# Implicit representations of high-codimension varieties

Ioannis Z. Emiris, Christos Konaxis, Clément Laroche

# Implicit representations of high-codimension varieties

Ioannis Z. Emiris[a,b], Christos Konaxis[a,b], Clément Laroche[a,b]

[a] *Department of Informatics & Telecommunications, National Kapodistrian University of Athens, Greece*
[b] *ATHENA Research Center, Maroussi, Greece*

**Abstract**

Implicitization usually focuses on plane curves and (hyper)surfaces, in other words, varieties of codimension 1. In this paper we shift the focus on space curves and, more generally, on varieties of codimension larger than 1, and discuss approaches that are not sensitive to base points.

Our first contribution is a direct generalization of an implicitization method based on interpolation matrices for objects of high codimension given parametrically or as point clouds. Our result shows the completeness of this approach which, furthermore, reduces geometric operations and predicates to linear algebra computations.

Our second, and main contribution is an implicitization method of parametric space curves and varieties of codimension $> 1$, which exploits the theory of Chow forms to obtain the equations of conical (hyper)surfaces intersecting precisely at the given object. We design a new, practical, randomized algorithm that always produces correct output but possibly with a non-minimal number of surfaces. For space curves, which is the most common case, our algorithm returns 3 surfaces whose polynomials are of near-optimal degree; moreover, computation reduces to a Sylvester resultant. We illustrate our algorithm through a series of examples and compare our Maple code with other methods implemented in Maple. Our prototype is not faster but yields fewer equations and is more robust than Maple's `implicitize`. Although not optimized, it is comparable with Gröbner bases and matrix representations derived from syzygies, for degrees up to 6.

*Keywords:* Implicitization, space curve, Chow form, resultant, conical hypersurface, randomized algorithm, Maple implementation

## 1. Introduction

In manipulating geometric objects, it is essential to possess robust algorithms for changing representation. This paper considers two fundamental representa-

*Email addresses:* `emiris@di.uoa.gr` (Ioannis Z. Emiris), `ckonaxis@di.uoa.gr` (Christos Konaxis), `claroche@di.uoa.gr` (Clément Laroche)

tions and, in particular, switching to implicit representation from the parametric representation, including the case of base points. One of our methods also treats point cloud representations as input. While many algorithms allow to switch representation of hypersurfaces of codimension 1, our algorithms construct implicit equations expressed as matrix determinants for parametric curves and (hyper)surfaces of codimension $> 1$. More precisely, for varieties of higher codimension, one cannot expect a single implicit equation. Instead, we compute a number of implicit (hyper)surfaces whose set-theoretic intersection equals the given variety. For instance, rational space curves are defined as the intersection of two or, at most, three implicit surfaces.

Implicit representations of geometric objects allow to answer to the membership and intersection problems. With such a representation, one can check easily whether given points are lying on the object or not. They are used for applications requiring geometric boolean operations such as intersections of several objects, differences and, in particular, use ray shooting and ray tracing methods.

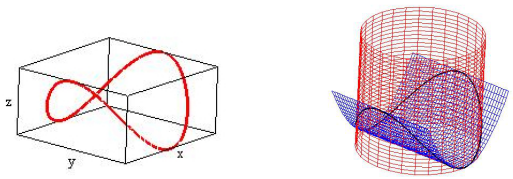



Figure 1: Left: The space curve of Example 3. Right: The two implicit surfaces defining the curve.

We introduce two implicitization methods for varieties of codimension higher than 1. Our first method is a direct generalization of interpolation matrices for objects given parametrically, but also as point clouds. Interpolation matrix methods usually are very simple but, since they can accept inputs represented as point clouds or can be used for approximate implicitization (as in [9]), they provide a very relevant, alternative representation especially for industrial applications today. In that context, interpolation matrices are sometimes called collocation matrices. The construction of the interpolation matrix representing the object reduces geometric operations to linear algebra.

The second implicitization method, and our main contribution, is the following. Given a variety in parametric form, and starting with the powerful and classic theory of Chow forms, which generalizes resultant theory, we design an original, practical algorithm that uses randomization in order to compute implicit hypersurfaces containing the variety. Computing the Chow form is expensive in the general case; high complexity is one reason why very few implementations exist.

Our algorithm cannot guarantee to yield the optimal implicit equations but is reasonably fast and usually its output is very close to being optimal. For space curves, our method yields 3 implicit surfaces defining the curve set-theoretically,

which is the minimum number of equations in general. The degree of the defining polynomials is close to being optimal. The output given by our method possesses strong geometric structure: all computed implicit equations correspond to conical (hyper)surfaces. The question is eventually reduced to the standard question of computing a (sparse) resultant; it is the complexity bottleneck of the algorithm.

The method generalizes for any variety of codimension higher than 1, even in the presence of base points or self-intersections. In particular, self-intersections and base points do not affect the degree of the output equations. We produce a number of hypersurfaces each containing the variety set-theoretically and whose degree is bounded in terms of mixed volume. The number of such hypersurfaces that define the variety set-theoretically is unknown in the general case but experiments indicate that $n+1$ are sufficient, where $n$ is the ambient dimension. This complies with a result by Kronecker [17] but uses one more hypersurface than the optimal number $n$ [16, Ch.V]. The specific output of our algorithm makes it difficult to exploit this result.

We implement our algorithm in Maple and Sage, illustrate it through a series of examples and compare our code with other methods implemented in Maple. We provide a comparison between various existing methods and ours in Table 1, specifying advantages and disadvantages of each method. Our prototype is not faster but yields fewer equations and is more robust than Maple's native command `implicitize`. Although not optimized, for curves in any ambient dimension its speed is comparable to Gröbner bases and matrix representations derived from syzygies, for degrees up to 6.

This paper extends certain results and ideas presented in preliminary form in the second part (Section 4) of [13].

The rest of the paper is organized as follows: The next section sketches the most relevant existing work. Section 3 generalizes interpolation matrices to codimension $> 1$. Sections 4 and 5 employ the Chow form to handle higher-codimension parametric varieties in 3 and higher dimension. Section 6 examines experimentally the performance of our methods and their implementation, and compares them to other methods of implicitization. We conclude with future work and open questions.

## 2. Preliminaries and previous work

This section introduces basic concepts and sketches the most relevant existing work.

Unless specified otherwise, we work in the projective or affine spaces of dimension $n$ over the complex field. A homogeneous parameterization of a projective variety $V$ of dimension $d$ is given as

$$\mathbb{P}^d \to \mathbb{P}^n$$
$$t = (t_0 : \cdots : t_d) \mapsto (F_0(t) : \cdots : F_n(t))$$

where $F_j$, $j = 0, \ldots, n$ is a homogeneous polynomial of some unique degree $\delta$. A rational parameterization is obtained by dehomogenization of a homogeneous parameterization:

$$f_j(t_1, \ldots, t_d) = \frac{F_j(1, t_1, \ldots, t_d)}{F_0(1, t_1, \ldots, t_d)},\ j = 1, \ldots, n.$$

There is no difficulty in switching between rational and homogeneous parameterizations but the latter deals with the points at infinity. Most of the time, we work in the projective space with homogeneous parameterization. For simplicity, we assume that $t_0$ is not a common factor of the polynomials $F_j$ hence $\delta = \max\{\deg(F_j(t)_{|t_0=1}),\ j = 0, \ldots, n\}$. We call $\delta$ the degree of the parameterization (both rational and homogeneous).

In general, approaches to implicitization include resultants, Gröbner bases, moving lines and surfaces (based on the theory of syzygies), and interpolation techniques. Resultants, and their matrix formulae, have led to practical methods to express the implicit surface equation, e.g., in [19, 18], but require the assumption that no base points exist.

Here, the resultant (resp. homogeneous resultant), without any more precision, should be understood as in the following:

**Definition 1.** *The resultant is the abstract map $R$ of $d+1$ polynomials $h_0, \ldots, h_d$ in $d$ variables with symbolic coefficients (resp. homogeneous polynomials in $d+1$ variables) satisfying:*

1. *$R$ is a homogeneous polynomial of degree $\prod_{i' \neq i} \deg h_{i'}$ in the symbolic coefficients of $h_i$ and*
2. *$R$ vanishes if and only if the polynomials $h_i$ have a common root (resp. a non-trivial common root).*

Such a map is known to exist and is unique up to a non-zero constant factor (see e.g. [14]). Its computation may vary: if $d = 1$, it is the determinant of the Sylvester, Bézout or hybrid matrices. In general it can be computed as the quotient of the determinant of the Macaulay or of the sparse resultant matrix divided by the determinant of one of its minors (cf. [8]).

Implicit matrix representations are quite robust, since they do not require developing the implicit equation; instead, they reduce geometric operations on the object to matrix algebra. This is the case of interpolation matrices below, but also of the method based on the theory of syzygies, e.g., [21, 2]. It handles base points and yields matrices whose entries are polynomials in the implicit variables and they indirectly represent implicit objects: their rank drops exactly on the curve or surface. Most of the time, the entries of matrices constructed with such techniques are linear in the implicit variables, though there exists methods for constructing implicit matrices with quadratic entries instead. They allow for geometric operations, such as surface-surface intersection [3] and, more recently, ray shooting [23], to be executed by linear algebra. The advantage of these matrices is that they are much smaller than interpolation matrices, and allow for inversion by solving an eigenproblem on them.

*2.1. Chow forms*

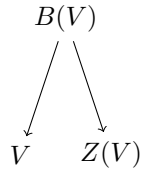
Chow forms have been studied in computer algebra, in particular for varieties of codimension $\geqslant 2$, since they provide a method to describe the variety by a single polynomial [6, 14]. The Chow form of a variety $V$ is basically a polynomial $R_V$ which indicates when linear subspaces intersect $V$. For example, the Chow form of a space curve in projective 3-dimensional space is a polynomial in the indeterminates $u_{ij}$ that vanishes whenever the planes

$$\begin{aligned} H_0 &= u_{00}x_0 + u_{01}x_1 + u_{02}x_2 + u_{03}x_3 = 0, \\ H_1 &= u_{10}x_0 + u_{11}x_1 + u_{12}x_2 + u_{13}x_3 = 0, \end{aligned} \tag{1}$$

intersect on the curve. If the space curve is given parametrically, the Chow form represents the variety in terms of $R_V$. It can be computed by a *symbolic resultant* of the system of linear equations (1) where the set of variables $X = (x_i)_i$ is substituted with the parametric equations; the resultant eliminates the parameters and yields a polynomial in the variables $U = (u_i)_i$. The implicit hypersurfaces in $X$ containing the variety have to be extracted through *rewriting rules*. These make implicitization algorithms that rely on the computation of $R_V$ impractical for varieties of high degree and/or dimension.

Due to their complexity, very few implementations exist for computing the Chow forms themselves. Among them, [24] is an implementation in Macaulay2, based on the formula of the Chow form in the Grassmannian space using the Plücker coordinates, and [15, Subroutine 7] is an algorithm using polynomial ring tools and based on a Poisson-like formula of the Chow form.

To formally define the Chow form let $Gr(k+1, n+1)$ denote the Grassmannian space of $k$-dimensional linear projective subspaces of $\mathbb{P}^n$. For a variety $V \subset \mathbb{P}^n$ of codimension $c$, let $B(V) \subset \mathbb{P}^n \times Gr(c, n+1)$ be the set of $(p, L)$ such that $p$ belongs both to $V$ and to the projective linear subspace $L$ of dimension $c-1$. Then we obtain $V$ by forgetting the second component in $B(V)$ and we obtain an hypersurface $Z(V) \coloneqq \{L \in Gr(c, n+1) \mid L \cap V \neq \emptyset\}$ of the Grassmannian space $Gr(c, n+1)$ by forgetting the first component in $B(V)$.




$Z(V)$ is called the *Chow variety* of $V$ and has the advantage of being an hypersurface in the Grassmannian space, so it is determined by a unique implicit equation up to a constant factor: the Chow form $R_V$. Despite being determined by a unique equation, $Z(V)$ describes the variety $V$ of unconstrained (co)dimension, see Proposition 1. Note that when $V$ is a variety of codimension 1, we have $Z(V) \simeq V$; this explains why the theory of Chow form is effective only for codimension $c > 1$. On the other hand, the Chow form of a zero-dimensional variety $V = \{v_1, \ldots, v_k\}$ is also known as the u-resultant.

**Definition 2.** *Let $V \subset \mathbb{P}^n$ be a $d$-dimensional irreducible variety and $H_0, \ldots, H_d$ be linear forms where*

$$H_i = u_{i0}x_0 + \cdots + u_{in}x_n, \quad i = 0, \ldots, d \tag{2}$$

*and $u_{ij}$ are new variables, $0 \leqslant i \leqslant d$, $0 \leqslant j \leqslant n$. The Chow form $R_V$ of $V$ is a polynomial in the variables $u_{ij}$ such that*

$$R_V(u_{ij}) = 0 \Leftrightarrow V \cap \{H_0 = 0, \ldots, H_d = 0\} \neq \emptyset.$$

The intersection of the $d+1$ hyperplanes $H_i$ defined in equation (2) is generically a $(n-d-1)$-dimensional linear subspace $L$ of $\mathbb{P}^n$, i.e., an element of the Grassmannian $Gr(n-d, n+1) = Gr(c, n+1)$, where $c$ is the codimension of $V$.

**Proposition 1.** [14, Prop.2.5,p.102] *A $d$-dimensional irreducible subvariety $V \subset \mathbb{P}^n$ is uniquely determined by its Chow form. More precisely, a point $\xi \in \mathbb{P}^n$ lies in $V$ if and only if any $(n-d-1)$-dimensional plane containing $\xi$ belongs to the Chow variety $Z(V)$ defined by $R_V$.*

One standard way to compute the Chow form would be to proceed as follows. Consider a variety $V$ as in Proposition 1, parameterized as

$$x_j = F_j(t),\ j = 0, \ldots, n,\ t = (t_0 : \cdots : t_d),$$

and $d+1$ hyperplanes $H_0 = \cdots = H_d = 0$, where $H_i$ is defined as in equation (2). Substituting $x_j = F_j(t)$ in every equation $H_i = 0$, we would obtain an overdetermined system of equations in the parameters $t$. We would then saturate these equations by the parameterization polynomials $F_j(t)$, $j = 0, \ldots, n$, that is, removing the common solutions of the parameterization (base points) e.g. by using a Gröbner basis method. This step can be ignored if $V$ has no base point (e.g. $V$ is a curve and $\gcd(F_0, \ldots, F_n) = 1$). Then we could eliminate the parameters $t$ by using resultants. This reduces the computation of any Chow form to the computation of a resultant:

**Corollary 2.** *Consider any $V \subset \mathbb{P}^n$ of dimension $d$, with parameterization $x_j = F_j(t)$, $j = 0, \ldots, n$. Then, the Chow form $R_V$ is the resultant of the hyperplane equations $H_i$ ($0 \leqslant i \leqslant d$), where one eliminates $t$.*

Here the resultant should be understood, in the sense of Definition 1, as a polynomial that eliminates the variables of the input equations. In other words, while the variables of $R_V$ are $u_{ij}$, its coefficients are polynomials in the coefficients of the parametric homogeneous functions $F_j$.

The coordinates used to represent points in the Grassmannian, and, hence, to describe $R_V$, are most commonly defined as the maximal minors of the $(d+1)\times(n+1)$ matrix whose rows are the normals to the hyperplanes $H_i$. These are known as *Plücker coordinates* or *brackets* and are the variables of $R_V$. Brackets are denoted as $[j_0, j_1, \ldots, j_d]$, where indices correspond to columns of the matrix. Equivalently, the *dual Plücker coordinates* or *dual brackets* can be used; these

are the maximal minors of a $(n-d)\times(n+1)$ matrix whose rows are $n-d$ points that span the intersection $L$ of the hyperplanes $H_i$. Dual brackets are denoted as $[[j_0, j_1, \ldots, j_{n-d-1}]]$, where indices correspond to columns of the matrix. Brackets and dual brackets with complementary index sets are equal up to sign. There are algorithms to recover the implicit or affine equations of hypersurfaces intersecting on $V$ from $R_V$ [14], [25].

Assuming that $R_V$ is a polynomial in the Plücker coordinates, to obtain a representation for $V$ as intersection of implicit hypersurfaces from its Chow form, one may apply a rewriting method. There are two such methods, namely [14, Cor.2.6,p.102], and [6, Prop.3.1], see also [25]. They are not straightforward procedures and, in the case of implicitization, typically yield more implicit polynomials than necessary. However, all implicit polynomials in $X$ are of optimal degree, equal to the degree of $V$.

To illustrate the approach in [6], let us focus on varieties of codimension 2 in $\mathbb{P}^3$, i.e., space curves. Consider the planes $H_0, H_1$ in equation (1). The Chow form $R_V$ is a polynomial in the brackets $[j_0, j_1]$, where $[j_0, j_1]$ denotes the maximal minor indexed by the columns $0 \leqslant j_0, j_1 \leqslant 3$, of the matrix

$$U \coloneqq \begin{bmatrix} u_{00} & u_{01} & u_{02} & u_{03} \\ u_{10} & u_{11} & u_{12} & u_{13} \end{bmatrix}.$$

$R_V$ is then rewritten as a polynomial in the dual brackets $[[j_0, j_1]]$, using the relations: $[0,1] = [[2,3]]$, $[0,2] = -[[1,3]]$, $[0,3] = [[1,2]]$, $[1,2] = [[0,3]]$, $[1,3] = -[[0,2]]$, $[2,3] = [[0,1]]$. The dual brackets are then substituted by the determinant $u_{0j_0}u_{1j_1} - u_{0j_1}u_{1j_0}$ of the corresponding minor of $U$. Finally, the result is expanded as a polynomial whose variables are polynomials in the $u_{10}, u_{11}, u_{12}, u_{13}$ and its coefficients are polynomials in the $u_{00}, u_{01}, u_{02}, u_{03}$. The latter polynomials are all of degree equal to the degree of $V$ and form a system of implicit equations of $V$.

**Example 1.** *As an example, the Chow form of the twisted cubic curve (see also Example 2) with parameterization*

$$(x_0 : x_1 : x_2 : x_3) = (s^3 : s^2t : st^2 : t^3), \quad (s,t) \in \mathbb{P}^1, \tag{3}$$

*is given by the following determinant in the primal brackets:*

$$\det \begin{bmatrix} [0,1] & [0,2] & [0,3] \\ [0,2] & [0,3]+[1,2] & [1,3] \\ [0,3] & [1,3] & [2,3] \end{bmatrix},$$

*which is also known as the Bézout resultant of the system of equations* (1) *where we have substituted the $X$ variables with the parameterization in* (3)*. Rewriting this determinant in the dual brackets we obtain:*

$$\det \begin{bmatrix} [[2,3]] & -[[1,3]] & [[1,2]] \\ -[[1,3]] & [[1,2]]+[[0,3]] & -[[0,2]] \\ [[1,2]] & -[[0,2]] & [[0,1]] \end{bmatrix}$$

*Substituting the dual brackets by the determinant of the corresponding minors of $U$ and collecting the terms in the variables $\{u_{0j_0}\}$ and $\{u_{1j_1}\}$, we obtain the Chow form of the twisted cubic:*

$$
\begin{aligned}
&(u_{02}^2u_{03}-u_{01}u_{03}^2)u_{10}^2u_{12}+(u_{01}u_{02}u_{03}-u_{02}^3)u_{10}^2u_{13}+(u_{01}u_{03}^2-u_{02}^2u_{03})u_{10}u_{11}^2\\
&+(u_{00}u_{03}^2-u_{01}u_{02}u_{03})u_{10}u_{11}u_{12}+(3\,u_{01}u_{02}^2-2\,u_{01}^2u_{03}-u_{00}u_{02}u_{03})u_{10}u_{11}u_{13}\\
&+(u_{00}u_{01}u_{03}-2\,u_{00}u_{02}^2-3\,u_{01}^2u_{02})u_{10}u_{12}u_{13}+(u_{03}^3-u_{00}u_{01}u_{02})u_{10}u_{13}^2\\
&+(u_{02}^3-u_{00}u_{03}^2)u_{11}^3+(3\,u_{00}u_{02}u_{03}-3\,u_{01}u_{02}^2)u_{11}^2u_{12}+(u_{00}^2u_{03}-u_{01}^3)u_{12}^3\\
&+(2\,u_{00}u_{01}u_{03}-2\,u_{00}u_{02}^2)u_{11}^2u_{13}+(3\,u_{01}^2u_{02}-3\,u_{00}u_{01}u_{03})u_{11}u_{12}^2\\
&+(u_{00}u_{01}u_{02}-u_{00}^2u_{03})u_{11}u_{12}u_{13}+(u_{00}^2u_{02}-u_{00}u_{01}^2)u_{11}u_{13}^2\\
&+(2\,u_{01}^2u_{03}-2\,u_{00}u_{02}u_{03})u_{10}u_{12}^2+(u_{00}u_{01}^2-u_{00}^2u_{02})u_{12}^2u_{13},
\end{aligned}
$$

*where the polynomial coefficients in the $\{u_{0j}\}$ variables are the defining implicit equations of the twisted cubic.*

Here we adopt a practical method that avoids conversion assuming we have a parametric representation of $V$. We compute a polynomial whose vanishing is a necessary but not always sufficient condition for a point to lie on the variety. The algorithm we propose in Sections 4 and 5 avoids the computation of the Chow form polynomial and the need for rewriting techniques. For space curves, we shall achieve the result of Proposition 1 by computing only a few selected subsets of $Z(V)$.

### *2.2. Interpolation matrices*

This section describes a direct method to reduce implicitization to linear algebra by constructing an interpolation matrix $M$, given a plane curve or a (hyper)surface in parametric form or as a point cloud. The matrix is indexed by all possible monomials in the implicit equation (columns) and different values (rows) at which all monomials get evaluated. The vector of coefficients of the implicit equation is in the kernel of $M$, even in the presence of base points. This idea has been extensively used, e.g. [1, 9]. The matrix is somewhat different in [4], which is the method implemented in Maple for implicitization via the `implicitize` command. The latter method consists of expressing the implicit equations as the kernel of a carefully chosen integral form. It accepts non-algebraic parameterization and can still return formal implicit formulae provided that the integral form can be expressed with a formal formula (the integrand is polynomial in the parametric input equations). Alternatively, this method can perform floating-point computations and return approximate implicit formulae.

In [12], sparse elimination theory is employed to predict the implicit monomials and build the interpolation matrix. The monomial set is determined quite tightly for parametric models, by means of the sparse resultant of the parametric polynomials, thus exploiting the sparseness of the parametric and implicit polynomials.

More specifically, if the input object is *affinely* parameterized by rational functions

$$x_j = f_j(t),\ j = 1, \ldots, n,\ t = (t_1, \ldots, t_{n-1}) \tag{4}$$

then it is possible to predict the implicit monomials. This set is included in the *predicted (implicit) polytope* computed by software `ResPol`[1] [10]. If the input is a point cloud, we consider a coarse estimation of the monomial set by guessing the total degree of the variety and taking all the monomials of that degree or lower. Let $S$ be the predicted set of implicit monomials and $|S|$ its cardinality.

The set $S$ is used to construct a numerical matrix $M$, expressing a linear system whose unknowns are the coefficients $c_i$ $(i = 1, \ldots, |S|)$ of the monomials $S$ in the implicit polynomial, as discussed above. If the input object is a parameterization, we obtain the linear system in the $c_i$ by substituting each $x_j$ by its rational parametric expression $x_j = f_j(t)$ in the equation $\sum_{i=1}^{|S|} c_i x^{a_i} = 0$, $x^{a_i} := x_1^{a_{i1}} \cdots x_n^{a_{in}}$. We then evaluate the parameters $t$ at *generic* points (randomized in practice) $\tau_k \in \mathbb{C}^{n-1},\ k = 1, \ldots, \mu,\ \mu \geqslant |S|$, avoiding values that make the denominators of the parametric expressions close to 0. Each evaluation gives a linear equation in the coefficients $c_i$.

Letting $m_i = m_i(t)$ denote the monomial $x^{a_i}$ after substituting each $x_j$ by its parametric expression in (4), and $m_i|_{t=\tau_k}$ its evaluation at $t = \tau_k$, we end up with a matrix $M$ of the form:

$$M = \begin{bmatrix} m_1|_{t=\tau_1} & \cdots & m_{|S|}|_{t=\tau_1} \\ \vdots & \cdots & \vdots \\ m_1|_{t=\tau_\mu} & \cdots & m_{|S|}|_{t=\tau_\mu} \end{bmatrix}. \tag{5}$$

Typically $\mu = |S|$ for performing exact kernel computation, and $\mu = 2|S|$ for approximate numeric computation.

If the input object is given as a point cloud, we take $\mu$ random points out of it $(\mu \geqslant |S|)$ and use them instead of the points evaluated at the parameters $\tau_k,\ k = 1, \ldots, \mu$.

Let $M'$ be the $(|S| - 1) \times |S|$ numeric matrix obtained by evaluating the monomials $S$ at $|S| - 1$ points $\tau_k$, $k = 1 \ldots, |S| - 1$. We obtain the $|S| \times |S|$ matrix $M(x)$, which is numeric except for its last row, by appending the row of monomials $S$ to matrix $M'$:

$$M(x) = \begin{bmatrix} M' \\ S(x) \end{bmatrix}, \tag{6}$$

where we use the notation $S(x)$ to emphasize that this is the only symbolic row of $M(x)$. Notice that matrices $M'$, $M$ and $M(q)$, for a point $q$ lying on the hypersurface, have the same kernel. Matrix $M(x)$ has an important property:

**Lemma 3.** [11, Lem. 7] *Assuming $M'$ is of full rank, then* $\det M(x)$ *equals the implicit polynomial up to a constant.*

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

## 3. Interpolation matrices in higher codimension

This section generalizes the construction of interpolation matrices described in Section 2.2 to the case of varieties of codimension greater than 1.

Let $V \subset \mathbb{C}^n$ be a variety of any codimension given parametrically or as a point cloud. Given a set of monomials $S$, we randomly pick $\mu$ ($\mu \geqslant |S|$) points $\overline{x}_k \in V$, $k = 1, \ldots, \mu$ on $V$ either from the input point cloud, or as evaluations $\overline{x}_k = (f_1(\tau_k), \ldots, f_n(\tau_k))$ of the input parametric equations at random points $\tau_k$. Then we construct the interpolation matrix $M(x) = \begin{bmatrix} M' \\ S(x) \end{bmatrix}$ as in Section 2.2, where $M'$ is a numeric submatrix, $x = (x_1, \ldots, x_n)$ are symbolic coordinates and $S(x)$ is a row of symbolic monomials in $x$.

Recall that the *support* of a polynomial is the set of monomials appearing with nonzero coefficient. We define the following set of polynomials:

$$\mathcal{P} \coloneqq \{p \in \mathbb{C}[x] \mid \text{support}(p) \subset S \text{ and } \forall\, \xi \in V,\, p(\xi) = 0\}.$$

$\mathcal{P}$ is a $\mathbb{C}$-vector space. We assume that $S$ contains all the monomials of a set of generators of the ideal $\mathcal{I}(V)$, i.e., $V = \{\xi \in \mathbb{C}^n \mid \forall p \in \mathcal{P},\, p(\xi) = 0\}$. This construction is summarized in Algorithm 1.

**Algorithm 1:** Interpolation matrix of a $d$-dimensional variety $V \subset \mathbb{C}^n$

**Input :** - $V$ parameterized as in (4), or
- A set of monomials $S$ and a point cloud $(\overline{x}_k)_k \subset V$ of at least $|S|$ points such that $S$ contains all monomials of a set of generators of $\mathcal{I}(V)$

**Output:** An interpolation matrix $M(x)$ whose rank drops on $V$

(1) If $S$ is not given then
Let $\delta \coloneqq \prod_{i=1}^{d} \max_j(\deg_{t_i} f_j)$ and $S = (m_1, \ldots, m_{|S|})$ be the monomial basis of polynomials of degree $\leqslant \delta$

(2) Pick $\mu$ random points $(\overline{x}_k)_{1 \leqslant k \leqslant \mu}$ on $V$ with $|S| \leqslant \mu \leqslant 2|S|$

(3) Construct $M' \coloneqq (m_i(\overline{x}_k))_{1 \leqslant i \leqslant |S|, 1 \leqslant k \leqslant \mu}$

(4) Construct $M(x) \coloneqq \begin{bmatrix} M' \\ S(x) \end{bmatrix}$

Suppose also that the points $\overline{x}_k$, $k = 1, \ldots, \mu$, defining the rows of $M'$ are chosen generically enough, so that for all $h \in \mathbb{C}[x]$ with monomials in $S$, we have

$$h(\overline{x}_k) = 0,\, \forall k \in \{1, \ldots, \mu\} \iff h \in \mathcal{P}.$$

Then, the matrix $M(x)$ has a drop-of-rank property but weaker than that of Lemma 3 in the sense that $M'$ is not of full rank.

**Lemma 4.** *Assume we built $M(x)$ using a set $S$ that contains all the monomials of a set of generators of the ideal $\mathcal{I}(V)$. Then, for $\xi \in \mathbb{C}^n$, $\xi$ belongs to $V$ if and only if* $\text{Rank}(M(\xi)) < \text{Rank}(M(x))$*, where it holds* $\text{Rank}(M(x)) = \text{Rank}(M') + 1$.

*Proof.* Using the basis $S$ of monomials, we consider the canonical complex space of dimension $|S|$, $\mathbb{C}^{|S|}$. It is isomorphic to the space of polynomials $\{p \in \mathbb{C}[x] \mid \text{support}(p) \subset S\}$.

By abuse of notation, the image of $\mathcal{P}$ under this isomorphism will also be called $\mathcal{P}$. By the hypothesis on genericity of $\overline{x}_k$, $k = 1, \ldots, \mu$, we have that $\text{Ker}(M') = \mathcal{P}$. So, for $\xi \in \mathbb{C}^n$, we have:

$$\begin{aligned} \xi \in V &\iff \\ \forall p \in \mathcal{P},\, p(\xi) = 0 \,(\text{by the hypothesis that } \mathcal{P} \text{ characterizes } V) &\iff \\ \text{Ker}(M(\xi)) = \mathcal{P} = \text{Ker}(M') &\iff \\ \text{Rank}(M(\xi)) = \text{Rank}(M') < \text{Rank}(M(x)). & \end{aligned}$$

□

In practice, taking $\mu = |S|$ and random points $\overline{x}_k$, $k = 1, \ldots, \mu$, is enough to satisfy the hypothesis of Lemma 4. The set of monomials $S$ is hard to determine optimally. One can estimate bounds on the degree of $V$ and take all monomials of degree up to that bound, which usually leads to more monomials and a larger matrix $M(x)$ than needed. If the input is a rational parameterization $(f_1, \ldots, f_n)$ of $V$, we have an upper bound of $\deg(V)$ given by $\deg(V) \leqslant \prod_{i=1}^{d} \max_j(\deg_{t_i} f_j)$ where $\deg_{t_i} f_j$ is the maximum of the degrees of the numerator and denominator of $f_j$ in the $i$-th parameter.

The drop-of-rank property readily leads to a computation of the implicit equations representing $V$ set-theoretically, either by computing all the maximal nonzero minors of $M(x)$ containing the last line, which is inefficient in practice, or by computing the nullspace of $M'$.

The matrix representation given in [3] has the same drop-of-rank property but is of smaller size. The construction of their matrix, unlike $M(x)$, relies on syzygy computations and is thus slower. However, that method is overall more efficient because of the faster rank computation at each point evaluation.

## 4. Space curves

This section derives implicit representations of parametric space curves by Chow forms. Our methods avoid complex computations, such as the rewriting algorithm. In particular, we avoid the explicit computation of the Chow form and instead focus on a proper subset of the Chow variety that is enough to describe the space curve. Indeed, the Chow form of a space curve vanishes on a space line $L$ if and only if $L$ intersects the space curve. The method presented here provides sets of such lines, not all of them, but enough to be able to retrieve the space curve from them. This is how we proceed:

Suppose that we have a space curve $V$ parameterized as

$$x_j = F_j(t), \quad j = 0, \ldots, 3, \quad t = (t_0 : t_1).$$

Let the line $L$ be defined by a symbolic point $\xi = (\xi_0 : \cdots : \xi_3)$ and a sufficiently generic point $G \notin V$. Define two planes $\text{Aff}(G, \xi, P_0)$ and $\text{Aff}(G, \xi, P_1)$

that intersect along $L$, by choosing two random points $P_0$ and $P_1$ and let $H_0(x_0 : \cdots : x_3)$ and $H_1(x_0 : \cdots : x_3)$ be their respective implicit equations, as in (1). The coefficients of $H_0$ and $H_1$ are now *linear polynomials* in $\xi$. The (homogeneous) Sylvester resultant (see[5, Ch.3, Prop.1.7]) of this system, where we set $x_j = F_j(t)$, eliminates $t$ and returns a polynomial in $\xi$ which vanishes on $V$ (but not only on $V$), thus offering a necessary but not sufficient condition, see Algorithm 2.

**Lemma 5.** *Let $\delta = \deg F_j(t)$, $j = 0, \ldots, 3$ and $R_G$ be the Sylvester resultant of*

$$H_0(F_0(t) : \ldots : F_3(t)), \quad H_1(F_0(t) : \ldots : F_3(t)), \tag{7}$$

*where $H_0, H_1$ are defined as above. Then $R_G$ is of degree $2\delta$ and factors into a degree $\delta$ polynomial defining a surface $\mathcal{S}_V^G \supset V$, and a polynomial $E_L^\delta$, where $E_L$ is a linear polynomial defining the plane passing through points $G, P_0, P_1$.*

*Proof.* The degree of the Sylvester resultant in the coefficients of each of the $H_0, H_1$, is $\delta$. $\xi$ is involved linearly in the coefficients of both $H_0$ and $H_1$, since it is taken to lie in the intersection of the two planes. Hence the degree of the sought polynomial in $\xi$ is $2\delta$.

It vanishes only in two cases: if $\xi$ belongs to the plane defined by $G$, $P_0$ and $P_1$, or if $\xi$ belongs to a line passing by $G$ and intersecting $V$. Hence we can divide (possibly several times) the sought polynomial in $\xi$ by the equation $E_L$ of the plane defined by $G$, $P_0$ and $P_1$, thus obtaining an equation of the conical surface $\mathcal{S}_V^G$ of vertex $G$ and directrix $V$. Since such a conical surface is of degree $\delta$, its equation is $R_G/E_L^\delta$. □

**Theorem 6.** *Let $F : \mathbb{P}^1 \to \mathbb{P}^3$, be a homogeneous parameterization of a space curve $V$ and $\mathcal{S}_V^{G_k}$, $k = 1, 2, 3$ be three conical surfaces obtained by the method above with 3 different random points $G_k \notin V$. We distinguish two cases.*

1. *If $V$ is not planar and the points $G_k$ are not collinear, then $V$ is the only 1-dimensional component of $\mathcal{S}_V^{G_1} \cap \mathcal{S}_V^{G_2} \cap \mathcal{S}_V^{G_3}$.*
2. *If $V$ is contained in a plane $\mathcal{P}$ and if $G_1$ is not in $\mathcal{P}$, then $V = \mathcal{P} \cap \mathcal{S}_V^{G_1}$.*

*Proof. Case (1).* Since $\mathcal{S}_V^{G_k}$ are three different cones - or cylinders when the vertices $G_k$ are at infinity -, they have no 2-dimensional component in common. We first reduce the problem to the case where the vertices are $P_x \coloneqq (0 : 1 : 0 : 0)$, $P_y \coloneqq (0 : 0 : 1 : 0)$ and $P_z \coloneqq (0 : 0 : 0 : 1)$. We shall prove that an algebraic space curve is the intersection of the 3 cylinders spanned by the curve itself and of directions $\vec{x}_1, \vec{x}_2$ and $\vec{x}_3$ respectively.

Let $\mathcal{C}$ be an irreducible component of $\left(\mathcal{S}_V^{G_1} \cap \mathcal{S}_V^{G_2} \cap \mathcal{S}_V^{G_3}\right)$. Since $G_1, G_2, G_3$ are not collinear, there exists a map $\phi \in \mathrm{PGL}(4, \mathbb{C})$ that sends $G_1$ to $P_x$, $G_2$ to $P_y$ and $G_3$ to $P_z$. By linearity of $\phi$ (in particular, $\phi$ preserves alignment), we have $\phi\left(\mathcal{S}_V^{G_1}\right) = \mathcal{S}_{\phi(V)}^{P_x}$ and similar equalities for $\phi\left(\mathcal{S}_V^{G_2}\right)$ and $\phi\left(\mathcal{S}_V^{G_3}\right)$. Also, $\phi(V)$ is a homogeneous variety parameterized by $\phi \circ F$.

Thus, if $\mathcal{C} \subset \mathcal{S}_V^{G_3}$, then $\phi(\mathcal{C}) \subset \mathcal{S}_{\phi(V)}^{P_z}$. By this argument, we only have to prove that there is no homogeneous space curve $\phi(V)$ for which $\mathcal{S}_{\phi(V)}^{P_x} \cap \mathcal{S}_{\phi(V)}^{P_y} \cap \mathcal{S}_{\phi(V)}^{P_z}$ contains a different curve than $\phi(V)$. For convenience, $\phi(V)$ is denoted by $V$ and $\phi(\mathcal{C})$ is denoted by $\mathcal{C}$ in what follows.

**Remark 1.** *Since $V$ does not lie in the plane at infinity ($x_0 = 0$), we can switch to the affine setting.*

The parameterization in this affine setting is

$$f : t_1 \in \mathbb{C} \mapsto \left( \frac{F_1(1:t_1)}{F_0(1:t_1)}, \frac{F_2(1:t_1)}{F_0(1:t_1)}, \frac{F_3(1:t_1)}{F_0(1:t_1)} \right) \in \mathbb{C}^3.$$

For convenience, we use the same notations for both the homogeneous parameterization and its restriction to this affine setting $t_0 = x_0 = 1$.

We now have simpler expressions for our surfaces:

- $\mathcal{S}_V^{P_x} = \{(x_1, f_2(t_1), f_3(t_1)) \mid x_1, t_1 \in \mathbb{C}\}$,
- $\mathcal{S}_V^{P_y} = \{(f_1(t_1), x_2, f_3(t_1)) \mid x_2, t_1 \in \mathbb{C}\}$,
- $\mathcal{S}_V^{P_z} = \{(f_1(t_1), f_2(t_1), x_3) \mid x_3, t_1 \in \mathbb{C}\}$.

Since $\mathcal{C} \subset \mathcal{S}_V^{P_x}$, there is a (not necessarily rational) parameterization of $\mathcal{C}$ given by $q : t_1 \in \mathbb{C} \mapsto (q_1(t_1), f_2(\varphi(t_1)), f_3(\varphi(t_1)))$, where $q_1$ and $\varphi$ are continuous piecewise smooth maps.

**Remark 2.** *We see that $\varphi$ (resp. $q_1$) is not locally constant: otherwise, a part of $\mathcal{C}$ would be included in a straight line (resp. a plane), which contradicts the fact that $V$ is not planar.*

We can thus pick a small open disc $I \subset \mathbb{C}$ such that $q_{|I}$ is injective and so, without loss of generality, we assume that $\varphi_{|I} = \mathrm{Id}_{|I}$. Also, since $V$ is not planar and $f$ is rational, the sets of singular points of the three maps $\pi_{x_1} : t_1 \mapsto (f_2(t_1), f_3(t_1))$, $\pi_{x_2} : t_1 \mapsto (f_1(t_1), f_3(t_1))$ and $\pi_{x_3} : t_1 \mapsto (f_1(t_1), f_2(t_1))$ are finite. Shrinking $I$ if necessary, we assume there is no such singular point in $q(I)$.

Now, we have a local curve $q(I) = \{(q_1(t_1), f_2(t_1), f_3(t_1)) \mid t_1 \in I\}$ included in $\left(\mathcal{S}_V^{P_x} \cap \mathcal{S}_V^{P_y} \cap \mathcal{S}_V^{P_z}\right)$. Using the fact that it lies on $\mathcal{S}_V^{P_y}$, we have another parameterization of $q(I)$ given by $r = (r_1, f_2, r_3)$ with $r_i$ injective on $I$ and $r_i(I) \subset f_i(\mathbb{C})$. Similarly, $q(I) \subset \mathcal{S}_V^{P_z}$ gives a third parameterization $s = (s_1, s_2, f_3)$ with $s_i$ injective on $I$ and $s_i(I) \subset f_i(\mathbb{C})$.

Comparing $s$ with $q$ and $r$, we have $s = (s_1, f_2, f_3)$. Lastly, since $\pi_{x_1}$ is regular on $I$, there is only one branch in $\pi_{x_1}(s(I))$ and so $s_1^{-1}(s(I)) = f_1^{-1}(s(I))$. The curve $\mathcal{C}$ is thus locally contained in $V$; it follows that $\mathcal{C} = V$.

*Case (2).* Since $G_1 \notin \mathcal{P}$, the conical surface $\mathcal{S}_V^{G_1}$ consists only of the union of lines transversal to $\mathcal{P}$: $\mathcal{S}_V^{G_1} = \cup_{x \in V} \mathrm{Line}(G_1, x)$. Each of these lines intersects $\mathcal{P}$ only in one point $x \in V$ so the curve $V$ is exactly $\mathcal{P} \cap \mathcal{S}_V^{G_1}$. □

Note that Theorem 6 is also valid over the reals, the key argument being the existence of local smooth maps around the (dense) set of regular points.

## 5. Varieties of arbitrary codimension

In this section we generalize the construction of Section 4 to varieties of arbitrary codimension. Let $V \subset \mathbb{P}^n$ be a $d$-dimensional variety parameterized as

$$x_j = F_j(t),\ j = 0, \ldots, n,\ t = (t_0 : \cdots : t_d),$$

and $\mathcal{G} = \{G_1, \ldots, G_{n-d-1}\}$ be a set of $n-d-1$ sufficiently generic points not in $V$. Choose $d+1$ sets of $d$ random points $P_i = \{P_{i1}, \ldots, P_{id}\}$, none of these points lying in $V$, such that for $i = 0, \ldots, d$ the points in $\mathcal{G}$ and in $P_i$ form an affinely independent set. Let $H_i$, $i = 0, \ldots, d$ be the hyperplane defined as the span of the points $\xi, \mathcal{G}, P_i$. Substitute $x_j = F_j(t)$ in each $H_i$ to obtain the system of equations

$$H_0(F_0(t) : \cdots : F_n(t)) = \ \cdots \ = H_d(F_0(t) : \cdots : F_n(t)) = 0. \tag{8}$$

The resultant of the polynomial system (8) eliminates $t$ and returns a polynomial $R_{\mathcal{G}}$ in $\xi$ which vanishes on $V$ (but not only on $V$), thus offering a necessary but not sufficient condition, see Algorithm 2.

There are three issues we have to examine when generalizing the algorithm. We do so in the following three subsections.

### *5.1. Computing the resultant in several variables*

The resultant computation is the bottleneck of the algorithm. To compute the resultant for $d = 1$, we can use Sylvester determinantal formula. For arbitrary $d$ there exist rational formulae yielding the resultant as the ratio of two determinants, namely the Macaulay determinant [5] or the sparse resultant matrix and one of its minors [7]. These formulae are optimal for generic coefficients. For arbitrary coefficients, an infinitesimal perturbation may be applied.

Another option is to use interpolation in conjunction with information on the resultant support. This might be obtained from degree bounds on $R_{\mathcal{G}}$, as explained below, or by the computation of the monomials of the polynomials in (8) using software `ResPol` from [10]. The latter is analogous to the basic approach for defining the interpolation matrix by the support obtained from the resultant polytope, see Section 2.2.

If we choose to interpolate the resultant, an issue arises at sampling: all generated points $\xi$ lie on $V$, whereas we are trying to compute a hypersurface containing $V$. But the kernel of $M$ should have large dimension and, among the kernel vectors, we may choose one or more "small" independent vectors to define the implicit equations. We use independent vectors so as to obtain distinct surfaces. Indeed, with independent vectors, the tangent spaces of the surfaces differ and thus the surfaces intersect transversally. Here "small" may refer to the number of non-zero vector entries, or to the total degree of the monomials corresponding to its non-zero entries.

Independently of the resultant algorithm used, though, there is always an extraneous factor in the resultant that is similar to the one pinpointed by Lemma 5. Which leads to:

*5.2. Identifying the extraneous factor in the resultant*

The resultant is indeed always reducible and only one of its irreducible components is relevant for describing $V$. To address this issue, Lemma 5 can be generalized as follows.

**Lemma 7.** *Let $\delta = \deg F_j(t)$, $j = 0, \ldots, n$ and $R_{\mathcal{G}}$ be the resultant of the equations* (8)*. Then $R_{\mathcal{G}}$ factors into a polynomial defining a hypersurface $\mathcal{S}_V^{\mathcal{G}}$ which is of degree at most $\delta^d$ (equality holds when there are no base points) and contains $V$, and an extraneous factor $E^p$, where $E$ is a polynomial of degree $d$ and $p \leqslant \delta^d$.*

*Proof.* We define the *generalized conical hypersurface* of directrix $V$ and vertices $\mathcal{G} = (G_1, \cdots, G_{n-d-1})$ as following:

$$\mathcal{S}_V^{\mathcal{G}} \coloneqq \cup_{x \in V} \operatorname{Aff}(G_1, \ldots, G_{n-d-1}, x)$$

By abuse of notation, we shall also denote by $\mathcal{S}_V^{\mathcal{G}}$ the square-free polynomial defining the hypersurface $\mathcal{S}_V^{\mathcal{G}}$ (unique up to a constant factor).

Let us first note that $R_{\mathcal{G}}$ has a total degree in $\xi$ of $(d+1)\delta^d$. Indeed, the degree in $t$ of every $H_i(F_0(t) : \cdots : F_n(t))$ is $\delta$, and the coefficients of the $H_i$'s are linear polynomials in $\xi$. The resultant of these polynomials has degree in the coefficients of each $H_i$ bounded by $\delta^d$, therefore total degree $\leqslant (d+1)\delta^d$, see e.g. [5, Thm.3.1].

Now, $R_{\mathcal{G}}$ vanishes if and only if the equations (8) have a common solution, which happens when:

i. either the hyperplanes defined $H_i = 0$, $i = 0, \ldots, d$, intersect along a linear subspace $L$ of dimension $n - d$ and $L \cap V \neq \emptyset$, or
ii. these hyperplanes intersect along a linear subspace of dimension $> n - d$.

The first case is dealt with by the condition $\xi \in \mathcal{S}_V^{\mathcal{G}}$. It remains to prove that the second case is equivalent to $\xi \in E$ with $E$ being a hypersurface of degree $d$.

In what follows, we use the theory of *exterior algebra* ([22, Ch.10]) to compute the equations of $E$. Let $(\Lambda\mathcal{G}) \coloneqq G_1 \wedge \ldots \wedge G_{n-d-1}$ for convenience.

Then,

$$H_i = \xi \wedge (\Lambda\mathcal{G}) \wedge P_{i1} \wedge \ldots \wedge P_{id}, \quad \text{for } i = 0, \ldots, d.$$

Thus,

$$\bigcap_{i=0}^{d} H_i = (\xi \wedge (\Lambda\mathcal{G}) \wedge P_{01} \wedge \ldots \wedge P_{0d}) \cdots (\xi \wedge (\Lambda\mathcal{G}) \wedge P_{d1} \wedge \ldots \wedge P_{dd}) =$$

$$\left[\sum_{i=1}^{d} (-1)^i \det(\xi, \mathcal{G}, P_{11}, \ldots, P_{1d}, P_{0i})(\xi \wedge (\Lambda\mathcal{G}) \wedge P_{01} \wedge \ldots \wedge P_{0(i-1)} \wedge P_{0(i+1)} \wedge \ldots \wedge P_{0d})\right] \cdot$$

$$\cdot (\xi \wedge (\Lambda\mathcal{G}) \wedge P_{21} \wedge \ldots \wedge P_{2d}) \cdots (\xi \wedge (\Lambda\mathcal{G}) \wedge P_{d1} \wedge \ldots \wedge P_{dd}).$$

By continuing developing $d$ times the exterior product, we obtain the relation

$$\cap_{i=0}^{d} H_i = E(\xi)(\xi \wedge (\Lambda\mathcal{G})),$$

where $E$ is a polynomial of degree $d$ in $\xi$. So the resultant vanishes if $\dim(\cap_i H_i) > n-d$, that is if $E(\xi) = 0$. Example 8 shows how this polynomial $E$ is computed for a concrete variety.

Since $R_{\mathcal{G}}(\xi) = 0$ if and only if $\mathcal{S}_V^{\mathcal{G}}(\xi) = 0$ or $E(\xi) = 0$, the factor $E$ is present and raised to a power $p \leqslant \delta^d$, in the expression of $R_{\mathcal{G}}$. □

*5.3. How many hypersurfaces are sufficient*

Computing the resultant and factoring out the extraneous factor given by Lemma 7, yields one implicit polynomial defining a hypersurface that contains the given variety. To achieve the hypothesis of Proposition 1, we must iterate for a few distinct pointsets $\mathcal{G}$ thus obtaining implicit polynomials $\mathcal{S}_V^{\mathcal{G}} = 0$ whose intersection is $V$.

Unfortunately, we do not yet have an a priori bound $\rho$ for the number of equations needed to describe the variety set-theoretically as in Theorem 6. Experimental results indicate that for curves in $\mathbb{C}^n$, $n+1$ hypersurfaces of the type $\mathcal{S}_V^{\mathcal{G}} = 0$ are required. This bound also extends to surfaces in $\mathbb{C}^4$, where 5 such hypersurfaces are sufficient. The theoretical result in [16, Ch.V] indicates that $n$ hypersurfaces suffice for any variety in $\mathbb{C}^n$. It is not clear how to apply this result to the specific type of hypersurfaces (conical) obtained by our method.

These equations are obtained using random pointsets $\mathcal{G}$: the hypothesis of genericity is important. Indeed, each pointset $\mathcal{G}_k$ $(k = 1, \ldots, \rho)$ must obviously consist of affinely independent points not in $V$ in order to define $\mathcal{S}_V^{\mathcal{G}_k}$ properly. It is also possible that more implicit polynomials are required for specific (bad) choices of pointsets $\mathcal{G}_k$. In particular, if there is a common affine subspace $L \subset (\cap_k \operatorname{Aff}(\mathcal{G}_k))$ and if $L \cap V \neq \emptyset$, then $L \subset \left(\cap_k \mathcal{S}_V^{\mathcal{G}_k}\right)$ and the equations of these conical hypersurfaces are not defining $V$ set-theoretically. To avoid these degenerate cases, it may be interesting to choose the random pointsets $\mathcal{G}_k$ such that any $n+1$ points picked from those pointsets are always affinely independent.

*5.4. Degree bounds*

Before stating the implicitization algorithm, we first examine the degrees of the factors of the resultant polynomial $R_{\mathcal{G}}$.

We showed that $E^p$ appears as a factor of $R_{\mathcal{G}}$, where $p$ is possibly very high. For curves $(d = 1)$, $p$ indeed achieves the upper bound $\delta^d$. However, in our tests with $1 < d < n$ and in presence of base points, the factor defining $\mathcal{S}_V^{\mathcal{G}}$ also appears to a power $q > 1$ in the expression of $R_{\mathcal{G}}$:

$$\underbrace{R_{\mathcal{G}}}_{\text{degree } \leqslant \delta^d + d\delta^d} = \underbrace{(\mathcal{S}_V^{\mathcal{G}})^q}_{\text{degree } \leqslant \delta^d \times q} \times \underbrace{E^p}_{\text{degree } d \times p}$$

Note that when $V$ is a properly parameterized curve, the inequalities become equalities and we have $q = 1$ and $p = \delta$. The algorithm works correctly on non-properly parameterized varieties. However, a non-proper parameterization decreases the degree of $\mathcal{S}_V^{\mathcal{G}}$ by some factor (the generic number of preimages) and increases the power degree $q$ by that same factor. In practice, the extraneous factor $E$ seems to always appear with some power $p$ close to its upper bound $\delta^d$.

A tighter bound on the degree of $R_{\mathcal{G}}$ can be obtained by considering sparse resultants and *mixed volumes* [5, Ch.7]. Let

$$MV_{-i} = MV(H_0, \ldots, H_{i-1}, H_{i+1}, \ldots, H_d),\ 0 \leqslant i \leqslant d,$$

be the mixed volume of all polynomials excluding $H_i$. The degree of the sparse resultant in the coefficients of $H_i$ is known to equal $MV_{-i}$, therefore its total degree equals $\sum_{i=0}^{d} MV_{-i}$.

For curves in $\mathbb{P}^n$, i.e., when $d = 1$, Algorithm 2 utilizes the Sylvester determinant for computing the resultant instead of the Macaulay or sparse resultant determinant in the general case. This fact, in conjunction with Lemma 7 for determining the extraneous factors of the resultant, make the algorithm much more efficient for $d = 1$ than in the general case of arbitrary $d$.

---

**Algorithm 2:** Implicit representations of a $d$-dimensional variety $V \subset \mathbb{P}^n$

---

**Input :** - $V$, parameterized by $x_j = F_j(t), j = 0, \ldots, n$,
- Number of iterations: $\rho\,(= 3$ when $n = 3)$

**Output:** $\rho$ polynomials vanishing on $V$

Repeat $\rho$ times:

(1) Define a $(n-d-1)$-dimensional linear subspace by affinely independent random points $\mathcal{G} = \{G_1, \ldots, G_{n-d-1}\}$ none $G_k \notin V$ $(1 \leqslant j \leqslant n-d-1)$, and consider symbolic point $\xi = (\xi_0, \ldots, \xi_n)$.

(2) Define $d+1$ hyperplanes $H_i$ through $\mathcal{G}, \xi, P_i$, for random point sets $P_i$ affinely independent from points in $\mathcal{G}$.

(3) Set $x_j = F_j(t)$ in the equations of $H_i$: their resultant $R_{\mathcal{G}}$, where we eliminate $t$, is the sought polynomial in $\xi$. Compute it by formula, or by interpolation as in Section 2.2.

(4) Compute the extraneous factor $E$ using the exterior algebra recursive formula. Then divide $R_{\mathcal{G}}$ by $E$ as many times as possible to obtain $\mathcal{S}_V^{\mathcal{G}}$.

---

## 6. Experiments and performance

This section presents the features of Algorithm 2 and its practical performance through a number of examples. In all these examples we use our Maple

implementation of Algorithm 2 [2] on a laptop with a 2GHz Intel Celeron processor running Maple 18. Our algorithm was also implemented in Sage with similar runtimes as in Maple. In some of these examples we compare different existing approaches for implicitization with ours:

(CF) our main contribution, Algorithm 2 based on Chow forms,

(MI) the matrix interpolation Algorithm 1 of Section 3,

(GB) a Gröbner bases computation using one of Maple's native commands:

(GBa) `Basis` from the package `Groebner` or

(GBb) `EliminationIdeal` from the package `PolynomialIdeals`,

(Mrep) a matrix representation algorithm based on syzygies [2, 3], using the Maple code of the authors, and

(Impl) Maple's command `implicitize` from the package `algcurves` which is an implementation of the algorithm presented in [4].

For all these algorithms the input is the parametric equations. In Figure 2, and whenever we need to consider generic parameterizations, Maple's command `randpoly` is used to generate these parametric equations. (MI) also needs a superset of the support of the implicit equations. For this, in the examples we use the exact set of monomials. Even though the algorithm's runtime is very competitive, its usage is slower than the usage of the other algorithms: (MI) becomes inefficient as soon as the user needs to check membership of more than $\sim 20$ points. This is due to the rank computations performed by (MI) vs the polynomial evaluations of all other algorithms except (Mrep). The latter's usage also requires rank computations but as its matrices are much more compact, these are done more efficiently than (MI). On the upside, (MI) can also accept a point cloud representation as input. Algorithm (Impl) also requires as input the target total degree of the implicit polynomial. Unless specified otherwise, we use the lowest degree that is sufficient for having a generating set of the variety's ideal.

Note that Algorithm 2, (CF), can be easily parallelized as the computation of each implicit equation can be done independently. This effectively reduces its runtime by a factor equal to the number of equations required (three for space curves). For a fair comparison, we do not use parallelization in the runtimes presented here because we have not studied the potential of parallelization of the other algorithms.

The output of (CF), (GB) and (Impl) are implicit polynomials. The output of the matrix representation (Mrep) and interpolation (MI) algorithms are symbolic matrices that have the drop of rank property. It is possible to retrieve

[2] A prototype version for space curves is available at:
`http://users.uoa.gr/~claroche/publications/ChowFormImplicitize.zip`

implicit equations from such matrices by computing their minors of size equal to the maximal rank of the matrix. However, this is not the aim of these algorithms, and quickly becomes too expensive. We could nonetheless compute the minors of the output of (Mrep) for space curves of parametric degree up to 7. These minors yield, up to constant factors, the same equations as the (GB) method. The minors of the matrices computed by (MI) depend on the set of monomials used as input. Computing these minors takes longer than 30 min already for space curves of parametric degree 4. Algorithm 2 outputs a different set of equations. For curves of moderate and high degrees, the Sylvester resultant computation represents more than 99% of our algorithm's computation time.

For curves parameterized by polynomials of degree up to 6 all methods are comparable in running time. For larger degrees Gröbner basis outperform our method by a factor that depends on the degree and the number of monomials in the parametric polynomials and ranges from 10 to $10^3$. In a few cases Algorithm 2 is also slower than the syzygy-based matrices. It is competitive with (Impl) in most situations.

On the upside, Algorithm 2 always outputs 3 implicit equations for space curves while their number depends on the input in the other methods and is typically much larger than 3. Also, our method works on parametric equations that contain additional formal parameters, unlike (Mrep).

The space curve example of the largest degree that we computed is of degree 18. It took about 2 min to implicitize it with (GBb), 3 min with (CF) and 4 min with (Impl).

| | **(CF)** | **(Impl)** | **(GB)** | **(Mrep)** |
|---|---|---|---|---|
| **Runtime (representation)** | Medium | Medium | Fast | Medium |
| **Runtime (membership)** | Immediate | Immediate | Immediate | May be slow |
| **# of outputs** | Low and fixed | None (failure) or very high | Medium, moderately varying | Matrix of full rank, varying size |
| **Degree of outputs** | Same as input | Set by user | Same as input or a bit lower | Matrix entries are linear polynomials |
| **Restrictions on the input** | Rational* parameterization, $n - d > 1$ | None (accepts analytic parameterizations) | Rational* parameterization, Non-parametric coefficients | Rational* parameterization, Non-parametric coefficients |
| **Retrieve point parameters** | ✗ | ✗ | ✗ | ✓ |
| **Randomized** | ✓** | ✗ | ✗ | ✓** |
| **Geometric Feature** | Conical surfaces | ✗ | ✗ | ✗ |

(*) Also accepts homogeneous parameterizations as the conversion rational ↔ homogeneous is straightforward
(**) Depends on the implementation

Table 1: Comparison of the different features of implicitization algorithms

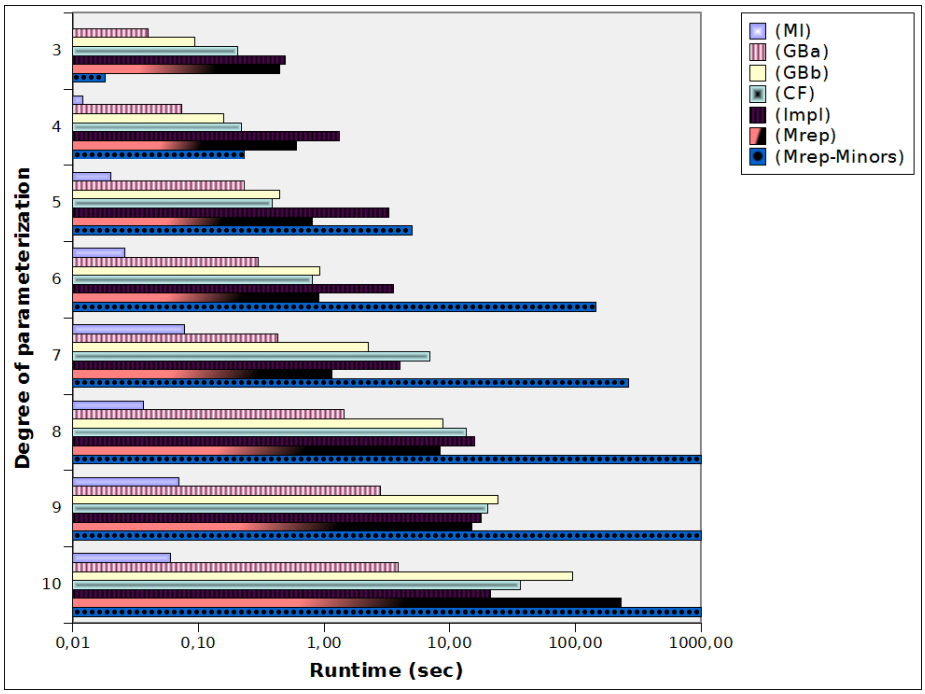



Figure 2: Timing (sec) of various algorithms on random space curves: (CF) implements Alg. 2, (MI) matrix interpolation Alg. 1, (GBa) Gröbner bases by Maple's `Basis`, (GBb) Gröbner bases by Maple's `EliminationIdeal`, (Mrep) Maple code for matrix representation algorithm using syzygies, (Impl) Maple's `implicitize` implementing [4]. (Mrep-Minors) is the time of computing Mrep minors, providing implicit equations. Computing minors by (MI) takes $> 20$ min at degree 5 thus there is no row for (MI-Minors).

Though the runtime per equation of (Impl) is lower than that of (CF), the number of implicit equations that (Impl) outputs cannot be predicted and it seems to be only dependent on the input degree and dimensions: for space curves of the same parametric degree it always returns the same number of implicit equations. For a space curve of parametric degree larger than 8 and requested outputs of the same degree, their number is greater than 100. In contrast, Algorithm 2 can return the implicit equations one by one, and Theorem 6 guarantees that 3 of them define the curve set-theoretically. For parameterizations of degree up to 4, (Impl) computes a small number of implicit equations whose total degree is lower than the degree of those computed by our method. On the other hand, Algorithm 2 always outputs equations of the same total degree equal to the degree of the parameterization, by Lemma 5.

### *6.1. Examples*

In the sequel we switch from the projective to the affine setting and set $\xi = (\xi_1, \ldots, \xi_n) \equiv (x_1, \ldots, x_n)$, for emphasizing these are the implicit variables. For curves in any ambient dimension we compute the resultant directly using the

Sylvester matrix, and also by interpolation, employing degree bounds relying on mixed volume. For $d > 1$ we use the Macaulay, or the sparse resultant matrix.

The Sylvester matrix leads to polynomials of degree twice the degree of the curve for any random points $\mathcal{G}$, as expected. Interpolating the resultant leads to matrices with very large kernels: on the upside, the polynomials we obtain from the kernel vectors are of degree no greater than those of the former method. Moreover, among them we can find a number of polynomials of small degree and, often, smaller than the degree predicted by degree bounds: these polynomials define the variety set-theoretically.

**Example 2.** *Consider the twisted cubic curve $V \subset \mathbb{C}^3$ affinely parameterized as:*

$$(x_1, x_2, x_3) = (t, t^2, t^3), \quad t \in \mathbb{C}.$$

*An optimal system of implicit equations for $V$ is*

$$x_1^2 - x_2 = x_2^2 - x_1 x_3 = x_1 x_2 - x_3 = 0. \tag{9}$$

*Let $L$ be the line passing through symbolic point $\xi = (x_1, x_2, x_3)$, and generic point $G \notin V$. We define two random planes $H_1(x_1, x_2, x_3)$, $H_2(x_1, x_2, x_3)$, intersecting at $L$ by considering additional random points $P_1, P_2 \notin L$, respectively. The Sylvester resultant of $H_1(t, t^2, t^3) = H_2(t, t^2, t^3) = 0$ is a polynomial of degree 6 in $\xi$ which factors into the degree 3 polynomial*

$$\begin{aligned} 32x_2 - 16x_3 + 56x_1x_3 + 16x_1x_2 - 80x_2^2 - 32x_1^2 - 40x_3^2 + 42x_2^3 - 2x_3x_2x_1 \\ +56x_3x_2 - 5x_3{x_2}^2 + 5{x_3}^2x_1 - 8x_3{x_1}^2 - 48{x_2}^2x_1 + 24x_2{x_1}^2 \end{aligned}$$

*and the extraneous linear factor raised to the power 3 predicted in Lemma 5.*

*This yields a surface containing $V$ but not of minimal degree. Repeating the procedure 3 times, the ideal of the resulting polynomials equals the ideal defined by the polynomials in* (9)*.*

*Alternatively, we may interpolate the Sylvester resultant above. We take as predicted support the lattice points in a 3-simplex of size 6. The $84 \times 84$ matrix constructed has a kernel of dimension 65. The corresponding 65 kernel polynomials are of degrees from 2 to 6. Among them, we can find the three polynomials in* (9) *(up to sign).*

| Algorithm | Time [ms] | Memory [MB] | Output | Minors [ms] |
|---|---|---|---|---|
| (CF) | 130 | 3.1 | 3 equations of degree 3 | — |
| (MI) | $< 10$ | 0.012 | $7 \times 6$ matrix (see (10)) | 180 |
| (GBb)* | 20 | 0.29 | $x^2 - y$; $x\,y - z$; $x\,z + y^2$ | — |
| (Mrep) | 160 | 3.88 | $\begin{bmatrix} -z & y & -x \\ y & -x & 1 \end{bmatrix}$ | $< 10$ |
| (Impl) | 390 | 11.42 | $x^2 - y$; $x\,y - z$; $x\,z + y^2$ | — |

(*) Only (GBb) is used since it is more efficient than (GBa) for degrees lower than 9.

Table 2: Comparison of the different algorithms on the twisted cubic (Example 2)

*The interpolation matrix built by (MI) is:*

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 4 & 8 & 4 & 16 & 8 & 16 \\ 9 & 27 & 9 & 81 & 27 & 81 \\ 16 & 64 & 16 & 256 & 64 & 256 \\ 25 & 125 & 25 & 625 & 125 & 625 \\ 36 & 216 & 36 & 1296 & 216 & 1296 \\ y & z & x^2 & y^2 & x\,y & x\,z \end{bmatrix} \tag{10}$$

*In contrast, computing the Chow form as in [6, Section 3.3] gives 16 (homogeneous) implicit equations, all of degree 3, see Example 1.*

*If we give up the computation of minors of the matrices given by (MI) and (Mrep), we can still check whether a point belongs to the curve with a rank computation. Computing the rank of the $2 \times 3$ matrix given by (Mrep) at a specified point costs about 2ms on average. For the $7 \times 6$ matrix given by (MI), it costs about 9ms on average. Consequently, (MI) is faster when one needs to check the ownership of a very small number of points (less than 20). If many points need to be checked or if the preimage parameters are required, then (Mrep) should be preferred over (MI).*

**Example 3.** *Consider the space curve $V$ in Figure 1 affinely parameterized as:*

$$(x_1, x_2, x_3) = \left( \frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}, \left( \frac{1-t^2}{1+t^2} \right)^2 \right), \quad t \in \mathbb{C}.$$

*It is the intersection of two cylinders:*

$$x_1^2 - x_3 = x_2^2 + x_3 - 1 = 0. \tag{11}$$

*Let line $L$ be defined from the symbolic point $\xi = (x_1, x_2, x_3)$, and "generic" point $G \notin V$. Define two random planes $H_1$ and $H_2$ that intersect at $L$, by choosing random points $P_1, P_2 \notin L$, respectively. Then, the Sylvester resultant of*

$$H_1 \left( \frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}, \left( \frac{1-t^2}{1+t^2} \right)^2 \right), \;\; H_2 \left( \frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}, \left( \frac{1-t^2}{1+t^2} \right)^2 \right)$$

*is a polynomial of degree 8 in $\xi$ which factors into the following degree 4 polynomial:*

$$\begin{aligned} 29 + 120x_1 - 56x_2 - 182x_3 + 168x_1x_2 - 110x_1^2 - 78x_2^2 + 90x_2^2x_3 - 12x_2x_3 + \\ 70x_1^2x_3 - 18x_1x_3 - 120x_1x_2^2 + 40x_1^2x_2 + +156x_3^2 + 56x_2^3 + 49x_2^4 + 25x_1^4 - \\ 48x_3^2x_2 - 72x_3^2x_1 + 12x_3^2x_2^2 + 28x_3x_2^3 + 12x_3^2x_1^2 - 168x_2^3x_1 + 222x_2^2x_1^2 - \\ 30x_3x_1^3 - 120x_2x_1^3 - 48x_3x_2x_1 - 102x_3x_2^2x_1 + 76x_3x_1^2x_2 \end{aligned}$$

*and the expected extraneous linear factor raised to the power 4. This yields a surface containing $V$ but not of minimal degree. Repeating the procedure 3 times, we obtain three surfaces that intersect on the curve.*

*Alternatively, we interpolate the Sylvester resultant above using as support the lattice points in a 3-simplex of size 8. The* $165 \times 165$ *matrix constructed has a kernel of dimension 133. The degrees of the corresponding 133 kernel polynomials vary from 2 to 8. Among them, we find the two quadratic polynomials in* (11).

*The equations above were obtained by choosing random points with integral coordinates and relatively close to the curve. When we increase the range or allow for non-integral points, in order to have better chances to avoid the non-generic points, the size of the coefficients increase. For example, using random points with integral coordinates in a box of size 100 around the curve yields equations as below, where we omit most terms:*

$$39834389985357559755785075 93x^4 + \cdots - 64216978805609810549754903 04.$$

*On this example, the algorithms (Impl), (GB) and (MI) compute the two equations in* (11). *This cannot be done by Algorithm 2 unless we know which special apex G for the conical surfaces should be used: for this example, points at infinity in the* $x_1$ *and* $x_3$ *directions allow to use only two equations instead of three. The algorithm (Mrep) computes a* $3 \times 5$ *matrix, resulting in 6 independent minors of degree 3.*

**Example 4.** *[20, Example 5.3] Consider the space curve parameterized as* $(\frac{p_1}{q}, \frac{p_2}{q}, \frac{p_3}{q})$, *where:*

$$p_1 = \frac{28775}{134878}\,t^4 - \frac{64513368541235902334413817941 2317}{44618662654079434352431999 3932400}\,t^3 - \frac{47828026434221466944680145374491240124}{56419462326158680749256153875975210675}\,t^2$$

$$+ \frac{10298677641229167982949337521716055081}{87926434794013528440399200845675653 00}\,t - \frac{28256478577625521695889619501560 6102373}{67703354791390416899107384651170252 8100},$$

$$p_2 = \frac{22552733768024491856640037074192574 87}{59000744916244372025365912550039436 00}\,t^4 - \frac{34811048949731884289969601722962523911 9}{33851677395695208449553692325585126405 0}\,t^3$$

$$+ \frac{60631257463078784542819156925667898391}{96719078273414881284439120930243218300}\,t^2 + \frac{25293180978708547795192839718157 27}{13830842023940351964026758319783100}\,t$$

$$- \frac{2160066846340}{11464617073833},$$

$$p_3 = -\frac{15952846667440940986442428354469281420281211399}{14948917889455551203561858304849170116912045200}\,t^3 + \frac{36546982604328063415870434419840722311473560 91}{18686147361819439004452322881061462646140056 50}\,t^2$$

$$- \frac{16105473218784719275242380230390811611718548457}{22423376834183326805342787457273755175368067 80}\,t - \frac{12484256529331711827822252688089878900804269 87}{44846753668366653610685574914547510350736135 60},$$

*and* $q = t^4 - \frac{350056078}{234205983}\,t^3 - \frac{142948855}{234205983}\,t^2 + \frac{458301406}{234205983}\,t - \frac{212187313}{234205983}$.

*The denominator* $q$ *has* 2 *real roots,* $-1.143$ *and* $1.13$. *Consequently, the curve has* 3 *connected components and so do the surfaces of the implicit equations. We computed the 3 implicit equations in 0.171 sec. Their coefficients are quite large; below we show one equation where the coefficients are rounded.*

$$1.23\,x_3 - 0.0595\,x_3^2 + 1.87\,x_3^2\,x_1\,x_2 + 78.9\,x_1^2\,x_3\,x_2 - 3.18\,x_2^2 - 26.5\,x_3\,x_1\,x_2 - 29.9\,x_3\,x_2^2\,x_1 + 2.24\,x_1 - 1.33\,x_2 - 67\,x_1^3\,x_3 + 124\,x_1^3\,x_2 - 9.1\,x_3\,x_1 + 0.266\,x_3^2\,x_1 +$$

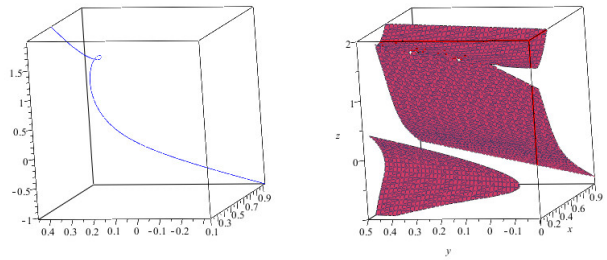


Figure 3: The figures depict the branch of the curve from Example 4 between the two poles (Left) and one of the surfaces computed by the algorithm (Right). The bottom-left component of the surface is "beyond" one of the poles and is disconnected from the other component.

$4.43\,x_3\,x_2^2 + 35.7\,x_1^2\,x_3 + 48\,x_1^3 - 13.2\,x_1^2 + 3.4\,x_3\,x_2 + 13.5\,x_1\,x_2 + 1.11\,x_3^2\,x_2 + 30.4\,x_1\,x_2^2 - 67.7\,x_1^2\,x_2 - 1.32\,x_3^3\,x_2 + 0.469\,x_3^2\,x_2^2 - 0.979\,x_3^3 + 2\,x_3^3\,x_1 + 3.54\,x_3\,x_2^3 - 5.88\,x_1^2\,x_3^2 + 24.9\,x_1\,x_2^3 - 84.6\,x_1^2\,x_2^2 - 4.34\,x_2^3 + 0.354\,x_3^4 - 2.66\,x_2^4 - 65.2\,x_1^4 + 0.00316 = 0.$
*See Fig. 3.*

**Example 5.** *We compare four implicitization algorithms on two space curves $V_1, V_2$ with parameterizations:*

$$\begin{aligned} V_1 : (x_1, x_2, x_3) =& (5\,t^3 - 2\,t^2 + 4, 2\,t^3 - t^2 + 3\,t + 6, -9\,t^3 - 4\,t^2 + 3\,t + 4), \\ V_2 : (x_1, x_2, x_3) =& (-6\,t^5 + 4\,t^4 + 9\,t^3 - 3\,t - 9, -5\,t^5 - 4\,t^4 + 9\,t^2 + 7\,t - 6, \\ & -7\,t^5 + 6\,t^4 - 7\,t^3 - t^2 - 9\,t - 9). \end{aligned}$$

*Tables 3 and 4 summarize the results.*

| Algorithm | Time [ms] | Memory [MB] | Output | Minors [ms] |
|---|---|---|---|---|
| (CF) | 140 | 3.31 | 3 equations of degree 3 | — |
| (MI) | <10 | 0.027 | 11x10 matrix of rank 8 | 65330 |
| (GBb) | 30 | 0.434 | 3 equations of degree 2 | — |
| (Mrep) | 430 | 9.87 | 2x3 matrix of rank 2 | <10 |
| (Impl) | 480 | 12.72 | 3 equations of degree 2 | — |

Table 3: Comparison of the different algorithms on curve $V_1$.

*The implicit polynomials for curve $V_2$ obtained as minors of the matrix by (Mrep) are all of degree 4. Since the entries of the matrices constructed by this method are always linear, the degree of the resulting implicit polynomials equals the rank of the matrix.*

**Example 6.** *Consider the curve in $\mathbb{C}^4$ with parameterization:*

$$(x_1, x_2, x_3, x_4) = (t^2 - t - 1,\ t^3 + 2\,t^2 - t,\ t^2 + t - 1,\ t^3 - 2\,t + 3),\ t \in \mathbb{C}.$$

| Algorithm | Time [sec] | Memory [MB] | Output | Minors [sec] |
|---|---|---|---|---|
| (CF) | 0.39 | 14.82 | 3 equations of degree 5 | — |
| (MI) | 0.02 | 0.179 | 27x26 matrix of rank 22 | >600 |
| (GBb) | 0.15 | 3.08 | 4 equations of degree 3 and 1 equation of degree 4 | — |
| (Mrep) | 0.78 | 18.66 | 4x7 matrix of rank 4 | 0.36 |
| (Impl) | 3.12 | 78.28 | 14 equations of degree 4 | — |

Table 4: Comparison of the different algorithms on curve $V_2$.

*The curve is defined by 5 implicit equations of degrees 1,2,2,2 and 3. Our algorithm computes 5 equations of degree 6 in 0.06 sec. These equations contain linear extraneous factors raised to the power 3. When these are divided out we obtain 5 degree 3 equations which define the curve set-theoretically.*

**Example 7.** *We tested our method on a surface in $\mathbb{C}^4$ with parameterization:*

$$\begin{aligned}
x_1 &= -1 + t_1 - 4\,t_2^2 - 5\,t_2,\\
x_2 &= 2 + 2\,t_1^2 + t_1\,t_2 - 2\,t_1 - 3\,t_2^2 - 4\,t_2,\\
x_3 &= -4 + 2\,t_1^2 - 3\,t_1\,t_2 - 2\,t_1 + 5\,t_2^2 + 3\,t_2,\\
x_4 &= 3 - 4\,t_1^2 - 4\,t_1\,t_2 - 3\,t_1 - 4\,t_2^2 + 3\,t_2.
\end{aligned}$$

*It has an implicit representation defined by 7 equations all of degree 3. When computing the resultant of the hyperplane equations in step 3 of Algorithm 2, we used the Macaulay matrix. While the resultant of the equations is of degree 12, the Macaulay determinant is of degree 15 and factors into 4 polynomials:*

$p_1 = 455\,x_3 + 750\,x_4 - 3099 + 97\,x_2 - 254\,x_1,$

$p_2 = -1231\,x_1^2 - 67428\,x_1 - 1368657 - 7911\,x_1\,x_4 - 238773\,x_4 + 120282\,x_4^2 - 7797\,x_2\,x_1 - 312183\,x_2 + 128844\,x_2\,x_4 + 30214\,x_2^2 - 13361\,x_1\,x_3 - 522219\,x_3 + 216972\,x_4\,x_3 + 98444\,x_2\,x_3 + 81846\,x_3^2,$

$p_3 = -9916630\,x_1\,x_4^3 + 9647284144\,x_2\,x_4^2 + 4038963040\,x_2^2\,x_1 + 337212316\,x_2\,x_4\,x_1^2 - 252641227648\,x_3 + 3366169952\,x_2^2\,x_3^2 + 2708928320\,x_2^3\,x_3 + 57096927668\,x_1 - 53684774940\,x_4 - 77758227688\,x_3^2 + 4536327935\,x_2\,x_3\,x_4^2 + 35030263974\,x_1\,x_3 - 122470859456\,x_2 + 3463514782\,x_2\,x_1^2 + 1675553082\,x_1\,x_2^2\,x_4 + 300052921\,x_2\,x_1^3 + 577364984\,x_1\,x_3^3 + 973095808\,x_2\,x_4^3 + 273084828\,x_4^4 + 475451312\,x_3^4 - 5750362055\,x_1^2 + 5758157904\,x_4\,x_2\,x_3^2 + 226483168\,x_1^2\,x_4^2 + 4022646056\,x_2\,x_1\,x_3\,x_4 + 17577663031\,x_4^2 + 634192304\,x_2^2\,x_3 + 1481456249\,x_2^2\,x_1^2 + 870615518\,x_1^2\,x_3^2 + 87651711\,x_1^3\,x_3 + 8195410478\,x_2\,x_3\,x_1 +$

$11932472704\,x_1\,x_4 + 1052921408\,x_2^4 - 20488463200\,x_2^2 + 9674045090\,x_1\,x_3\,x_4 + 5991087840\,x_4\,x_2^2\,x_3 - 5940646664\,x_2\,x_3^2 + 1965786032\,x_2\,x_3^3 - 263941933984 + 17138674\,x_1^3\,x_4 + 12449275102\,x_2\,x_4\,x_1 - 17345909250\,x_4\,x_3 + 1069083008\,x_2^3 + 707521563\,x_1^3 + 5581659148\,x_4^3 - 6156701992\,x_3^3 + 61103392\,x_1^4 + 532059077\,x_1\,x_4^2\,x_3 + 1965490737\,x_2\,x_3\,x_1^2 + 650614286\,x_1^2\,x_3\,x_4 + 2274927496\,x_1\,x_2\,x_3^2 + 3301748912\,x_1\,x_2^2\,x_3 + 1655223786\,x_1\,x_4\,x_3^2 + 982338067\,x_1\,x_2\,x_4^2 + 3614745108\,x_4^2\,x_3^2 + 1774467334\,x_4^3\,x_3 + 7406737452\,x_2\,x_1 - 17927257812\,x_2\,x_4 - 75703596848\,x_2\,x_3 + 1991048528\,x_4\,x_2^3 + 2510474960\,x_4\,x_3^3 + 10630864230\,x_2\,x_3\,x_4 + 8796789936\,x_2^2\,x_4 + 2284266257\,x_1\,x_4^2 + 7334934310\,x_1\,x_3^2 + 1319302559\,x_1^2\,x_3 + 1991564624\,x_1\,x_2^3 + 1984784289\,x_2^2\,x_4^2 + 17208929291\,x_4^2\,x_3 + 7927974534\,x_4\,x_3^2 + 2480674448\,x_1^2\,x_4,$

$p_4 = (-14670609 - 30942341\,x_3 - 11989497\,x_2 + 4731176\,x_1 - 28710187\,x_4 + 4015732\,x_3^2 + 3763632\,x_2\,x_3 + 933756\,x_2^2 - 1626083\,x_1\,x_3 - 788463\,x_2\,x_1 + 31801\,x_1^2 + 11404020\,x_4\,x_3 + 4795500\,x_2\,x_4 - 2469429\,x_1\,x_4 + 7274552\,x_4^2)^4.$

*Polynomial $p_3$ is irreducible of degree 4 and contains the surface; $p_4$ is a quadratic polynomial raised to the power 4; it is the extraneous factor predicted by the algorithm. The other two factors $p_1, p_2$, of degrees 1 and 2, respectively, are extraneous factors from the Macaulay's matrix construction. A total of 5 polynomials like $p_3$ define the surface set-theoretically.*

**Example 8.** *Algorithm 2 works even in the presence of base points. Consider the surface in $\mathbb{C}^4$ with projective parameterization*

$$(s:t:u) \mapsto \left( \frac{p_1(s:t:u)}{q(s:t:u)}, \frac{p_2(s:t:u)}{q(s:t:u)}, \frac{p_3(s:t:u)}{q(s:t:u)}, \frac{p_4(s:t:u)}{q(s:t:u)} \right),$$

*where:*

$$\begin{aligned} &p_1(s:t:u) = s^2 + t^2 - u^2, \quad p_2(s:t:u) = u(s+t-u), \\ &p_3(s:t:u) = su - (t-u)^2, \quad p_4(s:t:u) = (s-u)^2 + (t-u)^2 - u^2, \\ &q(s:t:u) = st. \end{aligned}$$

*Then $(0:1:1)$ and $(1:0:1)$ are two base points of the surface. The Macaulay matrix computed by the algorithm is of size 15 and of rank 13 while we expect the resultant to be of degree at most 12. Taking a non-zero minor of size 13 yields a polynomial that can be factored into the following:*

- *a factor $\mathcal{S}_V^G(x_0 : x_1 : x_2 : x_3 : x_4)$ of degree 2 that contains the surface,*
- *a factor of degree 2 that is the extraneous factor predicted by the algorithm, raised to a power 3,*
- *another factor of degree 2 and two factors of degree 1, one of which is squared.*

*The presence of the base points indeed reduced both the degree of the implicit equation containing the surface and the degree of the resultant. Thus we obtained more extraneous factors from the Macaulay construction.*

$$S_V^G(x_0 : x_1 : x_2 : x_3 : x_4) = x_2^2 + x_4^2 + 3x_3x_2 - 2x_4x_3 + 3x_3^2 - 2x_0x_2 + 2x_4x_0 - \\ - 5x_0x_3 + 2x_0^2 + 3x_1x_2 - 5x_4x_1 + 9x_3x_1 - 8x_0x_1 + 8x_1^2.$$

*The extraneous factor predicted by the algorithm can be computed without factoring, by using the exterior algebra formulae. For the algorithm, we chose 7 random points: $G$ and $P_{ij}, 0 \leqslant i \leqslant 2, 1 \leqslant j \leqslant 2$. The formulae is the following:*

$$(\xi \wedge G \wedge P_{01} \wedge P_{02}) \cdot (\xi \wedge G \wedge P_{11} \wedge P_{12}) \cdot (\xi \wedge G \wedge P_{21} \wedge P_{22}) = \\ [\det(\xi, G, P_{11}, P_{12}, P_{01})(\xi \wedge G \wedge P_{02}) - \det(\xi, G, P_{11}, P_{12}, P_{02})(\xi \wedge G \wedge P_{01})] \cdot \\ \cdot (\xi \wedge G \wedge P_{21} \wedge P_{22}) = \\ [\det(\xi, G, P_{11}, P_{12}, P_{01}) \det(\xi, G, P_{21}, P_{22}, P_{02}) - \\ - \det(\xi, G, P_{11}, P_{12}, P_{02}) \det(\xi, G, P_{21}, P_{22}, P_{01})](\xi \wedge G)$$

*So the extraneous factor is obtained by computing these four $5 \times 5$ determinants. Note that they can be reduced to $4 \times 4$ determinants since their last row consists of ones.*

**Example 9.** *Algorithm 2 can also handle parametric equations containing additional formal parameters.*

a. *Consider for instance a parameterized cubic:*

$$(x_1, x_2, x_3) = \left( at, bt^2, \frac{a+b}{2} t^3 \right), \, t \in \mathbb{C}.$$

*Algorithm (CF) computes 3 implicit equations of degree 3 in the $x_i$ and of degree 5 in $\{a, b\}$. Those equations match the ones obtained for the twisted cubic when $a = b = 1$.*

b. *Consider the following curve of degree 4 in $\mathbb{C}^3$:*

$$x_1(t) = -at^{12} - t^7 - bt^6 - 8t^3 - t + 9 \\ x_2(t) = (b-3)t^{11} + 3t^{10} - at^8 - 8t^7 + 8t^6 - t \\ x_3(t) = (a-b)t^{12} - 2t^{11} - t^9 - 3t^7 - 7t^6 + t^2 + b$$

*(CF) computes 3 implicit equations of conical surfaces, of degree 12 in the $x_i$, in about 10h. None of the other algorithms tested (Impl), (GBa), (GBb) and (Mrep) was able to implicitize that curve, either because they cannot handle formal parameters ((Mrep)) or because it took too long to compute ((GB) and (Impl)).*

c. *Consider Viviani's curve parameterized as:*
$x_1 = 128\, a^5\, t^2 / (256\, a^8 + 32\, a^4\, t^2 + t^4)$,

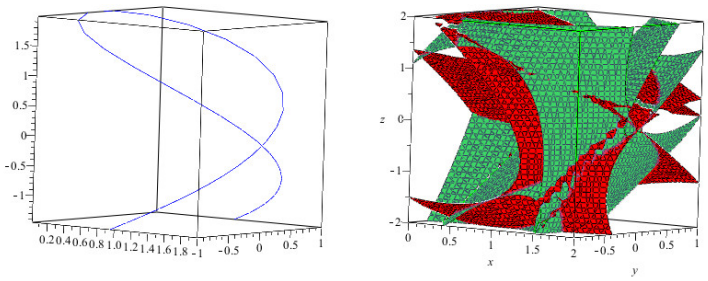


Figure 4: (Left) The curve of Example 9(c.) for $a = 1$. (Right) Two of its three defining equations found by Algorithm 2. A third equation is needed in order to remove the extraneous surface intersections.

$x_2 = (256\,a^7\,t - 16\,a^3\,t^3)/(256\,a^8 + 32\,a^4\,t^2 + t^4)$,
$x_3 = (32\,a^5 - 2\,a\,t^2)\,(16\,a^4 + t^2)/(256\,a^8 + 32\,a^4\,t^2 + t^4)$.
*Its implicit equations are:* $x_1^2 + x_2^2 + x_3^2 = 4\,a^2$, $(x_1 - a)^2 + x_2^2 = a^2$.

*The 3 implicit equations computed by Algorithm 2 in 7,72 sec are of total degree 4 in $x_i$. Since they are quite large, we show only one of them which is of degree 74 in $a$ and omit most of its terms:*

$281474976710656\,a^{36}\,(-1048576\,a^{19}\,x_1\,x_3^2 - 4\,a^4\,x_3^2 - 393216\,a^{19}\,x_1^2\,x_3^2 + \cdots + 524288\,a^{18}\,x_3^2\,x_1^2 - 786432\,a^{18}\,x_3\,x_1^3) = 0$.
*See Fig. 4.*

## 7. Future work

Interesting mathematical questions arise in studying the complexity of our method, namely how to find tighter bounds for the degree of the computed conical implicit surfaces and how to bound their number in the general case by results similar to Theorem 6. This also requires to study in more detail the extraneous factors of the computed equations and to distinguish them into those coming from the chosen method to compute the resultant, and those inherited in our implicitization method.

As it stands, the algorithm chooses random vertices $G$ for the hypersurfaces. A question arises of how to choose these vertices more efficiently, either by using a better random distribution or by finding particularly interesting points. The answer may depend on the tasks at hand: checking membership is faster when the coefficients are integral and of small size, computing intersections is easier when the same vertices are always used, when evaluating the distance to the variety perpendicular hypersurfaces are better, etc.

A possible improvement of the algorithm at least in 3D space, could be to always compute cylindrical hypersurfaces instead of conical hypersurfaces.

For space curves, we can reduce the computation of three conical surfaces to the computation of three cylinders by using projective linear maps, as in the proof of Theorem 6. The computation of a cylinder in $\mathbb{P}^3$ is equivalent to the computation of its intersection with a perpendicular plane, i.e., a planar curve. Using cylinders would thus allow the use of methods that are specialized to planar curve implicitization.

An interesting direction for future research concerns the possibility of defining a sparse version of the Chow form. In Corollary 2, consider the polynomials in $(t_0 : \cdots : t_d)$ obtained after substituting in the equations of the hyperplanes $H_i$, the $x_i$'s by the parametric expressions. These $d+1$ polynomial equations in $d+1$ homogeneous variables may have Newton polytopes smaller than the simplex of size $\delta$ and thus be suitable for a sparse resultant computation. In the sparse resultant method, the monomial bases of the input polynomials are analyzed very closely to deduce a sparse basis of the output resultant. By analogous means, we expect that a sparse Chow form of an input variety $V$ of codimension $c$ in $\mathbb{P}^n$ would be a hypersurface of a carefully picked subspace of $Gr(c, n+1)$. In Definition 2, the linear forms $H_i$ would have fewer degrees of freedom (by forcing a linear relation between some of the parameters $u_{ij}$, for instance). It is not obvious how to correctly pick subspace $D \subset Gr(c, n+1)$ such that $Z_D(V) \coloneqq \{L \in D \mid L \cap V \neq \emptyset\}$ would still satisfy Proposition 1. This potential generalization of the Chow form may lead to better algorithms for computing Chow forms.

**Acknowledgements:** All authors belong to team AROMATH, joint between INRIA Sophia-Antipolis (France) and NKUA. This project has received funding from the European UnionâĂŹs Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 675789 (ARCADES). We thank L. Busé and B. Mourrain for discussions.

[1] C. Bajaj and I. Ihm. Hermite interpolation of rational space curves using real algebraic surfaces. In *Proc. ACM Symp. on Comput. Geometry*, pages 94–103, 1989.

[2] L. Busé. Implicit matrix representations of rational Bézier curves and surfaces. *J. CAD*, 46:14–24, 2014. Spec. Issue 2013 SIAM Conf. Geometric & Physical Modeling.

[3] L. Busé and T. Luu Ba. The surface/surface intersection problem by means of matrix based representations. *J. CAGD*, 29(8):579–598, 2012.

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

[5] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Number 185 in GTM. Springer, New York, 2nd edition, 2005.

[6] J. Dalbec and B. Sturmfels. Introduction to chow forms. In N. L. White, editor, *Invariant Methods in Discrete and Computational Geometry: Proc. Curaçao Conference, 13–17 June, 1994*, pages 37–58. Springer, 1995.

[7] C. D'Andrea. Macaulay-style formulas for sparse resultants. *Trans. Amer. Math. Soc.*, 354:2595 – 2629, 2002.

[8] C. D'Andrea and A. Dickenstein. *Explicit formulas for the multivariate resultant*. Journal of Pure and Applied Algebra, pages 59–86, 2001.

[9] T. Dokken. Approximate implicitization. In *Mathematical methods for curves and surfaces (Oslo 2000)*, Innov. Appl. Math., pages 81–102. Vanderbilt Univ. Press, Nashville, 2001.

[10] I. Z. Emiris, V. Fisikopoulos, C. Konaxis, and L. Peñaranda. An oracle-based, output-sensitive algorithm for projections of resultant polytopes. *Intern. J. Comp. Geometry & Appl.*, 23:397–423, 2014.

[11] I. Z. Emiris, T. Kalinka, and C. Konaxis. Geometric operations using sparse interpolation matrices. *Graphical Models*, 82:99–109, Nov. 2015.

[12] I. Z. Emiris, T. Kalinka, C. Konaxis, and T. Luu Ba. Sparse implicitization by interpolation: Characterizing non-exactness, and an application to computing discriminants. *J. CAD*, 45:252–261, 2013.

[13] I. Z. Emiris, C. Konaxis, I. Kotsireas, and C. Laroche. Matrix representations by means of interpolation. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, pages 149–156, New York, NY, USA, 2017. ACM.

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

[15] G. Jeronimo, T. Krick, J. Sabia, and M. Sombra. The computational complexity of the chow form. *Foundations of Computational Mathematics*, 4(1):41–117, Feb 2004.

[16] E. Kunz. *Introduction to Commutative Algebra and Algebraic Geometry*. Modern Birkhäuser Classics. Birkhäuser Basel, 2013.

[17] L. Kronecker. *Grundzüge einer arithmetischen Theorie der algebraischen Grössen.*. J. reine angew. Math. 92 (1882), 1-123.

[18] D. Manocha and J. F. Canny. Algorithms for implicitizing rational parametric surfaces. *J. CAGD*, 9(1):25–50, 1992.

[19] D. Manocha and J. F. Canny. The implicit representation of rational parametric surfaces. *J. Symbolic Computation*, 13:485–510, 1992.

[20] S. L. Rueda, J. Sendra, and J. R. Sendra. An algorithm to parametrize approximately space curves. *J. Symbolic Computation*, 56:80 – 106, 2013.

[21] T. Sederberg and F. Chen. Implicitization using moving curves and surfaces. In R. Cook, editor, *Proc. SIGGRAPH*, pages 301–308. Addison Wesley, 1995.

[22] I. R. Shafarevich and A. O. Remizov. *Linear Algebra and Geometry.* Springer, New York, 2012.

[23] J. Shen, L. Busé, P. Alliez, and N. Dodgson. A line/trimmed NURBS surface intersection algorithm using matrix representations. Technical report, INRIA, 2016.

[24] G. Staglianò. Macaulay2 package "resultants". Available at `http://www2.macaulay2.com/Macaulay2/doc/Macaulay2-1.14/share/doc/Macaulay2/Resultants/html/index.html`.

[25] B. Sturmfels. *Algorithms in Invariant Theory.* Texts and Monographs in Symbolic Computation. Springer, 2008.