# Changing Representation of Curves and Surfaces: Exact and Approximate Methods.

Tatjana Kalinka*

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
kalinkat@di.uoa.gr

**Abstract.** In this thesis we explore recent methods for computing the Newton polytope of the implicit equation and study their applicability to the representation change from the parametric form to implicit. Computing a (super)set of the monomials appearing in the implicit equation allows us to determine the interpolation space. Following this phase we implement interpolation by exact or numeric linear algebra (applying singular value decomposition). We evaluate the monomials at the points, most suitable for the task, thus building a numeric matrix, ideally of corank 1, whose kernel vector contains the coefficients of the implicit equation. We propose techniques for handling the case of higher corank. This yields an efficient, output-sensitive algorithm for computing the implicit equation. The method can be applied to polynomial or rational parameterizations of planar curves or (hyper)surfaces of any dimension including parameterizations with base points. Moreover, this technique can be used for problems such as the computation of the discriminant of a multivariate polynomial or the resultant of a system of multivariate polynomials.

**Keywords:** implicitization, interpolation, Newton polytope, sparse resultant, linear algebra.

## 1 Introduction

The modern CAGD and CAM systems operate with several different representations of geometric objects, where each is more suitable for some applications. For instance, parametric and implicit representations have complementary features: with parametrization it is easy to obtain points on the geometric object while the implicit equation allows to check quickly if a given point is inside or outside a given object. Hence the need for the robust methods to change between the two representations.

In this thesis we focus on implicitization, i.e. process of changing the representation of a geometric object from parametric to implicit (algebraic). The main objective of our work was exploring applicability of the recently developed method for computing Newton polytope of a resultant [1,2] to computing the implicit equation.

**Definition 1.** *Given a polynomial* $f = \sum_a c_a t^a \in \mathbb{R}[t_1, \ldots, t_n]$, $t^a = t_1^{a_1} \cdots t_n^{a_n}, a \in \mathbb{N}^n, c_a \in \mathbb{R}$, *its support is the set* $\{a \in \mathbb{N}^n : c_a \neq 0\}$; *its Newton polytope* $N(f)$ *is the convex hull of its support.*

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:

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

* Dissertation Advisor: Ioannis Z. Emiris, Professor

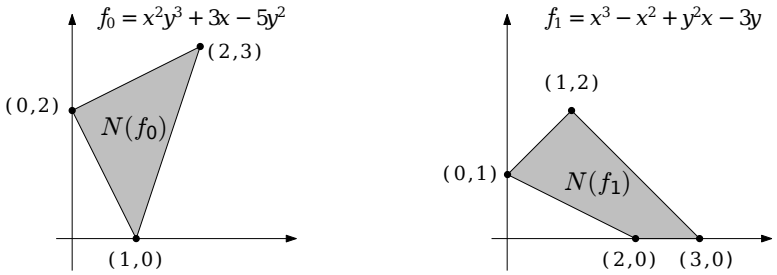


**Fig. 1.** Newton polygons $N(f_i)$ of polynomials $f_i \in \mathbb{Z}[x, y]$.

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 $\Omega := \Omega_1 \times \cdots \times \Omega_n,\ \Omega_i \subseteq \mathbb{R}$.

**Definition 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 $\Omega$. We restrict ourselves to the case when this is a principal ideal, and we wish to compute its defining polynomial*

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

*given its Newton polytope, or a polytope that contains it.*

There have been numerous approaches to implicitization, including those based on Gröbner bases, resultants, residues, moving lines and surfaces, and $\mu$-bases. Our approach, presented in [3,4,5,6], follows the standard method of interpolating the unknown coefficients of the implicit polynomial given a superset of its monomials.

Using the recently developed method to determine potential monomials of the implicit equation we build a numeric matrix, ideally, of codimension 1, whose kernel yields (up to a nonzero scalar multiple) the coefficients corresponding to the predicted implicit support. This is a standard case of *sparse interpolation* of the polynomial from its values.

Since the kernel can be computed numerically, our approach also yields an approximate sparse implicitization method.

The kernel space of the numerical matrix may be of high dimension. We relate it to the geometry of the predicted support, which is a superset of the true implicit support. Another reason for obtaining a high-dimensional kernel is that the numeric evaluation of the support monomials may not be sufficiently generic.

Our implicitization method can be applied to planar curves, surfaces, or hypersurfaces of any dimension, given by a polynomial, rational or trigonometric parametrization, including those with base points. Moreover, our method can be used to compute discriminants of well-constrained systems as well as resultants, since the latter can be viewed as a special case of discriminants.

### 1.1 Prior work

Our implicitization algorithm is closely related to the interpolation-based method presented in [7]. We employ the same, most direct, method to reduce implicitization to linear algebra: 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, and compute kernel vector $p$ of $M$ containing coefficients of the implicit equation. This idea was used also in [8,9]; the approach was extended to an approximate implicitization as well.

Evaluation of the potential monomials at unitary $\tau \in (\mathbb{C}^*)^n$, one of the evaluation strategies examined in our work, was proposed in [10], Another approach, described in [11], is based on integration of matrix $M = SS^T$, over each parameter $t_1, \ldots, t_n$. Then, $p$ is in the kernel of $M$. This method covers a wide class of parametrizations, including polynomial, rational, and trigonometric representations, but the size of $M$ is large and matrix entries take big values, so it is difficult to control its numeric corank.

In practical applications of CAGD, precise implicitization often can be impossible or very expensive to obtain. Moreover, exact implicit equations usually are of high degree and contain unwanted branches and singularities. Hence the need for approximate implicitization proposed by T. Dokken [12,13]. The idea is to interpolate the coefficients using successively larger bounds on the total degree of the target polynomial, starting with a quite small support and extending it until a satisfying accuracy of the approximation is reached.

In order to determine the space of interpolation we have used the Newton polytope of the implicit equation, or *implicit polytope*.

There are several methods for computing the implicit polytope, such as those based on tropical geometry, or mixed fiber polytopes, for instance [14,15,9]. In this thesis the implicit polytope is computed from the Newton polytope of the sparse (or toric) resultant, or *resultant polytope*, of polynomials defined by the parametric equations. In particular, we use software tool `ResPol` [2] to compute the implicit support in general case and, in the case of curves, the method presented in [1] and implemented in `Maple`. We shall note, however, that our implicitization method does not depend on the approach used to compute the implicit polytope.

## 2 Algorithm and implementation

In this section we present our implicitization algorithm, discuss the importance of suitable evaluation points for building the matrix and give some details on the implementation.

The main steps of our algorithm are the following:

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

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

1. Determine (a polytope containing) the implicit polytope.
2. Compute all lattice points $S \subseteq \mathbb{N}^{n+1}$ in the polytope.
3. Repeat $\geq |S|$ times: Select value $\tau$ for $t$, evaluate $x_i(t)$, $i = 0, \ldots, n$, thus evaluating each monomial with exponent in $S$. This yields a matrix $M$.
4. Given matrix $M$, solve $Mp = 0$ for kernel $p$.
5. Let the kernel vector $p_i$ correspond to a polynomial of least total degree.
6. Return the primitive part of polynomial $p_i^\top \cdot m$, where $m$ is the set of monomials with exponent in $S$.

The complexity of our algorithm is $O(\mu\,|S|^2)$.

Let us describe the construction of matrix $M$ in Step 3.

Consider $S := \{s_1, \ldots, s_{|S|}\}$; 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 $x_i = f_i(t)/g_i(t)$. We evaluate $m_j$ at some $\tau_k$, $k = 1, \ldots, \mu$, avoiding values that make the denominators of the parametric expressions

close to 0, and obtain $m_j|_{t=\tau_k} := \prod_i \left(\frac{f_i(\tau_k)}{g_i(\tau_k)}\right)^{s_{ji}}$. Thus, we build an $\mu \times |S|$ matrix $M$ with rows indexed by $\tau$'s and columns indexed by $m_j$'s:

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

We compute the kernel of the matrix $M$ either symbolically or numerically, by applying singular value decomposition (SVD).

### 2.1 Choosing the evaluation points

Experiments with curves and surfaces in the monomial basis as well as in the Bernstein basis, show that when building the matrix $M$, it is important to choose $\tau$ values that are suitable for the specific instance.

Choosing $\tau$ for implicitization of classical algebraic curves and surfaces, we have experimented with

- random integers in the range $-\mu^2 \dots \mu^2$,
- random rational numbers,
- complex $\mu$-th roots of unity,
- random complex numbers modulo 1.

Random integers offer the most numerically stable results, however with large matrices they result in fast growth of matrix entries. Random rational values have proved to be unreliable when implicitizing classical algebraic curves and surfaces, although complex values are numerically stable.

In the case of curves and surfaces in the Bernstein basis we have used evaluation by

- random rational numbers,
- uniformly distributed rational numbers,
- complex roots of unity,
- Chebyshev nodes in $[0, 1]$:

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

While for classical algebraic curves and surfaces rational numbers led to a loss of numerical stability, here rational numbers chosen randomly in $[0, 1]$ provide the fastest results.

Our experiments affirm the results of [13], as the evaluation with Chebyshev nodes allows to minimize the approximation error in numerical computations. Complex roots of unity gave the slowest timings and introduced complex coefficients into the resulting approximate implicit equation.

### 2.2 Implementation

Our algorithm is implemented in `Maple`[1] and `SAGE`, based on the software for computing implicit polytopes [2], available as a C++ implementation[2]. The main functions are *imgen* (general

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

implicitization, applicable for curves, surfaces and hypersurfaces, requires parametric equations and predicted polytope vertices as an input) and *imcurve* (for curves only, support prediction is part of the routine).

For exact computations we prefer `Maple`, while for numerical ones `SAGE`. In our `Maple` implementation the computation of the lattice points in Step 2 is done, for up to four dimensions, by routines that utilize the `Maple` package `convex` [16], whereas our `SAGE` implementation uses its built-in functions for the same task. For higher dimensions we have employed the software package `Normaliz`.

When the kernel computation in Step 4 is done numerically, we build a rectangular overconstrained matrix $M$ in order to increase the numerical stability.

### 2.3 Accuracy of the approximate implicitization

It is important to estimate the numeric accuracy, or quality, of the result when solving numerically. We use the matrix condition number and the ratio between the two smallest singular values to evaluate the error in the coefficient vector computed by SVD.

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

(a) Coefficient difference: measured as the Euclidean norm of the difference of the two coefficient vectors $V_{exact}, V_{app}$, obtained from exact and approximate implicitization, after padding with zero the entries of each vector which do not appear in the other.
(b) Evaluation norm: measured by considering the maximum norm of the approximate implicit equation when evaluated at a set of sampled points on the given parametric object.

Both accuracy measures feature in Table 2 where we compare running time and accuracy of our approximate implicitization against another method.

## 3 Results

In this section we discuss some of the key results of our work. First we prove the relation between the size of the predicted implicit polytope and the presence of the extraneous factors in the resulting expression. Next, we also talk about performance of the implementation of our method compared with others and present examples of alternative, non-geometrical application of the method.

### 3.1 The extraneous factors cases

By the construction of matrix $M$ using values $\tau$ that correspond to points on the parametric surface, we have the following:

**Lemma 1.** *Any polynomial in the basis of monomials indexing $M$, with coefficient vector in the kernel of $M$, is a multiple of the implicit polynomial $p$.*

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

**Theorem 1.** *Let $P = N(p)$ be the implicit polytope and $Q$ the predicted polytope. Then, assuming $M$ has been built using sufficiently generic evaluation points, the dimension of its kernel space equals $\#\{m \in \mathbb{Z}^n : m + P \subseteq Q\} = \#\{m \in \mathbb{Z}^n : N(x^m \cdot p) \subseteq Q\}$.*

We assume genericity of the resultant whose symbolic coefficients are then specialized to the actual coefficients of the parametric equations. If this does not hold, then the actual implicit equation divides the specialized resultant.

In order to produce the exact implicit equation in the instance when the matrix $M$ has corank $> 1$ we propose the following:

- Reduce the predicted Newton polytope.
- Compute gcd of two or more polynomials corresponding to kernel vectors. In case of numeric solving approximate methods for computing the gcd can be applied.
- Apply factoring, then determine which of the factors vanishes when the $x_i$ variables are substituted by the parametric expressions.
- In practice, an actual implicit equation usually is present among the polynomials corresponding to kernel vectors. Hence the solution: sort the polynomials, return the one of the least degree.

*Example 1.* Consider its parametrization:

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

Predicted implicit polytope has vertices: $(0,0,0)$, $(0,0,2)$, $(0,0,4)$, $(0,2,0)$, $(0,4,0)$, $(4,0,0)$. Implicit equation of the sphere being quadratic, here implicit polytope $P \subset Q$, where $Q$ is predicted polytope, which contains the actual implicit polytope. It contains 35 lattice points. We build $M$ of size $\mu \times 35\,(\mu \geq 35)$ of corank 10. The polynomials corresponding to the kernel vectors are:
$g_1 = -y^2 + y^2z^2 + y^4 + x^2y^2$,
$g_2 = -z^2 + z^4 + y^2z^2 + x^2z^2$,
$g_3 = -1 + z^2 + x^2 + y^2$,
$g_4 = -x + xz^2 + xy^2 + x^3$,
$g_5 = -yz + yz^3 + y^3z + x^2yz$,
$g_6 = -y + yz^2 + y^3 + x^2y$,
$g_7 = -xz + xz^3 + xy^2z + x^3z$,
$g_8 = -z + z^3 + y^2z + x^2z$,
$g_9 = -xy + xyz^2 + xy^3 + x^3y$,
$g_{10} = -1 + 2z^2 - z^4 + 2y^2 - 2y^2z^2 - y^4 + x^4$.

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

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

The approximate gcd of the first two is:
$-0.9999998548199414+0.9999999857259533x^2+1.000000000052092y^2+1.000000000000000z^2$, which is accurate to 7 decimal digits.

### 3.2 Comparison to other methods

Here we report on a comparison of our method, implemented in `Maple`, against existing implicitization software. All the experiments mentioned have been performed on an Intel©Core2 Duo CPU, 2.20GHz, 3Gb memory, `Maple 14`.

Table 1 features the running times for implicitization of some examples of algebraic curves by different methods, all implemented in `Maple`. Namely our function *imcurve*, only for curves, that includes support prediction routine, $\mu$-bases method only for curves [17], and *Maple* function *Implicitize*, which employs integration of matrix $M$ over each parameter [11] and can be run in exact and numerical mode.

| Curve | degree | *Implicitize* exact | *Implicitize* numeric | Our software | $\mu$-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 1.** Comparing runtimes (sec) of: Maple function *Implicitize* (exact and numeric), our method, and $\mu$-bases.

Of the three methods *Implicitize*, even in numerical mode, is the slowest, however the method has less restrictions on the parametrization accepting non-rational representations. Our method is faster than *Implicitize* but slower than the $\mu$-bases method.

While in case of curves our method may not be the best choice, experiments show that for geometric objects of higher degree and dimension it is competitive to the popular Gröbner bases method.

Consider Table 2, where we show the results of our experiments with surfaces. Here we compare our `Maple` implementation *imgen* against `Maple`'s native function *Implicitize* in numerical mode and implicitization using Gröbner bases in `Maple`.

The input consists of a family of classical algebraic surfaces, the so called Plücker's conoid. $x_0 = t,\ x_1 = s,\ x_2 = \frac{Re((t+I\cdot s)^a)}{|(t+I\cdot s)^a|}$. By choosing appropriate values of parameter $a = 2b$ we obtain rational parameterizations of the surfaces with desired total degree. While implicitization of the Plücker's conoid is trivial task, we have chosen this example to demonstrate robustness of our method when the properties of the surface family allow us to compute comparatively small implicit polytope. This is the reason our exact implicitization shows here better results that the Gröbner bases method. We should note that, compared with the `Implicitize` function, our approximate method is not only faster but also more precise (we use accuracy measures (a) and (b) as defined in [2.3]).

In general, the results of our experiments show that for low degree curves ($\leq 6$) or surfaces ($\leq 4$), Gröbner bases outperform our software. The situation is reversed for higher degree: for instance, the ranunculoid curve (degree 12) was computed in 1.3 sec. by our method and in 7.3 sec. using Gröbner bases. For the standard benchmark of the bicubic surface (degree 18) the timings are 42 min. and over 4 hours, respectively.

### 3.3 Non-geometrical applications

Our algorithm finds other applications besides the implicitization. Since the support prediction software `ResPol` actually computes a resultant support, its straightforward application is to

**Table 2.** Comparison of our method (exact and numerical) to `Maple`'s function *Implicitize()* and Gröbner bases. Runtimes are given in seconds.

| Surface | Our exact | Gröbner | Our numerical | | | `Implicitize`(numerical) | | |
|---|---|---|---|---|---|---|---|---|
| degree | runtime | | runtime | accuracy (a) | accuracy (b) | runtime | accuracy (a) | accuracy (b) |
| 3 | 0.016 | 0.031 | 0.031 | $10^{-15}$ | $9.07 \cdot 10^{-10}$ | 46.07 | $10^{-15}$ | $1.98 \cdot 10^{-9}$ |
| 5 | 0.016 | 0.046 | 0.032 | $10^{-10}$ | $3.57 \cdot 10^{-8}$ | 85.43 | $3.67 \cdot 10^{-7}$ | $6.83 \cdot 10^{-6}$ |
| 7 | 0.031 | 0.078 | 0.046 | $10^{-11}$ | $9.97 \cdot 10^{-8}$ | 359.49 | $9.06 \cdot 10^{-7}$ | $2.94 \cdot 10^{-4}$ |
| 9 | 0.046 | 0.078 | 0.063 | $10^{-10}$ | $1.35 \cdot 10^{-7}$ | 695.65 | $2.86 \cdot 10^{-6}$ | $7.55 \cdot 10^{-3}$ |
| 11 | 0.078 | 0.141 | 0.078 | $10^{-11}$ | $1.07 \cdot 10^{-6}$ | $> 2000$ | - | - |

reduce resultant computation to interpolation; this is also the premise of [18,19]. The main difference with interpolating the implicit equation is the absence of a parametric form, however, the latter can be derived using Horn-Kapranov parametrization [20], as demonstrated below.

*Example 2.* 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 [2] 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, $-t_1^2t_3^4t_5^2(-2t_1-2t_2)^2t_4^2$, $t_1^2t_3^4t_5^2t_2^2t_4^2$, $-t_1^2t_3^4t_5^2t_2t_4^2(2t_1+t_2)$, $(2t_1+t_2)^2t_3^4t_4^2t_1^2t_5^2$, evaluate at 4 sufficiently random $t_i$'s, and obtain a matrix whose kernel vector $(1, 1, -2, 1)$ yields $\mathcal{R} = a_1^2b_1b_0 + a_0^2b_1^2 - 2a_2a_0b_1b_0 + a_2^2b_0^2$.

Another possible application is computing the discriminant of a multivariate polynomial.

Computation of the discriminant is a difficult problem, since explicit formulas only exist for low-degree uni-variate polynomials. We reduce discriminant computation to sparse implicitization.

**Definition 3.** *$A$-discriminant is an irreducible polynomial $D_A = D_A(c)$ with integer coefficients in the vector of coefficients $c = (c_a : a \in A)$, defined up to sign, which vanishes for each choice of $c$ for which $F_A$ and all $\partial F_A/\partial t_i$ have a common root in $(\mathbb{C}\backslash\{0\})^n$.*

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

Let $B = (b_{ij}) \in \mathbb{Z}^{n\times(m-n-1)}$ be a matrix whose column vectors are a basis of the integer kernel of matrix $A$. Then $B$ is of full rank. Since the first row of $A$ equals $(1, \ldots, 1)$, the column vectors of $B$ add up to 0.

We illustrate computation of $A$-discriminant by the following example from [5].

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

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

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

We build the matrix A:

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

then the matrix B is as follows:

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

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

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

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

We perform substitution following

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

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

## 4 Conclusions

We have developed an algorithm for computing implicit equations that combines linear algebra with promising support prediction methods. The method applies to polynomial, rational and trigonometric parameterizations of classical algebraic equations of curves and (hyper)surfaces. Moreover, it can be used for implicitization of geometric objects represented in NURBS form, after converting them to the monomial base. The method works even in the presence of base points, which are known to raise important issues for some other implicitization methods.

Our method has its limits: geometric objects have to be presented using the monomial basis and in the case of trigonometric parameterizations they have to be convertible to rational functions.

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

While our implicitization algorithm was intended and applied in this work to implicitize curves and surfaces of codimension 1 only, interpolation can be sufficiently applied to implicitize space curves; this can be a challenge for future work.

Moreover, it is possible that our algorithm can be adapted in a way that, while bypassing actual computing of the equation, effectively provides an answer if the point belongs to the the geometric object. Our method represents an implicit (hyper)surface by a kernel vector. It is challenging to devise suitable CAGD algorithms that exploit this representation, for instance to compute surface-surface intersection, as in [12,21].

Yet another topic for future work is approximate implicitization of piecewise parametrized Bézier and NURBS curves and surfaces. In this thesis we have limited our experiments to implicitization of single patches of the objects represented in Bernstein basis, however the same interpolation principle can be applied to computing implicit equations of curve or surface splines. As

demonstrated in the thesis, the need for conversion from Bernstein basis to power basis presents a major drawback: in the NURBS format curves and surfaces are usually given by floating point coefficients and recalculating the parametrization in power basis furthers the precision loss.

## References


1. Emiris, I. Z., Konaxis, C.,Palios, L.: 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)
2. Emiris, I. Z., Fisikopoulos, V., Konaxis, C., Peñaranda, L.: An output-sensitive algorithm for computing projections of resultant polytopes. In: Proceedings of the 2012 symposuim on Computational Geometry SoCG '12 New York, NY, USA: ACM. Final version to appear in IJCGA pp. 179–188 (2012)
3. Emiris, I. Z., Kalinka, T., Konaxis, C.: Implicitization of curves and surfaces using predicted support. In: Electr. Proc. Inter. Works. Symbolic-Numeric Computation San Jose, Calif. (2011)
4. Emiris, I. Z., Kalinka, T., Konaxis, C., Luu Ba, T.: Implicitization of curves and (hyper)surfaces using predicted support. Theoretical Computer Science (2012)
5. Emiris, I. Z., Kalinka, T., Konaxis, C., Luu Ba, T.: Sparse implicitization by interpolation: Characterizing non-exactness and an application to computing discriminants. Computer-Aided Design. 45(2), 252–261 Special Issue Conference on SPM (2013)
6. Emiris, I. Z., Kalinka, T., Konaxis, C.: Sparse implicitization via interpolation. To appear in SAGA Volume (Springer) (2013).
7. Emiris, I. Z., Kotsireas, I. S.: Implicit polynomial support optimized for sparseness. In: Proc. Intern. Conf. Computational science appl.: Part III Berlin: Springer. pp. 397–406 (2003)
8. Marco, A., Martínez, J.-J.: Implicitization of rational surfaces by means of polynomial interpolation. CAGD. 19, 327–344 (2002)
9. Sturmfels, B., Yu, J.: Tropical implicitization and mixed fiber polytopes. In: Software for Algebraic Geometry, volume 148, of IMA Volumes in Math. & its Applic. pp. 111–131 Springer New York (2008)
10. Sturmfels, B., Tevelev, J., Yu, J.: The Newton polytope of the implicit equation. Moscow Math. J. 7(2) (2007)
11. Corless, R. M., Giesbrecht, M., Kotsireas, I. S., Watt, S. M.: Numerical implicitization of parametric hypersurfaces with linear algebra. In Proc. AISC. pp. 174–183 (2000)
12. Dokken, T., Thomassen, J. B.: Overview of approximate implicitization. Topics in algebraic geometry and geometric modeling. 334, 169–184 (2003)
13. Barrowclough, O. J. D., Dokken, T.: Approximate implicitization of triangular Bézier surfaces. In Proceedings of the 26th Spring Conference on Computer Graphics SCCG New York, NY, USA. pp. 133–140 (2010)
14. D'Andrea, C., Sombra, M.: The Newton polygon of a rational plane curve. Math. in Computer Science. 4(1), 3–24 (2010)
15. Esterov, A., Khovanskiĭ, A.: Elimination theory and newton polytopes. arXiv:0611107[math] (2006)
16. Franz, M.: Convex: a maple package for convex geometry, version 1.1.3. Available at: http://www-math.uwo.ca (2009)
17. Busé, L.. Luu Ba, T.: Matrix-based implicit representations of algebraic curves and applications. Computer Aided Geometric Design. 27(9), 681–699 (2010)
18. Cueto, M. A., Dickenstein, A.: Some results on inhomogeneous discriminants. In Proc. XVI Latin Amer. Algebra Colloq., Bibl. Rev. Mat. Iberoamericana. arXiv:math/0610031v2 [math.AG] pp. 41–62 (2007)
19. Tanabé, S.: On Horn-Kapranov uniformisation of the discriminantal loci. Adv. Studies Pure Math. 46, 223–249 (2007)
20. Kapranov, M.: A characterization of A-discriminantal hypersurfaces in terms of the logarithmic Gauss map. Mathematische Annalen. 290, 277–285 (1991)
21. Dokken, T., Thomassen, J. B.: Weak approximate implicitization. In Proc. IEEE Intern. Conf. Shape Modeling Appl. p. 31 (2006)