Project number IST-25582

CGL

Computational Geometric Learning

An output-sensitive algorithm for computing projections of resultant polytopes

STREP

Information Society Technologies

Period covered:	November 1, 2010–October 31, 2011
Date of preparation:	October 27, 2011
Date of revision:	October 27, 2011
Start date of project:	November 1, 2010
Duration:	3 years
Project coordinator name:	Joachim Giesen (FSU)
Project coordinator organisation:	Friedrich-Schiller-Universität Jena Jena, Germany

An output-sensitive algorithm for computing projections of resultant polytopes

Ioannis Z. Emiris* Vissarion Fisikopoulos* Christos Konaxis*

October 27, 2011

Abstract

We develop an incremental algorithm to compute the Newton polytope of the resultant, aka resultant polytope, or its orthogonal projection along a given direction. Given $n+1$ polynomials in n variables, expressed by their supports in $\mathbb{Z}^n$, the Cayley trick defines a new pointset $\mathcal{A} \subset \mathbb{Z}^{2n}$. Its regular triangulations can be realized as vertices of the secondary polytope of $\mathcal{A}$; we consider an equivalence relation on the secondary vertices based on the volume of mixed simplices of the corresponding triangulations. The class representatives are vertices of the resultant polytope. Alternatively, the latter is a Minkowski summand of the secondary polytope. Our algorithm avoids walking on the secondary polytope, and is output-sensitive in the sense that it computes one regular triangulation per equivalence class.

Our method computes both V- and H-representations of the polytope in the exact computation model. The code is CGAL-based, and also yields a triangulation of the polytope, which is useful for the targeted applications; it is shown to be competitive to recent software implementing tropical geometry. The resultant is the most fundamental tool in variable elimination and is instrumental in implicitizing parametric hypersurfaces: our code computes the Newton polytope of a bicubic surface's implicit equation in 30 msec, whereas the secondary polytope is intractable. It runs in the order of seconds or minutes in computing projections of resultant polytopes in 5-8 dimensions, hence we expect that it leads to competitive algorithms for interpolating such resultants; the projection corresponds to specializing some of the resultant's parameters.

Keywords. regular triangulation, mixed subdivision, secondary polytope, general dimension, Newton polytope, resultant, practical complexity, CGAL implementation

1 Introduction

Given pointsets $A_0, \ldots, A_n \subset \mathbb{Z}^n$, we define the Cayley pointset

$$\mathcal{A} := \bigcup_{i=0}^{n} (A_i \times \{e_i\}) \subset \mathbb{Z}^{2n}, \tag{1}$$

where e_i are an affine basis of $\mathbb{R}^n$, e.g. $e_0 = \mathbf{0}, e_i = (0, \ldots, 0, 1, 0, \ldots, 0), i \geq 1$. Clearly, $|\mathcal{A}| = |A_0| + \cdots + |A_n|$, where $|\cdot|$ denotes cardinality. By Cayley's trick, the regular mixed subdivisions of the Minkowski sum $A_0 + \cdots + A_n$ are in bijection with the regular triangulations of $\mathcal{A}$. The latter are the vertices of the *secondary polytope* $\Sigma(\mathcal{A})$ of $\mathcal{A}$.

The *Newton polytope* of a polynomial is the convex hull of its *support*, i.e. the exponent vectors corresponding to monomials with nonzero coefficient. Given $n+1$ polynomials in n variables, with fixed supports A_i and symbolic coefficients, their *sparse (or toric) resultant* $\mathcal{R}$ is

*National and Kapodistrian University of Athens, Department of Informatics and Telecommunications, Athens, Greece. {emiris,vissarion,ckonaxis}@di.uoa.gr

a polynomial in these coefficients which vanishes exactly when the polynomials have a common root. The resultant is the most fundamental tool in elimination theory, and is instrumental in system solving; it is also important in implicitizing parametric hypersurfaces. The Newton polytope of the resultant, or *resultant polytope*, is the object of our study, especially when some of the input coefficients are not symbolic, in which case we seek a projection of the resultant polytope.

We exploit an equivalence relation defined on the secondary vertices, whose representatives are vertices of a polytope which is strongly isomorphic (in the same affine space, with same normal fan) to the resultant polytope. Alternatively, the resultant polytope is strongly isomorphic to a Minkowski summand of the secondary polytope. The resultant polytope is much smaller than the secondary polytope, hence the naive approach of computing the former from the latter is largely inefficient.

Example 1. A standard benchmark in geometric modeling is the implicitization of the bicubic surface defined by 3 polynomials in 2 parameters; the supports have cardinalities 7,6,14, and total degrees 3,3,6, respectively. This yields $\mathcal{A} \subset \mathbb{Z}^4$ with 27 points. TOPCOM needs more than a day to compute $1,806,467$ regular triangulations which correspond to 29 resultant vertices using 9GB of RAM, and crashes. Our algorithm yields the vertices of a 3-dimensional projection of the resultant polytope in 30msec: $\{(0,0,1),(0,1,0),\ (1,0,0),(0,0,9),(0,18,0),(18,0,0)\}$.

Our main contribution is the design of an output-sensitive algorithm for computing the Newton polytope of $\mathcal{R}$, or of specializations of $\mathcal{R}$. The algorithm computes both vertex- and facet-representations of the output polytope, which is important for the targeted applications. Its incremental nature implies that we also obtain a triangulation of the polytope, which is useful for counting or enumerating its lattice points in subsequent applications. The complexity is proportional to the number of output vertices; our approach needs to compute as many regular triangulations of $\mathcal{A}$. We actually work in the space of the projected resultant polytope and revert to the high-dimensional space of secondary polytopes only whenever needed.

We conclude with an efficient implementation based on CGAL, comparisons with a tropical approach, and several experiments. Our method is particularly efficient when the output polytope lies in ≤ 5 dimensions, with typical runtimes in the range of seconds. For up to 7 dimensions, our implementation runs in the range of minutes for inputs with a few tens of points.

The number of coefficients of the $n+1$ polynomials ranges from $2n+2$ to $O(n^d d^n)$, where d bounds the total degree of the polynomials. Typically, one does *not* need to compute $\mathcal{R}$ as a polynomial with all coefficients being indeterminate. In important applications, such as system solving and implicitization of parametric hypersurfaces, one computes $\mathcal{R}$ when all but $O(n)$ of the coefficients are specialized to constants. The lattice points in the resultant polytope yield a superset of the support of $\mathcal{R}$; this reduces implicitization to sparse interpolation. It also greatly enhances existing methods for computing $\mathcal{R}$, whose bottleneck is determinant interpolation, see [CLO05, DE05]. Deterministic sparse interpolation requires a bound τ on support cardinality, and runs in $O(t^3 \delta m \log m + \tau^3)$, where δ bounds the output degree per variable, t is the actual support cardinality, and m the number of variables [BOT88]. For resultant matrices, we perform structured matrix operations in $O^*(\tau^2)$ [CKL89]. Here, $O^*(\cdot)$ implies that polylog factors have been ignored.

The roadmap of the paper is as follows. We next refer to related work. Sect. 2 describes the combinatorics of resultants, and the following section presents our algorithm. Sect. 4 discusses the implementation and experimental results. We conclude with future work.

Previous Work. The computation of secondary polytopes has been efficiently implemented in TOPCOM [Ram02], which has been the reference software for computing regular or all triangulations. It computes all Orientation determinants that will be needed and stores their

signs, i.e. computes the pointset's chirotope. Clearly, the approach is limited by space usage. To address this, reverse search was proposed [IMTI02], but the implementation cannot compete with TOPCOM.

The foundations of toric (or sparse) elimination were laid in [GKZ94], where the discriminant of a multivariate polynomial with support $\mathcal{A}$ is studied: its Newton polytope is shown to be a Minkowski summand of $\Sigma(\mathcal{A})$. Since sparse resultants are special instances of discriminants, the same holds for the resultant polytope. In [GKZ90] they describe the face lattice of the Newton polytope of the resultant of two univariate polynomials of degrees d_0, d_1: in particular, it has $\binom{d_0+d_1}{d_0}$ vertices and $d_0d_1 + 3$ facets.

In [Stu94], $\Sigma(A)$ is shown to be strongly isomorphic to a fiber polytope defined by the input supports, and the resultant polytope is shown to be a Minkowski summand of the latter. The same work associates to the resultant edges certain bistellar flips. Our attempt to compute resultant polytopes based on this approach did not lead to efficient algorithms [EFK10]. In [Stu94, sec.6] is proven that the only planar resultant polytope is the triangle, whereas the only 3-dimensional ones are the tetrahedron, the square-based pyramid, and the polytope of two univariate quadratics.

In [MC00] they describe all Minkowski summands of $\Sigma(A)$. In [MV99], the authors define an equivalence class over secondary vertices having the same mixed cells. These classes map in a many-to-one fashion to resultant vertices; our algorithm exploits a stronger equivalence relationship.

Tropical geometry is a new and very active combinatorial approach to algebraic geometry which, in some sense, generalizes toric elimination. In [JY11] they develop two algorithms to compute the resultant of tropical polynomials, which coincides with the tropicalization of $\mathcal{R}$. The latter corresponds to the union of codimension-one cones in the normal fan of $N(\mathcal{R})$. Assuming $\mathcal{R}$ defines a hypersurface, their best method for computing $N(\mathcal{R})$ is based on a description of the tropical resultant as a union of (possibly overlapping)Then, they compute the normal cones to the vertices of $N(\mathcal{R})$. This approach also computes tropicalizations of specialized resultants.

2 The combinatorics of resultants

We introduce tools from combinatorial geometry [LRS10, Zie95] to describe resultants [CLO05, DE05, GKZ94]. Let $\mathrm{vol}(\cdot)$ denote Euclidean volume, $(\mathbb{R}^m)^\times$ the linear m-dimensional functionals, and $\mathrm{CH}(\cdot)$ the convex hull operation.

Let $\mathcal{A} \subset \mathbb{R}^d$ be any pointset with $\dim(\mathrm{CH}(\mathcal{A})) = d$. For any triangulation T of $\mathcal{A}$, define vector $\phi_T \in \mathbb{R}^{|\mathcal{A}|}$ with coordinate, for $a \in \mathcal{A}$:

$$\phi_T(a) = \sum_{\sigma \in T : a \in \sigma} \mathrm{vol}(\sigma) \ \in \mathbb{N}, \tag{2}$$

summing over all simplices $\sigma \in T$ having a as a vertex; $\Sigma(\mathcal{A})$ is the convex hull of ϕ_T for all triangulations T. Now consider *regular triangulations* of $\mathcal{A}$: these are obtained by projecting the upper (or lower) hull of the pointset lifted to $\mathbb{R}^{d+1}$ via a generic lifting function in $(\mathbb{R}^{|\mathcal{A}|})^\times$.

Proposition 1. [GKZ94] *The dimension of $\Sigma(\mathcal{A})$ is $|\mathcal{A}|-d-1$. The vertices of $\Sigma(\mathcal{A})$ correspond to the regular triangulations of $\mathcal{A}$, while its face lattice corresponds to the poset of regular (polyhedral) subdivisions of $\mathcal{A}$, ordered by refinement.*

A lifting produces regular triangulation T iff the corresponding vector lies in the normal cone of vertex ϕ_T of $\Sigma(\mathcal{A})$. A lifting produces a regular subdivision iff the vector lies in the normal cone of the corresponding face of $\Sigma(\mathcal{A})$.

The family $A_0, \ldots, A_n \subset \mathbb{Z}^n$ is *essential* if they jointly affinely span $\mathbb{Z}^n$ and every subset of cardinality $j, 1 \leq j < n$, spans a space of dimension $\geq j$. In the sequel, the input $A_0, \ldots, A_n \subset$

$\mathbb{Z}^n$ is supposed to be essential. Let $P_0, \ldots, P_n \subset \mathbb{R}^n$ be their convex hulls, and

$$P = P_0 + \cdots + P_n$$

be their Minkowski sum. A *Minkowski (maximal) cell* of P is any full-dimensional convex polytope $B = \sum_{i=0}^n B_i$, where each B_i is a convex polytope with vertices in A_i. Minkowski cells $B, B' = \sum_{i=0}^n B'_i$ intersect properly when $B_i \cap B'_i$ is a face of both and their Minkowski sum descriptions are compatible, i.e. coincide on the common face.

Definition 1. A *mixed subdivision* of P is any family of Minkowski cells which partition P and intersect properly. A cell is called *i-mixed* or *v_i-mixed*, if it is the Minkowski sum of n one-dimensional segments from P_j, $j \neq i$, and vertex $v_i \in P_i$.

Mixed subdivisions contain *faces* of all dimensions between 0 and n, the maximum dimension corresponding to cells. Every face of a mixed subdivision of P has a unique description as Minkowski sum of $B_i \subset P_i$. A mixed subdivision is called *regular* if it is obtained as the projection of the upper (or lower) hull of the Minkowski sum of lifted polytopes $\{(p_i, w_i(p_i)) \mid p_i \in P_i\}$, for lifting $w_i : P_i \to \mathbb{R}$. If the lifting function $w := \{w_0 \ldots, w_n\}$ is sufficiently generic, then the induced mixed subdivision is called *tight*, and $\sum_{i=0}^n \dim B_i = \dim \sum_{i=0}^n B_i$, for every cell.

Given $A_0, \ldots, A_n$ we define the Cayley pointset $\mathcal{A} \subset \mathbb{Z}^{2n}$ as in equation (1).

Proposition 2. [The Cayley Trick] [GKZ94] *There exist bijections between: (a) the regular tight mixed subdivisions of P and the regular triangulations of $\mathcal{A}$, (b) the tight mixed subdivisions of P and the triangulations of $\mathcal{A}$, (c) the mixed subdivisions of P and the polyhedral subdivisions of $\mathcal{A}$.*

Definition 2. Let $A_0, \ldots, A_n \subset \mathbb{Z}^n$ be an essential family of sets, and $f_0, \ldots, f_n \in \mathbb{C}[x_1, \ldots, x_n]$ polynomials with these supports and symbolic coefficients $c_{ij}, i = 0, \ldots, n,\ j = 1, \ldots, |A_i|$:

$$f_i = \sum_{a \in A_i} c_{ij} x^a, \quad x^a = x_1^{a_1} \cdots x_n^{a_n}.$$

The *sparse (or toric) resultant* of the f_i's is a polynomial, defined up to sign,

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

vanishing iff $f_0 = f_1 = \cdots = f_n = 0$ has a common root in $(\mathbb{C}^*)^n$, $\mathbb{C}^* = \mathbb{C} \setminus \{0\}$.

This is the single most important object in several important algebraic algorithms. For linear systems, $\mathcal{R}$ equals the determinant of the $(n+1) \times (n+1)$ coefficient matrix. For $n = 1$, $\mathcal{R}$ is known as Sylvester's resultant. Moreover, the discriminant of an n-variate polynomial $F(x_1, \ldots, x_n)$ differs by a power product of coefficients from the resultant of $F, \partial F/\partial x_1, \ldots, \partial F/\partial x_n$.

Let us focus on 3 important applications. The first concerns interpolation of the resultant as a polynomial in all input coefficients.

Example 2. Let $f_0 = a_2 x^2 + a_1 x + a_0$, $f_1 = b_1 x^2 + b_0$, 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. Our algorithm computes its Newton polytope with vertices $(0, 2, 0, 1, 1)$, $(0, 0, 2, 2, 0)$, $(2, 0, 0, 0, 2)$; it contains 4 lattice points, corresponding to 4 potential monomials $a_1^2 b_1 b_0$, $a_0^2 b_1^2$, $a_2 a_0 b_1 b_0$, $a_2^2 b_0^2$. There is a parameterization of resultants [Kap91], which yields: $a_2 = (2t_1 + t_2) t_3^2 t_4$, $a_1 = (-2t_1 - 2t_2) t_3 t_4$, $a_0 = t_2 t_4$, $b_1 = -t_1 t_3^2 t_5$, $b_0 = t_1 t_5$, where $t = (t_1, t_2, t_3, t_4, t_5)$ are parameters. We substitute these expressions to the monomials, evaluate at 4 sufficiently random t's, and obtain a matrix whose kernel vector $(1, 1, -2, 1)$ yileds $\mathcal{R} = a_1^2 b_1 b_0 + a_0^2 b_1^2 - 2 a_2 a_0 b_1 b_0 + a_2^2 b_0^2$. □

The second application concerns polynomial system solving by means of the Rational Univariate Representation (RUR) of all common roots [BPR03]. Given $f_1, \ldots, f_n \in \mathbb{C}[x_1, \ldots, x_n]$, we define an overconstrained system by adding a linear polynomial $u_0 + u_1 x_1 + \cdots + u_n x_n$ with symbolic u_i's. If all coefficients $c_{ij}, i \geq 1$, take specific values, the resultant of the new system $\mathcal{R}_u$ is the u-resultant. If the common roots of $f_1 = \cdots = f_n = 0$ are isolated, denoted by $r_i = (r_{i1}, \ldots, r_{in})$, then

$$\mathcal{R}_u = c \prod_{r_i} (u_0 + u_1 r_{i1} + \cdots + u_n r_{in})^{m_i},\ c \in \mathbb{C}^*,$$

where m_i is the multiplicity of r_i. Computing $\mathcal{R}_u$ is the bottleneck of RUR.

Example 3. Let $f_1 = x_1^2 + x_2^2 - 4$, $f_2 = x_1 - x_2 + 2$, and $f_0 = u_0 + u_1 x_1 + u_2 x_2$. Our algorithm computes a polygon with vertices $\{(2,0,0), (0,2,0), (0,0,2)\}$, which contains $N(\mathcal{R}_u) = \mathrm{CH}\{(2,0,0), (1,1,0), (1,0,1),\ (0,1,1)\}$. The coefficient specialization is not generic, hence $N(\mathcal{R}_u)$ is strictly contained in the computed polygon. Proceeding as in Example 2, $\mathcal{R}_u = 2u_0^2 + 4u_0u_1 - 4u_0u_2 - 8u_1u_2$, which factors as $2(u_0 + 2u_1)(u_0 - 2u_2)$. □

The third application comes from geometric modeling, where:

$$y_i = f_i(x),\ i = 0, \ldots, n,\ x = (x_1, \ldots, x_n) \in \Omega \subset \mathbb{R}^n$$

defines a parametric hypersurface over domain Ω. Many applications require the equivalent implicit representation of the (hyper)surface as the zero-set of a polynomial $F(y_1, \ldots, y_n) = 0$. This amounts to eliminating x from the overconstrained system $y_i - f_i(x)$, where the y_i's belong to the coefficient field. Hence, it is crucial to compute the resultant of $y_i - f_i(x), i = 0, \ldots, n$, when all coefficients are specialized except for the y_i's. The Newton polytope of F is contained in the orthogonal projection of the resultant polytope of $y_i - f_i(x)$, projected to the $n+1$ coordinates corresponding to the y_i's; cf. Example 5, and [EKK11].

Example 4. A practical implicitization problem concerns a surface whose parametric expressions have total degrees 9, 9, 6, respectively. This yields a pointset $\mathcal{A}$ with 60 points in $\mathbb{R}^4$. Our algorithm yields 6 vertices of the 3-dimensional projection of the resultant polytope: $\{(0,0,1), (0,1,0),\ (2,0,0), (0,0,36), (0,36,0), (36,0,0)\}$ in 0.09 seconds.

Resultant polytopes. The Newton polytope $N(\mathcal{R})$ of $\mathcal{R}$ is a lattice polytope called *resultant polytope*. Any monomial corresponding to a vertex of $N(\mathcal{R})$ is an *extreme term* of $\mathcal{R}$.

Proposition 3. [GKZ94, Stu94] *For a sufficiently generic lifting function $w \in (\mathbb{R}^{|\mathcal{A}|})^\times$, the w-extreme monomial of $\mathcal{R}$ has an exponent vector which maximizes inner product with w. This monomial equals*

$$\pm \prod_{i=0}^{n} \prod_{\sigma} c_{i,v_i}^{\mathrm{vol}(\sigma)}, \tag{3}$$

where σ ranges over all v_i-mixed cells of the regular tight mixed subdivision of P induced by w, and c_{i,v_i} is the coefficient of monomial x^{v_i} in f_i.

Example 5. Let $f_0 := c_{00} - c_{01}x_1x_2$, $f_1 := c_{10} - c_{11}x_1x_2^2$, $f_2 := c_{20} - c_{21}x_1^2$, with Newton polygons in fig. 1. The A_i's form an essential set, so $\mathcal{R} = -c_{00}^4 c_{11}^2 c_{21} + c_{01}^4 c_{10}^2 c_{20}$. The Minkowski sum of the 3 Newton polygons has 2 mixed subdivisions yielding the extreme terms of $\mathcal{R}$. □

If T is the corresponding regular triangulation, via the Cayley trick, we denote by $\rho_T \in \mathbb{N}^{|\mathcal{A}|}$ the exponent of this monomial. There exists a many-to-one surjection from secondary vertices to extreme monomials of $\mathcal{R}$. One defines an *equivalence relationship* on all regular tight mixed subdivisions, where equivalent subdivisions yield the same monomial in $\mathcal{R}$. Thus, equivalent vertices of $\Sigma(\mathcal{A})$ correspond to the same resultant vertex.

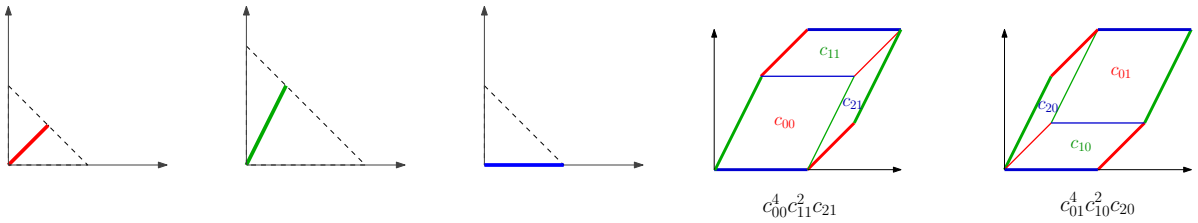


Figure 1: Left: Dashed are the Newton polygons of fully dense polynomials of same total degree. Right: Mixed subdivisions giving the extreme terms of $\mathcal{R}$.

Proposition 4. [GKZ94, MC00, Stu94] *$N(\mathcal{R})$ is Minkowski summand of a polytope which is strongly isomorphic to $\Sigma(\mathcal{A})$. Both $\Sigma(\mathcal{A})$, $N(\mathcal{R})$ have dimension $|\mathcal{A}| - 2n - 1$.*

Consider any vector $w \in (\mathbb{R}^{|\mathcal{A}|})^\times$ lying in the union of outer-normal cones of equivalent vertices of $\Sigma(\mathcal{A})$. They all project to the same resultant vertex, whose outer-normal cone contains w; this also defines an w-extremal resultant monomial. If w is non-generic, it specifies a sum of extremal monomials in $\mathcal{R}$, known as an *initial form*. This has Newton polytope equal to a face of $N(\mathcal{R})$, whose open outer-normal cone contains w; this face is a Minkowski summand of a face of $\Sigma(\mathcal{A})$ whose open outer-normals cone contains w.

3 The resultant polytope projection algorithm

This section describes our algorithm for computing the Newton polytope of a specialization of the resultant, where some c_{ij} remain symbolic whereas all others take values. It is possible that all c_{ij} remain symbolic and the set of specialized coefficients is empty. A specialization of the resultant corresponds to an orthogonal projection of the resultant polytope in a space of lower dimension.

Any projection of the resultant polytope could be computed naively by applying Prop. 3, and projecting the polytope. This is extremely inefficient even for small systems. Instead, we propose an algorithm that efficiently computes directly this orthogonal projection.

Let Q denote the lattice polytope defined as the orthogonal projection of $N(\mathcal{R})$ in some m-dimensional space, $m < |\mathcal{A}| - 2n - 1$:

$$\pi : \mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}^m : N(\mathcal{R}) \mapsto Q.$$

Wlog, this is the subspace of the first m coordinates. We first present the core procedures of our algorithm: the construction of a regular subdivision of a pointset given a linear lifting (Alg. 1) which essentially is a procedure that computes the lower hull of the lifted pointset and the construction of a placing triangulation (Alg. 2). Our algorithm also uses a incremental procedure (`conv` in Alg.4) that computes the convex hull of a pointset. We then describe the algorithm for computing a single vertex of Q (or a point on its boundary), extreme wrt a given $w \in (\mathbb{R}^m)^\times$.

Algorithm 1: reg_subdivision $(\mathcal{A}, w)$

Input : pointset $\mathcal{A}$, vector $w \in (\mathbb{R}^m)^\times$
Output: the regular subdivision of $\mathcal{A}$ wrt the lifting w

$\widetilde{\mathcal{A}} \leftarrow \{ (a_i, w_i) \mid i \in \{1, |\mathcal{A}|\} \}$
$\pi : \mathbb{R}^{|\mathcal{A}|+1} \to \mathbb{R}^{|\mathcal{A}|}$; `/* discards the last coordinate */`
$S \leftarrow \pi(\texttt{lower\_hull}(\widetilde{\mathcal{A}}) \cap \widetilde{\mathcal{A}})$
return S

Algorithm 2: placing_triang $(\mathcal{A}, J)$

Input : pointset $\mathcal{A}$, a set of labels $J = \{1, \ldots, |\mathcal{A}|\}$
Output: the placing triangulation of $\mathcal{A}$ for the ordering given by J

$T \leftarrow \emptyset$
foreach $a \in \mathcal{A}$ *with order given by* J **do**
  if $a \notin T$ **then**
    $T \leftarrow T \cup \{B \cup \{a\} \mid B \in T \text{ and is visible from } a\}$
return T

Algorithm 3: VertexQ $(\mathcal{A}, w, \pi)$

Input : pointset $\mathcal{A}$, vector $w \in (\mathbb{R}^m)^\times$, projection π
Output: a lattice pointon the boundary ∂Q of Q, extremal in the direction of w

$\widehat{w} \leftarrow (w, \vec{0}) \in (\mathbb{R}^{|\mathcal{A}|})^\times$
$S \leftarrow$ `reg_subdivision`$(\mathcal{A}, \widehat{w})$
foreach $s \in S$ **do**
  if s *not simplex* ***and*** $dim(s) = dim(S)$ **then**
    $T \leftarrow T \cup$ `placing_triang`$(s, \{1, \ldots, |s|\})$
  else $T \leftarrow T \cup s$
compute $\rho_T \in \mathbb{N}^m$ by formula (3) of Prop. 3 using the mixed subdivision corresponding (by the Cayley trick) to T
return $\pi(\rho_T)$

For preprocessing, we apply:

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

A simple initialization of W in the first step of Alg. 4 is $W = \{\pm e_i \,|\, i = \{1, \ldots, m\}\}$ with the $2m$ vectors of the standard basis of $(\mathbb{R}^m)^\times$. For Q' computed in this step it may hold that $\dim(Q') \neq \dim(Q)$. To overcome this, we add into W normals perpendicular to the affine hull of Q'. Thus, we ensure $\dim(Q') = \dim(Q)$ after step 1.

Lemma 6. *Triangulation T constructed in Alg. 3 is regular, and corresponds to some secondary vertex ϕ_T which maximizes inner product with $\widehat{w}$.*

Proof. The subdivision S constructed in Alg. 3 corresponds to a face of $\Sigma(\mathcal{A})$ whose outer normal is $\widehat{w}$. T is clearly a triangulation and $T = S$ iff S is tight, in which case it corresponds to a secondary vertex which, by construction, maximizes inner product with $\widehat{w}$. If S is not tight, since S is regular, any regular subdivision of a cell s in S yields an overall regular subdivision. Note that Alg. 2 induces a regular triangulation on s, i.e. every placing triangulation is regular. Further, $\widehat{w}$ is outer normal to some face $f \subset \Sigma(\mathcal{A})$, $\dim(f) \geq 1$. T corresponds to a vertex of $\Sigma(A)$: Since it is a refinement of S, this is a vertex of f, hence it belongs to the vertex set maximizing inner product with $\widehat{w}$. □

Now we prove the correctness of Alg. 4.

Lemma 7. *In Alg. 4, $v \notin F \Leftrightarrow F$ is illegal.*

Algorithm 4: ComputeQ $(A_0, \ldots, A_n, \pi)$

Input : pointsets $A_0, \ldots, A_n \subset \mathbb{Z}^n$ generating $\mathbb{Z}^n$, projection π
Output: the projection Q of the $N(\mathcal{R})$ wrt π

$\mathcal{A} \leftarrow Cayley(A_0, \ldots, A_n)$

```
// Initialization phase
```
$W \leftarrow \{w \,|\, w \in (\mathbb{R}^m)^\times\}$
$Q \leftarrow \emptyset$
foreach $w \in W$ **do**
  $v \leftarrow$ `VertexQ`$(\mathcal{A}, w)$
  $Q \leftarrow$ `conv`(Q, v)

```
// Incremental phase
```
foreach $F \in Q$ **do** legal$(F) \leftarrow false$; ; `/* F is a facet of Q */`
foreach $F \in Q$ ***and*** legal$(F) = false$ **do**
  w is the normalized outer normal vector of F
  if $w \notin W$ **then**
    $W \leftarrow W \cup w$
    $v \leftarrow$ `VertexQ`$(\mathcal{A}, w)$
    if $v \notin Q$ **then**
      $Q' \leftarrow$ `conv`(Q, v)
      foreach $F \notin \{Q' \setminus Q\}$ **do** legal$(F) \leftarrow false$
      $Q \leftarrow Q'$
    else legal$(F) \leftarrow true$
return Q

Proof. Let $v = \pi(\rho_T) = VertexQ(\mathcal{A}, w)$, where T is a triangulation refining subdivision S in Alg. 3. It is clear that if $v \notin F$, then $v \notin Q'$, hence F is illegal and $v \in \partial Q$ is extremal wrt w.

To prove the other direction, suppose that v lies in $F \subset Q'$. Let f, f' be the faces of $\Sigma(\mathcal{A})$ and $N(\mathcal{R})$, respectively, wrt the normal vector $\widehat{w}$. By the proof of Lem. 6, $\phi_T \in f$, hence $\rho_T \in f'$. Now, the supporting hyperplanes of f, f' lie in the subspace $\mathbb{R}^{|\mathcal{A}|-m}$ of $\mathbb{R}^{|\mathcal{A}|}$, which is complementary to $\mathbb{R}^m$ and parallel to the projection π. Hence, $\pi(\rho_{T'})$ of every resultant vertex $\rho_{T'}$ in f' lies also in F, and $\pi(f') = F$, which shows that F is a facet of Q, hence legal. □

It follows from Lemma 6 that Alg. 3 computes a point on tha boundary of the projection π of the secondary polytope and thus of Q.

Corollary 8. *All points computed by Alg. 4 lay on ∂Q.*

This proof implies that any triangulation T, computed in step 2 of Alg. 3, is good to determine whether a facet $F \subset Q'$ wrt normal w is a facet of Q or not.

Complexity analysis. We now bound the asymptotic time complexity of our algorithm. We can use a balanced search tree, such as the red-black tree with logarithmic search and insertion time as a data structure to represent W. Thus, checking $w \notin W$ and updating $W = W \cup w$ takes $O(\log |W|)$. Given some $\widehat{w} \in \mathbb{R}^{2n+1}$, Alg. 3 lifts $\mathcal{A}$, computes its upper hull, and projects it to $\mathbb{R}^{2n}$ in order to obtain a regular triangulation of $\mathcal{A}$. The complexity lies in $O^*(|\mathcal{A}|^{\lfloor (2n+1)/2 \rfloor})$, which is the complexity of computing the upper hull.

Alg. 4 given a vector w at every step (initialization or incremental), uses Alg. 3 to compute a lattice point on the boundary ∂Q of Q, extremal in the direction of w. Thus, it either (a) finds a new vertex of Q, (b) finds a lattice point on a face of Q, or (c) concludes that a facet of Q is

legal. Thus, the number of steps is bounded by the number of lattice points on the boundary ∂Q of Q, which also bounds the number of facets in Q. The asymptotic time complexity of Alg. 4 becomes:

$$O^*\left(|\partial Q \cap \mathbb{Z}^m| \cdot |\mathcal{A}|^n\right).$$

This bound is essentially output sensitive, since $|\partial Q \cap \mathbb{Z}^m|$ is related to the volume $\mathrm{vol}(Q)$ or the number of lattice points in Q. In the following section we show that in practice the above statement holds.

The total time complexity of a convex hull algorithm is a function in the number of vertices of the input and facets of the output that bounds the running time of the algorithm. It is known that any incremental convex hull algorithm has a worst-case super-polynomial total time complexity [Bre96]. The beneath-beyond algorithm constructs a placing triangulation of Q for some ordering of the input points. The algorithm's complexity is bounded by the size of the constructed triangulation. The maximum size of a triangulation of a d dimensional polytope with n vertices is $O(n^{\lfloor (d+1)/2 \rfloor})$ [Sta96]. Thus, beneath-beyond has worst-case exponential total time complexity for Q.

4 Implementation and Experiments

We implement Alg. 4 in C++ to compute the projection of $N(R)$ in $\mathbb{R}^m$; our code can be obtained from `http://respol.sourceforge.net`. We mainly use CGAL [CGA], including its experimental package `Triangulation` for triangulations in general dimensional spaces; it was shown to be the fastest in up to 6 dimensions [BDH09]. It provides the same functionality as CGAL lower-dimensional triangulation packages, with many algorithmic improvements and a new efficient triangulation data structure. We modified it in order to benefit from the hashed determinants scheme of [EFP11].

We use `Triangulation` to compute regular triangulations in $2n+1$ dimensions in Alg. 3, as well as to maintain the projection of the resultant polytope. All convex hull computations in the algorithm are incremental and maintain a polytope where both boundary and interior are triangulated. Thus, projecting the upper hull in Alg. 2 always results in a triangulation.

We perform an experimental analysis of our algorithm on an Intel Core i5-2400 3.1GHz, with 6MB L2 cache and 8GB RAM running 64-bit Debian GNU/Linux. We start by comparing our algorithm with the naive one that uses TOPCOM to enumerate all regular triangulations, applies Prop. 3 to compute the vertices of $N(\mathcal{R})$, then projects. Until recently, this was the only efficient implementation for computing an arbitrary projection of $N(\mathcal{R})$ without computing the resultant polynomial. Example 1 shows that this method is quite inefficient.

As a next step we design experiments parameterized on: the total number of input points $|\mathcal{A}|$, the dimension n of the original pointsets A_i, and the dimension of the projection space m. Recall that the algorithm computes regular triangulations in dimension $2n+1$ and the projection of $N(\mathcal{R})$ in dimension m.

First, we examine our algorithm on random inputs for the implicitization and the u-resultant problems. We fix a degree δ and select random lattice points in the triangle $(0,0),(0,\delta),(\delta,0)$ (including its vertices) to generate dense inputs, and lattice points in the square $(0,0),(0,\delta/2)$, $(\delta/2,0),(\delta/2,\delta/2)$ to generate sparse inputs.

Recall that for implicitization $m = n+1$ which corresponds to the constant term of each polynomial, i.e. point $(0,\ldots,0)$ in each input support. Similarly, when computing the u-resultant polytope, $m = n+1$ but now the non-specialized coefficients all come from the linear polynomial f_0, i.e. the points $(1,\ldots,0),\ldots,(0,\ldots,1) \in A_0$. Thus, in these problems the only free parameters are $|\mathcal{A}|$ and n. The graph in Fig. 2 (left) shows the execution time of our algorithm as a function of $|\mathcal{A}|$ when $n = 2$. Note that $|\mathcal{A}| < 60$ and $n = 2$ for both the implicitization challenges (see Examps. 1 and 4) and all the implicitization problems discussed in [EKK11], which means that we can compute their resultant polytopes in less than a second.

Finally, we examine the effect of the hashed determinants method [EFP11] to the execution time of our algorithm. Fig. 2 (right) shows an impressive gain when $n = 2$. For $n > 2$ this method allows our algorithm to compute instances of the problem that would be intractable otherwise.

Emphasizing more on the structure of $m = n+1$ problems, Fig. 3,4 illustrates the behaviour of our algorithm when n increases. In these graphs the y axis are in logarithmic scale. We can observe that Fig. 3 (left) shows that execution time as a function of the input becomes exponential when $n > 3$. However, Fig. 4 shows that the object we compute becomes exponential in the size of the input when $n > 3$ and Fig. 3 (right) that the execution time of our algorithm seems to be polynomial as a function to the output. That is, the exponential behaviour for $n > 3$ is because it has to compute an exponentially large object. This discussion gives evidences that our algorithm is also output-sensitive in practice. Finally, we perform some experiments for $m > n + 1$ which corresponds to arbitrary specialized resultants. The table in Fig. 4 (right) shows the limits of out implementation. By fixing n, m it shows the largest $|\mathcal{A}|$ that we can compute (timings varying from minutes to an hour).

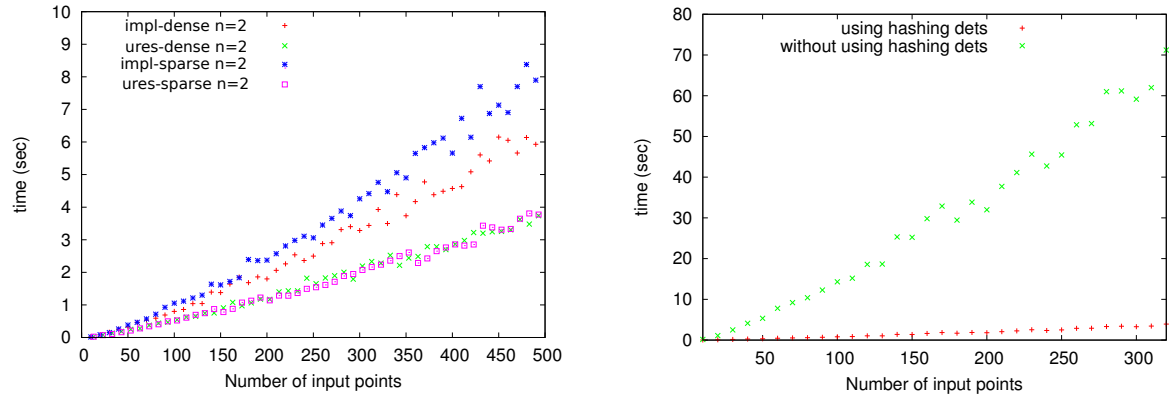


Figure 2: Comparisons tests for $n = 2$ (left). Performance comparisons for hashing versus non hashing determinants when $n = 2$ (right).

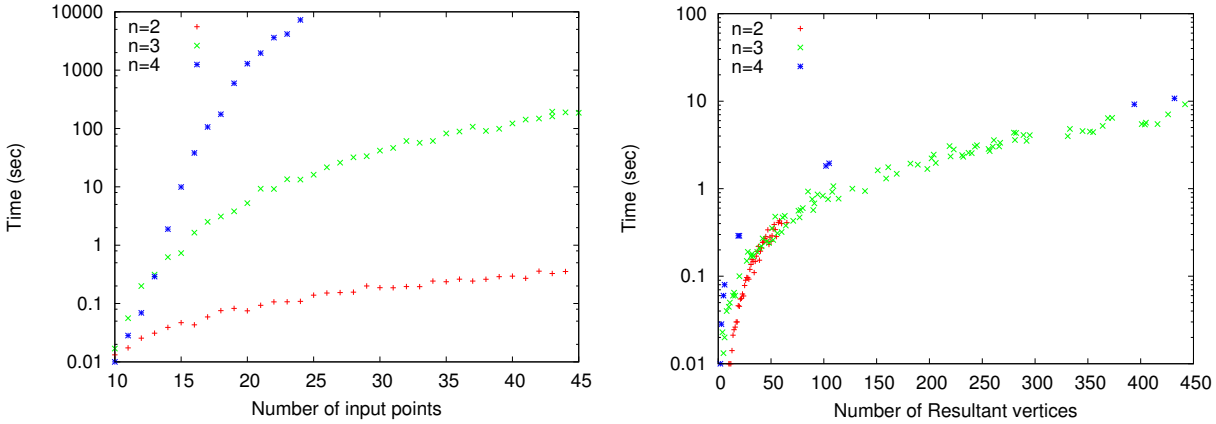
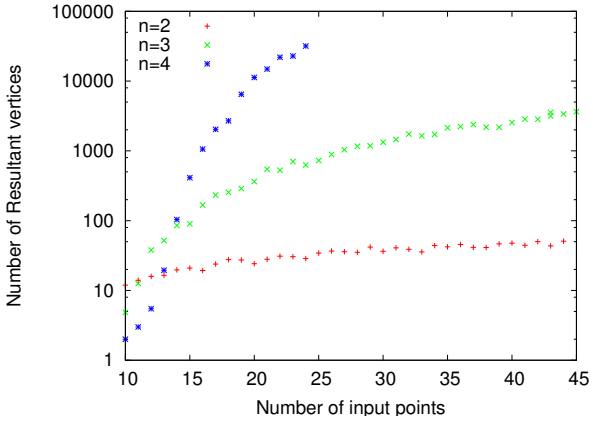


Figure 3: Performance of proposed algorithm for $n = 2, 3, 4$ as a function of input (left) and output (right). The y-axis are in logarithmic scale.

Comparing algorithms. We compare the implementation of our algorithm against the implementation of tropical algorithms in [JY11]. The comparison is based on the set of examples ((a) - (i)) appearing in [JY11], but their tests are performed on a slightly faster machine (3.1GHz vs 2.66GHz). Examples (f,h) correspond to the bicubic surface with or without internal points.



$n=3$	$m=5$	$\lvert\mathcal{A}\rvert=27$
	$m=6$	$\lvert\mathcal{A}\rvert=17$
	$m=7$	$\lvert\mathcal{A}\rvert=13$
$n=4$	$m=6$	$\lvert\mathcal{A}\rvert=16$
	$m=7$	$\lvert\mathcal{A}\rvert=15$

Figure 4: The output (number of Resultant polytope's vertices) as a function of the input (left). The largest $|\mathcal{A}|$ for fixed n, m that we can perform computations (right).

In each example we report only the timing of their best performing method, see Table 1. Our code seems faster in medium dimensions but slower to higher dimensions, probably due to the data-structure we compute, which includes a triangulation, hence several internal cells.

example	tropical(secs)	ResPol(secs)
a	1.4	1.32
b	6	16.19
c	55	89.39
d	0.7	0.27
e	1.3	1.24
f	0.4	0.03
h	2.6	1.2
i	184	358.31

Table 1: Comparison of our implementation to that of [JY11].

5 Future work

We conclude with ideas for future improvements. When F turns out to be illegal and $T \neq T'$, can we use T' to define a resultant face of positive dimension and enlarge Q' faster? One heuristic is to choose a subset of the Q' vertices (or edges) "covering" all the facets. We apply Alg. 3 to these vertices, which are fewer than the facets. We pick w (in the normal cone of each vertex) and hope it extends many incident facets. This is a fast heuristic for extending illegal faces and facets, especially when new vertices lie far from the facets. If we cannot extend Q', this does not imply the adjacent facets are legal.

In order to study experimentally the size of placing triangulations of Q, but also in order to assess our choice of CGAL, we compare the execution time of 4 convex hull packages relying on exact arithmetic: (a) `cdd` implements the double description method based on Fourier-Motzkin elimination [Fuk08], (b) `lrs` [Avi98] implements revearch search [AF91], and is especially efficient with simple polytopes, (c) `BeneathBeyond` implemented in Polymake [GJ01], with random order typically being the best insertion strategy [Jos03], (d) `CGAL` [BDH09] implements the randomized incremental algorithm of [CMS93], with a *biased randomized insertion order*. The last two also maintain a triangulation of the polytope. All packages can be called as a library. However, only the CGAL implementation offers a function for adding points one by one. CGAL triangulates the sphere of the same dimension as the polytope, by keeping a point at infinity

connected to all polytope vertices. In our case, every new point conflicts with at least one cell having the point at infinity as vertex.

Acknowledgment Authors enjoy partial support from project "Computational Geometry Learning", which acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 255827. We also thank S. Hornus for discussing the `Triangulation` package.

References

[AF91] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. In *Proc. ACM Symp. Comp. Geom.*, pages 98–104. ACM, 1991.

[Avi98] D. Avis. lrs: A revised implementation of the reverse search vertex enumeration algorithm, 1998.

[BDH09] J.-D. Boissonnat, O. Devillers, and S. Hornus. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In *SoCG*, pages 208–216, 2009.

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

[BPR03] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in real algebraic geometry*. Springer-Verlag, Berlin, 2003.

[Bre96] D. Bremner. Incremental convex hull algorithms are not output sensitive. In *Proc. 7th Intern. Symp. Algorithms and Comput.*, pages 26–35, London, UK, 1996. Springer.

[CGA] CGAL: Computational geometry algorithms library. http://www.cgal.org.

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

[CLO05] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Number 185 in GTM. Springer, New York, 2nd edition, 2005.

[CMS93] K.L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Comput. Geom.: Theory & Appl.*, 3:185–121, 1993.

[DE05] A. Dickenstein and I.Z. Emiris, editors. *Solving Polynomial Equations: Foundations, Algorithms and Applications*, volume 14 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, May 2005.

[EFK10] I.Z. Emiris, V. Fisikopoulos, and C. Konaxis. Regular triangulations and resultant polytopes. In *Proc. Europ. Workshop Computat. Geometry*, Dortmund, Germany, 2010.

[EFP11] I.Z. Emiris, V. Fisikopoulos, and L. Pearanda. Optimizing the computation of sequences of determinantal predicates. Technical Report CGL-TR-14, NKUA, 2011.

[EKK11] I.Z. Emiris, T. Kalinka, and C. Konaxis. Implicitization using predicted support. In *Proc. Inter. Works. Symbolic-Numeric Computation*, 2011.

[Fuk08] K. Fukuda. cdd and cdd+ home page. ETH Zürich. http://www.ifor.math.ethz.ch/∼fukuda/cdd_home/, 2008.

[GJ01] E. Gawrilow and M. Joswig. Polymake: an approach to modular software design in computational geometry. In *Proc. Annual ACM Symp. Computational Geometry*, pages 222–231. ACM Press, 2001.

[GKZ90] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. Newton polytopes of the classical resultant and discriminant. *Advances in Math.*, 84:237–254, 1990.

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

[IMTI02] H. Imai, T. Masada, F. Takeuchi, and K. Imai. Enumerating triangulations in general dimensions. *Intern. J. Comput. Geom. Appl.*, 12(6):455–480, 2002.

[Jos03] M. Joswig. Beneath-and-beyond revisited. In M. Joswig and N. Takayama, editors, *Algebra, Geometry, and Software Systems*, Mathematics and Visualization. Springer, Berlin, 2003.

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

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

[LRS10] J.A. De Loera, J. Rambau, and F. Santos. *Triangulations: Structures for Algorithms and Applications*, volume 25 of *Algorithms and Computation in Mathematics.* Springer, 2010.

[MC00] T. Michiels and R. Cools. Decomposing the secondary cayley polytope. *Discr. Comput. Geometry*, 23:367–380, 2000.

[MV99] T. Michiels and J. Verschelde. Enumerating regular mixed-cell configurations. *Discr. Comput. Geometry*, 21(4):569–579, 1999.

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

[Sta96] R. P. Stanley. Combinatorics and commutative algebra. In *Birkhuser, Boston, 2. edition*, 1996.

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

[Zie95] G.M. Ziegler. *Lectures on Polytopes.* Springer, 1995.