LE MATEMATICHE
Vol. LXXV (2020) – Issue II, pp. 471–487
doi: 10.4418/2020.75.2.5

THE NEWTON POLYTOPE OF THE DISCRIMINANT OF A QUATERNARY CUBIC FORM

LARS KASTNER - ROBERT LÖWE

We determine the 166 104 extremal monomials of the discriminant of a quaternary cubic form. These are in bijection with D-equivalence classes of regular triangulations of the 3-dilated tetrahedron. We describe how to compute these triangulations and their D-equivalence classes in order to arrive at our main result. The computation poses several challenges, such as dealing with the sheer number of triangulations effectively, as well as devising a suitably fast algorithm for computation of a D-equivalence class.

1. Introduction

The A-discriminant Δ_A is a homogeneous polynomial associated to an integral point configuration $A \subset \mathbb{Z}^d_{\geq 0}$ that detects singularities of polynomials supported on A. The most popular example is the case of a quadratic polynomial $f = ax^2 + bx + c$ supported on $A = \{0,1,2\}$, in which case the A-discriminant is $\Delta_A = b^2 - 4ac$. Here the discriminant vanishes if and only if f has a double root.

Received on September 12, 2019

AMS 2010 Subject Classification: 14Q10; 52B55
Keywords: triangulations of point configurations; mptopcom; discriminant; quaternary cubic; gkz vectors
Research by the first author is supported by DFG via SFB-TRR 195: "Symbolic Tools in Mathematics and their Application". The second author is supported by DFG via SFB-TRR 109: "Discretization in Geometry and Dynamics".

In general, Δ_A can be defined as follows. Let $f = \sum_{a \in A} c_a \mathbf{x}^a$ be a polynomial supported on A. Then the associated hypersurface $V(f)$ is either singular or smooth, depending on the choice of coefficients $c \in \mathbb{C}^A$. The subset of $\mathbb{C}^A$ consisting of those coefficients that define singular A-hypersurfaces has the structure of an affine cone over a projective hypersurface $\nabla_A \subset \mathbb{P}(\mathbb{C}^A)$, called the *discriminantal hypersurface* of A. Then the *A-discriminant* $\Delta_A \in \mathbb{C}[c_a \mid a \in A]$ is defined as the irreducible integral polynomial defining the discriminantal hypersurface, i.e. $V(\Delta_A) = \nabla_A$. In this way Δ_A is defined uniquely up to sign.

Except for a few special cases, it is quite cumbersome to write down A-discriminants in an expanded form, simply because they are too large. For example, the discriminant of a ternary cubic form, in which case $A = \{z \in \mathbb{Z}^3_{\geq 0} \mid z_1 + z_2 + z_3 = 3\}$, is a degree 12 polynomial of 2040 monomials [8, Chapter 11].

In this article we are concerned with the discriminant $\Delta_{\mathcal{A}}$ of a quaternary cubic form supported on

$$\mathcal{A} = \{z \in \mathbb{Z}^4_{\geq 0} \mid z_1 + z_2 + z_3 + z_4 = 3\} \ , \tag{1}$$

the set of integral points of the 3-dilated standard tetrahedron embedded in $\mathbb{R}^4$. Throughout this note $\mathcal{A}$ denotes the point configuration defined in (1), whereas A is used for arbitrary point configurations.

We call the monomials of $\Delta_{\mathcal{A}}$ corresponding to vertices of its Newton polytope *extremal monomials*. The symmetric group on four letters S_4 acts on $\mathcal{A}$ by coordinate permutation. This action naturally extends to the discriminant $\Delta_{\mathcal{A}}$ and its Newton polytope. We are concerned with the following questions:

Question 1.1. How many extremal monomials does the discriminant $\Delta_{\mathcal{A}}$ have? Furthermore, how many S_4-orbits of extremal monomials are there?

To answer this question we use combinatorial tools developed in [8, Chapter 11]. The key observation of Gel′fand, Kapranov and Zelevinsky is that the extremal monomials of the A-discriminant are in bijection with so called D-equivalence classes of regular triangulations of A. With this at hand, it suffices to compute all regular triangulations of A. The latter task turns out to be a computationally challenging problem in general. However, using the recently developed software `MPTOPCOM` [11] we manage to compute all regular triangulations of the point configuration $\mathcal{A}$ in question. Finally, we efficiently compute the number of D-equivalence classes to arrive at our main result.

Theorem 1.2. *The discriminant* $\Delta_{\mathcal{A}}$ *has* 166 104 *extremal monomials that split into* 7 132 *S_4-orbits. Equivalently,* $\mathcal{A}$ *admits* 166 104 *D-equivalence classes of regular triangulations.*

Our computation yields a list of exponent vectors of the extremal monomials. Once these exponent vectors are known, it is also possible to explicitly write down the corresponding coefficients [8, Theorem 11.3.2], but we will not be concerned with this.

Representations of the discriminant $\Delta_{\mathcal{A}}$ are known in various fashions. By definition, $\Delta_{\mathcal{A}}$ equals the resultant of the four quadratic partial derivatives f_{x_1}, f_{x_2}, f_{x_3}, f_{x_4} of a generic polynomial f supported on $\mathcal{A}$. In 1899 Nanson [12] gave a representation of this resultant (and thereby of $\Delta_{\mathcal{A}}$) as the determinant of an 20×20 matrix. Recently $\Delta_{\mathcal{A}}$ was represented as the Pfaffian of an 16×16 matrix [4]. However, with current computer algebra software these matrices are too large to compute their determinants, making this approach to answer Question 1.1 impractical.

Let $P = \mathcal{N}(\Delta_{\mathcal{A}})$ denote the Newton polytope of the discriminant $\Delta_{\mathcal{A}}$. Theorem 1.2 tells us that P has 166 104 vertices, split in 7 132 S_4-orbits. In general, the structure of the Newton polytope of a polynomial is closely related to the geometry of the associated hypersurface. For example, the vertices of the Newton polytope are in bijection with the unbounded regions of the amoeba associated to the hypersurface, which can be viewed as a logarithmic shadow of the hypersurface; see [8, Chapter 6]. In this way, Theorem 1.2 sheds light on the discriminantal hypersurface $\nabla_{\mathcal{A}}$.

2. D-equivalence of regular triangulations

Let $A = (a_1, \dots, a_n)$ be a finite point configuration in $\mathbb{R}^d$ and let $Q = \mathrm{conv}(A)$ denote its convex hull. We always assume that Q has full dimension $\dim(Q) = d$, and that the lattice spanned by A is $\mathbb{Z}^d$. That way we do not have to consider the volume with respect to the lattice spanned by A, but can rather use the ordinary lattice volume vol everywhere.

A *triangulation* of A is a collection of simplices $T = (\sigma)_{\sigma \in T}$ with vertices in A such that

1. T is closed under taking faces, i.e. if $\sigma \in T$ and $\sigma' < \sigma$ then also $\sigma' \in T$,
2. T covers Q, meaning $\bigcup_{\sigma \in T} = Q$, and
3. for any two $\sigma, \sigma' \in T$ the intersection $\sigma \cap \sigma'$ is a face of both σ and σ'.

For each triangulation T of A we define its GKZ *vector* $\Phi_T \in \mathbb{Z}^A$ by

$$\Phi_T(a_i) = \sum_{\sigma \ni a_i} \mathrm{vol}(\sigma) \ , \tag{2}$$

where the summation is over all maximal simplices of T for which a_i is a vertex. Since A is finite it only admits finitely many triangulations. Hence the convex hull of all GKZ vectors

$$\Sigma\text{-poly}(A) = \operatorname{conv}\{\Phi_T \mid T \text{ triangulation of } A\} \subset \mathbb{R}^A$$

is a convex polytope, called the *secondary polytope* of A. A triangulation of A is called *regular* (some prefer *coherent*) if its GKZ vector is vertex of the secondary polytope.

The theory of regular triangulations and secondary polytopes was introduced by Gel'fand, Kapranov and Zelevinsky [8] and has numerous applications to questions in classic algebraic geometry; see the monograph of De Loera, Rambau and Santos [6]. For instance, the Newton polytope of the A-discriminant Δ_A, denoted by $\mathcal{N}(\Delta_A)$, is a Minkowski summand of the secondary polytope of A, i.e.

$$\Sigma\text{-poly}(A) = \mathcal{N}(\Delta_A) + R_A \quad , \tag{3}$$

where R_A is a polytope uniquely defined by this equation [8, Chapter 10]. It follows that for any regular triangulation T its GKZ vector Φ_T can be written as

$$\Phi_T = \eta_T + r_T \tag{4}$$

where η_T and r_T are vertices of $\mathcal{N}(\Delta_A)$ and R_A, respectively. In the following we will describe which regular triangulations T of A have the same $\mathcal{N}(\Delta_A)$-summand η_T, leading to the notion of D-equivalence of regular triangulations. We stick with the notation of [8, Chapter 11.3].

Let T be a triangulation of A. A j-dimensional face (simplex) σ of T is called *massive* if it is contained in some j-dimensional face of $Q = \operatorname{conv}(A)$. In that case the face of Q is unique and we denote it by $\Gamma(\sigma)$. In particular, any d-face σ of T is massive with $\Gamma(\sigma) = Q$. Denote by $M_T^j(a_i)$ the set of massive j-simplices of T that have a_i as a vertex. We set

$$\eta_{T,j}(a_i) = \sum_{\sigma \in M_T^j(a_i)} \operatorname{vol}(\sigma) \quad .$$

Here $\operatorname{vol}(\sigma)$ is lattice volume of σ in the intersection lattice $\mathbb{Z}^d \cap \operatorname{aff}(\sigma)$. Note that $\eta_{T,d}$ coincides with the GKZ vector Φ_T defined in (2).

Definition 2.1. We define the *massive* GKZ *vector (*mGKZ*)* $\eta_T \in \mathbb{Z}^A$ of T by

$$\eta_T = \sum_{j=0}^{d} (-1)^{d-j} \eta_{T,j} \quad . \tag{5}$$

Finally, we say two regular triangulations of A are *D-equivalent* if they have the same mGKZ vector.

Theorem 2.2. *[8, Chapter 11, Theorem 3.2] Let $\mathcal{N}(\Delta_A)$ be the Newton polytope of the A-discriminant Δ_A. The vertices of $\mathcal{N}(\Delta_A)$ are exactly the points η_T for all regular triangulations T of A. Thus, they are in one-to-one correspondence with the D-equivalence classes of regular triangulations of A.*

In view of (3) this means that a D-equivalence class with mGKZ vector η consists of all those regular triangulations whose GKZ vectors have η as a unique $\mathcal{N}(\Delta_A)$-summand.

Example 2.3. Let Q be the triangle in $\mathbb{R}^2$ with vertices $(0,0),(2,0),(0,2)$ and let A be the set of lattice points contained in Q,

$$A = \{(0,0),(0,1),(0,2),(1,0),(1,1),(2,0)\} \ .$$

In Figure 1 we show all 14 regular triangulations of A together with both their mGKZ vectors as well as their ordinary GKZ vectors. We see that there are five different D-equivalence classes of regular triangulations with associated mGKZ vectors

$$\begin{aligned}
(1,0,1,0,0,1) &= \eta_{T_0} \ , \\
(1,0,0,0,2,0) &= \eta_{T_1} = \eta_{T_2} = \eta_{T_3} = \eta_{T_4} \ , \\
(0,0,1,2,0,0) &= \eta_{T_5} = \eta_{T_6} = \eta_{T_7} = \eta_{T_8} \ , \\
(0,2,0,0,0,1) &= \eta_{T_9} = \eta_{T_{10}} = \eta_{T_{11}} = \eta_{T_{12}} \ , \\
(0,1,0,1,1,0) &= \eta_{T_{13}} \ .
\end{aligned}$$

By Theorem 2.2 we know that $\mathcal{N}(\Delta_A)$ has five vertices given by these mGKZ vectors. Indeed, the A-discriminant of a quadratic form

$$f(x,y) = a_{00} + a_{01}y + a_{02}y^2 + a_{10}x + a_{11}xy + a_{20}x^2$$

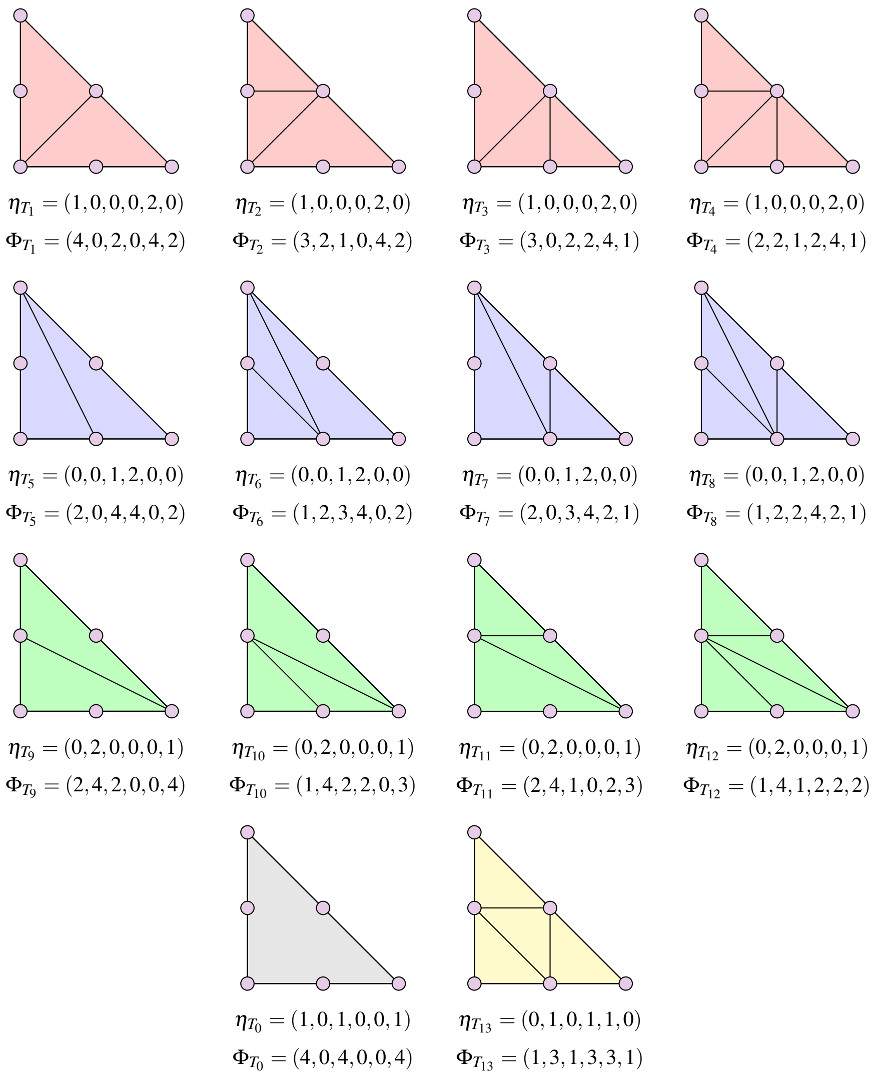
is well known to be [8, Chapter 13]

$$\Delta_A = a_{00}a_{11}^2 + a_{01}^2 a_{20} + a_{02}a_{10}^2 - a_{01}a_{10}a_{11} - 4a_{00}a_{02}a_{20} \ .$$

We observe that the exponent vectors of the A-discriminant above are exactly the mGKZ vectors that we found. Here the variables $a_{00}, a_{01}, \ldots, a_{20}$ are ordered lexicographically. In this case all five monomials of Δ_A correspond to vertices of its Newton polytope $\mathcal{N}_{\Delta_A}$ (a pentagon). Figure 2 depicts the secondary polytope of A. Here the vertex colors indicate the D-equivalence classes of regular triangulations.

From now on we are only concerned with the point configuration $\mathcal{A}$ that supports cubic quaternary forms, i.e.

$$\mathcal{A} = \{z \in \mathbb{Z}_{\geq 0}^4 \mid z_1 + z_2 + z_3 + z_4 = 3\} \ .$$



$\eta_{T_1} = (1,0,0,0,2,0)$ $\Phi_{T_1} = (4,0,2,0,4,2)$

$\eta_{T_2} = (1,0,0,0,2,0)$ $\Phi_{T_2} = (3,2,1,0,4,2)$

$\eta_{T_3} = (1,0,0,0,2,0)$ $\Phi_{T_3} = (3,0,2,2,4,1)$

$\eta_{T_4} = (1,0,0,0,2,0)$ $\Phi_{T_4} = (2,2,1,2,4,1)$

$\eta_{T_5} = (0,0,1,2,0,0)$ $\Phi_{T_5} = (2,0,4,4,0,2)$

$\eta_{T_6} = (0,0,1,2,0,0)$ $\Phi_{T_6} = (1,2,3,4,0,2)$

$\eta_{T_7} = (0,0,1,2,0,0)$ $\Phi_{T_7} = (2,0,3,4,2,1)$

$\eta_{T_8} = (0,0,1,2,0,0)$ $\Phi_{T_8} = (1,2,2,4,2,1)$

$\eta_{T_9} = (0,2,0,0,0,1)$ $\Phi_{T_9} = (2,4,2,0,0,4)$

$\eta_{T_{10}} = (0,2,0,0,0,1)$ $\Phi_{T_{10}} = (1,4,2,2,0,3)$

$\eta_{T_{11}} = (0,2,0,0,0,1)$ $\Phi_{T_{11}} = (2,4,1,0,2,3)$

$\eta_{T_{12}} = (0,2,0,0,0,1)$ $\Phi_{T_{12}} = (1,4,1,2,2,2)$

$\eta_{T_0} = (1,0,1,0,0,1)$ $\Phi_{T_0} = (4,0,4,0,0,4)$

$\eta_{T_{13}} = (0,1,0,1,1,0)$ $\Phi_{T_{13}} = (1,3,1,3,3,1)$

Figure 1: All 14 triangulations of the point configuration A from Example 2.3 are regular. There are five different D-equivalence classes, distinguished by coloring.

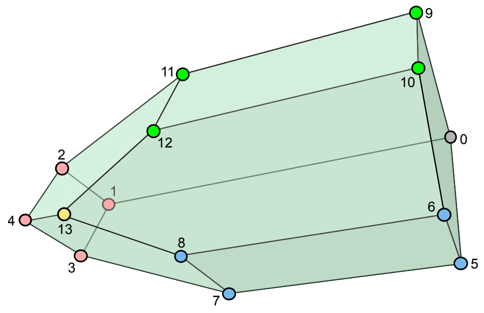


Figure 2: The secondary polytope of the point configuration A from Example 2.3 with vertex colors indicating the D-equivalent triangulations.

These are the lattice points of the 3-dilated standard tetrahedron embedded in $\mathbb{R}^4$. Our goal is to determine all D-equivalence classes of regular triangulations of A. Proceeding as in Example 2.3 one way to approach this problem is to first find all regular triangulations of A. This turns out to be a computationally challenging task as the number of regular triangulations explodes even for rather ordinary point configurations containing not too many points in rather low dimensions; see [6, Chapter 8]. However, using the recently developed software framework `MPTOPCOM` [11] we managed to compute all 910 974 879 regular triangulations of $\mathcal{A}$ up to S_4-symmetry (Theorem 4.1). This will be explained later in section 4. We first continue by describing how we use this intermediate result to arrive at the total number of 166 104 D-equivalence classes of regular triangulations of $\mathcal{A}$ from Theorem 1.2.

3. Massive chains

The goal is to compute mGKZ vectors efficiently. There are 910 974 879 regular triangulations of $\mathcal{A}$ up to S_4-symmetry, and we have to compute the mGKZ vector for one representative of every orbit. For a given triangulation T, the formula given in (5) involves

- computing the Hasse diagram of T in order to get all faces of T,
- checking which faces of T are massive, and
- computing the lattice volume of the massive faces.

We estimate based on an ad-hoc implementation in `polymake` that this would take roughly $1/10$-th of a second for every triangulation, amounting to 2.9 years total computation time. This computation is trivially parallelized, but even on 100 cores it would still take 1.5 weeks.

Remark 3.1. Of course 1.5 weeks on 100 cores is perfectly feasible resource wise. However, at the time we developed the algorithm below, the enumeration of all triangulations was not finished yet, so we did not know how many mGKZ vectors we would have to compute. Hence we devised a fast method. Nevertheless the algorithm below is still interesting, since it also works for computing mGKZ vectors in other examples. Furthermore it is an interesting fact that mGKZ vectors can be decomposed into a sum of vectors associated to the simplices of the given triangulation.

By saying that we want to design the computation more efficiently, we mean that we want to allow longer preprocessing, which is done only once, if in return the computation of a single mGKZ vector is sped up. This means that computation of a single mGKZ vector might become much slower, whereas computation of 900 million mGKZ vector becomes much faster.

The principle is demonstrated in `MPTOPCOM`. For computing the ordinary GKZ vector, first the volume of any d-simplex with vertices in in A is computed and stored. Then no volume computation is necessary anymore throughout the computation, just cache accesses.

The goal of this section is for a given triangulation T to find a decomposition

$$\eta_T = \sum_{\sigma^d \in T} \eta(\sigma^d) \ , \tag{6}$$

where the sum runs over the full-dimensional simplices of $\sigma^d \in T$. Then we can cache the $\eta(\sigma^d)$ for all possible σ^d that can occur in A and speed up computations.

Our main tools are massive chains:

Definition 3.2. Let T be a triangulation of a point configuration A. A sequence $\sigma^j \leq \sigma^{j+1} \leq \ldots \leq \sigma^{d-1} \leq \sigma^d$ of faces of T is called a *massive chain* if for all $k = j, \ldots, d$ we have that σ^k is a massive face of dimension $\dim \sigma^k = k$.

We say a massive chain *starts in* σ, if $\sigma^j = \sigma$. This does not mean that there cannot be a σ^{j-1}, it is just a notion for convenience. Denote by $\mathrm{MC}(\sigma, T)$ the massive chains starting in σ contained in T.

Example 3.3. Consider the triangulation T_1 from Example 2.3. Figure 3 depicts three sequences of faces of T_1. The first one is not a massive chain since the starting edge is not massive. Although all faces of the second sequence are massive faces, it is not a massive chain because the vertex and the triangle differ by two dimensions. Finally, the last sequence is a massive chain that starts on a vertex.

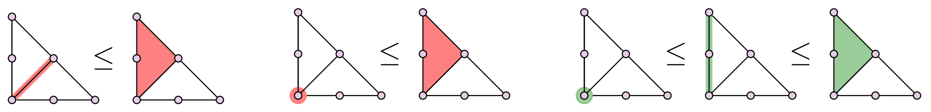


Figure 3: Three sequences of faces of the triangulation T_1. Only the last one depicts a massive chain.

Lemma 3.4. *Every massive face σ of a triangulation T is part of some massive chain. The number of massive chains starting in σ is determined by the smallest face of $Q = \operatorname{conv}(A)$ containing it.*

Proof. We will proceed by induction over $j = \dim \sigma$.

Firstly let $j = d$. Then σ is a full-dimensional simplex of the triangulation T. It forms a massive chain itself, proving the first statement. Furthermore there is no other chain starting in σ, so the number of chains is 1, even independently of Q.

Let us explain the induction step from j to $j-1$. Take σ^{j-1} a massive face of T. Then σ^{j-1} is contained in a unique face F^{j-1} of Q. Now F^{j-1} is the intersection of all the j-dimensional faces of $\operatorname{conv}(A)$ containing it, i.e.

$$F^{j-1} = \bigcap_{k=1}^{m} F_k^j \ .$$

Fix a face F_k^j and consider its triangulation induced by T. Then there is a unique i-dimensional simplex $\sigma_k^j \subset F_k^j$ containing the massive face σ^{j-1}. But then σ_k^j is a massive face of T. Since by induction hypothesis the first statement is true for σ_k^j, we have just proven it for σ^{j-1}.

To proof the second statement let $\#\mathrm{MC}(\sigma, T)$ denote the number of massive chains starting in a face σ. By the induction hypothesis, for the faces σ_k^j these numbers are determined by the faces F_k^j, so we write $\#\mathrm{MC}(\sigma_k^j, T) = \#\mathrm{MC}(F_k^j)$. Thus we get the following recursive formula

$$\#\mathrm{MC}(\sigma^{j-1}, T) = \sum_{k=1}^{m} \#\mathrm{MC}(\sigma_k^j, T) = \sum_{k=1}^{m} \#\mathrm{MC}(F_k^j) \ ,$$

with the recursion anchor $\#\mathrm{MC}(\sigma^d, T) = \#\mathrm{MC}(Q) = 1$. The last sum runs over all the j-dimensional faces of Q containing F^{j-1}. Hence it only depends on F^{j-1}, concluding the proof. □

Proposition 3.5. *Fix a dimension j, $0 \le j \le d$, and a point $a_i \in A$. Let T be a triangulation of A and denote by $M_T^j(a_i)$ the set of all massive j-dimensional faces of T that have a_i as a vertex. Let $\mathrm{MC}(\sigma^j, \sigma^d)$ be the massive chains* *starting in σ^j that are contained in a full-dimensional simplex $\sigma^d \in T$, and define $M^j_{\sigma^d}(a_i)$ analogously. Then*

$$\eta_{T,j}(a_i) \;=\; \sum_{\sigma^j \in M^j_T(a_i)} \operatorname{vol}(\sigma^j) \;=\; \sum_{\sigma^d \in T} \left(\sum_{\sigma^j \in M^j_{\sigma^d}(a_i)} \left(\frac{\#\mathrm{MC}(\sigma^j,\sigma^d)\cdot \operatorname{vol}(\sigma^j)}{\#\mathrm{MC}(\sigma^j,T)} \right) \right)$$

Proof. First we need to make sure that $\#\mathrm{MC}(\sigma^j,T) \neq 0$, but this is the first statement of Lemma 3.4. Now every massive chain in T is contained in a unique full-dimensional simplex $\sigma^d \in T$ by definition. Rewriting

$$\sum_{\sigma^j \in M^j_T(a_i)} \operatorname{vol}(\sigma^j) \;=\; \sum_{\sigma^j \in M^j_T(a_i)} \left(\frac{1}{\#\mathrm{MC}(\sigma^j,T)} \cdot \sum_{C \in \mathrm{MC}(\sigma^j,T)} \operatorname{vol}(\sigma^j) \right),$$

and inserting that every massive chain is contained in a unique simplex, one arrives at the desired formula. □

Note that the denominator $\#\mathrm{MC}(\sigma^j,T)$ does not depend on the triangulation T, but rather on $Q = \operatorname{conv}(A)$. Hence, let us write $\#\mathrm{MC}(\sigma^j,Q)$ for this factor. Then we can define

$$\eta^j_i(\sigma^d) \;:=\; \sum_{\sigma^j \in M^j_{\sigma^d}(a_i)} \left(\frac{\#\mathrm{MC}(\sigma^j,\sigma^d)\cdot \operatorname{vol}(\sigma^j)}{\#\mathrm{MC}(\sigma^j,Q)} \right)$$

and we get $\eta_{T,j}(a_i) = \sum_{\sigma^d \in T} \eta^j_i(\sigma^d)$ as desired in (6). The main advantage is that for any simplex with vertices in A we can precompute and cache all $\eta^j_i(\sigma^d)$. This reduces the effort for computing a massive GKZ vector of a triangulation to some lookups and additions. We avoid determining massivity of faces and intersecting with boundaries completely. Since the triangulations usually vastly outnumber the simplices we can form from A, this will increase performance drastically, if one needs to compute many massive GKZ vectors.

Example 3.6. Let us use Proposition 3.5 to recalculate the mGKZ vector of the triangulation T_1 from Example 2.3. There are two full-dimensional simplices in T_1 which we denote by σ_L and σ_R (left and right). The mGKZ vector of T_1 decomposes as $\eta_{T_1} = \eta(\sigma_L) + \eta(\sigma_R)$. Let us compute $\eta(\sigma_R)$. It is defined by

$$\begin{aligned}\eta(\sigma_R) &= \eta^0(\sigma_R) - \eta^1(\sigma_R) + \eta^2(\sigma_R)\\ &= (1/2,0,0,0,0,1) - (2,0,0,0,1,3) + (2,0,0,0,2,2)\\ &= (1/2,0,0,0,1,0)\ .\end{aligned}$$

Table 1: Timings for computing mGKZ vectors of a batch of 4 215 120 triangulations via different implementations

Method	Time in s
Loop with C++ client (full batch)	103 809.90
perl script with massive chains (full batch)	10 177.12
perl script with massive chains (full batch, second run)	4 677.97
estimated preprocessing time	ca. 6 000.00
Loop with C++ client (1000 triangulations)	40.55
perl script with massive chains (1000 triangulations)	1 137.20

Let us focus on the sixth (last) entry of this vector corresponding to the bottom right vertex $\sigma^0 = (2,0)$. As a vertex σ^0 is a massive 0-face of induced lattice volume $\mathrm{vol}(\sigma^0) = 1$. Any massive chain that starts at σ^0 terminates in the right triangle σ_R and therefore $\eta_6^0(\sigma_R) = 1$. Furthermore, there are two massive edges in T_1 incident to the vertex σ^0, whose lattice volumes are 1 and 2, respectively. Both edges are uniquely completed to a massive chain by adding the triangle σ_R, yielding $\eta_6^1(\sigma_R) = 3$. Finally, $\eta_6^2(\sigma_R) = 2$ since σ_R has lattice volume 2. Similarly we get $\eta(\sigma_L) = (1/2,0,0,0,1,0)$. In total we obtain the desired mGKZ vector $\eta_{T_1} = (1,0,0,0,2,0)$.

3.1. Implementations and timings

There are two implementations of mGKZ vectors in `polymake`. One is a C++ client using the formula of Definition 2.1 directly. The other is a simple perl script using massive chains. The preprocessing time is not directly extractable, since the cache of vectors $\eta(\sigma)$ is populated on demand, which eliminates the task of constructing all possible simplices on startup. Since we will always be provided with triangulations, we know that we will only encounter valid simplices. We ran each of the methods on a batch of 4 215 120 triangulations, and the resulting times are depicted in Table 1.

The perl script took a little under 3 hours, while the C++ method took about 28 hours, a speedup of roughly factor 10. For larger batches the speedup becomes even greater, since the preprocessing time stays constant, as demonstrated with the second run of the perl script on the same batch, using the stored cache from the first run. We can use this to estimate the preprocessing time to be about 6000 seconds, then we get a speedup factor of more than 20 for the actual computation. If we extrapolate for 189 batches, then the C++ method would take almost a year, while the perl script finishes within 10 days.

On the contrary consider the timings on a small batch of 1000 triangulations in the last two lines of Table 1. Here the perl script is almost 30 times slower than the C++ implementation. The timings in Table 1 are for one run only and do not depict proper benchmarkings, but they provide an idea of the strength of using massive chains. Note that since we populate the cache on the fly, running the perl script on 1000 triangulations will not populate the full cache.

4. Computing triangulations with `MPTOPCOM`

In order to compute all regular triangulations of $\mathcal{A}$, we used `MPTOPCOM`, a software framework for enumerating triangulations in parallel [10, 11]. It is based on `polymake` [7], `TOPCOM` [13] and `mts` [2]. The core idea of `MPTOPCOM` is as follows. The regular triangulations of a point configuration form the vertices of the secondary polytope and the regular flips are the edges of the polytope. Ordering the vertices lexicographically we get a so-called *reverse search structure* [1]. In essence a reverse search structure gives a rooted tree within a graph, such that the membership of an edge in the tree can be determined locally, i.e. only knowing its endpoints. In our setting this means that we get two triangulations with a flip between them, and the reverse search condition can tell us, whether this flip constitutes an edge in the reverse search tree. This tree is then traversed in a depth first search.

We will not go into detail about the flip graph of triangulations and triangulation orbits here, but refer the reader to [6] and [9]. Instead, we will just describe the interesting ingredients around the particular example of $\mathcal{A}$, leading to the following result.

Theorem 4.1. *The point configuration $\mathcal{A} = \{z \in \mathbb{Z}^4_{\geq 0} \mid z_1 + z_2 + z_3 + z_4 = 3\}$ admits* 910 974 879 *S_4-orbits of regular triangulations.*

4.1. Parallelization

Parallelization in `MPTOPCOM` is done via the so-called budgeted reverse search [2, 3]. This means that every worker gets a budget, which basically is the maximum depth that he is allowed to explore in the depth first search. Unexplored nodes are then returned to the master, who stores them in a queue, until they can be redistributed among the idle workers. The `mts` framework will dynamically adjust the budget in order to maintain a constant high load to exhaust the given resources optimally.

4.2. Exploiting group action

The symmetric group S_4 acts on $\mathcal{A}$ by coordinate permutations. This group action naturally expands to triangulations. An important property of this action is that it keeps volumes of simplices constant. Thus, it acts on GKZ vectors via permutation and `MPTOPCOM` can exploit this group action. Instead of triangulations, we want to consider orbits of triangulations. In order to do this, it is necessary to be able to get a canonical representative from the orbit of a triangulation and we pick the triangulation with the lex-maximal GKZ vector from the same orbit as canonical representative. Finding this triangulation is equivalent to sorting a vector descending with a restricted set of permutations, namely those from G. This problem has been solved effectively in [11] and implemented in `MPTOPCOM`, making it very unlikely that a full orbit ever needs to be computed.

Even though we are interested in the actual triangulations, rather than just orbits, it is much more effective to have `MPTOPCOM` utilize the group action and expand orbits later on. For instance, regularity then only needs to be checked on the canonical representative, vastly improving performance.

4.3. Checkpointing

An important feature of `mts` is its ability to write checkpoint files and to restart from these. The principle is to let the single workers finish their jobs without assigning them new jobs. When all the workers have finished, the master queue is written to a file. Long lasting computations on big clusters are prone to a variety of interferences: power and network outages, disc failures, random restarts, and so forth. In our concrete case the computation took ca. 80 days. It is very unrealistic to expect a resource intensive computation to be able to run for 80 days without any disturbance. Thus, we chose to produce a checkpoint every 12 hours. Depending on the input parameters it can take a long time to produce the checkpoint, i.e. for all workers to finish. We triggered the end of the computation after 8 hours and then it took ca. 4 hours for all workers to finish.

4.4. Estimating result size

Another advantage of checkpointing is that one can analyze the intermediate results. In our case we already know that $\mathcal{A}$ has 21 125 102 regular and full triangulations, i.e. regular and using all points of $\mathcal{A}$, up to symmetry [11]. These triangulations will reappear as a subset of the regular triangulations. Since the sum over all entries of any GKZ vector is a constant, and the GKZ vector of a full triangulation does not have any zeros, we can deduce that the lexicographically largest GKZ vectors belong to non-full triangulations. Hence, we expect the full triangulations to appear rather towards the end of our computation. Still, the budgeted reverse search is different from reverse search in certain aspects and the curve we observe is depicted in Figure 4.

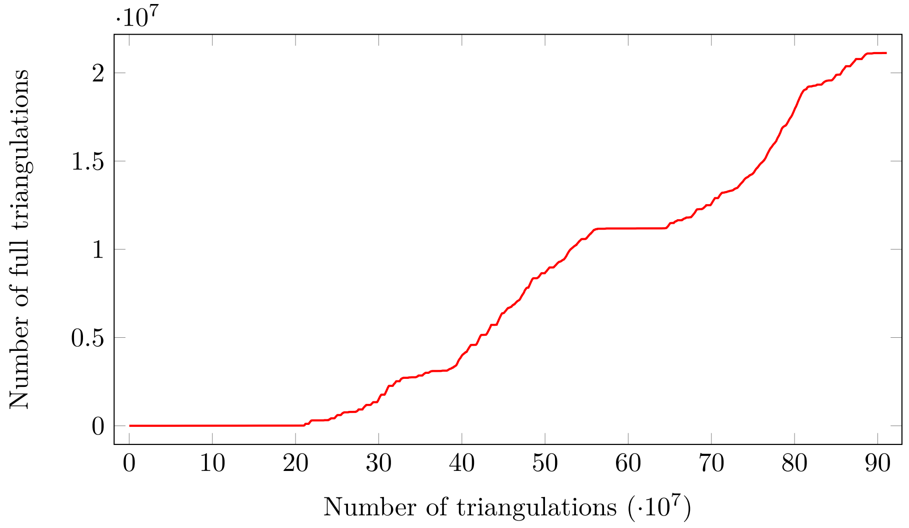


Figure 4: Number of full triangulations during computation

4.5. Scheduling on large clusters

Often computations on large clusters are managed via a software scheduler that one submits jobs to. The scheduler will then arrange those jobs in a way that optimizes efficiency, i.e. such that idle time is low, jobs with high priority are executed faster, etc. When submitting a job, one has to specify the resources needed, like number of nodes, amount of memory, execution time, and so on. The priority of a job is computed by the scheduler upon submission and grows with the amount of time that the job has been waiting. Jobs with high resource demands will get lower priority and will then have to wait until their priority becomes high enough. This means that if one requires 300 nodes for 220 hours (the maximal available time on the cluster), the job will have to queue for a long time. On the other hand, if jobs are short, then the scheduler will often be able to squeeze them in, even if the other resource demands are high. This is why we chose to make a checkpoint every 12 hours. In total the cluster ran 189 jobs on 128 nodes. This took 6 892 460 seconds, that is approximately 80 days. On a laptop with four cores, this would be roughly 7 years.

The result is stored in 189 compressed files, with a total size of 16.5 GiB. For compression we used xz [5]. The uncompressed total size is 338.2 GiB.

4.6. Incomplete result files

Sometimes it would happen that the number of a triangulations written to the result file would not agree with the number reported by `MPTOPCOM`. There are several possible reasons for this. Network problems can have prevented a worker from sending the triangulations to the output worker. The output worker might have gotten shut down at the end before he managed to write the buffer completely. When we noticed this, already several subsequent jobs had run. Due to its parallel nature, the output one gets from one job is not deterministic, the order of triangulations may vary and also the number. Thus we had to devise a way to find all triangulations between two checkpoints. One can boil this down to a problem of graph theory: One has two sets of nodes, a set of starting nodes and a set of target nodes. To find all nodes in the reverse search tree between these two sets, one can run a depth first search from every starting node and never go deeper once a node belongs to the target set. Back to our original problem, we just needed to manipulate the `MPTOPCOM` source to check for every triangulation that it found whether it belongs to the target checkpoint, and if it does, exit.

5. Datasets and code

`MPTOPCOM` can be downloaded at `https://polymake.org/mptopcom`. The triangulations of $\mathcal{A}$ are available at `https://polymake.org/doku.php/dequivalence` as 189 “.xz” files, together with the perl script producing their mGKZ vectors and the sets of mGKZ vectors stored as `polymake` sets.

Acknowledgements

This article addresses the third of the 27 *questions on cubic surfaces* [14]. We would like to thank Dominic Bunnett, Michael Joswig, Marta Panizzut, and Bernd Sturmfels for bringing this problem to our attention and for helpful discussions. We are very grateful to Benjamin Lorenz for help and advice during setting up our software on the TU Berlin clusters and for help when implementing mGKZ vectors in `polymake`.

REFERENCES

[1] David Avis - Komei Fukuda, *Reverse search for enumeration.*, Discrete Appl. Math. 65 no. 1-3 (1996), 21–46 (English).

[2] David Avis - Charles Jordan, *A parallel framework for reverse search using `mts`*, 2016, Preprint `arXiv:1610.07735`.

[3] David Avis - Charles Jordan, *`mplrs`: A scalable parallel vertex/facet enumeration code*, Mathematical Programming Computation 10 no. 2 (2018), 267–302.

[4] Dominic Bunnet - Hanieh Keneshlou, *Determinantal representations of the cubic discriminant*, this volume.

[5] Lasse Collin, *XZ Utils*, 2019, https://tukaani.org/xz/.

[6] Jesús A. De Loera - Jörg Rambau - Francisco Santos, *Triangulations*, Algorithms and Computation in Mathematics, vol. 25, Springer-Verlag, Berlin, 2010, Structures for algorithms and applications.

[7] Ewgenij Gawrilow - Michael Joswig, *`polymake`: a framework for analyzing convex polytopes*, Polytopes—combinatorics and computation (Oberwolfach, 1997), DMV Sem., vol. 29, Birkhäuser, Basel, 2000, pp. 43–73.

[8] I. M. Gel′fand - M. M. Kapranov - A. V. Zelevinsky, *Discriminants, resultants, and multidimensional determinants*, Mathematics: Theory & Applications, Birkhäuser Boston, Inc., Boston, MA, 1994.

[9] Hiroshi Imai - Tomonari Masada - Fumihiko Takeuchi - Keiko Imai, *Enumerating triangulations in general dimensions*, Internat. J. Comput. Geom. Appl. 12 no. 6 (2002), 455–480.

[10] Charles Jordan - Michael Joswig - Lars Kastner, *Parallel enumeration of triangulations.*, Electron. J. Comb. 25 no. 3 (2018), research paper p3.6, 27 (English).

[11] Charles Jordan - Michael Joswig - Lars Kastner, ***mptopcom**, version 1.1*, Open source software for the parallel enumeration of triangulations (2019), https://polymake.org/mptopcom.

[12] E. J. Nanson, *On the eliminant of a set of quadrics, ternary or quaternary.*, Proc. R. Soc. Edinburgh 22 (1899), 353–358 (English).

[13] Jörg Rambau, *`TOPCOM`: triangulations of point configurations and oriented matroids*, Mathematical software (Beijing, 2002), World Sci. Publ., River Edge, NJ, 2002, pp. 330–340.

[14] Bernd Sturmfels - Kristian Ranestad, *Twenty-seven questions about the cubic surface*, this volume.

LARS KASTNER
Institut für Mathematik
TU Berlin
Str. des 17. Juni 136, 10623 Berlin, Germany
e-mail: `kastner@math.tu-berlin.de`

ROBERT LÖWE
Institut für Mathematik
TU Berlin
Str. des 17. Juni 136, 10623 Berlin, Germany
e-mail: `loewe@math.tu-berlin.de`