$ is a polynomial in the c_{ij} with integer coefficients, namely*

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

which is unique up to sign and vanishes if and only if the system $F'_0 = F'_1 = \cdots = F'_{n+1} = 0$ has a common root in a specific variety. This variety is the projective variety $\mathbb{P}^n$ over the algebraic closure of the coefficient field in the case of projective (or classical) resultants, or the toric variety defined by the A_i's.

The resultant polytope is denoted by $N(\mathcal{R})$.

The implicit equation of the parametric hypersurface defined in (3) equals the resultant $\mathrm{Res}(F_0, \ldots, F_{n+1})$, provided that the latter does not vanish identically. Thus, the latter can be obtained from $\mathrm{Res}(F'_0, \ldots, F'_{n+1})$ by specializing the symbolic coefficients of the F'_i's to the actual coefficients of the F_i's, provided that this specialization is generic enough. In this case, the implicit polytope equals the resultant polytope projected to the space of the implicit variables, i.e. the Newton polytope of the specialized resultant, up to some translation. When this condition fails for the given specialization of the c_{ij}'s, the support of the specialized resultant is a superset of the support of the actual implicit polynomial modulo a translation. This follows from the fact that the method computes the same resultant polytope as the tropical approach, where the latter is specified in [22]. Note that there is no exception even in the presence of base points.

Proposition 3 [22, 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 I of algebraic relations is principal, say $I = \langle p \rangle$, and let $P_i \subset \mathbb{R}^n$ be the Newton polytope of f_i. Then the resultant polytope which is constructed combinatorially from $P_0, \ldots, P_n$ contains a translate of the Newton polytope of p.*

2.1. *Support prediction - The software* `ResPol`

Our method is based on the computation of the implicit polytope, given the Newton polytopes of the polynomials in (3). Then the implicit support is a subset of the set of lattice points contained in the computed implicit polytope.

There are methods for the computation of the implicit polytope based on tropical geometry [22,23], see also [5]. Our method relies on sparse elimination theory. In the case of curves, the implicit support is directly determined in [12]. In general, the implicit polytope is obtained from the projection of the resultant polytope of the polynomials in (4) defined by the specialization of their symbolic coefficients to those of the polynomials in (3).

In [9], they develop an incremental algorithm to compute the resultant polytope, or its orthogonal projection along a given direction. It is implemented in package `ResPol` [4].

The algorithm exactly computes vertex- and halfspace-representations of the target polytope and it is output-sensitive. It also computes a triangulation of the polytope, which may be useful in enumerating the lattice points. It is efficient for inputs relevant to implicitization: it computes the polytope of surface equations within 1 second, assuming there are less than 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.

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

ResPol takes as input three lines:

– The dimension n of the input supports (in our case, this equals the number of parametric variables).
– The cardinality of each support | support points defining the projection (in our case, these are the exponents of monomials in t having coefficient x_i).
– The supports of the polynomials defined by the parametric expressions.

Example 4 Consider the standard benchmark of bicubic surface, and define the following in $(\mathbb{R}[x_i])[t_1, t_2]$:

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

and prepare the input file for ResPol:

2
7 6 14
$[[0,0], [0,1], [1,0], [0,2], [2,0], [0,3], [3,0], [0,0], [0,1], [1,0], [2,0], [0,3], [3,0], [0,0], [0,1], [1,0], [0,2], [1,1], [2,0], [1,2], [2,1], [1,3], [2,2], [3,1], [2,3], [3,2], [3,3]]$

Alternatively, in 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. These 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 second 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 5 Consider the rational parametric curve known as folium of Descartes:

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

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

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

ResPol outputs seven 4-dimensional vertices: $(0,0,2,1)$, $(3,0,0,3)$, $(0,3,3,0)$, $(1,2,0,0)$, $(1,0,0,1)$, $(0,2,2,0)$, $(0,0,2,1)$. The first two 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 is demonstrated below.

Example 6 (Cont'd from Example 5) We introduce a new variable w expressing the common denominator t^3+1 and rewrite the system:

$$F_0 := -x_0w + 3t^2,\ F_1 := -x_1w + 3t,\ F_2 := 1 - w + t^3.$$

The Newton polygons of the F_i's are shown in Fig. 1.

ResPol gives implicit vertices $(0,3), (3,0), (3,3), (1,1)$ in (x_0, x_1)-space which are directly used in our implicitization routine.

3. Kernel of Higher Dimension

This section describes our implicitization algorithm 1, then focuses on the case of high-dimensional kernels.

Algorithm 1: Sparse Implicitization

Input : Polynomial or rational parameterization $x_i = f_i(t)$, $i = 0, \ldots, n$,
Predicted implicit polytope Q, if $n \geq 2$
Output: Implicit polynomial $p(x_0, \ldots, x_n)$ in its monomial basis.

```
ℕ^{n+1} ⊇ S ← lattice points in Q
foreach s_i ∈ S do m_i ← x^{s_i}          // x := (x_0, ..., x_n)
m ← (m_1, ..., m_|S|)       // vector of monomials in x
Initialize μ × |S| matrix M, μ ≥ |S|:
for i ← 1 to μ do
    select τ_i ∈ ℂ^{n+1}
    for j ← 1 to |S| do
        M_ij ← m_j|_{t=τ_i}
{v_1, ..., v_k} ← Basis of Nullspace(M)
if k = 1 then p ← g_1
else
    for i ← 1 to k do g_i ← primpart(v_i·m)// inn.prod.
    p ← gcd(g_1, ..., g_k)
return p
```

Let us describe in more detail the construction of matrix M. Let $S := \{s_1, \ldots, s_{|S|}\}$; each $s_j = (s_{j0}, \ldots, s_{jn})$ is an exponent of a (potential) monomial $m_j := x^{s_j} = x_0^{s_{j0}} \ldots x_n^{s_{jn}}$ of the implicit polynomial, where x_i is given in (2) . We evaluate m_j at some τ_k, $k = 1, \ldots, \mu$, $\mu \geq |S|$. 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 $\mu \times m$ 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 the construction of matrix M using values τ that correspond to points on the parametric surface, we have the following:

Lemma 7 *Any polynomial in the basis of monomials indexing M, with coefficient vector in the kernel of M, is a multiple of the implicit polynomial p.*

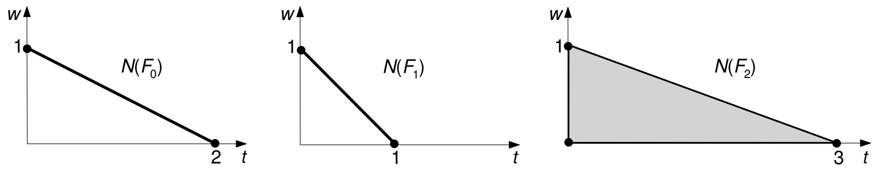

Fig. 1. Newton polytopes of F_0, F_1, F_2 in Example 6.

As in [11], one of the main difficulties is to build M whose corank, or kernel dimension, equals 1, i.e. its rank is 1 less than its column dimension. Of course we have to avoid values that make the denominators of the parametric expressions close to 0. 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. 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. 10 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 two or more polynomials g_i obtained from kernel vectors. There exist many algorithms for the exact [20,21] or approximate gcd of multivariate polynomials. The first approximate approach, given polynomials f, g and error tolerance $\epsilon > 0$, computes the maximum degree gcd of polynomials $\hat{f}, \hat{g}$ where $|f - \hat{f}|, |g - \hat{g}| < \epsilon$ [10]. The second minimizes ϵ such that $\hat{f}, \hat{g}$ have gcd of at least a given degree r [17]. There exist similar techniques for several univariate polynomials [8]. Our software uses Maple's command `gcd` for exact, and package ApaTools [26] for approximate gcd computations.

Example 8 (Cont'd from Example 5) The method in [12] yields the implicit vertices: $(1, 1)$, $(0, 3)$, $(3, 0)$. This polygon contains five 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 parameterization, substituting t by t^2, we obtain

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

then the algorithm in [12] predicts an implicit polytope with vertices: $(2, 2), (0, 6), (6, 0)$, containing twelve 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: $g_1 = x^2y(y^3 - 3yx + x^3)$, $g_2 = (y^3 - 3yx + x^3)(x^3 + 3yx - y^3)$, $g_3 = xy^2(y^3 - 3yx + x^3)$, $g_4 = yx(y^3 - 3yx + x^3)$, $g_5 = y^3(y^3 - 3yx + x^3)$. Their gcd is the implicit equation.

Example 9 (Unit Sphere) Consider its parameterization:

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

`ResPol` predicts an implicit polytope with vertices: $(0, 0, 0)$, $(0, 0, 2)$, $(0, 0, 4)$, $(0, 2, 0)$, $(0, 4, 0)$, $(4, 0, 0)$. 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: $g_1 = y^2(-1 + z^2 + x^2 + y^2)$, $g_2 = z^2(-1 + z^2 + x^2 + y^2)$, $g_3 = -1 + z^2 + x^2 + y^2$, $g_4 = x(-1 + z^2 + x^2 + y^2)$, $g_5 = yz(-1 + z^2 + x^2 + y^2)$, $g_6 = y(-1 + z^2 + x^2 + y^2)$, $g_7 = xz(-1 + z^2 + x^2 + y^2)$, $g_8 = z(-1 + z^2 + x^2 + y^2)$, $g_9 = xy(-1 + z^2 + x^2 + y^2)$, $g_{10} = (x^2 + 1 - y^2 - z^2)(-1 + z^2 + x^2 + y^2)$. Computing the gcd of two randomly chosen polynomials we obtain either the actual implicit equation $p = -1 + z^2 + x^2 + y^2$, or a multiple of p of degree 3.

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 seven decimal digits.

The following theorem establishes the relation between the dimension of the kernel 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 P is expected to be much smaller of Q.

Theorem 10 *Let $P = N(p)$ be the Newton polytope of the implicit equation, and Q the predicted polytope. Assuming M has been built using sufficiently generic evaluation points, the dimension of its kernel 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. 7, the kernel of M consists of the coefficient vectors $\boldsymbol{c}$ 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 $\boldsymbol{c_i}$ of g_i lies in the kernel of M because g_i vanishes on all evaluation points $m_i(\tau_i),\ i = 1, \ldots, k$ used for constructing M, since p vanishes on these points. Moreover, the vectors $\boldsymbol{c_i}$ in the set $\{\boldsymbol{c_1}, \ldots, \boldsymbol{c_r}\}$ are linearly independent. Obviously, every coefficient vector $\boldsymbol{c}$ of a polynomial of the form fp, where $N(fp) \subset Q$, can be written as a linear combination of the vectors $\boldsymbol{c_i}$, hence $\mathrm{corank}(M) = r$.

Let the P, Q be as in Thm. 10 and assume $Q \supsetneq P + R$, 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 11 *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 $pf \in \mathbb{R}[x_0, \ldots, x_n]$, 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 $x^{m_i}p, i = 1, \ldots, r$, 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, $\mathrm{corank}(M) = \dim_{\mathbb{R}}(I)$.

Corollary 12 *Let M be the matrix from Alg. 1, built with sufficiently generic evaluation points, and suppose the specialization of the polynomials in (4) to the parametric equations is sufficiently generic. Let $v_1, \ldots, v_k$ be a basis of the kernel of M and $g_1, \ldots, g_k$ be the corresponding polynomials (Step 4 of Alg. 1). Then the gcd of $g_1, \ldots, g_k$ equals the implicit equation.*

Some examples where M is of corank > 1 are shown in the Appendix; Table B.1, contains parametric and implicit representations. Table B.2 shows: the vertices of the actual implicit polytope, the number of its lattice points, the degree and the number of monomials in the implicit equation, the vertices of the predicted implicit polytope, the number of its lattice points, 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.

4. Maple implementation

We have implemented our method in `Maple 13`. A beta-version is publicly available. [5] Our release's main functions are `imcurve` and `imgen`. Both functions operate similarly: first they construct a square or rectangular M by evaluating the implicit monomials to random integers, random complex numbers of modulus one, or complex roots of unity evaluated as floating point numbers. To compute the nullspace of M we use Maple's commands `LinearSolve` and `Nullspace`; approximate results are obtained by numerical methods, in particular SVD, using `SingularValues`. The user can choose the method of solving as well as the way of evaluating the potential monomials. To compute all lattice points contained in the predicted implicit polytope Q, we rely on the external Maple package `convex` [6]. More specialized software for this task, e.g. `Normaliz` [7], may improve the performance.

Function `imcurve` concerns planar curves only and computes the implicit polygon following [12].Function `imgen` is more general since it can compute the implicit equation of parametric curves, surfaces or hypersurfaces in 4-dimensional space. It is not self-contained as it reads the implicit polytope from an external method, such as `ResPol`. These functions take as arguments:
– The list of parametric expressions
– (`imgen` only) The set of the predicted implicit vertices,
– The solving method parameter: "n" stands for `Nullspace`, "l" `LinearSolve`, and "s" for `SingularValues`.
– The evaluation parameter: "int" stands for integers, "unc" for random complex numbers of modulus 1, and "ruf" for roots of unity evaluated as floating point numbers. Note that the latter can only be used with SVD.
– The ratio between number of rows and columns of the matrix, which is at least 1.

Compared to the preliminary release in [11], our software has many improvements, among which are:
– Improved handling of cases when $\mathrm{corank}(M) > 1$: rectangular matrices are allowed and gcd of two randomly chosen polynomials (corresponding to kernel vectors) is employed.
– New function `writeRespolInput` for creating input files for `ResPol`.
– New functions for generating complex τ's.

In the sequel all experiments, unless otherwise stated, were performed on a Celeron 1.6 GHz linux machine with 2 GB of memory.

Example 13 We demonstrate the use of our two implicitization functions with the curve of Example 5. Let $f_1 := 3t^2/(t^3+1)$ and $f_2 := 3t/(t^3+1)$ and call function `imcurve` as `imcurve`$([f_1, f_2]$, "l", "int", 1). In 0.012 seconds we obtain the implicit equation $y^3 - 3xy + x^3$.

The same curve can be implicitized using function `imgen`: `imgen`$([f_1, f_2], \{[1, 1], [0, 3], [3, 0]\}$,"l","int",1) which yields the same implicit equation in 0.044 seconds.

Example 14 Consider the polynomial parametric surface

$$x_0 = \frac{1}{2}t^2 - \frac{1}{2}s^2 - \frac{1}{4}t^4 + \frac{3}{2}t^2s^2 - \frac{1}{4}s^4,$$
$$x_1 = -ts - t^3s + ts^3,$$
$$x_2 = \frac{2}{3}t^3 - 2ts^2.$$

We define the polynomials $f_1 := 1/2t^2 - 1/2s^2 - 1/4t^4 + 3/2t^2s^2 - 1/4s^4$, $f_2 := -ts - t^3s + ts^3$, and $f_3 := 2/3t^3 - 2ts^2$. `ResPol` predicts implicit vertices $(3, 2, 2)$, $(9, 0, 4)$, $(0, 12, 0)$,$(0, 0, 16)$,$(4, 4, 0)$,$(0, 0, 6)$,$(8, 4, 0)$,$(0, 8, 0)$, $(3, 0, 4)$, $(0, 2, 4)$,$(3, 2, 2)$. This polytope contains 400 lattice points.

Let S denote the set of predicted implicit vertices. Issuing the following command in Maple `imgen`$([f_1, f_2, f_3], S$, "l", "int", 1), we obtain the implicit equation of the surface in 9.4 seconds.

Example 15 (Cont'd from Example 4) The implicit equation of the bicubic surface is computed in 42 seconds; it is a polynomial of degree 18 containing 715 terms which correspond exactly to the predicted implicit support.

5. Discriminant computation

This section computes the discriminant of a multivariate polynomial, which characterizes the existence of multiple roots. It subsumes the discriminant of a well-constrained $n \times n$ system as well as 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. The vanishing of the discrimi-

[5] http://ergawiki.di.uoa.gr/index.php/Implicitization
[6] http://www.math.uwo.ca/∼mfranz/convex
[7] http://www.mathematik.uni-osnabrueck.de/normaliz/

nant partitions coefficient space to cells of values for which the underlying polynomial has a fixed number of real roots. For mechanical, robotics, molecular or vision systems expressed by polynomials, the discriminant variety partitions configuration space to instances that are connected by continuous movement without singularities, e.g. [15].

It is well known 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:

$$D(c_0+c_1t+c_2t^2+c_3t^3) = c_1^2c_2^2-4c_1^3c_3-4c_0c_2^3-27c_0^2c_3^2+18c_0c_1c_2c_3.$$

More generally, consider a polynomial $f(t_1,\dots,t_n)$ in n variables.

Definition 16 *A multiple root of f is a point where f vanishes together with all its first derivatives $\partial f/\partial t_i$. The* discriminant *$D(f)$ is a polynomial in the coefficients of f, which vanishes whenever f has 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 (Laurent) 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. It is shown in [16] that there exists 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, called the *A-discriminant*, 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 PDE's. 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, such as the resultant and the determinant. If, for instance, $A = \{(0,0),(1,0),\dots,(m,0),(0,1),(1,1),\dots,(n,1)\} \subset \mathbb{Z}^2$, 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 polynomials $f_0,\dots,f_k$ 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 $x_iy_j, i,j = 1,\dots,n$, i.e. a bilinear form $F_A = \sum c_{ij}x_iy_j$. Then its A-discriminant is the determinant of the matrix (c_{ij}). Moreover, D_A is a factor of the resultant of F_A and $\partial F_A/\partial t_i, i = 1,\dots,n$. The extraneous factors in this resultant are powers of discriminants associated to certain subsets of A.

Computing A-discriminants may be reduced to implicitization. Given the set of m points $A \subset \mathbb{Z}^n$, 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 $B = (b_{ij}) \in \mathbb{Z}^{m\times(m-n-1)}$ 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 entries of each column vector of B add up to 0.

Set $d = m-n-1$. The, so called, Horn-Kapranov parameterization [16,18], is defined as:

$$x_j = \prod_{i=1}^m (b_{i1}y_1 + \cdots + b_{id}y_d)^{b_{ij}},\ j = 1,2,\dots,d, \tag{7}$$

where $y_i, i = 1,\dots,d$ are homogeneous parameters. In the examples, we shall set $y_1 = 1$ in order to dehomogenize the parameterization. We denote by $l_i, i = 1,\dots,m$ the inner product of the i-th row of B and the parameter vector $(1,y_2,\dots,y_d)$, hence

$$x_j = \prod_{i=1}^m l_i^{b_{ij}},\ j = 1,2,\dots,d. \tag{8}$$

The l_i correspond bijectively to the coefficients c_i of polynomial F_A and are thus the discriminant variables.

The implicit equation of the image of parameterization (8) is a polynomial Δ_B in $x := (x_1,\dots,x_d)$ which in fact is the dehomogenized version of the A-discriminant $D_A(c)$ of F_A. In particular, Δ_B and D_A have the same number of monomials and the same coefficients.

To obtain $D_A(c)$ (up to a monomial) from $\Delta_B(x)$ we use relation (8) and substitute each x_i in Δ_B by the corresponding power product of linear forms $l_i(\equiv c_i)$:

$$D_A(c) = \Delta_B\Big(\prod_{i=1}^m c_i^{b_{i1}},\dots,\prod_{i=1}^m c_i^{b_{id}}\Big).$$

This reduces the computation of D_A to implicitizing the parametric hypersurface (7). Thanks to our support prediction approach, the complexity of our method 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 $d = 2$ or $d = 3$, i.e. $m = n+3$ or $m = n+4$ [2,4,6], although our algorithm may compute discriminants for any d. In particular, we implicitize the parametric curve and surface given, after dehomogenization, 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 $l_i, i = 1,\dots,m$ 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 $l_i := b_{i1} + b_{i2}s + b_{i3}t$.

Example 17 Let $A = \{(1,0),(0,1),(1,1),(2,0),(3,0)\} \subset \mathbb{Z}^2$, and consider the generic polynomial in t_1, t_2 with this support $F_A(t_1,t_2) = c_1t_1 + c_2t_2 + c_3t_1t_2 + c_4t_1^2 + c_5t_1^3$. Then

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} -1 & -1 \\ 1 & 2 \\ -1 & -2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Here $l_1 = -1-s, l_2 = 1+2s, l_3 = -1-2s, l_4 = 1, l_5 = s$. We have the parameterization

$$x_1 = \frac{l_2l_4}{l_1l_3} = \frac{1+2s}{(-1-2s)(-1-s)}, \quad x_2 = \frac{l_2^2l_5}{l_1l_3^2} = \frac{(1+2s)^2s}{(-1-s)(-1-2s)^2}.$$

The predicted implicit polygon has vertices $(0,0), (2,0), (3,0), (3,2)$ and contains seven lattice points. Applying `imcurve`, we obtain the implicit equation $x_1^2(x_1 - x_2 - 1)$, so $\Delta_B(x_1,x_2) = x_1 - x_2 - 1$ because x_1^2 is always nonzero. [8] Then

$$D_A(c_1,c_2,c_3,c_4,c_5) = \Delta_B\left(\frac{c_2c_4}{c_1c_3}, \frac{c_2^2c_5}{c_1c_3^2}\right),$$

so the A-discriminant is $D_A = c_2c_3c_4 - c_2^2c_5 - c_1c_3^2$.

Example 18 Let $A = \{(1,1,0),(1,0,1),(0,1,1),(2,0,0),(0,3,0),(0,0,3)\} \subset \mathbb{Z}^3$, and $F_A(t_1,t_2) = c_1t_1t_2 + c_2t_1t_3 + c_3t_2t_3 + c_4t_1^2 + c_5t_2^3 + c_6t_3^3$. Then

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 & 3 & 0 \\ 0 & 1 & 1 & 0 & 0 & 3 \end{pmatrix}, \quad B = \begin{pmatrix} 3 & -1 \\ -3 & -1 \\ 0 & 1 \\ 0 & 1 \\ -1 & 0 \\ 1 & 0 \end{pmatrix}.$$

Here $l_1 = 3-s, l_2 = -3-s, l_3 = s, l_4 = s, l_5 = -1, l_6 = 1$, and we have the parameterization

$$x_1 = \frac{l_1^3l_6}{l_2^3l_5} = \frac{(3-s)^3}{(3+s)^3}, \quad x_2 = \frac{l_3l_4}{l_1l_2} = \frac{s^2}{(3-s)(3+s)}.$$

The predicted polygon contains twelve lattice points and yields a matrix M of corank 1. The implicit equation is

$$\Delta_B(x_1,x_2) = 1 - 2x_1 - 36x_1x_2 - 96x_1x_2^2 - 64x_1x_2^3 + x_1^2$$

and the A-discriminant is $D_A = \Delta_B(\frac{c_1^3c_6}{c_2^3c_5}, \frac{c_3c_4}{c_1c_2}) = c_2^6c_5^2 - 2c_1^3c_6c_2^3c_5 - 36c_1^2c_6c_3c_4c_2^2c_5 - 96c_1c_6c_3^2c_4^2c_2c_5 - 64c_6c_5c_3^3c_4^3 + c_1^6c_6^2$.

Using approximate computation, namely complex evaluation points and applying SVD for computing the kernel, we obtain:

$1 - 2x_1 - 36.0001x_1x_2 - 96.0001x_1x_2^2 - 64x_1x_2^3 + x_1^2 + 1.3921 \cdot 10^{-21}I + (-2.1482 \cdot 10^{-16} + 3.2297 \cdot 10^{-15}I)x_2 + (2.3068 \cdot 10^{-16} - 2.8561 \cdot 10^{-15}I)x_2^2 + (-4.8344 \cdot 10^{-17} + 1.9862 \cdot 10^{-15}I)x_2^3 - 5.3777 \cdot 10^{-19}Ix_1 - 6.4659 \cdot 10^{-15}Ix_1x_2 + -1.1053 \cdot 10^{-13}Ix_1x_2^2 - 2.1119 \cdot 10^{-13}Ix_1x_2^3 + 1.6829 \cdot 10^{-19}Ix_1^2 + (-2.1857 \cdot 10^{-16} + 3.2281 \cdot 10^{-15}I)x_1^2x_2 + (2.0665 \cdot 10^{-16} - 2.8528 \cdot 10^{-15}I)x_1^2x_2^2 + (-1.7033 \cdot 10^{-16} + 1.8923 \cdot 10^{-15}I)x_1^2x_2^3$.

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

$$1 - 2x_1 - 36.0001x_1x_2 - 96.0001x_1x_2^2 - 64x_1x_2^3 + x_1^2$$

and the approximate A-discriminant: $c_2^6c_5^2 - 2c_1^3c_6c_2^3c_5 - 36.0001c_1^2c_6c_3c_4c_2^2c_5 - 96.0001c_1c_6c_3^2c_4^2c_2c_5 - 64c_6c_5c_3^3c_4^3 + c_1^6c_6^2$, which has the correct support and whose coefficients are accurate up to three decimal digits.

Example 19 [4] Consider the discriminant computation with matrix

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

It gives the parameterization

$$x_1 = \frac{(1-s)^2(1-s+t)}{(-1+2s)(-1+s-2t)(-1+t)},$$

$$x_2 = \frac{(-1+2s)^2(-1+s-2t)}{(1-s)^2(1-s+t)},$$

$$x_3 = \frac{(1-s+t)(-1+t)}{(-1+s-2t)^2}.$$

As in Example 6, we employ the following useful technique: we introduce three new variables $u := (-1+2s)(-1+t), v := (1-s)^2(1-s+t), w := -1+s-2t$ and define polynomials $F_0 = 1+s+t-s^2+2st-s^3+s^2t-xuw$, $F_1 = -1+5s-2t-8s^2+8st+4s^3-8s^2t-yv$, $F_2 = -1+s-st+t^2-zw^2$. `ResPol` yields an implicit polytope with vertices $(0,6,7)$, $(6,0,0)$, $(0,6,0)$, $(0,0,7)$, $(0,0,0)$, $(6,6,4)$, $(6,0,4)$, $(6,6,0)$ containing 308 lattice points. The interpolation matrix has corank 8. The gcd of the polynomials corresponding to two randomly chosen kernel vectors equals the actual implicit equation:

$\Delta_B(x_1,x_2,x_3) = 16x_1^5x_2^5x_3^3 + 80x_1^4x_2^4x_3^3 - 8x_1^4x_2^4x_3^2 + 500x_1^4x_2^3x_3^2 + 3125x_1^4x_2^2x_3^2 + 160x_1^3x_2^3x_3^3 - 32x_1^3x_2^3x_3^2 + x_1^3x_2^3x_3 + 1000x_1^3x_2^2x_3^2 - 225x_1^3x_2^2x_3 + 160x_1^2x_2^2x_3^3 - 48x_1^2x_2^2x_3^2 + 3x_1^2x_2^2x_3 + 500x_1^2x_2x_3^2 - 225x_1^2x_2x_3 + 27x_1^2x_2 + 80x_1x_2x_3^3 - 32x_1x_2x_3^2 + 3x_1x_2x_3 + 16x_3^3 - 8x_3^2 + x_3$.

Let $l_1 = 1-s, l_2 = 1-s+t, l_3 = 1-s, l_4 = -1+2s, l_5 = -1+s-2t, l_6 = -1+t$. Then we can rewrite the parameterization as

$$(x_1,x_2,x_3) = \left(\frac{l_1^2l_2}{l_4l_5l_6}, \frac{l_4^2l_5}{l_1l_2l_3}, \frac{l_2l_6}{l_5^2}\right),$$

and obtain the A-discriminant as

$D_A(c) = \Delta_B(\frac{c_1^2c_2}{c_4c_5c_6}, \frac{c_4^2c_5}{c_1c_2c_3}, \frac{c_2c_6}{c_5^2}) = c_5^4c_6^3c_3^5 - 8c_2c_5^2c_6^4c_3^5 + 16c_2^2c_6^5c_3^5 + 3c_5^4c_6^2c_3^4c_1c_4 - 32c_2c_5^2c_6^3c_3^4c_1c_4 + 80c_2^2c_6^4c_3^4c_1c_4 + 27c_5^5c_3^4c_1^3 - 225c_2c_5^3c_6c_3^4c_1^3 + 500c_2^2c_5c_6^2c_3^4c_1^3 + 3c_5^4c_6c_3^3c_1^2c_4^2 - 48c_2c_5^2c_6^2c_3^3c_1^2c_4^2 + 160c_2^2c_6^3c_3^3c_1^2c_4^2 - 225c_2c_5^3c_3^3c_1^4c_4 + 1000c_2^2c_5c_6c_3^3c_1^4c_4 + c_5^4c_3^2c_1^3c_4^3 - 32c_2c_5^2c_6c_3^2c_1^3c_4^3 + 160c_2^2c_6^2c_3^2c_1^3c_4^3 + 3125c_2^3c_3^3c_1^6 + 500c_2^2c_5c_3^2c_1^5c_4^2 - 8c_2c_5^2c_3c_1^4c_4^4 + 80c_2^2c_6c_3c_1^4c_4^4 + 16c_2^2c_1^5c_4^5$.

Example 20 Consider the discriminant computation with matrix

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

[8] ck: why?

which gives $l_1 = 3, l_2 = -1-s-t, l_3 = -1-s, l_4 = -s+t, l_5 = 2s+t, l_6 = -1+s-t$. We have the parameterization

$$x_1 = \frac{l_1^3}{l_2 l_3 l_6} = \frac{27}{(-1+s-t)(-1-s-t)(-1-s)},$$
$$x_2 = \frac{l_5^2 l_6}{l_2 l_3 l_4} = \frac{(2s+t)^2(-1+s-t)}{(-1-s-t)(-1-s)(-s+t)},$$
$$x_3 = \frac{l_4 l_5}{l_2 l_6} = \frac{(-s+t)(2s+t)}{(-1+s-t)(-1-s-t)}.$$

`ResPol` yields Newton polytope vertices $(6,4,3)$, $(6,0,0)$, $(0,6,0)$, $(0,0,9)$, $(0,0,0)$, $(4,6,5)$, $(6,0,3)$, $(6,4,0)$, $(0,6,9)$, $(4,6,0)$. We build a matrix M of corank 6 and obtain Δ_B by computing the gcd of polynomials corresponding to two randomly chosen kernel vectors. It is a polynomial of degree 10 containing 74 terms shown in Appendix A. Substituting x_i's by the corresponding rational functions in l_i's and renaming each l_i as c_i, we get the discriminant D_A.

Example 21 [2] We compute the A-discriminant when $A = \{(0,2,0),(0,0,6),(0,1,2),(1,2,0),(1,1,3),(1,2,2),(1,1,2)\}$. Then

$$A = \begin{pmatrix} 1&1&1&1&1&1&1 \\ 0&0&0&1&1&1&1 \\ 2&0&1&2&1&2&1 \\ 0&6&2&0&3&2&2 \end{pmatrix}, \quad B = \begin{pmatrix} 1&0&1 \\ 0&1&1 \\ -1&-1&-2 \\ 0&2&1 \\ 2&0&0 \\ -1&-1&-1 \\ -1&-1&0 \end{pmatrix}$$

Here $l_1 = 1+t, l_2 = s+t, l_3 = -1-s-2t, l_4 = 2s+t, l_5 = 2, l_6 = -1-s-t, l_7 = -1-s$, and we have the parameterization

$$x_1 = \frac{l_1 l_5^2}{l_3 l_6 l_7} = \frac{4(1+t)}{(-1-s-2t)(-1-s-t)(-1-s)},$$
$$x_2 = \frac{l_2 l_4^2}{l_3 l_6 l_7} = \frac{(s+t)(2s+t)^2}{(-1-s-2t)(-1-s-t)(-1-s)},$$
$$x_3 = \frac{l_1 l_2 l_4}{l_3^2 l_6} = \frac{(1+t)(s+t)(2s+t)}{(-1-s-2t)^2(-1-s-t)}$$

The predicted implicit polytope has vertices: $(0,3,9)$, $(9,0,0)$, $(0,9,0)$, $(0,0,9)$, $(0,0,0)$, $(9,0,3)$, $(0,9,3)$, $(3,0,9)$, $(0,3,9)$. The kernel of M has dimension 20. Computing the gcd of two randomly chosen polynomials gives Δ_B which is of degree nine.

After factoring Δ_B and substituting x_1, x_2, x_3 by the corresponding rational functions in l_i's and renaming each l_i as c_i, we obtain D_A. The latter seems irreducible because Maple cannot factor it even when we specialize all but one c_i to $\mathbb{Z}$. Both D_A and Δ_B are shown in Appendix A.

6. Conclusions and future work

Sparse implicitization by interpolation and by using predicted support seems to be an effective tool, both for classical geometric implicitization as well as for computing discriminants and resultants. An advantage of our method is that it can seamlessly handle base points.

We focused on the case that the kernel dimension exceeds 1. If this is due to insufficient genericity at evaluating M, one increases the randomness of evaluation points, and employs rectangular matrices with sufficiently more rows than columns, which corresponds to oversampling the given parametric object. Otherwise, the predicted polytope is a superset of the actual one. We characterized this case in terms of sparse elimination theory and discussed methods to obtain a smaller multiple or the exact implicit equation by applying multivariate polynomial gcd, either exact or apprximate. By factoring, one can determine which of the factors vanishes when the x_i variables are substituted by the parametric expressions. For larger problems, we employ approximate computation.

Our matrices have 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 [14,24].

Employing the Bernstein base representation of multivariate polynomials may improve the numerical stability of our algorithms. However, one has to cope with conversion issues and the potential increase of size of the interpolation matrix.

Acknowledgement. I. Z. Emiris, Tatjana Kalinka, and Thang Luu Ba are partially supported by Marie-Curie Initial Training Network "SAGA" (ShApes, Geometry, Algebra), FP7-PEOPLE contract PITN-GA-2008-214584. Christos Konaxis is supported from the European Union's Seventh Framework Programme (FP7-REGPOT-2009-1) under grant agreement n° 245749. We thank Alicia Dickenstein for stating and proving Thm. 10, while visiting our group, in the framework of SAGA.

References

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

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

[3] R.M. Corless, M. Giesbrecht, Ilias S. Kotsireas, and S.M. Watt. Numerical implicitization of parametric hypersurfaces with linear algebra. In *Proc. Art. Intell. Scient. Comp.*, pages 174–183, 2000.

[4] 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.

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

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

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

[8] 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.

[9] I.Z. Emiris, V. Fisikopoulos, C. Konaxis, and L. Penaranda. An output-sensitive algorithm for computing projections of resultant polytopes. In *Proc. ACM Symp. on Computational*

Geometry, Chapel Hill, North Carolina, pages 179–188, June 2012.

[10] I.Z. Emiris, A. Galligo, and H. Lombardi. Certified approximate univariate GCDs. *J. Pure Appl. Algebra*, 117-118:229–251, 1997.

[11] I.Z. Emiris, T. Kalinka, and C. Konaxis. Implicitization of curves and surfaces using predicted support. In *Electr. Proc. Inter. Works. Symbolic-Numeric Computation*, pages 137–146, San Jose, Calif., 2011.

[12] I.Z. Emiris, C. Konaxis, and L. Palios. Computing the Newton polygon of the implicit equation. *Math. in Comp. Science, Spec. Issue Comp. Geometry & CAD*, 4:25–44, 2010.

[13] 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.

[14] I.Z. Emiris and V.Y. Pan. Symbolic and numeric methods for exploiting structure in constructing resultant matrices. *J. Symbolic Computation*, 33:393–413, 2002.

[15] 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.

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

[17] 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.

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

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

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

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

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

[23] 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.

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

[25] E. Wurm, J.B. Thomassen, B. Juttler, and T. Dokken. Comparative benchmarking of methods for approximate implicitization. In *Geom. Modeling & Computing*, pages 537–548. 2004.

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

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

Appendix A. Omitted results

Results from Example 20

$\Delta_B(x_1,x_2,x_3) = -14348907x_2^3 + 314928x_2^2x_3^8 + 43046721x_2^3x_3 - 239112x_1x_2^4x_3^5 + 451980x_1x_2^4x_3^4 + 731916x_1x_2^4x_3^3 - 1023516x_1x_2^4x_3^2 + 393660x_1x_2^4x_3 + 62208x_1^2x_2x_3^5 + 93312x_1^2x_2x_3^4 + 23328x_1x_2^2x_3^7 + 7912566x_1x_2^2x_3^5 + 98415x_1x_2^2x_3^4 - 13994613x_1x_2^2x_3^3 + 27103491x_1x_2^2x_3^2 + 1102248x_1x_2x_3^6 + 17537553x_1x_2x_3^4 + 1417176x_1x_2x_3^5 + 414072x_1x_2^3x_3^6 - 125388x_1x_2^3x_3^5 - 1062882x_1x_2^3x_3^4 + 5334093x_1x_2^3x_3^3 - 1200663x_1x_2^3x_3^2 + 3011499x_1x_2^3x_3 - 729x_1^2x_2^4x_3^4 + 2187x_1^2x_2^4x_3^3 - 2187x_1^2x_2^4x_3^2 + 729x_1^2x_2^4x_3 + 25272x_1^2x_2^3x_3^5 - 6804x_1^2x_2^3x_3^4 - 657666x_1^2x_2^3x_3^3 + 19683x_1^2x_2^3x_3^2 + 104976x_1^2x_2^3x_3 + 432x_1^2x_2^2x_3^6 - 864x_1^2x_2^2x_3^5 + 1368576x_1^2x_2^2x_3^4 + 1465776x_1^2x_2^2x_3^3 + 2511405x_1^2x_2^2x_3^2 + 432x_1^3x_2^3x_3^4 - 864x_1^3x_2^3x_3^3 - 1512x_1^3x_2^3x_3^2 + 1944x_1^3x_2^3x_3 + 66816x_1^3x_2^2x_3^3 + 86400x_1^3x_2^2x_3^2 + 1024x_1^3x_2x_3^4 + 1024x_1^4x_2^2x_3^2 + 314928x_2^5x_3^5 + 944784x_2^5x_3^3 - 944784x_2^5x_3^4 - 314928x_2^5x_3^2 + 944784x_2^4x_3^6 + 5196312x_2^4x_3^4 - 1889568x_2^4x_3^5 + 12754584x_2^4x_3^2 - 12754584x_2^4x_3^3 - 4251528x_2^4x_3 - 944784x_2^3x_3^6 + 944784x_2^3x_3^7 - 25509168x_2^3x_3^4 + 12754584x_2^3x_3^5 - 43046721x_2^3x_3^2 + 27103491x_2^3x_3^3 - 12754584x_2^2x_3^5 + 12754584x_2^2x_3^6 + 43046721x_2^2x_3^2 - 86093442x_2^2x_3^3 + 43046721x_2^2x_3^4 + 4251528x_2x_3^7 + 43046721x_2x_3^5 - 43046721x_2x_3^4 - 729x_1^3x_2^3 - 59049x_1^2x_2^3 + 14348907x_3^6 - 1594323x_1x_2^3.$

$D_A = 314928c_5^9c_6^6c_3c_4^3 + 14348907c_4^9c_2^4c_3^6 - 1434890c_6^9c_2^7c_3^3 + 432c_1^9c_5^4c_4^4c_6^2 + 23328c_1^3c_5^5c_4^8c_3^3 + 432c_1^6c_5^4c_4^7c_3^2 - 729c_1^6c_5^6c_6^4c_4^3 + 729c_1^6c_6^7c_5^3c_2^3 - 314928c_5^6c_6^9c_2^3c_3 + 944784c_5^8c_4^5c_6^4c_3^2 - 4251528c_5^3c_6^9c_2^5c_3^2 + 944784c_5^7c_4^7c_6^2c_3^3 + 4251528c_5^3c_4^9c_2^2c_3^5 - 729c_1^9c_2^4c_6^6 - 657666c_1^6c_5^3c_2^2c_6^4c_3c_4^3 + 19683c_1^6c_5^2c_2^3c_6^5c_3c_4^2 + 5334093c_1^3c_5^3c_2^3c_6^5c_3^2c_4^3 + 3011499c_1^3c_6^7c_5c_2^5c_3^2c_4 + 86400c_1^9c_2^3c_6^3c_3c_4^3 + 1465776c_1^6c_5c_4^4c_2^3c_6^3c_3^2 + 98415c_1^3c_5^2c_4^5c_2^3c_6^3c_3^3 - 13994613c_1^3c_5c_4^4c_2^4c_6^4c_3^3 + 7912566c_1^3c_5^3c_4^6c_2^2c_6^2c_3^3 + 1417176c_1^3c_5c_4^7c_2^3c_6c_3^4 + 27103491c_1^3c_2^5c_6^5c_3^3c_4^3 + 17537553c_1^3c_4^6c_2^4c_6^2c_3^4 + 93312c_1^6c_4^6c_2^3c_6c_3^3 + 731916c_1^3c_5^5c_2^2c_6^6c_3c_4^2 - 1023516c_1^3c_6^7c_5^4c_2^3c_3c_4 - 864c_1^6c_5^3c_4^6c_2c_6c_3^2 - 6804c_1^6c_5^4c_4^4c_2c_6^3c_3 + 104976c_1^6c_5c_2^4c_6^6c_3c_4 + 2511405c_1^6c_2^4c_6^4c_3^2c_4^3 - 1062882c_1^3c_5^4c_4^4c_2^2c_6^4c_3^2 - 125388c_1^3c_5^5c_4^5c_2c_6^3c_3^2 - 1200663c_1^3c_5^2c_2^4c_6^6c_3^2c_4^2 + 1368576c_1^6c_5^2c_4^5c_2^2c_6^2c_3^2 + 451980c_1^3c_5^6c_2c_6^5c_3c_4^3 + 66816c_1^9c_5c_4^4c_2^2c_6^2c_3 + 5196312c_5^6c_2^2c_6^6c_3^2c_4^3 + 27103491c_5^3c_2^4c_6^6c_3^3c_4^3 + 43046721c_2^6c_6^6c_3^4c_4^3 + 393660c_1^3c_6^8c_5^3c_2^4c_3 - 864c_1^9c_5^3c_2c_6^3c_4^3 + 1024c_1^9c_4^6c_2^2c_3^2 + 414072c_1^3c_5^6c_4^6c_6^2c_3^2 - 239112c_1^3c_5^7c_4^4c_6^4c_3 + 62208c_1^6c_5c_4^7c_2^2c_3^3 + 25272c_1^6c_5^5c_4^5c_6^2c_3 - 2187c_1^6c_5^4c_2^2c_6^6c_4 + 43046721c_5c_6^8c_2^6c_3^3c_4 + 944784c_5^7c_6^8c_2^2c_3c_4 - 944784c_5^8c_6^7c_2c_3c_4^2 - 1889568c_5^7c_4^4c_2c_6^5c_3^2 + 12754584c_5^4c_6^8c_2^4c_3^2c_4 - 12754584c_5^5c_6^7c_2^3c_3^2c_4^2 - 944784c_5^6c_4^6c_2c_6^3c_3^3 - 25509168c_5^4c_4^4c_2^3c_6^5c_3^3 + 12754584c_5^5c_4^5c_2^2c_6^4c_3^3 - 43046721c_5^2c_6^7c_2^5c_3^3c_4^2 - 12754584c_5^3c_4^6c_2^3c_6^3c_3^4 + 12754584c_5^4c_4^7c_2^2c_6^2c_3^4 - 86093442c_5c_4^4c_2^5c_6^5c_3^4 + 43046721c_5^2c_4^5c_2^4c_6^4c_3^4 + 43046721c_5c_4^7c_2^4c_6^2c_3^5 - 43046721c_4^6c_2^5c_6^3c_3^5 - 59049c_1^6c_6^7c_2^5c_3 - 1594323c_1^3c_6^8c_2^6c_3^2 + 314928c_5^6c_4^9c_3^4 + 1024c_1^{12}c_2^2c_6^2c_4^3 + 1944c_1^9c_5c_2^3c_6^5c_4 - 1512c_1^9c_5^2c_2^2c_6^4c_4^2 + 1102248c_1^3c_5^2c_4^8c_2^2c_3^4 + 2187c_1^6c_5^5c_2c_6^5c_4^2.$

Results from Example 21

$\Delta_B(x_1,x_2,x_3) = 512x_1x_2x_3^3 - 576x_1x_2x_3^5 - 1024x_1x_2^2x_3^2 + 3712x_1x_2^2x_3^3 + 320x_1x_2^2x_3^4 - 1664x_1x_2^3x_3^2 + 320x_1x_2^3x_3^3 - 64x_1x_2^4x_3^2 - 608x_1^2x_2x_3^3 + 368x_1^2x_2x_3^4 - 960x_1^2x_2x_3^5 + 1824x_1^2x_2^2x_3^2 + 880x_1^2x_2^2x_3^3 + 1088x_1^2x_2^2x_3^4 - 64x_1^2x_2^2x_3^5 - 1296x_1^2x_2^3x_3 + 64x_1^2x_2^3x_3^2 - 64x_1^2x_2^3x_3^3 + 64x_1^2x_2^3x_3^4 - 16x_1^2x_2^4x_3 + 144x_1^3x_2x_3^3 - 640x_1^3x_2x_3^4 - 128x_1^3x_2x_3^5 + 108x_1^3x_2^2x_3 + 60x_1^3x_2^2x_3^2 + 784x_1^3x_2^2x_3^3 + 128x_1^3x_2^2x_3^4 - 16x_1^3x_2^3x_3^2 + 16x_1^3x_2^3x_3^3 - 16x_1^4x_2x_3^3 + 64x_1^4x_2x_3^4 - 27x_1^4x_2^2x_3 + 128x_1^4x_2^2x_3^2 + 32x_1^4x_2^2x_3^3 + 16x_1^5x_2x_3^3 + 2048x_2^2x_3^3 - 144x_1^2x_3^5 + 192x_1^3x_3^5 - 216x_1^3x_2^3 - 64x_1^4x_3^5$

$D_A = -512c_1c_5^2c_2c_3^6 + 576c_1^3c_5^2c_6^2c_2^3c_4^2c_3^2 + 1024c_5^2c_7c_2c_4c_3^7 - 320c_1^2c_5^2c_6^2c_7c_2^3c_4^3c_3^3 - 3712c_1c_5^2c_6c_7c_2^2c_4^2c_3^5 + 608c_1^2c_5^4c_6c_7c_2c_3^5 + 1664c_5^2c_6c_7^2c_2^2c_4^3c_3^6 - 320c_1c_5^2c_6^2c_7^2c_2^3c_4^4c_3^4 - 368c_1^3c_5^4c_6^2c_7c_2^2c_4c_3^3 + 960c_1^4c_5^4c_6^3c_7c_2^3c_4^2c_3 - 880c_1^2c_5^4c_6^2c_7c_2^2c_4^2c_3^4 - 1824c_1c_5^4c_6c_7^2c_2c_4c_3^6 - 1088c_1^3c_5^4c_6^3c_7^2c_2^3c_4^3c_3^2 + 1296c_5^4c_6c_7^3c_2c_4^2c_3^7 - 64c_1c_5^4c_6^2c_7^3c_2^2c_4^3c_3^5 + 64c_1^2c_5^4c_6^3c_7^3c_2^2c_4^3c_3^3 - 64c_1^3c_5^4c_6^4c_7^3c_2^4c_4^5c_3 - 784c_1^3c_5^6c_6^3c_7^3c_2^2c_4^2c_3^3 + 16c_5^4c_6^2c_7^4c_2^2c_4^4c_3^6 + 216c_5^6c_6c_7^4c_4c_3^8 + 64c_1^4c_5^4c_6^4c_7^2c_2^4c_4 + 128c_1^5c_5^6c_6^4c_7^2c_2^3c_4^2 + 64c_1^6c_5^8c_6^4c_7^2c_2^2 - 64c_1^5c_5^8c_6^4c_7^3c_2^2c_4c_3 + 64c_5^2c_6^2c_7^3c_2^3c_4^5c_3^5 - 2048c_2^2c_4^2c_3^6 + 640c_1^4c_5^6c_6^3c_7^2c_2^2c_4c_3^2 - 108c_1c_5^6c_6c_7^3c_3^7 - 60c_1^2c_5^6c_6^2c_7^3c_2c_4c_3^5 + 16c_1^4c_5^8c_6^3c_7^3c_2c_3^3 - 128c_1^4c_5^6c_6^4c_7^3c_2^3c_4^3c_3 + 16c_1^2c_5^6c_6^3c_7^4c_2^2c_4^3c_3^4 - 16c_1^3c_5^6c_6^4c_7^4c_2^3c_4^4c_3^2 + 27c_1^2c_5^8c_6^2c_7^4c_3^6 - 128c_1^3c_5^8c_6^3c_7^4c_2c_4c_3^4 - 32c_1^4c_5^8c_6^4c_7^4c_2^2c_4^2c_3^2 - 16c_1^5c_5^10c_6^4c_7^4c_2c_3^2 - 192c_1^5c_5^6c_6^3c_7c_2^2c_3 - 144c_1^3c_5^6c_6^2c_7^2c_2c_4^3 + 144c_1^4c_5^4c_6^2c_2^2c_3^2.$

Appendix B. Tables

Table B.1
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	$x^2+y^2+z^2-w^2$

Table B.2
The table shows the vertices of the actual implicit polytope, the number of its lattice points, the degree and number of monomials of the implicit equation, the vertices of the predicted implicit polytope, the number of its lattice points, and the corank of M.

Geometric object	Implicit				Predicted			
	Newton polytope vertices	lattice points	degree	mono-mials	Newton polytope vertices	lattice points	corank of M	# g_i's of (degree)
Trifolium curve	$(4,0),(1,2),(0,4),(3,0)$	8	4	5	$(8,0),(0,8),(1,0),(0,2)$	43	15	1(4),2(5),3(6),4(7),5(8)
Cayley sextic	$(6,0),(0,6),(0,4),(3,0)$	19	6	11	$(0,2),(1,0),(0,12),(12,0)$	89	28	1(6),2(7),3(8),4(9),5(10),6(11),7(12)
Sphere	$(0,0,0),(0,2,0),(2,0,0),$ $(0,0,2)$	10	2	4	$(0,0,2),(4,0,0),(0,4,0),$ $(0,0,4),(0,2,0),(0,0,2)$	35	10	1(2),3(3),6(4)
Double sphere	$(0,0,0),(0,2,0),(2,0,0),$ $(0,0,2)$	10	2	4	$(4,0,0),(0,4,0),(0,0,8),$ $(2,0,0),(0,0,1),(4,0,4),$ $(0,4,4)$	125	45	3(4),4(5),9(6),11(7),18(8)
Eight surface	$(0,2,0),(2,0,0),(0,0,2),$ $(0,0,4)$	10	4	4	$(4,0,0),(0,4,0),(0,0,16),$ $(1,0,0),(0,0,1),(4,0,8),$ $(0,2,0),(0,4,8)$	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	$(0,2,0,0),(2,0,0,0),$ $(0,0,2,0),(0,0,0,2)$	10	2	4	$(0,0,0,8),(0,0,8,0),$ $(0,8,0,0),(8,0,0,0)$	165	84	84(8)