Implicitization of curves and (hyper)surfaces using predicted support

Ioannis Z. Emiris[a], Tatjana Kalinka[a], Christos Konaxis[b], Thang Luu Ba[a,c] *

[a]Department of Informatics & Telecommunications, University of Athens, Greece
[b]Archimedes Center for Modeling, Analysis & Computation (ACMAC), University of Crete, Heraklio, Greece
[c]Department of Mathematics, Hanoi National University of Education, Hanoi, Vietnam

May 31, 2012

Abstract

We reduce implicitization of rational planar parametric curves and (hyper)surfaces to linear algebra, by interpolating the coefficients of the implicit equation. For predicting the implicit support, we focus on methods that exploit input and output structure in the sense of sparse (or toric) elimination theory, namely by computing the Newton polytope of the implicit polynomial, via sparse resultant theory. Our algorithm works even in the presence of base points but, in this case, the implicit equation shall be obtained as a factor of the produced polynomial. We implement our methods on Maple, and some on Matlab as well, and study their numerical stability and efficiency on several classes of curves and surfaces. We apply our approach to approximate implicitization, and quantify the accuracy of the approximate output, which turns out to be satisfactory on all tested examples; we also relate our measures to Hausdorff distance. In building a square or rectangular matrix, an important issue is (over)sampling the given curve or surface: we conclude that unitary complexes offer the best tradeoff between speed and accuracy when numerical methods are employed, namely SVD, whereas for exact kernel computation random integers is the method of choice. We compare our prototype to existing software and find that it is rather competitive.

1 Introduction

Implicitization is the problem of changing the representation of parametric objects to implicit form. It 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. Implicitization is also of independent interest, since certain questions in areas as diverse as robotics or statistics reduce to deriving the implicit form. For instance, in [CTY10] they implicitize a 16-dimensional hypersurface by interpolation, once they have computed the implicit polytope. Another motivation is to compute the dual of a curve by interpolation [Vol97]: our approach would exploit structure to yield a faster method in sparse cases.

Here we follow a classical symbolic-numeric method, which reduces implicitization to interpolating the coefficients of the defining equation. We implement interpolation by exact or numeric linear algebra following an exact phase which computes a (super)set of the monomials appearing in the implicit equation. These monomials are then suitably evaluated to build a numeric matrix, ideally of corank 1, whose kernel vector contains their coefficients in the implicit equation. We give techniques for handling the case of higher corank.

One contribution of this paper is to exploit sparse (or toric) variable elimination theory to predict the Newton polytope of the implicit equation.

Definition 1. Given a polynomial

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

its *support* is the set $A_i = \{a\in\mathbb{N}^n : c_{ia}\neq 0\}$; its *Newton polytope* is the convex hull of the support. We shall call the support of the implicit equation, *implicit support*, and its Newton polytope, *implicit polytope*.

One reason for revisiting interpolation of the implicit coefficients is the current increase of activity around various approaches capable of predicting the implicit support. Our team has been focusing on sparse elimination theory [EFKP12, EKP10, EK03, EKK11]. Theorem 4 settles the general case by showing that this approach

*Email: {emiris,kalinkat,thanglb}@di.uoa.gr, ckonaxis@acmac.uoc.gr

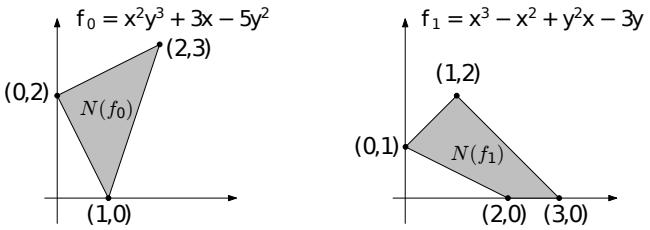

Figure 1: Examples of Newton polygons $N(f_i)$ of polynomials $f_i \in \mathbb{Z}[x, y]$.

yields a superset of the implicit support. Recent support prediction methods notably include tropical geometry methods, e.g. [Cue10, DS10, JY11, STY07, SY08]; see section 3. The present work can use the implicit support predicted by any method. In fact, [SY08, 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 [...] "* Our implementations interface linear algebra interpolation with the implicit support predictors of [EFKP12, EKP10]. In the sequel, we juxtapose the use of exact and numerical linear algebra.

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 [DT03, SJ08, BD10a, BD10b]. We discuss approximate implicitization, in the setting of sparse elimination. Approximate implicitization is one of the main motivations for reducing implicitization to interpolation of the implicit coefficients. We offer a publicly available Maple implementation [1], which is based on the software for computing implicit polytopes from [EFKP12]. The latter is also available as a C++ implementation[2]. We study the numerical stability and efficiency of our algorithms on several classes of curves and surfaces. 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. It appears that complex unitary numbers offer the best tradeoff of efficiency and accuracy for numerical computation, whereas random integers are preferred for exact kernel computation.

Let us now define the problem formally. A *parametrization* of a geometric object of *co-dimension one*, in a space of dimension $n+1$, can be described by parametric map:

$$f : \mathbb{R}^n \to \mathbb{R}^{n+1} : t = (t_1, \ldots, t_n) \mapsto x = (x_0, \ldots, x_n),$$

where t 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 $\Omega := \Omega_1 \times \cdots \times \Omega_n,\ \Omega_i \subseteq \mathbb{R}$. In the case of trigonometric input, we restrict our study to those functions that may be converted to polynomials 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 $t = \tan\theta/2$.

The *implicitization problem* asks for the smallest algebraic variety containing the image of the parametric map $f : t \mapsto f(t)$. This image is contained in the variety defined by the ideal of all polynomials $p(x_0, \ldots, x_n)$ 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. If the parametrization is not proper but has degree $\theta > 1$, our method computes the implicit equation to the power θ.

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

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

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

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

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

which can be represented as polynomials

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

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

$$\bar{F}_{n+1} = 1 - g_0(t) \cdots g_n(t) y, \tag{4}$$

where y is a new variable.

Several algorithms exist for implicitization, including methods based on resultants, Gröbner bases, μ-bases and moving surfaces, and residues. Our approach can use any support prediction method, see Section 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 [EKP10]. In the general case, the implicit support is provided by that of a symbolic sparse resultant related to the polynomials in (3), whose Newton polytope is projected to the space of the x_i's [EFKP12]. The theoretical foundations of our approach are given in Theorem 4, Lemma 8 and Corollary 9.

We can impose the additional assumption $\gcd(f_0, \ldots, f_n, g_0, \ldots, g_n) = 1$, on (3), which assures that the set of base points of the parametrization is finite. This is not necessary because our method, being based on a *symbolic* resultant, works even in the presence of base points of arbitrary dimension. In this case however, its output is, usually, a large multiple of the actual implicit equation. Our approach can also handle f_i with (certain) symbolic nonzero coefficients, thus computing the implicit polytope for entire families of parametric objects.

Having reduced implicitization to interpolation, we employ standard methods to determine the unknown coefficients by linear algebra. These are divided in two main categories, dense and sparse methods. The former require only a bound on the total degree of the target polynomial, whereas the latter require a bound on the number of its terms, thus exploiting any sparseness of the target polynomial. A priori knowledge of the support helps significantly, by essentially answering the first step of sparse interpolation algorithms.

The rest of the paper is structured as follows. Previous work is discussed in the next two sections: Section 2 discusses existing methods for interpolating the implicit polynomial's coefficients by linear algebra. Section 3 discusses implicit support prediction and sparse elimination theory. Our algorithm is detailed in Section 4, where we discuss complexity issues and mention possible algorithmic extensions. The algorithm's implementation, and performance are described in Section 5, where we compare it to other implementations and present some examples. Numerical issues in sampling the objects, the accuracy in the case of approximate implicitization, and how our quality measures are related to Hausdorff distance are in Section 6. We conclude with future work in Section 7. The Appendix contains examples of exact and approximate implicit equations of parametric curves and surfaces used in our experiments, and further experimental results.

A preliminary version of partial results from this paper appeared as [EKK11]. Here all experiments have been revisited using an improved method for computing the implicit polytope [EFKP12]. This work also includes a better discussion of sparse elimination theory, including Theorem 4, Lemma 8 and Corollary 9, comparison with μ-bases, Groebner bases, and the Maple implicitization routine, as well as a discussion on how to measure the accuracy of approximation when the implicit equation is obtained by numerical computation. New experiments consider downscaling the implicit polytope and emphasize symbolic-numeric aspects such as oversampling.

2 Existing interpolation methods

This section examines how implicitization had been reduced to interpolation. Throughout the paper, we use interpolation to refer to the method of determining the implicit coefficients from the implicit support and its evaluations on points of our choice. Of course, these points lie in the space of parameters.

Let S be (a superset of) the support of the implicit polynomial $p(x_0, \ldots, x_n) = 0$, and p be the $|S| \times 1$ vector of its unknown coefficients. We refer to S as implicit support, with the understanding that it may be a superset of the actual support.

Sparse interpolation is the problem of interpolating a multivariate polynomial when information of its support is given [Zip93]. This may simply be a bound σ on support cardinality, then sparse interpolation is achieved in $O(|S|^3 \delta n \log n + \sigma^3)$, where δ bounds the output degree per variable, $|S|$ here is the actual support cardinality, and n the number of variables [BOT88, KL89]. A probabilistic approach runs in $O(|S|^2 \delta n)$ [Zip90] and requires as input only δ.

For the sparse interpolation of resultants, the quasi-Toeplitz structure of the matrix allows us to reduce complexity by one order of magnitude, when ignoring polylogarithmic factors, and arrive at a quadratic complexity in matrix size [CKL89]. This was extended to the case of sparse resultant matrices [EP05].

Our matrices reveal what we call quasi-Vandermonde structure, since the matrix columns are indexed by monomials and the rows by values on which the monomials are evaluated. This reduces matrix-vector multiplication to multipoint evaluation of a multivariate polynomial. It is unclear how to achieve this post-multiplication

in time quasi-linear in the size of the polynomial support when the evaluation points are arbitrary, as in our case. Existing work achieves quasi-linear complexity for specific points [EP02, Pan94, Sau04, vdHS10].

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 monomials with exponents in S (columns) and $|S|$ different values (rows) at which all monomials get evaluated. Then, vector p is in the kernel of M. This idea was used in [EK03, MM02, SY08]; it is the approach explored in this paper, extended to an approximate implicitization as well.

In [STY07], 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 [CGKW00], based on integration of matrix $M = SS^T$, over each parameter $t_1, \ldots, t_n$. Then, p is in the kernel of M. In fact, the authors propose to consider successively larger supports in order to capture sparseness. This method covers a wide class of parametrizations, including polynomial, rational, and trigonometric representations, but the size of M is quite big and matrix entries take big values, so it is difficult to control its numeric corank. In some cases, its corank is ≥ 2. Thus, the accuracy, or quality, of the approximate implicit polynomial is unsatisfactory. The resulting matrix has Henkel-like structure [KL03]. When it is computed over floating-point numbers, the resulting implicit polynomial does not necessarily have integer coefficients. In [CGKW00], they discuss some post-processing to yield the integer relations among the coefficients, but only for small examples.

2.2 Approximate implicitization

Approximate implicitization over floating-point numbers was introduced by T. Dokken and co-workers in a series of papers. Today, there are direct [DT03, WTJD04] and iterative techniques [APJ12]. We describe the basic direct method [DT03]: Given a parametric (spline) curve or surface $x(t),\ t \in \Omega \subset \mathbb{R}^n$, the goal is to find polynomial $q(x)$ such that $q(x(t) + \eta(t)g(t)) = 0$, where $g(t)$ is a continuous direction function with Euclidean norm $\|g(t)\| = 1$ and $\eta(t)$ a continuous error function with $|\eta(t)| \leq \epsilon$. Now, $q(x(t)) = (Mp)^T\alpha(t)$, where matrix M is built from monomials in x. It may be constructed as in this paper, or it may contain a subset of the monomials of the implicit support. Moreover, p is the vector of implicit coefficients, hence $Mp = 0$ returns the exact solution, and $\alpha(t)$ is the basis of the space of polynomials which describes $q(x(t))$, and is assumed to form a partition of unity and to be nonnegative over Ω: $\sum_i \alpha_i = 1,\ \alpha_i \geq 0,\ \forall i, t \in \Omega$. One may use the Bernstein-Bézier basis with respect to Ω, in the case of curves, or a triangle which contains Ω, in the case of surfaces.

In [DT03, 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 complex values leads to better numerical stability. Since both our and their methods rely on SVD, our experiments confirm their findings.

The idea of the above methods is to interpolate the coefficients using successively larger supports, starting with a quite small support and extending it so as to reach the exact one. All existing approaches, e.g. [CGKW00], have used upper bounds on the total implicit degree, thus ignoring any sparseness structure. This fails to take advantage of the sparseness of the input in order to accelerate computation, and [3] Our methods provide a formal manner to examine different supports, in addition to exploiting sparseness.

In the context of sparse elimination, the Newton polytope captures the notion of degree. Given an implicit polytope we can naturally define candidates of smaller support, the equivalent of lower degree in classical elimination, by an inner offset of the implicit polytope. The operation of scaling down the polytope can be repeated, thus producing a list of implicit supports yielding smaller implicit equations with larger approximation error. See Examples 3 and 4 for when the predicted implicit polytope is much larger than the true one.

3 Support prediction

This section describes our methods for computing the implicit support, which is based on the sparse resultant, and any information we obtain from this computation towards computing the implicit equation.

In order to exploit sparseness in the implicit polynomial in the sense of Proposition 6, the problem of computing the Newton polytope of a rational hypersurface was posed in [SY94] for generic Laurent polynomial parametrizations, in the framework of sparse elimination theory.

Algorithms based on tropical geometry have been offered in [DFS07, STY07, SY08]. This method computes the abstract tropical variety of a hypersurface parametrized by generic Laurent polynomials in any number of variables, thus yielding its implicit support; it is implemented in `TrIm`. For non-generic parametrizations of rational curves, the implicit polygon is predicted. In higher dimensions, the following holds:

[3] c: the phrase ends unexpectedly

Proposition 1. [STY07, 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 constructed combinatorially from $P_0, \ldots, P_n$ using tropical geometry contains a translate of the Newton polytope of g.*

The tropical approach was improved in [Cue10] to yield the precise implicit polytope in $\mathbb{R}^3$ for generic parametrizations of surfaces in 3-space. In [JY11], they describe efficient algorithms implemented in the GFan library for the computation of Newton polytopes of specialized resultants, which may then be applied to predict the implicit polytope. Sparse elimination has been used for the same task [EFKP12], as detailed below. The latter is faster on dimensions relevant here, namely for projected polytopes in up to 5 dimensions.

The Newton polygon of a curve parametrized by rational functions, without any genericity assumption, is determined in [DS10]. In a similar direction, an important connection with combinatorics was described in [EK06], 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 [EKP10], sparse elimination is applied to determine the vertex representation of the implicit polygon of planar curves. The method relies on the study of the Newton polytope of a resultant. It can be applied to polynomial and rational parametrizations, where the latter may have the same or different denominators. In case of non-generic inputs, the predicted polygon is guaranteed to contain the Newton polygon of the implicit equation. The method can be seen as a special case of the general approach based on sparse elimination.

In [EK03] 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 [Ram02]. This approach was quite expensive and, hence, applicable only to small examples; it is refined and improved in this paper.

3.1 Sparse elimination theory

Sparse, or toric, elimination subsumes classical, or dense, elimination in the sense that, when Newton polytopes equal the corresponding simplices, the former bounds become those of the classical theory [EK03, sec.3], [SY94, thm.2(2)].

Consider the polynomial system $\bar{F}_0, \ldots, \bar{F}_n$ as in expression (3), defining a hypersurface, and let $A_i \subset \mathbb{Z}^n$ be the support of $\bar{F}_i$ and $P_i \subset \mathbb{R}^n$ the corresponding Newton polytope. The family $A_0, \ldots, A_n$ is *essential* if they jointly affinely span $\mathbb{Z}^n$ and every subset of cardinality $j, 1 \leq j < n$, spans a space of dimension $\geq j$. It is straightforward to check this property algorithmically and, if it does not hold, to find an essential subset. In the sequel, the input $A_0, \ldots, A_n \subset \mathbb{Z}^n$ is supposed to be essential.

For simplicity, in what follows we do not consider the extra polynomial in expression (4) as part of our polynomial systems. This is equivalent to considering polynomial parametrizations. However, it is straightforward to generalize the discussion below to the rational case.

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

$$F_i = \sum_{j=1}^{|A_i|} c_{ij} t^{a_{ij}} \in K[t],\ a_{ij} \in A_i,\ i = 0, \ldots, n. \tag{5}$$

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

Now we introduce our main tool, namely the sparse resultant of an overconstrained polynomial system. The *sparse resultant* of polynomial system in expression (5) is an irreducible polynomial

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

defined up to sign, vanishing iff $F_0 = F_1 = \cdots = F_n = 0$ has a common root in the torus $(\mathbb{C}^*)^n$. The Newton polytope $N(\mathcal{R})$ of the resultant polynomial is the *resultant polytope*. We call any monomial which corresponds to a vertex of $N(\mathcal{R})$ an *extreme term* of $\mathcal{R}$.

The *Minkowski sum* $A + B$ of convex polytopes $A, B \subset \mathbb{R}^n$ is the set $A + B = \{a + b \mid a \in A, b \in B\} \subset \mathbb{R}^n$. A *tight mixed subdivision* of $P = P_0 + \cdots + P_n$, is a collection of n-dimensional convex polytopes σ, called (Minkowski) *cells*, s.t.: They form a polyhedral complex that partitions P, and every cell σ is a Minkowski sum of subsets $\sigma_i \subset P_i$: $\sigma = \sigma_0 + \cdots + \sigma_n$, where $\dim(\sigma) = dim(\sigma_0) + \cdots + dim(\sigma_n) = n$.

A cell σ is called v_i-*mixed* if it is the Minkowski sum of n one-dimensional segments $E_j \subset P_j$ and one vertex $v_i \in P_i : \sigma = E_0 + \cdots + v_i + \cdots + E_n$. A mixed subdivision is called *regular* if it is obtained as the projection of the lower hull of the Minkowski sum of lifted polytopes $\widehat{P}_i := \{(p_i, \omega(p_i)) \,|\, p_i \in P_i\}$. If the lifting function ω is sufficiently generic, then the induced mixed subdivision is tight.

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

Theorem 2. [Stu94] *Given a polynomial system as in expression (5) and a regular tight mixed subdivision of the Minkowski sum $P = P_0 + \cdots + P_n$ of the Newton polytopes of the system polynomials, an extreme term of the resultant $\mathcal{R}$ equals*

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

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

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

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

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

Proposition 3. [Cayley trick] [GKZ94] *There exist bijections between: the regular tight mixed subdivisions, the tight mixed subdivisions, or the mixed subdivisions of the convex hull of $A_0 + \cdots + A_n$ and, respectively, the regular triangulations, the triangulations, or the polyhedral subdivisions of A.*

The set of all regular triangulations corresponds to the vertices of the secondary polytope $\Sigma(A)$ of A [GKZ94]. To compute the resultant polytope, one can enumerate all regular triangulations of A: it is equivalent to enumerating all regular tight mixed subdivisions of the convex hull of $A_0 + \cdots + A_n$. Each such subdivision yields a vertex of $N(\mathcal{R})$. This method is proven to be inefficient even for medium sized inputs [EK03]; instead, we follow a different approach.

3.2 The implicit polytope

To predict the implicit polytope we use the algorithm in [EFKP12] for the computation of resultant polytopes and their orthogonal projections. Note that the latter correspond to generic specializations of the resultant.

Given the supports $A_i,\ i = 0, \ldots, n$ of the polynomials in expression (5), the algorithm in [EFKP12] computes the resultant polytope $N(\mathcal{R})$ of their sparse resultant $\mathcal{R}$ without enumerating all mixed subdivisions of the convex hull of $A_0 + \cdots + A_n$. More precisely, it is an incremental algorithm to compute $N(\mathcal{R})$ by considering an equivalence relation on mixed subdivisions, where two subdivisions are equivalent iff they specify the same resultant vertex. The class representatives are vertices of the resultant polytope. The algorithm exactly computes vertex- and halfspace-representations of the resultant polytope or its projection. It avoids computing $\Sigma(A)$, but uses the above relationships to define an oracle producing resultant vertices in a given direction. It is output-sensitive as it computes one mixed subdivision per equivalence class, and is the fastest today in dimension up to 5; in higher dimensions it is competitive with the implementation of [JY11], relying on the GFan library. Moreover, there is an approximate variant that computes polytopes whose volume differs by $\leq 10\%$ from the true volume, with a speedup of up to 25 times.

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

$$\phi : K \to K' :\ c_{ij} \mapsto c'_{ij}, \tag{6}$$

yielding a generic specialization of the coefficients c_{ij} of the polynomial system in expression (5). We denote by $F'_i := \phi(F_i),\ i = 0, \ldots, n$, the images of F_i's under ϕ. Let $\mathcal{R} := \mathrm{Res}(F_0, \ldots, F_n)$ be the resultant of polynomial system in (5) over K and $H := \mathrm{Res}(F'_0, \ldots, F'_n)$ be the resultant of $F'_0, \ldots, F'_n$ over K'. Then, the specialized sparse resultant $\phi(\mathcal{R})$ coincides (up to a scalar multiple from K') with the resultant H of the system of specialized polynomials provided that H does not vanish, a certain genericity condition is satisfied, and the parametrization is generically 1-1 [CLO98],[SY94, thm.3]:

$$\phi(\mathcal{R}) = c \cdot H,\ c \in K'. \tag{7}$$

If the latter condition fails, then $\phi(\mathcal{R})$ is a power of H. When the genericity condition fails for a specialization of the c_{ij}'s, the support of the specialized resultant $\phi(\mathcal{R})$ is a superset of the support of H modulo a translation, provided the sparse resultant does not vanish. This follows from the fact that the method computes the same polytope as the tropical approach, whereas the latter is characterized in Proposition 1. In particular, the resultant polytope is a Minkowski summand of the *fiber polytope* $\Sigma_\pi(\Delta, P)$, where polytope Δ is a product of simplices, each corresponding to a support A_i, $P = \sum_{i=0}^{n} P_i$, and π is a projection from Δ onto P. Then, $\Sigma(\Delta, P)$, is strongly isomorphic to the secondary polytope of the point set obtained by the Cayley embedding of the A_i's, [Stu94, sec.5]. The algorithm in [EFKP12] provides the Newton polytope of $\phi(\mathcal{R})$.

When specialization ϕ yields the coefficients of the polynomials in (3), i.e. $\phi(F_i) = \bar{F}_i$, then $H = \mathrm{Res}(\bar{F}_0, \ldots, \bar{F}_n) = p(x_0, \ldots, x_n)$, where $p(x_0, \ldots, x_n)$ is the implicit equation of the hypersurface defined by (3). Equation (7) reduces to

$$\phi(\mathcal{R}) = c \cdot p(x_0, \ldots, x_n),\ c \in \mathbb{C}[x_0, \ldots, x_n], \tag{8}$$

hence [EFKP12] yields a superset of the vertices of the implicit polytope. The coefficients of the polynomials in (5), which define the projection ϕ, are those who are specialized to linear polynomials in the x_i's.

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

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

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

Lemma 5. [JY11, lem.3.20] *If $a_{ij} \in A_i$ corresponds to a specialized coefficient of F_i, and lies in the convex hull of the other points in A_i corresponding to specialized coefficients, then removing a_{ij} from A_i does not change the Newton polytope of the specialized resultant.*

Furthermore, in order to eliminate some extraneous monomials predicted by our support prediction method, we may apply the following well-known degree bounds, generalized in the context of sparse elimination. For a proof, the reader may refer to [EK03].

Proposition 6. *The total degree of the implicit polynomial of the hypersurface corresponding to system (3) 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 $\bar{F}_i$, $i \neq j$, seen as polynomials in t.*

The classical results for the dense case follow as corollaries. Take a surface parametrized by polynomials of degree d, then the implicit polynomial is of degree d^2. For tensor parametrizations of bi-degree (d_1, d_2), the implicit degree is $2d_1d_2$. We use these bounds to reduce the predicted Newton polytope in certain cases, see also Corollary 10.

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

Proposition 7. [GKZ94] *$N(\mathcal{R})$ is of dimension $|A| - 2n - 1$. The inner product of any coordinate vector of $N(\mathcal{R})$ with row i of $\mathcal{A}$ is: constant, for $i = 1, \ldots, n$, and equals the mixed volume of $F_0, \ldots, F_{j-1}, F_{j+1}, \ldots, F_n$, for $j = i - (n+1)$, $i = n+1, \ldots, 2n+1$.*

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

4 Implicitization algorithm

The main steps of our algorithm are given below.
Input: Polynomial or rational parametrization $x_i = f_i(t_1, \ldots, t_n)$, $i = 0, \ldots, n$.
Output: Implicit polynomial $p(x_i)$ in the monomial basis in $\mathbb{N}^{n+1}$.

1. We obtain (a superset of) the implicit polytope.
2. Compute all lattice points $S \subseteq \mathbb{N}^{n+1}$ in the polytope.
3. Repeat $\mu \geq |S|$ times: Select value $\tau \in \mathbb{C}^n$ for t, evaluate $x_i(\tau)$, $i = 0, \ldots, n$, then evaluate each monomial in S.
4. Construct the $\mu \times |S|$ matrix M, and compute vector p in the kernel of M, where some entry of p is set to 1. Return the primitive part of polynomial $p^\top S$.

Typically $\mu = 2|S|$. If p is not unique or, equivalently, the kernel null(M) has dimension ≥ 2, hence the polynomial $p^\top S$ is a multiple of the true implicit equation. Alternatively this can be detected by factoring the polynomial and determining that it possesses more than one nontrivial factors. Then, we may repeat the entire algorithm with a scaled down copy of the polytope used in the first run of the algorithm by scaling the polytope by 1/2. If there is no solution p, then the polytope does not contain the implicit polytope and we should use an integral multiple of this polytope, by multiplying the polytope by 2. Overall, we try a constant number of polytopes following a binary search scheme.

4.1 Building the matrix

We focus on two support prediction methods. The first applies only to curves and is described in [EKP10]. The second is general and computes the support of the resultant of system (3) and of its arbitrary specializations [EFKP12]. Both methods provide us a (super)set of the implicit vertices: the set of vertices of the polytope $N(\phi(\mathcal{R}))$ of the specialized resultant $\phi(\mathcal{R})$, where $\mathcal{R}$ is the resultant of the system of polynomials in (5). This polytope is then intersected with the halfspaces described in Proposition 6; for the specifics of this operation see Corollary 10 and the discussion afterwords. In the following, we abuse notation and denote this intersection also by $N(\phi(\mathcal{R}))$.

We compute all lattice points s_j contained in $N(\phi(\mathcal{R})) \subset \mathbb{N}^{n+1}$ to obtain the set $S := \{s_1, \ldots, s_{|S|}\}$; each $s_j = (s_{j0}, \ldots, s_{jn})$ is an exponent of a (potential) monomial $m_j := x^{s_j} = x_0^{s_{j0}} \cdots x_n^{s_{jn}}$ of the implicit polynomial, where x_i is given in (2). We evaluate $m_j, j = 1, \ldots, |S|$ at some $\tau_k,\ k = 1, \ldots, \mu,\ \mu \geq |S|$; we use $\mu > |S|$ evaluation points to improve the numerical stability of our algorithm. Let $m_j|_{t=\tau_k} := \prod_i \left(\frac{f_i(\tau_k)}{g_i(\tau_k)}\right)^{s_{ji}}$ denote the evaluated j-th monomial m_j at τ_k. Thus, we construct an $\mu \times |S|$ matrix M with rows indexed by $\tau_1, \ldots, \tau_\mu$ and columns by $m_1, \ldots, m_{|S|}$:

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

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

Lemma 8. *Any polynomial in the basis of monomials indexing M, with coefficient vector in the kernel of M, is a multiple of the implicit polynomial.*

Corollary 9. *Assume that the predicted polytope equals the actual one and that we construct a $\mu \times |S|$ matrix M, as above. Then, the vector of the implicit coefficients lies in the matrix kernel, hence rank$(M) < |S|$. If the points $x(\tau_i)$, $i = 1, \ldots, \mu$ are sufficiently generic, then M has corank 1, i.e. rank$(M) = |S| - 1$. Then, if we solve $Mp = 0$ for p, such that one of its entries is set to 1, this yields the coefficients of the implicit equation in a unique fashion.*

Note that the above result follows also from relation (8). In view of Lemma 8, we have the following corollary of Proposition 6.

Corollary 10. *An appropriate translate of the Newton polytope of the irreducible factor of the polynomial obtained from any kernel vector of matrix M, that corresponds to the implicit equation, is bounded by the halfspaces defined in Proposition 6.*

As a consequence of the previous Corollary, we cannot directly apply the degree bounds on the predicted polytope since we don't know a priori its Minkowski summands. However, we can do so if the intersection of the predicted polytope with the halfspaces defined by the degree bounds contains (a translate) of the implicit polytope. If this is not the case, then our algorithm will not return any solution and we have to fallback to the initial larger predicted polytope.

4.2 Complexity

In this section we briefly analyze the asymptotic complexity of the main subroutines of our algorithm.

The complexity of the support prediction algorithm is given in [EFKP12, thm.10]. The second part of the procedure is the computation of the lattice points contained in the predicted polytope. It is a NP-hard problem to detect a lattice point in a polytope when the dimension of the polytope is an input variable. When the dimension is fixed the algorithm in [BP99] counts the number of lattice points in a polytope within polynomial time in the size of the input. The software LattE [LHH$^+$03] implements Barnivok's algorithm. The software package Normaliz [BIS] computes lattice points in polytopes, and is very fast in practice; this is the one interfaced to our software. Based on these algorithms, one can enumerate all lattice points in output-sensitive manner, i.e.

in polynomial time in the output size, which of course can be exponential in the input size. The computation up to this point is essentially offline, because it does not require knowledge of the specific coefficients.

Suppose that, for the predicted support S, the exponent of every monomial in the i-th variable lies in $[0, \delta]$, for $i = 1, 2, \ldots, n$. Let $O^*(\cdot)$ denote asymptotic bounds when ignoring polylogarithmic factors in the arguments.

Proposition 11. [EP02, lem.4.3] *Consider a set S of monomials in n variables. Given n scalar values $p_1, p_2, \ldots, p_n$, the algorithm of [EP02] evaluates all the monomials of S at these values in $O^*(|S|n + n\sqrt{\delta})$ arithmetic operations and $O(|S|n)$ space.*

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

Corollary 12. *Assume our algorithm builds a rectangular matrix $\mu \times |S|$, $\mu \geq |S|$. Then, all $\mu\,|S|$ entries are computed in $O^*(\mu\,|S|\,n)$ operations.*

Once constructed, the kernel computation costs $O(m^{2.376})$ arithmetic operations, which follows from the current record for matrix multiplication. Our bound can be improved if matrix multiplication is improved. On $\mu \times |S|$ rectangular matrices, the kernel computation has complexity $O(\mu\,|S|^2)$. Hence

Theorem 13. *The overall complexity of our implicitization algorithm is $O(\mu\,|S|^2)$.*

5 Implementation and experimental results

This section looks at the actual symbolic and numeric computations once the problem has been reduced to a question in linear algebra. We start with software for the matrix operations, then detail several examples. Our algorithms are implemented in Maple and Matlab. We report on a comparison of our implementation against existing methods.

Let us refer to Corollary 9 and assume M has corank 1. For exact computing, solving the linear system

$$Mp = 0,$$

yields the kernel vector, where one entry is set to 1. Hence we obtain all implicit coefficients p_i for each predicted monomial m_i. Exact methods can treat indefinite parameters encountered in the parametric expressions.

For larger examples, we trade exactness for speed and apply Singular Value Decomposition (SVD), thus computing

$$Mp^\top = (U\Sigma V^\top)p^\top = 0^\top \Leftrightarrow \Sigma v^\top = 0^\top, \text{ where } Vv^\top = p^\top,$$

where $UU^\top = VV^\top = I$ and Σ is diagonal. 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$(M) = 1$, $v = [0, \ldots, 0, 1]$ and the last row of $V^\top$ gives p. The same derivation holds if M is rectangular, say $\mu \times |S|, \mu \geq |S|$. Then Σ is of the same dimensions, U is $\mu \times \mu$, and V is $|S| \times |S|$, where its last column is the sought vector.

Our algorithm is implemented in Maple 13, as functions `imgen` (general implicitization, applicable for 2D, 3D and 4D geometrical objects,) and `imcurve` (for curves only, support prediction is part of the routine). The functions take the following arguments:

- list of parametric expressions;
- only `imgen`: vertices of the predicted support;
- solving method parameter: "l" for `LinearSolve`, "n" for `Nullspace`, "s" for SVD;
- evaluation parameter: "int" for integers, "unc" for random complex numbers modulo 1, "ruf" for roots of unity evaluated as floating point numbers;
- ratio between number of rows and columns of the matrix.

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

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

All experiments, unless otherwise stated, were performed on a Celeron 1.6 GHz linux machine with 1 GB of memory. Most curves and surfaces in our experiments are in Tables 8 and 9 in the Appendix. The tables show the parametric, the exact implicit and the approximate implicit representation. The last two equations are primitive and for the latter we omit terms with very small coefficients. Runtimes (sec) for the various approaches to implicitization of these curves and surfaces are given, respectively, in Tables 1 and 2, in Maple.

In both tables, we used random integers for exact computation, with functions `NullSpace` and `LinearSolve`, and unitary complexes for numeric computation with SVD.

We also used dense and sparse Bézier curves of various degrees; the runtimes for this family of curves are shown in Table 6. In this set of experiments we show the size $\mu \times |S|$, $\mu > |S|$ of matrices used in numerical computation; the corresponding matrices for exact computation are $|S| \times |S|$.

Table 10 in the Appendix shows results of our experiments with the industrial examples in [STW+06]. Converting the input from the Bernstein to the monomial basis, the resulting equation contains monomials (usually of high degree) with coefficients close to zero. Removing them, we get a smaller Newton polytope compared to the one obtained from the original equation; the latter is referred in the table as "raw".

In order to use LinearSolve we round the coefficients to integers, otherwise the kernel vector contains only zeros. In some cases (Self_ucurves_no_cut, Simplesweep) there is no solution even after that. Note that these two examples in the paper are described as obtained from industrial source. Lastly, in some cases our method outputs an equation of smaller degree than the actual one: Nested nodal and Simplesweep have actual degree 6. It seems that for implicitization of NURBS patches, approximate solving methods prove to be more reliable, for these parametrizations have floating point coefficients.

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

Curve	Exact			SVD			#impl.
	`NullSpace`	`LinearSolve`	matrix size	time	accuracy (a)	matrix size	monom.
Descartes' Folium	0.016	0.012	5×5	0.012	$1.29 \cdot 10^{-12}$	10×5	3
Tricuspoid	0.076	0.044	15×15	0.028	$6.05 \cdot 10^{-6}$	30×15	8
Talbot's curve	1.625	0.324	28×28	0.132	$8.06 \cdot 10^{-16}$	56×28	8
Nephroid	1.656	0.312	28×28	0.17	$2.31 \cdot 10^{-21}$	56×28	10
Fifth heart	5.3	0.104	33×33	0.124	$1.09 \cdot 10^{-5}$	66×33	43
Trifolium	19.7	0.26	45×45	0.188	$6.37 \cdot 10^{-37}$	90×45	37
Ranunculoid	8414.8	1.376	91×91	2.224	$7.71 \cdot 10^{-6}$	182×91	43

Table 1: Runtimes (sec) and accuracy of approximation for curves.

Surface	Exact		SVD		matrix	# implicit
	`NullSpace`	`LinearSolve`	time	accuracy (a)	size	monomials
Quartoid	0.06	0.036	0.036	$9.72 \cdot 10^{-14}$	16×16	4
Peano	0.028	0.024	0.024	$3.05 \cdot 10^{-14}$	10×10	4
Swallowtail	0.24	0.108	0.096	$1.52 \cdot 10^{-11}$	25×25	6
Sine	2224.5	1.164	0.3	$1.03 \cdot 10^{-5}$	125×125	7
Bohemian dome	2150.23.4	1.181	0.292	$1.68 \cdot 10^{-5}$	125×125	7
Enneper	310.14	0.766	0.42	$8.51 \cdot 10^{-9}$	103×103	23
Bicubic surface	$>$ 4hours	42.059	74.63	$5.69 \cdot 10^{-5}$	715×715	715

Table 2: Runtimes (sec) and accuracy of approximation for surfaces.

5.1 Examples

Example 1 (Folium of Descartes)**.** Let us consider the following curve:

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

The algorithm in [EKP10] yields 3 implicit polytope vertices: $(1,1)$, $(0,3)$, $(3,0)$. This polygon contains 5 lattice points which yield the potential implicit monomials $x_1^3, x_0x_1, x_0x_1^2, x_0^2x_1, x_0^3$ indexing the columns of matrix M in this order. To fill the rows of matrix M, we plug in to each monomial the parametric expressions and evaluate

using 5 random integer τ's: $19, 17, 10, 6, 16$. Then,

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

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

Example 2 (Bicubic surface)**.** We consider the benchmark challenge of the bicubic surface [GV97]:

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

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

The implicit degree in x_0, x_1 is 18, and 9 in x_2. The approach of [EK03] could not handle it because it generates 737129 regular triangulations (by TOPCOM) in a file of 383MB; our method computes the optimal support. The implicit polytope has vertices $(0,0,0), (18,0,0), (0,18,0), (0,0,9)$, and 715 lattice points. The nullvector of matrix M, computed in 42sec, contains 715 non-zero entries which correspond precisely to the actual implicit support.

Example 3 (Hypercone)**.** We illustrate our method on a hypersurface of dimension 4, whose rational parametric representation is:

$$x_0 = \frac{t_3(1-t_1^2)(1-t_2^2)}{(1+t_1^2)(1+t_2^2)},\ x_1 = \frac{2t_3(1-t_1^2)t_2}{(1+t_1^2)(1+t_2^2)},\ x_2 = \frac{2t_3t_1}{1+t_1^2},\ x_3 = t_3. \tag{9}$$

This is an example where the resultant of the system (3) is a multiple of the implicit equation, hence it defines a variety strictly containing the image of the parametrization. In order to facilitate the computation of the predicted support, we set up an equivalent system:

$$\begin{aligned} F_0 = x_0w - t_3(1-t_1^2)(1-t_2^2),\ F_1 = x_1w - 2t_3(1-t_1^2)t_2, \\ F_2 = x_2w - 2t_3t_1(1+t_2^2),\ F_3 = x_3 - t_3,\ F_4 = w - (1+t_2^2)(1+t_1^2), \end{aligned} \tag{10}$$

where w is a new variable. We then compute the support of the system's sparse resultant wrt the t_i's and w. The software in [EFKP12] predicts 4 implicit vertices: $(8,0,0,0), (0,8,0,0), (0,0,8,0), (0,0,0,8)$. Proposition 6, applied to the polynomials defined by the parametric expressions in (9), gives the following degree bounds: total degree ≤ 24, $\deg_{x_0} \leq 4$, $\deg_{x_1} \leq 4$, $\deg_{x_2} \leq 8$, $\deg_{x_3} \leq 16$ which improve the predicted support.

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

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

Example 4. We examine the hypersurface described in [KM00, 4.2].

$$\begin{aligned} x_0 = \frac{1}{(1-t_3)((1/2)t_2^2 + 1/4) + t_3(-(1/2)t_1^2 - 1/4)},\ x_1 = \frac{(1-t_3)((1/2)t_2^2 - 1/4) + t_3(-(1/2)t_1^2 + 1/4)}{(1-t_3)((1/2)t_2^2 + 1/4) + t_3(-(1/2)t_1^2 - 1/4)}, \\ x_2 = \frac{(1-t_3)t_2}{(1-t_3)((1/2)t_2^2 + 1/4) + t_3(-(1/2)t_1^2 - 1/4)},\ x_3 = \frac{t_3t_1}{(1-t_3)((1/2)t_2^2 + 1/4) + t_3(-(1/2)t_1^2 - 1/4)}. \end{aligned} \tag{11}$$

To facilitate the computation of the predicted support, we express the common denominator introducing a new variable w:

$$\begin{aligned} F_0 = x_0w - 1,\ F_1 = x_1w - (1/2)t_2^2 + 1/4 + (1/2)t_3t_2^2 - (1/2)t_3 + (1/2)t_3t_1^2, \\ F_2 = x_2w - t_2 + t_2t_3,\ F_3 = x_3w - t_3t_1,\ F_4 = w - (1/2)t_2^2 + 1/4 + (1/2)t_3t_2^2 - (1/2)t_3 + (1/2)t_3t_1^2. \end{aligned} \tag{12}$$

The predicted implicit polytope has vertices $(0,2,0,4),(6,0,0,0),(2,4,0,0),(0,0,6,0),(0,0,0,6),\ (2,0,0,0)$, $(0,0,4,0),(0,0,0,4)$ and contains 144 lattice points. Proposition 6, when applied to the polynomials defined from the parametric expressions in 11, gives the following degree bounds: total degree ≤ 8, $\deg_{x_i} \leq 8, \forall i = 0,1,2,3$, which do not improve the predicted support. The matrix M has corank 8. Choosing an arbitrary nullvector, the corresponding polynomial is a multiple of the actual implicit equation.

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

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

equivalent to the one in [KM00, sec.4.2]. In Maple 13, our software with input the initial predicted polytope takes 2.2sec, while with the scaled down polytope by 0.75 takes 0.348sec.

Our last example concerns resultant computation. The support prediction software actually computes a resultant support so its straightforward application is to reduce resultant computation to interpolation; this is also the premise of [CD07, Tan07]. The main difference with interpolating the implicit equation is the absence of a parametric form of the resultant. But, this is provided by the parametrization of the resultant hypersurface, known as Horn-Kapranov parametrization [Kap91], illustrated below.

Example 5. Let $f_0 = a_2x^2 + a_1x + a_0,\ f_1 = b_1x^2 + b_0$, with supports $A_0 = \{2,1,0\}, A_1 = \{1,0\}$. Their (Sylvester) resultant is a polynomial in a_2, a_1, a_0, b_1, b_0. The algorithm in [EFKP12] computes its Newton polytope with vertices $(0,2,0,1,1),\ (0,0,2,2,0),\ (2,0,0,0,2)$; it contains 4 points, corresponding to 4 potential monomials $a_1^2b_1b_0,\ a_0^2b_1^2,\ a_2a_0b_1b_0,\ a_2^2b_0^2$. The Horn-Kapranov parametrization of the resultant yields: $a_2 = (2t_1 + t_2)t_3^2t_4,\ a_1 = (-2t_1 - 2t_2)t_3t_4,\ a_0 = t_2t_4,\ b_1 = -t_1t_3^2t_5,\ b_0 = t_1t_5$, where the t_i's are parameters. We substitute these expressions to the predicted monomials, evaluate at 4 sufficiently random t_i's, and obtain a matrix whose kernel vector $(1,1,-2,1)$ yields $\mathcal{R} = a_1^2b_1b_0 + a_0^2b_1^2 - 2a_2a_0b_1b_0 + a_2^2b_0^2$.

The complexity of interpolating resultants is $O^*(|S|^2)$ where S is the set of lattice points in the predicted resultant support, because the dominating stage is a kernel computation for a structured matrix M. Using Weidemann's approach, the main oracle is post-multiplication of M by a vector, which amounts to evaluating a $(n+1)$-variate polynomial at *chosen* points, and this can be done in quasi-linear complexity in $|S|$ [vdHS10, Pan94]. For certain classes of polynomial systems, when one computes the resultant in one or more parameters, this may be competitive to current methods for resultant computation. The best such methods rely on developing the determinant of a resultant matrix in these parameters [CE00, D'A02]. The matrix dimension is in $O^*(t^n \deg \mathcal{R})$ [Emi96], where $\deg \mathcal{R}$ is the total degree of $\mathcal{R}$ in all input coefficients, and t is the scaling factor relating the input Newton polytopes, which is bounded by the maximum degree of the input polynomials f_i in any variable. Then, developing *univariate* resultants has complexity in $O^*(t^{3.5n}(\deg \mathcal{R})^{3.5})$ [Emi96, EP05]. Hence, our approach improves the complexity when the predicted support is small compared to t and $\deg \mathcal{R}$.

5.2 Comparisons to other methods

We report on a comparison of our implementation against existing implicitization software, namely software implementing μ-bases [CSC98] only for curves [BB10], and Maple function `Implicitize()`, which is based on integration of matrix M over each parameter, see [CGKW00] and Section 2.1.

Table 3 summarizes the total time to implicitize a curve, given its parametrization. We used the same algebraic curves as in other tables, grouped by degree; for each degree, the table shows the average runtime. In our experiments, μ-bases yield the fastest runtimes, whereas `Implicitize()` is the slowest of the three when run in exact mode or when the parametrization is rational.

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

Gröbner bases give an effective symbolic method to solve the implicitization problem. Let J be the ideal $J =< \bar{F}_0, \cdots, \bar{F}_n, \bar{F}_{n+1} > \subset \mathbb{C}[t,x]$ generated by the polynomials in (3) and polynomial (4), and let $J_n = J \cap \mathbb{C}[x]$ be the n-th elimination ideal. Then $V(J_n)$ is the smallest variety containing the image of the parametrization. The method computes the Gröbner basis of I wrt to any lexicographic ordering where every t_i is greater than every x_i, then, the elements of this basis not involving t_i form a basis of I_n and define $V(J_n)$. In case this is a principal ideal, then there is a unique element of the Gröbner basis that generates it. This is the implicit equation of the parametric hypersurface and equals the resultant wrt t of the input polynomials.

Example 6. Consider the parametrized Enneper's surface:

$$f_1 := t_1/3 - (1/9)t_1^3 + t_1t_2^2/3,\ f_2 := -t_2/3 + (1/9)t_2^3 - t_1^2t_2/3,\ f_3 := t_1^2/3 - t_2^2/3.$$

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

Table 3: Comparing runtimes (sec) of: Maple function `Implicitize` (exact and numeric), our method (`LinearSolve`, random integers), and μ-bases.

We compute a Gröbner basis G of ideal $I =< x - f_1, y - f_2, z - f_3 >$ with respect to the lexicographic ordering $t_1 > t_2 > x > y > z$. The unique polynomial depending only x, y, z is the implicit equation of this surface:
$128z^7 - 27y^6 + 702x^2y^2z^3 - 9x^4z - 9y^4z - 48x^2z^3 - 64z^5 + 432x^2z^5 + 240x^2z^4 + 135y^4z^3 + 432y^2z^5 - 240y^2z^4 + 27x^6 + 81x^2y^4 - 162y^4z^2 + 144x^2z^6 - 144y^2z^6 - 81x^4y^2 + 135x^4z^3 + 162x^4z^2 - 64z^9 + 18x^2zy^2 - 48z^3y^2$

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

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

Table 4: Comparing runtimes (sec) of Gröbner bases method implemented in Maple and our method (`LinearSolve`, random integers).

6 Numerical issues

This section discusses different ways to evaluate the matrix entries, how to measure the accuracy of approximate implicitization, and compares our measures to Hausdorff distance.

6.1 Point sampling

A central part in our linear system construction is held by the evaluation of matrix M at convenient τ. This section describes our approaches and experimental results.

We have experimented with both 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. complex numbers of modulus equal to 1. The roots of unity when used with approximate methods were evaluated as floats. When examining approximate methods we used the ratio of the last two singular values σ_m/σ_{m-1}, which indicates how close to having corank 1 is matrix M.

Table 5 shows representative timings about these options, which we examined with our implementation on Maple, optimized for the specific task. Our experiments show that runtimes do not vary significantly in small examples but in larger ones, the best results are given by random integers for the exact method, and unitary complexes and roots of unity evaluated as floats for the numeric method, with the former having a slightly better overall performance over the latter, both in terms of stability and speed.

As expected, random integers give matrices which are closer to having corank 1. Note that in Table 5, the Trifolium's matrix M, when computed using random integers, was of corank > 1 so we computed a multiple of its implicit equation. Namely, any kernel vector supplied a multiple of the implicit equation. The degree of the extraneous factor varied depending on the vector chosen.

For the family of Bézier curves, using random integers we obtain better values of the ratio of singular values, compared to other evaluation methods.

Curve	implicit degree	lattice points	SVD (σ_m/σ_{m-1}) root of 1	SVD (σ_m/σ_{m-1}) unitary $\mathbb{C}$	SVD (σ_m/σ_{m-1}) rand.$\mathbb{Z}$	NullSpace rand.$\mathbb{Z}$	LinearSolve rand.$\mathbb{Z}$
Folium	3	5	0.032 (10^{-12})	0.012 (10^{-12})	0.012 (10^{-13})	0.016	0.012
Conchoid	4	10	0.112 (10^{-14})	0.144 (10^{-14})	0.072 (10^{-13})	0.036	0.028
Bean curve	4	13	0.352 (10^{-11})	0.4 (10^{-14})	0.116 (10^{-13})	0.048	0.036
Tricuspoid	4	15	0.356 (10^{-14})	0.028 (10^{-14})	0.168 (10^{-14})	0.076	0.044
Cardioid	4	15	0.28 (10^{-14})	0.29 (10^{-14})	0.2 (10^{-12})	0.068	0.044
Nephroid	6	28	0.248 (10^{-15})	0.17 (10^{-34})	0.192 (10^{-21})	1.656	0.312
Talbot's curve	6	28	0.416 (10^{-14})	0.132 (10^{-17})	0.196 (10^{-16})	1.625	0.324
Trifolium	4	45	0.284 (10^{-34})	0.188 (10^{-77})	0.876 (10^{-20})	19.7	0.26
Fifth heart	8	33	0.224 (10^{-37})	0.124 (10^{-56})	0.63 (10^{-26})	5.3	0.104
Ranunculoid	12	91	3.764 (10^{-359})	2.224 (10^{-58})	79.853 (10^{-2})	8414.8	1.376
Dense Bézier	8	45	6.4 (10^{-169})	5.928 (10^{-150})	5.66 (10^{-151})	16.37	0.152
Sparse Bézier	8	33	2.42 (10^{-101})	2.2 (10^{-102})	2.1 (10^{-112})	2.73	0.12

Table 5: Comparison of matrix evaluation methods. Runtimes on Maple (sec), whereas the parenthesis contains $\sigma_{|S|}/\sigma_{|S|-1}$.

6.2 Accuracy of approximate implicitization

In this section, we evaluate the numeric accuracy, or quality, of the approximate implicit equation obtained by our method, by comparing it to the exact implicit equation.

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

We employ two measures to quantify the accuracy of approximate implicitization:

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

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

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

The approximate implicit equation in all experiments below is obtained using the command `SingularValues()`, where the matrix is instantiated by unitary complex values τ, whereas the exact one is obtained using command `NullSpace()` using random integers. We used several parametric curves and surfaces. The computed approximate implicit equations are given in Tables 8 and respectively 9. The runtimes of approximate and exact methods, and the accuracy of approximation using measure (a) above, are shown in Table 1 and Table 2. These results confirm that SVD can give very good approximations of the actual implicit equation on most inputs.

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

We present some specific examples, using both dense and sparse Bézier curves of varying degree (see Example 7), yielding dense and sparse implicit equations. These are polynomial parametrizations, where the implicit equation is of the same total degree. We compare the runtimes for exact and approximate methods, and the accuracy of the latter using both measures: (a) in Table 6, and (b) in Table 7. Both measures give overall very encouraging results.

Table 6 also juxtaposes the efficiency of our algorithm on dense and sparse Bézier inputs. It appears that we are able to exploit sparseness, since the matrix size is smaller in sparse inputs, and not very far from the actual size of the implicit support. This translates into faster runtimes and better accuracy of the approximate implicit equation. For the dense curve of degree 8, the accuracy of the approximate polynomial is rather large, but may be acceptable given the large norm of the coefficient, namely at least 10^6. Figure A in the Appendix summarizes the accuracy estimation, using criterion (a), for approximating dense Bézier curves. The figure shows the hardness of approximation as the parametric and implicit degree grows.

Implicit degree	SVD		NullSpace		Accuracy (a)		Matrix size (SVD)		# nonzero terms	
	dense	sparse	dense	sparse	dense	sparse	dense	sparse	dense	sparse
4	0.128	0.052	0.128	0.048	$1.14 \cdot 10^{-12}$	$2 \cdot 10^{-24}$	30×15	22×11	15	8
5	0.616	0.192	0.532	0.164	$1.6 \cdot 10^{-5}$	$6.01 \cdot 10^{-22}$	42×21	32×16	21	14
6	2.88	0.368	3.064	0.428	$3.19 \cdot 10^{-4}$	$1.58 \cdot 10^{-11}$	56×28	38×19	28	19
8	5.928	2.2	16.37	2.73	$1.68 \cdot 10^{-6}$	$6.46 \cdot 10^{-4}$	90×45	66×33	45	32

Table 6: Maple runtimes (sec) and accuracy for dense and sparse Bézier curves.

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

Table 7: Accuracy of approximation under measure (b) over 100 sampled points

Example 7. We consider a family of dense and sparse Bézier polynomial curves. Their implicit degree equals the maximum degree of their parametric polynomials. The dense Bézier curve of degree 8 has both parametric polynomials of maximum degree:

$$x(t) = 4t-42t^2+168t^3-385t^4+532t^5-406t^6+140t^7-11t^8,\ y(t) = 1/2-28t^3+105t^4-196t^5+210t^6-120t^7+29t^8.$$

The sparse Bézier curve of degree 8 is missing certain terms, namely one of the parametric polynomials is of lower degree:

$$x(t) = 1 + 112t^3 - 630t^4 + 1344t^5 - 1344t^6 + 592t^7 - 75t^8,\ y(t) = 2 - 16t + 280t^3 - 420t^4 - 392t^5 + 546t^6.$$

Further curves of this family are used in our experiments and are generated in a similar fashion. The corresponding accuracy of approximation and the runtimes are shown in Table 6. Figure A in the Appendix summarizes the approximation accuracy for the dense family.

6.3 Hausdorff distance

The Hausdorff distance is a fundamental tool in measuring the distance between two hypersurfaces. However, its computation is a hard problem. To our knowledge, there is no effective algorithm for this problem in general.

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

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

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

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

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

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

where $S(t,u)$ is the vector dot product $(f(t) - g(u)) \cdot (f(t) - g(u))$. [4] In this case, the computation reduces to solving a nonlinear system, which is quite hard [PM01]. In the case of curves, when the nearest points are both inner points, the system becomes $S'_t(t,u) = S'_u(t,u) = 0$. In [CCW+09, CMXP10], the authors gave effective algorithms to compute the Hausdorff distance for B-spline curves and Bézier curves. Effective algorithms for the computation of the Hausdorff distance of surfaces do not exist.

Suppose V_1, V_2 are given in implicit form, namely $V_i := \{x \in \mathbb{R}^{n+1} : p_i(x) = 0, \quad p_i \in \mathbb{R}[x], i = 1, 2\}$. The computation of $d_H(V_1, V_2)$ seems to be much harder, even if the V_i are curves. In [Jüt00], the author presented a bound and algorithms for the Hausdorff distance of two spline curves C_1, C_2 and algorithms when C_1 is close to C_2. By setting $p(x_1, x_2) := p_1(x_1, x_2) - p_2(x_1, x_2)$, where p_1, p_2 are spline bivariate functions, they have $d_H(C_1, C_2) < cM$, where c is a constant and M is the maximum of absolute coefficients of p. There is no general connection between Hausdorff distance and the coefficients of $p(x_1, x_2)$, though we expect that, if the latter are sufficiently small, then $d_H(C_1, C_2)$ is small. Hence we develop our first method (a) to evaluate the quality of approximate implicitization.

Suppose V_1 is parametrized by $(f_0(t), \ldots, f_n(t))$, $t \in \Omega$, and V_2 is implicit, namely $V_2 = \{x \in \mathbb{R}^{n+1} : p(x) = 0, \ p \in \mathbb{R}[x]\}$. Then $d_H(V_1, V_2)$ is close relation to the values of the function $p(f_0(t), \ldots, f_n(t))$ by the Łojasiewicz inequality [JKS92]. The Łojasiewicz inequality asserts $\exists\, c, \alpha > 0 : d(x, V_2)^\alpha \le c\,|p(x)|$, for every x in the compact domain $\subset \mathbb{R}^{n+1}$. Let $p(x) \in \mathbb{R}[x]$ be an approximate implicit equation of V_1, and $V_2 := \{x \in \mathbb{R}^{n+1} |\, p(x) = 0\}$. Then

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

If $\max|p(f_0(t), \ldots, f_n(t))| \to 0$ then $p(x)$ is a good approximate implicit equation for V_1. Finding the best value for α in terms of the degrees of polynomials is an interesting problem [JKS92]. Computing c in terms of values of p is open and seems very hard [JKS92]. In [Jüt00], if C_1 is a B-spline and $p(x)$ is a spline bivariate function, there is a bound of c supposing C_1 is close enough to C_2.

Our second measure (b) for evaluating the quality of approximate implicitization has followed this approach. Evaluating the exact maximal of $|p(f_0(t), \ldots, f_n(t))|, t \in \Omega$ is complicated if V_2 is a surface. To avoid this, we try to evaluate the maximum absolute value of a function by choosing a large sample of values for the parameters t and find $\max|f(t)|$.

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

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

where $\rho, \sigma : [0,1] \to [0,1]$ range over all continuous and non-decreasing functions (reparametrizations) with $\rho(0) = \sigma(0) = 0$ and $\rho(1) = \sigma(1) = 1$. Obviously, $d_H(C_1, C_2) \le d_F(C_1, C_2)$, but the ratio d_H/d_F is not bounded. It is possible for two curves to have small Hausdorff but large Fréchet distance. In [AKW04], the authors showed that, for closed convex curves, the Hausdorff equals the Fréchet distance, while the latter is $\le \kappa+1$ times the Hausdorff distance for κ-bounded curves. In [BBW08], the authors studied the Fréchet distance for simple polygons but it seems likely that the Fréchet distance between general surfaces is not computable.

Computing the Hausdorff (and Fréchet) distance of the parametric variety and the computed implicit approximation is an interesting and challenging task. Now, we give some illustrative examples.

Example 8. We consider a parametrized curve C_1 of degree 4

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

The exact implicit equation of this curve is : $103 - 81y - 13x - 12x^2 + 2x^3 + x^4$.
Its approximate implicit equation is:

$$p(x,y) := 103.000000000009 - 80.999999999993y - 13.0000000000138x - 11.9999999999934x^2 + 1.99999999999878x^3 + x^{\cdot}4$$

[4] Bernard also suggested our method (b) measure (sometime called algebraic distance). I didn't find any paper that use this distance

Let C_2 be the curve with implicit equation $p(x, y)$. The accuracy of approximation under measures (a) and (b) are: $\approx 2.07562746307138 \cdot 10^{-11}$ and $\approx 9.192 \cdot 10^{-10}$, respectively. Note that the reparametrizations of C_1 and C_2 are respectively

$$f(t) = (t, \frac{103 - 13t - 12t^2 + 2t^3 + t^4}{81}), g(u) := (u, \frac{103.000000000009 - 13.0000000000138u - 11.9999999999934u^2 + 1.99999999999878u^3 + u^4}{80.999999999993})$$

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

Example 9. We consider a parametrized curve C_1 of degree 5

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

The exact implicit equation of this curve is : $-5 + y - 2x - 4xy + 14x^2 + 6x^2y - 10x^3 - 4x^3y + 2x^4 + x^4y$.
Its approximate implicit equation is:
$p(x, y) := -5.00000003073011 + 0.999999998287437y - 1.99999993893286x - 3.99999999565477xy + 13.9999999566394x^2 + 5.99999999627234x^2y - 9.99999998715587x^3 - 3.99999999892032x^3y + 1.99999999866029x^4 + x^4y$.
Let C_2 be the curve with implicit equation $p(x, y)$. The accuracy of approximation under measures (a) and (b) are: $\approx 8.245330463 \cdot 10^{-8}$ and $\approx 1.351124722 \cdot 10^{-9}$, respectively. The reparametrizations of C_1 and C_2 are respectively

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

We compute the Hausdorff distance $d_H(C_1, C_2)$ by evaluating the function $S(t, u) :=< f(t) - g(u), f(t) - g(u) >$, where $(t, u) \in [\frac{3}{2}, 2] \times [\frac{3}{2}, 2]$. We obtain $d_H(C_1, C_2) \leq 1.58529999888284 \cdot 10^{-11}$.

Example 10. We consider a cylinder surface $S_1 : x = t+1,\ y = t^3+3t^2+5t+3,\ z = s$ where $\Omega := (t, s) \in [0, 1] \times [0, 1]$. Its implicit equation is x^3+2x-y and its approximate implicit equation (S_2) is $4.18399 \cdot 10^{-8} - y + 2x + x^3$. The accuracy of approximation under measures (a) and (b) is $\approx 4.18399 \cdot 10^{-8}$.

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

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

we obtain $d_H(S_1, S_2) \leq 1.871141684 \cdot 10^{-8}$.

Example 11. We consider Peano's surface $S_1 : x = t,\ y = s + 1,\ z = -s^2 + s(3t^2 - 2) - 1 + 3t^2 - 2t^4$ where $\Omega := (t, s) \in [0, 1] \times [0, 1]$. Its implicit equation is $z + y^2 - 3x^2y + 2x^4$ and its approximate implicit equation (S_2) is $-3.725155334 \cdot 10^{-8} + z - 4.414233699 \cdot 10^{-10}y + y^2 - 2.879451183 \cdot 10^{-10}x + 1.830328567 \cdot 10^{-12}xy + 2.350959794 \cdot 10^{-11}x^2 - 3x^2y + 5.645612348 \cdot 10^{-14}x^3 + 2x^4$. The accuracy of approximation under measures (a) and (b) are $\approx 3.725528887 \cdot 10^{-8}$ and $\approx 2.983 \cdot 10^{-9}$, respectively. The reparametrizations of S_1 and S_2 are:
$f(t, s) = (t, s, -s^2 + 3t^2s - 2t^4)$, and $g(u, v) = (u, v, -3.725155334 \cdot 10^{-8} + 4.414233699 \cdot 10^{-10}v - v^2 + 2.879451183 \cdot 10^{-10}u - 1.830328567 \cdot 10^{-12}uv - 2.350959794 \cdot 10^{-11}u^2 + 3u^2v - 5.645612348 \cdot 10^{-14}u^3 - 2v^4)$.
To compute the Hausdorff distance $d_H(S_1, S_2)$, we need to evaluate the polynomial

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

We know that $d_H(S_1, S_2) \leq \max_{(t,s) \in [0,1] \times [1,2]} \sqrt{S(t, s; t, s)}$. We compute the maximum value of $S(t, s)$, $(t, s) \in [0, 1] \times [1, 2]$, and obtain $d_H(S_1, S_2) \leq 3.813440008 \cdot 10^{-8}$.

7 Conclusion and further work

In order to tackle large problems we plan to employ state-of-the-art software libraries for matrix operations. For exact computation, LinBox implements asymptotically fast algorithms and seems to be of choice for $\dim M > 100$. For approximate implicitization, we plan to use LAPACK for further examination of numerical stability and, in particular, in order to apply matrix preconditioning. An alternative approach to improve stability is to change from the monomial to the Bernstein basis. Lastly, we may juxtapose least squares as an alternative technique.

An interesting aspect is that matrix M has the structure of quasi-Vandermonde matrices. In particular, multiplying M by a vector v on the right-hand side is equivalent to evaluating a $(n + 1)$-variate polynomial with support S and coefficient vector v at all points defining the rows of M. If this complexity were quasilinear in m (i.e. linear when ignoring polylogarithmic factors in m), then the kernel computation of M should have complexity quasi-quadratic in m, or even faster, when computing only one eigenvector by Lanczos' method.

We have restricted attention to hypersurfaces, but the algorithms discussed in this paper apply to surfaces of codimension ≥ 2, such as space curves. In this case, the generalization of the resultant is the Chow form, and our methods could interpolate this form, thus offering information about the implicit representation of the surface.

It is possible to approximate manifolds given by k parametric pieces with a single implicit equation, by applying SVD on $M^\top = [M_1 \cdots M_k]^\top$, where M_i is the matrix constructed by the algorithms of this paper for the i-th piece, for $i = 1, \ldots, k$. This includes planar curve or surface splines defined by k segments or patches, respectively. We assume the k parametric representations yield implicit polynomials with (roughly) the same Newton polytope, which always happens if the parametric representation of each piece uses polynomials with the same supports. Matrix M is then evaluated over points spanning all k segments or patches.

Acknowledgements I.Z. Emiris, T. Kalinka, and T. Luu Ba are partially supported by Marie-Curie Initial Training Network "SAGA" (ShApes, Geometry, Algebra), FP7-PEOPLE contract PITN-GA-2008-214584. C. Konaxis' research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7-REGPOT-2009-1) under grant agreement n° 245749.

References

[AKW04] H. Alt, C. Knauer, and C. Wenk. Comparison of Distance measures for planar curves. *Algorithmica*, 38:45–58, 2004.

[APJ12] M. Aigner, A. Poteaux, and B. Jüttler. Approximate implicitization of space curves. In U. Langer and P. Paule, editors, *Symbolic and Numeric Computation*. Springer, Vienna, 2012. To appear.

[BB10] L. Busé and T. Luu Ba. Matrix-based implicit representations of algebraic curves and applications. *Computer Aided Geometric Design*, 27(9):681–699, 2010.

[BBW08] K. Buchin, M. Buchin, and C. Wenk. Computing the Fréchet distance between simple polygons. *Comput. Geom.*, 41(1-2):2–20, 2008.

[BD10a] O.J.D. Barrowclough and T. Dokken. Approximate implicitization and approximate null spaces. The 16th Conference of the International Linear Algebra Society (ILAS), Pisa, Italy, 2010.

[BD10b] O.J.D. Barrowclough and T. Dokken. Approximate implicitization of triangular Bézier surfaces. In *Proceedings of the 26th Spring Conference on Computer Graphics*, SCCG '10, pages 133–140, New York, NY, USA, 2010.

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

[BOT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proc. ACM Symp. Theory of Computing*, pages 301–309. ACM Press, New York, 1988.

[BP99] A. Barvinok and J. Pommersheim. An algorithmic theory of lattice points in polyhedra. *Complexity*, 38:91–147, 1999.

[CCW+09] X-D. Chen, L. Chen, Y. Wang, G. Xu, J-H. Yong, and J-C. Paul. Computing the minimum distance between two Bézier curves. *J. Computational Applied Math*, 229:294–301, 2009.

[CD07] M.A. Cueto and A. Dickenstein. Some results on inhomogeneous discriminants. In *Proc. XVI Latin Amer. Algebra Colloq., Bibl. Rev. Mat. Iberoamericana*, pages 41–62, 2007. arXiv:math/0610031v2 [math.AG].

[CE00] J.F. Canny and I.Z. Emiris. A subdivision-based algorithm for the sparse resultant. *J. ACM*, 47(3):417–451, May 2000.

[CGKW00] R.M. Corless, M. Giesbrecht, Ilias S. Kotsireas, and S.M. Watt. Numerical implicitization of parametric hypersurfaces with linear algebra. In *Proc. AISC*, pages 174–183, 2000.

[CKL89] J.F. Canny, E. Kaltofen, and Y. Lakshman. Solving systems of non-linear polynomial equations faster. In *Proc. ACM Intern. Symp. on Symbolic & Algebraic Comput.*, pages 121–128, 1989.

[CLO98] D.A. Cox, J.B. Little, and D. O'Shea. *Using Algebraic Geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer-Verlag, NY, 1998.

[CMXP10] X-D. Chen, W. Ma, G. Xu, and J-C. Paul. Computing the Hausdorff distance between two B-spline curves. *Computer-Aided Design*, 42:1197–1206, 2010.

[CSC98] D.A. Cox, T.W. Sederberg, and F. Chen. The moving line ideal basis of planar rational curves. *Comput. Aided Geom. Design*, 15 (8):803–827, 1998.

[CTY10] M.A. Cueto, E.A. Tobis, and J. Yu. An implicitization challenge for binary factor analysis. *J. Symbolic Computation*, 45(12):1296–1315, 2010.

[Cue10] M.A. Cueto. *Tropical Implicitization.* PhD thesis, Dept Mathematics, UC Berkeley, 2010.

[D'A02] C. D'Andrea. Macaulay-style formulas for the sparse resultant. *Trans. of the AMS*, 354:2595–2629, 2002.

[DFS07] A. Dickenstein, E.M. Feichtner, and B. Sturmfels. Tropical discriminants. *J. AMS*, pages 1111–1133, 2007.

[DS10] C. D'Andrea and M. Sombra. The Newton polygon of a rational plane curve. *Math. in Computer Science*, 4(1):3–24, 2010.

[DT03] T. Dokken and J.B. Thomassen. Overview of approximate implicitization. *Topics in algebraic geometry and geometric modeling*, 334:169–184, 2003.

[EFKP12] I.Z. Emiris, V. Fisikopoulos, C. Konaxis, and L. Peñaranda. An output-sensitive algorithm for computing projections of resultant polytopes. Proc. Symp. Comput. Geom. To appear and also arXiv:1108.5985v2 [cs.SC], 2012.

[EK03] 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.

[EK06] A. Esterov and A. Khovanskiĭ. Elimination theory and newton polytopes. arXiv:0611107[math], 2006.

[EKK11] I.Z. Emiris, T. Kalinka, and C. Konaxis. Implicitization of curves and surfaces using predicted support. In *Electr. Proc. Inter. Works. Symbolic-Numeric Computation*, San Jose, Calif., 2011.

[EKP10] I.Z. Emiris, C. Konaxis, and L. Palios. Computing the Newton polygon of the implicit equation. *Mathematics in Computer Science, Special Issue on Computational Geometry and Computer-Aided Design*, 4(1):25–44, 2010.

[Emi96] I.Z. Emiris. On the complexity of sparse elimination. *J. Complexity*, 12:134–166, 1996.

[EP02] 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.

[EP05] I.Z. Emiris and V.Y. Pan. Improved algorithms for computing determinants and resultants. *J. Complexity, Special Issue*, 21:43–71, 2005. Special Issue on FOCM-02.

[GKZ94] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, Resultants and Multidimensional Determinants.* Birkhäuser, Boston, 1994.

[GV97] L. Gonzalez-Vega. Implicitization of parametric curves and surfaces by using multidimensional Newton formulae. *J. Symbolic Comput.*, 23(2-3):137–151, 1997. Parametric algebraic curves and applications (Albuquerque, NM, 1995).

[JKS92] S. Ji, J. Kollár, and B. Shiffman. A global Łojasiewicz inequality for algebraic varieties. *Trans. Am. Math. Soc.*, 329(2):813–818, 1992.

[Jüt00] B. Jüttler. Bounding the Hausdorff distance of implicitly defined and/or parametric curves. In T. Lyche and L.L. Schumaker, editors, *Math. Methods in CAGD*, pages 1–10. 2000.

[JY11] A. Jensen and J. Yu. Computing tropical resultants. arXiv:1109.2368v1[math.AG], 2011.

[Kap91] M. M. Kapranov. A characterization of A-discriminantal hypersurfaces in terms of the logarithmic Gauss map. *Mathematische Annalen*, 290:277–285, 1991.

[KL89] E. Kaltofen and Y. Lakshman. Improved sparse multivariate polynomial interpolation algorithms. In P. Gianni, editor, *Proc. ACM Intern. Symp. on Symbolic & Algebraic Comput. 1988*, volume 358 of *Lect. Notes in Comp. Science*, pages 467–474. Springer-Verlag, 1989.

[KL03] I.S. Kotsireas and E.S.C. Lau. Implicitization of polynomial curves. In *Proc. ASCM*, pages 217–226, Beijing, 2003.

[KM00] R. Krasauskas and C. Mäurer. Studying cyclides with Laguerre geometry. *Comput. Aided Geom. Des.*, 17(2):101–126, 2000.

[LHH+03] J.A. De Loera, D. Haws, R. Hemmecke, P. Huggins, J. Tauzer, and R. Yoshida. A user's guide for latte v1.1. Software package LattE is available at http://www.math.ucdavis.edu/~latte/, 2003.

[MM02] A. Marco and J.J. Martinez. Implicitization of rational surfaces by means of polynomial interpolation. *CAGD*, 19:327–344, 2002.

[Pan94] V.Y. Pan. Simple multivariate polynomial multiplication. *J. Symb. Comp.*, 18:183–186, 1994.

[PM01] N. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer-Verlag, New York, 2001.

[Ram02] J. Rambau. TOPCOM: Triangulations of point configurations and oriented matroids. In Arjeh M. Cohen, Xiao-Shan Gao, and Nobuki Takayama, editors, *Intern. Conf. Math. Software*, pages 330–340. World Scientific, 2002.

[Sau04] T. Sauer. Lagrange interpolation on subgrids of tensor product grids. *Math. of Comput.*, 73:181–190, January 2004.

[SJ08] M. Shalaby and B. Jüttler. Approximate implicitization of space curves and of surfaces of revolution. In R. Piene and B. Jüttler, editors, *Algebraic Geometry and Geometric Modeling*, pages 215–228. Springer, 2008.

[Stu94] B. Sturmfels. On the Newton polytope of the resultant. *J. Algebraic Combin.*, 3:207–236, 1994.

[STW+06] M. Shalaby, J. Thomassen, E. Wurm, T. Dokken, and B. Jüttler. Piecewise approximate implicitization: Experiments using industrial data. In B. Mourrain, M. Elkadi, and R. Piene, editors, *Algebraic Geometry and Geometric Modeling*, pages 37–52. Springer, 2006.

[STY07] B. Sturmfels, J. Tevelev, and J. Yu. The Newton polytope of the implicit equation. *Moscow Math. J.*, 7(2), 2007.

[SY94] B. Sturmfels and J.T. Yu. Minimal polynomials and sparse resultants. In F. Orecchia and L. Chiantini, editors, *Proc. Zero-dimensional schemes (Ravello, 1992)*, pages 317–324. De Gruyter, 1994.

[SY08] 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.

[Tan07] S. Tanabe. On Horn-Kapranov uniformisation of the discriminantal loci. *Adv. Studies Pure Math.*, 46:223–249, 2007.

[vdHS10] J. van der Hoeven and E. Schost. Multi-point evaluation in higher dimensions. Technical Report 00477658, HAL, 2010.

[Vol97] E. Volcheck. On computing the dual of a plane algebraic curve. In *Proc. ISSAC*, pages 356–358, 1997.

[WTJD04] E. Wurm, J.B. Thomassen, B. Juettler, and T. Dokken. Comparative benchmarking of methods for approximate implicitization. In M. Neamtu and M. Lucian, editors, *Geometric Modeling and Computing, Seattle 2003*, pages 537–548. Nashboro Press, 2004.

[Zip90] R. Zippel. Interpolating polynomials from their values. *J. Symbolic Computation*, 9:375–403, 1990.

[Zip93] R. Zippel. *Effective Polynomial Computation*. Kluwer Academic Publishers, Boston, 1993.

A Appendix

Curve	Parametric form	Exact implicit polynomial	Approximate implicit polynomial
Nephroid	$\frac{-(-1+t^2)(1+10t^2+t^4)}{(1+t^2)^3}$, $\frac{32t^3}{(1+t^2)^3}$	$-64-60y^2-12y^4+y^6$ $+48x^2-24x^2y^2+3x^2y^4-12x^4$ $+3x^4y^2+x^6$	$-1.81402.10^{-5}x^2y-23.99999x^2y^2$ $-11.99999x^4-59.99999y^2$ $-2.17512.10^{-5}y^3-11.99999y^4$ $+47.99999x^2+3x^4y^2+y^6$ $+x^6-63.99999+3x^2y^4$
Talbot's curve	$\frac{-(1+6t^2+t^4)(-1+t^2)}{(1+t^2)^3}$, $\frac{-2t(1-2t^2+t^4)}{(1+t^2)^3}$	$x^6+3y^2x^4-x^4+3y^4x^2$ $-20y^2x^2+y^6+8y^4+16y^2$	$0.00318649y+0.0019125x^2y^3$ $+2.999989x^2y^4+0.00010255x^4y$ $+2.999994x^4y^2-0.004140395x^2y$ $-20.00909x^2y^2+15.9979y^2$ $+0.0008008y^3+7.999837y^4$ $+0.00079661x^2+0.9999945y^6+x^6$ $-1.0001x^4+0.8906879.10^{-3}y^5$
Tricuspoid	$\frac{-t^4-6t^2+3}{(1+t^2)^2}, \frac{8t^3}{(1+t^2)^2}$	$-27+18y^2+y^4+24xy^2+$ $+18x^2+2x^2y^2-8x^3+x^4$	$-27+17.99999y^2+0.99999y^4$ $+23.99999xy^2+18x^2+x^4$ $+1.99999x^2y^2-8x^3$
Ranunculoid	$-\frac{1-1092t^6+423t^8-54t^{10}}{(1+t^2)^6}$ $+\frac{13t^{12}-102t^2+363t^4}{(1+t^2)^6}$, $\frac{8t^3(-29+108t^2-78t^4+44t^6+3t^8)}{(1+t^2)^6}$	$-52521875-1286250x^2-1286250y^2$ $-32025(x^2+y^2)^2+93312x^5$ $-933120x^3y^2+466560xy^4$ $-812(x^2+y^2)^3-21(x^2+y^2)^4$ $-42(x^2+y^2)^5+(x^2+y^2)^6$	

Table 8: Parametric, implicit, and approximate implicit representation of curves; for the latter, we do not show coefficients of absolute value $< 10^{-6}$.

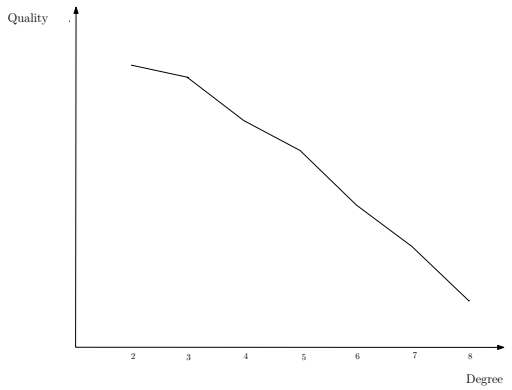

Figure 2: Accuracy of implicitization of dense Bézier curves

Surface	Parametric form	Exact implicit polynomial	Approximate implicit polynomial
Quartoid	$t, s, -(t^2+s^2)^2$	$z + x^4 + 2x^2y^2 + y^4$	$z + x^4 + 2x^2y^2 + y^4$
Sine surface	$\frac{2t}{1+t^2}$, $\frac{2s}{1+s^2}$, $\frac{2s+2t-2st^2-2ts^2}{1+s^2+t^2+s^2t^2}$	$-2y^2z^2 + 4x^2y^2z^2 - 2x^2y^2$ $-2x^2z^2 + z^4 + y^4 + x^4$	$-0.23681.10^{-5}y + 0.20275.10^{-5}x$ $-0.35873.10^{-5}x^2y^3 + 0.18891.10^{-5}x^2y$ $-0.58171.10^{-5}x^3y + 0.55752.10^{-5}xy$ $-0.98381.10^{-5}xy^2 + 0.22139.10^{-5}xy^3$ $-2x^2y^2 - 2y^2z^2 - 2x^2z^2$ $-0.59775.10^{-5}x^2 + 4x^2y^2z^2 + z^4$ $+0.20153.10^{-5}y^2 + 0.2.10^{-4}Ix^2z^2$ $+0.2.10^{-4}Ix^2y^2 + 0.2.10^{-4}Iy^2z^2$ $+0.42669.10^{-5}x^3 + x^4 + y^4$
Bohemian dome	$\frac{1-t^2}{1+t^2}$, $\frac{1+2t+t^2-s^2-s^2t^2+2ts^2}{1+s^2+t^2+s^2t^2}$, $\frac{2s}{1+s^2}$	$2x^2y^2 - 2x^2z^2 - 4y^2$ $+x^4 + z^4 + 2y^2z^2 + y^4$	$1.9999x^2y^2 + 1.9999y^2z^2$ $+z^4 + y^4 - 3.9999y^2$ $+x^4 - 1.9999x^2z^2$
Swallowtail surface	$ts^2 + 3s^4,\ -2ts - 3s^3,\ t$	$-15xy^2z + 3y^4 + y^2z^3$ $-4xz^4 + 12x^2z^2 - 9x^3$	$-4xz^4 - 10^{-4}Ix^3 + y^2z^3$ $+10^{-4}Ixz^4 + 12x^2z^2$ $-8.9999x^3 - 15xy^2z$ $+10^{-4}Ix^2z^2 + 2.9999y^4$
Enneper's surface	$\frac{t}{3} - \frac{t^3}{9} + \frac{ts^2}{3}$, $-\frac{s}{3} + \frac{s^3}{9} - \frac{t^2s}{3}$, $\frac{t^2}{3} - \frac{s^2}{3}$	$-64z^5 + 128z^7 - 64z^9$ $-48y^2z^3 - 240y^2z^4 + 432y^2z^5$ $-144y^2z^6 - 9y^4z - 162y^4z^2$ $+135y^4z^3 - 27y^6 - 48x^2z^3$ $+240x^2z^4 + 432x^2z^5 + 144x^2z^6$ $+18x^2y^2z + 702x^2y^2z^3$ $+81x^2y^4 - 9x^4z + 162x^4z^2$ $+135x^4z^3 - 81x^4y^2 + 27x^6$	

Table 9: Parametric, implicit, and approximate implicit representation of surfaces; for the latter, we do not show coefficients of absolute value $< 10^{-6}$.

Surface	degree	matrix size	LinearSolve time	SVD				"raw"
				time	accuracy	"raw" time	"raw" accuracy	matrix size
Looped patch	3	6×6	0,012	0.012	$6.24 \cdot 10^{-8}$	0.048	$3.53 \cdot 10^{-6}$	23×23
Nested nodal	5	139×139	1.904	fail		fail		319×319
Quartic	4	22×22	0,088	0.864	$4.46 \cdot 10^{-7}$	36.447	$2.07 \cdot 10^{-6}$	71×71
Self_ucurves_no_cut		129×129	no solution	fail		fail		158×158
Simplesweep	4	36×36	no solution	5.337	$6.23 \cdot 10^{-9}$	fail		2457×2457

Table 10: Runtimes (sec) of NURBS patches: exact and numerical interpolation ("raw" refers to parametric equation derived from NURBS representration where terms with coeefficients $< 10^{-2}$ are not removed).