Algebraic Polytopes in Normaliz

Winfried Bruns(✉)

Institut für Mathematik, Universität Osnabrück, 49069 Osnabrück, Germany
wbrunas@uos.de
http://www.home.uni-osnabrueck.de/wbruns/

Abstract. We describe the implementation of algebraic polyhedra in Normaliz. In addition to convex hull computation/vertex enumeration, Normaliz computes triangulations, volumes, lattice points, face lattices and automorphism groups. The arithmetic is based on the package *e-antic* by V. Delecroix.

Keywords: Polyhedron · Real algebraic number field · Computation

Algebraic polytopes lacking a rational realization are among the first geometric objects encountered in high school geometry: at least one vertex of an equilateral triangle in the plane has non-rational coordinates. Three of the five Platonic solids, namely the tetrahedron, the icosahedron and the dodecahedron are non-rational, and, among the 4-dimensional regular polytopes, the 120-cell and the 600-cell live outside the rational world.

But algebraic polytopes do not only appear in connection with Coxeter groups. Other contexts include enumerative combinatorics [17], Dirichlet domains of hyperbolic group actions [8], $\mathrm{SL}(2,\mathbb{R})$-orbit closures in the moduli space of translation surfaces, and parameter spaces and perturbation polyhedra of cut-generating functions in integer programming.

1 Real Embedded Algebraic Number Fields

The notion of convexity is defined over any ordered field, not only over the rationals $\mathbb{Q}$ or the reals $\mathbb{R}$. *Real embedded algebraic number fields* are subfields of the real numbers (and therefore ordered) that have finite dimension as a $\mathbb{Q}$-vector space. It is well known that such a field $\mathbb{A}$ has a primitive element, i.e., an element a such that no proper subfield of $\mathbb{A}$ contains a. The minimal polynomial of a is the least degree monic polynomial μ with coefficients in $\mathbb{Q}$ such that $\mu(a) = 0$. It is an irreducible polynomial, and $\dim_{\mathbb{Q}} \mathbb{A} = \deg \mu$. In particular, every element b of $\mathbb{A}$ has a unique representation $b = \alpha_{n-1}a^{n-1} + \cdots + \alpha_1 a + \alpha_0$ with $\alpha_{n-1}, \ldots, \alpha_0 \in \mathbb{Q}$, $n = \deg \mu$. The arithmetic in $\mathbb{A}$ is completely determined by μ: addition is the addition of polynomials and multiplication is that of polynomials followed by reduction modulo μ. The multiplicative inverse can be computed by the extended Euclidean algorithm. The unique determination of the coefficients α_i allows one

A. M. Bigatti et al. (Eds.): ICMS 2020, LNCS 12097, pp. 193–201, 2020.
https://doi.org/10.1007/978-3-030-52200-1_19

to decide whether $b = 0$. Every element of $\mathbb{A}$ can be written as the quotient of a polynomial expression $\alpha_{n-1}a^{n-1} + \cdots + \alpha_1 a + \alpha_0$ with $\alpha_i \in \mathbb{Z}$ for all i and an integer denominator; this representation is used in the implementation.

However, the algebraic structure alone does not define an ordering of $\mathbb{A}$. For example, $\sqrt{2}$ and $\sqrt{-2}$ cannot be distinguished algebraically: there exists an automorphism of $\mathbb{Q}[\sqrt{2}]$ that exchanges them. For the ordering we must fix a real number a whose minimal polynomial is μ. (Note that not every algebraic number field has an embedding into $\mathbb{R}$.) In order to decide whether $b > 0$ for some $b \in \mathbb{A}$ we need a floating point approximation to b of controlled precision.

Normaliz [4] uses the package *e-antic* of V. Delecroix [7] for the arithmetic and ordering in real algebraic number fields. The algebraic operations are realized by functions taken from the package *antic* of W. Hart and F. Johansson [11] (imported to *e-antic*) while the controlled floating point arithmetic is delivered by the package *arb* of F. Johansson [13]. Both packages are based on W. Hart's *Flint* [12].

In order to specify an algebraic number field, one chooses the minimal polynomial μ of a and an interval I in $\mathbb{R}$ such that μ has a unique zero in I, namely a. An initial approximation to a is computed at the start. Whenever the current precision of b does not allow to decide whether $b > 0$, first the approximation of b is improved, and if the precision of a is not sufficient, it is replaced by one with twice the number of correct digits.

2 Polyhedra

A subset $P \subset \mathbb{R}^d$ is a *polyhedron* if it is the intersection of finitely many affine halfspaces:

$$P = \bigcap_{i=0}^{s} H_i^+, \qquad H_i^+ = \{x : \lambda_i(x) \geq \beta_i\}, \qquad i = 1, \ldots, s,$$

where λ_i is a linear form and $\beta_i \in \mathbb{R}$. It is a *cone* if one can choose $\beta_i = 0$ for all i, and it is a *polytope* if it is bounded.

By the theorem of Minkowski-Weyl-Motzkin [2, 1.C] one can equivalently describe polyhedra by "generators": there exist $c_1, \ldots, c_t \in \mathbb{R}^d$ and $v_1, \ldots, v_u \in \mathbb{R}^d$ such that

$$P = C + Q$$

where $C = \{c\gamma_1 c_1 + \cdots + \gamma_t c_t : \gamma_i \in \mathbb{R}, \gamma_i \geq 0\}$ is the *recession cone* and $Q = \{\kappa_1 v_1 + \cdots + \kappa_u v_u : \kappa_i \in \mathbb{R}, \kappa_i \geq 0, \sum \kappa_i = 1\}$ is a polytope. These two descriptions are often called *H-representation* and *V-representation*. The conversion from H to V is *vertex enumeration* and the opposite conversion is *convex hull computation*.

For theoretical and computational reasons it is advisable to present a polyhedron P as the intersection of a cone and a hyperplane. Let $C(P)$ be the *cone over* P, i.e., the smallest cone containing $P \times \{1\}$, and $D = \{x : x_{d+1} = 1\}$ the *dehomogenizing hyperplane*. Then P can be identified with $C(P) \cap D$. After this

step, convex hull computation and vertex enumeration are two sides of the same coin, namely the dualization of cones.

In the definition of polyhedra and all statements following it, the field $\mathbb{R}$ can be replaced by an arbitrary subfield (and even by an arbitrary ordered field), for example a real algebraic number field $\mathbb{A}$. The smallest choice for $\mathbb{A}$ is $\mathbb{Q}$: for it we obtain the class of *rational polyhedra*. For general $\mathbb{A}$ we get *algebraic polyhedra*.

For the terminology related to polyhedra and further details we refer the treader to [2].

3 Normaliz

Normaliz tackles many computational problems for rational and algebraic polyhedra:

- dual cones: convex hulls and vertex enumeration
- projections of cones and polyhedra
- triangulations, disjoint decompositions and Stanley decompositions
- Hilbert bases of rational, not necessarily pointed cones
- normalizations of affine monoids (hence the name)
- lattice points of polytopes and (unbounded) polyhedra
- automorphisms (euclidean, integral, rational/algebraic, combinatorial)
- face lattices and f-vectors
- Euclidean and lattice normalized volumes of polytopes
- Hilbert (or Ehrhart) series and (quasi) polynomials under $\mathbb{Z}$-gradings
- generalized (or weighted) Ehrhart series and Lebesgue integrals of polynomials over rational polytopes

Of course, not all of these computation goals make sense for algebraic polyhedra. The main difference between the rational and the non-rational case can be described as follows: the monoid of lattice points in a full dimensional cone is finitely generated if and only if the cone is rational.

Normaliz is based on a templated C++ library. The template allows one to choose the arithmetic, and so it would be possible to extend Normaliz to more general ordered fields. The main condition is that the arithmetic of the field has been coded in a C++ class library. There is no restriction on the real algebraic number fields that Normaliz can use.

Normaliz has a library as well as a file interface. It can be reached from CoCoA, GAP [9], Macaulay2, Singular, Python [10] and SageMath. The full functionality is reached on Linux and Mac OS platforms, but the basic functionality for rational polyhedra is also available on MS Windows systems.

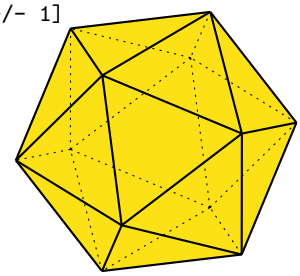
Its history goes back to the mid 90ies. For recent developments see [3] and [6]. The extension to algebraic polytopes was done in several steps since 2016. We are grateful to Matthias Köppe for suggesting it.

The work on algebraic polytopes has been done in cooperation with Vincent Delecroix (*e-antic*), Sebastian Gutsche (PyNormaliz), Matthias Köppe and Jean-Phiippe Labbé (integration into SageMath). A comprehensive article with these coauthors is in preparation.

4 The Icosahedron

Let us specify the icosahedron, a Platonic solid, by its vertices:

```
amb_space 3
number_field min_poly (a^2 - 5) embedding [2 +/- 1]
vertices 12
0 2 (a + 1) 4
0 -2 (a + 1) 4
2 (a + 1) 0 4
...
(-a - 1) 0 -2 4
Volume
LatticePoints
FVector
EuclideanAutomorphisms
```



The first line specifies the dimension of the affine space. The second defines the unique positive sqare root of 5 as the generator of the number field. It is followed by the 12 vertices. Each of them is given as a vector with 4 components for which the fourth component acts as a common denominator of the first three. Expressions involving a are enclosed in round brackets. The last lines list the computation goals for Normaliz. (Picture by J.-P. Labbé)

Normaliz has a wide variety of input data types. For example, it would be equally possible to define the icosahedron by inequalities. Now we have a look into the output file. (We indicate omitted lines by ...)

```
Real embedded number field:
min_poly (a^2 - 5) embedding [2.23606797...835961152572 +/- 5.14e-54]

1 lattice points in polytope
12 vertices of polyhedron
0 extreme rays of recession cone
20 support hyperplanes of polyhedron (homogenized)
f-vector:
1 12 30 20 1
embedding dimension = 4
affine dimension of the polyhedron = 3 (maximal)
rank of recession cone = 0 (polyhedron is polytope)
...
volume (lattice normalized) = (5/2*a+15/2 ~ 13.090170)
volume (Euclidean) = 2.18169499062
Euclidean automorphism group has order 120
**********************************************************************
1 lattice points in polytope:
0 0 0 1
12 vertices of polyhedron:
...
0 extreme rays of recession cone:
20 support hyperplanes of polyhedron (homogenized):
(-a+1 ~ -1.236068) (-2*a+4 ~ -0.472136)                    0 1
```

```
...
  (a-1 ~ 1.236068)   (2*a-4 ~ 0.472136)                    0 1
```

The output (in homogenized coordinates) is self-explanatory. Note that non-integral numbers in the output are printed as polynomials in a together with a rational approximation. At the top we can see to what precision $\sqrt{5}$ had to be computed. The automorphism group is described in another output file:

```
Euclidean automorphism group of order 120
**************************************************************************
3 permutations of 12 vertices of polyhedron
Perm 1: 1 2 4 3 7 8 5 6 10 9 11 12
Perm 2: 1 3 2 5 4 6 7 9 8 11 10 12
Perm 3: 2 1 3 4 6 5 8 7 9 10 12 11
Cycle decompositions
Perm 1: (3 4) (5 7) (6 8) (9 10) --
Perm 2: (2 3) (4 5) (8 9) (10 11) --
Perm 3: (1 2) (5 6) (7 8) (11 12) --
1 orbits of vertices of polyhedron
Orbit 1 , length 12:  1 2 3 4 5 6 7 8 9 10 11 12
**************************************************************************
3 permutations of 20 support hyperplanes
Perm 1: 2 1 5 6 3 4 7 8 11 12 9 10 13 14 17 18 15 16 20 19
...
Cycle decompositions
Perm 1: (1 2) (3 5) (4 6) (9 11) (10 12) (15 17) (16 18) (19 20) --
...
1 orbits of support hyperplanes
Orbit 1 , length 20:  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

5 Computation Goals for Algebraic Polyhedra

The basic computation in linear convex geometry is the dualization of cones. We start from a cone $C \subset \mathbb{R}^d$, given by generators $x_1, \dots, x_n$. The first (easy) step is to find a coordinate transformation that replaces $\mathbb{R}^d$ by the vector subspace generated by $x_1, \dots, x_n$. In other words, we can assume $\dim C = d$.

The goal is to find a minimal generating set $\sigma_1, \dots, \sigma_s \in (\mathbb{R}^d)^*$ of the dual cone $C^* = \big\{\lambda : \lambda(x_i) \geq 0,\ i = 1, \dots, n\big\}$. Because of $\dim C = d$, the linear forms $\sigma_1, \dots, \sigma_s$ are uniquely determined up to positive scalars: they are the extreme rays of C^*. By a slight abuse of terminology we call the hyperplanes $S_i = \{x : \sigma_i(x) = 0\}$ the *support hyperplanes* of C.

Let C_k be the cone generated by $x_1, \dots, x_k$. Normaliz proceeds as follows:

1. It finds a basis of $\mathbb{R}^d$ among the generators $x_1, \dots, x_n$, say $x_1, \dots, x_d$. Computing C_d^* amounts to a matrix inversion.
2. Iteratively it extends the cone C_k to C_{k+1}, and shrinks C_k^* to C_{k+1}^*, $k = d \dots, n-1$.

Step 2 is done by *Fourier-Motzkin elimination*: if $\sigma_1, \dots, \sigma_t$ generate C_k^*, then C_{k+1}^* is generated by

$$\big\{\sigma_i : \sigma_i(x_{k+1}) \geq 0\big\} \cup \big\{\sigma_i(x_{k+1})\sigma_j - \sigma_j(x_{k+1})\sigma_i : \sigma_i(x_{k+1}) \ > 0, \sigma_j(x_{k+1}) < 0\big\}.$$

From this generating set of C_{k+1}^* the extreme rays of C_{k+1}^* must be selected.

This step is of critical complexity. Normaliz has a sophisticated implementation in which *pyramid decomposition* is a crucial tool; see [5]. It competes very well with dedicated packages (see [14]). The implementation is independent of the field of coefficients. As said above, $\mathbb{R}$ can be replaced by an algebraic number field $\mathbb{A}$. In this case Normaliz uses the arithmetic over the field $\mathbb{A}$ realized by *e-antic*, whereas arithmetic over $\mathbb{Q}$ is avoided in favor of arithmetic over $\mathbb{Z}$.

In addition to the critical complexity caused by the combinatorics of cones, one must tame the coordinates of the linear combination $\lambda = \sigma_i(x_{k+1})\sigma_j - \sigma_j(x_{k+1})\sigma_i$. For example, if, over $\mathbb{Z}$, both σ_i and σ_j are divisible by 2, then λ is divisible by 4. If this observation is ignored, a doubly exponential explosion of coefficients will happen. One therefore extracts the gcd of the coordinates. But there is usually no well-defined gcd of algebraic integers, and even if one has unique decomposition into prime elements, there is in general no Euclidean algorithm. Normaliz therefore applies two steps:

1. λ is divided by the absolute value of the last nonzero component (or by another "norm").
2. All integral denominators are cleared by multiplication with their lcm.

Computational experience has shown that these two steps together are a very good choice.

Normaliz tries to measure the complexity of the arithmetic in $\mathbb{A}$ and to control the algorithmic alternatives of the dualization by the measurements. There are several "screws" that can be turned, and it is difficult to find the optimal tuning beforehand.

Normaliz computes lexicographic triangulations of algebraic cones in the same way as triangulations of rational cones. Their construction is interleaved with the extension from C_k to C_{k+1}: the already computed triangulation of C_k is extended by the simplicial cones generated by x_{k+1} and those subcones in the triangulation of C_k that are "visible" from x_{k+1}.

An algebraic polytope P contains only finitely many integral points. They are computed by Normaliz' project-and-lift algorithm. The truncated Hilbert basis approaches, which Normaliz can also use for rational polytopes, are not applicable in the algebraic case. Once the lattice points are known, one can compute their convex hull, called the *integer hull* of P.

At present Normaliz computes volumes only for full-dimensional algebraic polytopes. The volume is the sum of the volumes of the simplices in a triangulation, and these are simply (absolute values of) determinants. We do not see any reasonable definition of "algebraic volume" for lower dimensional polytopes that could replace the lattice normalized volume. The latter is defined for all rational polytopes and is a rational number that can be computed precisely.

It would certainly be possible to extend the computation of the approximate Euclidean volume to all algebraic polytopes, and this extension may be included in future Normaliz versions. Note that the Euclidean volume does in general not belong to $\mathbb{A}$ if P is lower dimensional. Its precise computation would require an extension of $\mathbb{A}$ by square roots.

The computation of automorphism groups follows the suggestions in [1]. First one transforms the defining data into a graph, and then computes the automorphism group of this graph by *nauty* [15]. For algebraic polytopes the Euclidean and the algebraic automorphism groups can be computed, and the combinatorial automorphism group is accessible for all polyhedra.

The Euclidean automorphism group is the group of rigid motions of the ambient space that map the polytope to itself, and the algebraic automorphism group is the group of affine transformations over $\mathbb{A}$ stabilizing the polytope. Both groups are finite, as well as the combinatorial automorphism group, the automorphism group of the face lattice, which can be computed from the facet-vertex incidence vectors, just as in the rational case.

We do not try to define the algebraic (or Euclidean) automorphism group for unbounded polyhedra. First of all, the algebraic automorphism group is infinite in general. Second, it would have to be realized as the permutation group of a vector configuration, and there seems to be no reasonable way to norm the involved vectors. But for polytopes we can and must use the vertices.

6 Scaled Convex Hull Computations

We illustrate the influence of the algebraic number field on the computation time by some examples. For each of them we start from a cone (over a polyhedron) that is originally defined over the integers. Then we scale some coordinates by elements of the field $\mathbb{A}$. This transformation preserves the combinatorial structure throughout. It helps to isolate the complexity of the arithmetic operations. The types of arithmetic that we compare are

int: original input, computation with machine integers,
mpz: same input as int, but computation with GMP mpz_class integers,
rat: same input as int, but computation in $\mathbb{Q}[\sqrt{5}]$,
sc2: scaled input in $\mathbb{Q}[\sqrt{5}]$,
sc8: scaled input in $\mathbb{Q}[\sqrt[8]{5}]$,
p12: scaled input in $\mathbb{Q}[a]$, $a^{12} + a^6 + a^5 + a^2 - 5 = 0$, $a > 1$.

The test candidates are A553 (from the Ohsugi-Hibi classification of contingency tables [16]), the cone q27f1 from [14], the linear order polytope for S_6, and the cyclic polytope of dimension 15 with 30 vertices. The last two are classical polytopes. While the other three cones are given by their extreme rays, q27f1 is defined by 406 equations and inequalities (Table 1).

The Normaliz version is 3.8.4, compiled into a static binary with gcc 5.4 under Ubuntu 16-04. The computations use 8 parallel threads (the default choice of Normaliz). They were taken on the author's PC with an AMD Ryzen 7 1700X

Table 1. Combinatorial data of the test candidates

	amb_space	dim	ext rays	supp hyps
A553	55	43	75	306,955
q27f1	30	13	68,216	92
lo6	16	16	720	910
cyc15-30	16	16	30	341088

at 3.2 GHz. Table 2 lists wall times in seconds. As a rule of thumb, for a single thread the times must be multiplied by 6.

Table 2. Wall times of scaled convex hull computations in seconds

Coeff	A553	q27f1	lo6	cyc15-30
int	57	16	5	–
mpz	299	58	5	7
rat	277	40	5	7
sc2	783	166	4	14
sc8	1272	475	15	28
p12	2908	905	31	42

The cyclic polytope and all intermediate polytopes coming up in its computation are simplicial. Therefore it profits from Normaliz' special treatment of simplicial facets—almost everything can be done by set theoretic operations. Also lo6 is combinatorially not complicated. That lo6 is fastest with sc2, is caused by the fine tuning of the pyramid decomposition, which is not always optimal.

Surprisingly, rat is faster than mpz for A553 and q27f1. This can be explained by the fact that linear algebra over $\mathbb{Z}$ must use the Euclidean algorithm, and therefore needs more steps than the true rational arithmetic of rat.

References

1. Bremner, D., Sikirić, M.D., Pasechnik, D.V., Rehn, T., Schürmann, A.: Computing symmetry groups of polyhedra. LMS J. Comput. Math. **17**, 565–581 (2014)
2. Bruns, W., Gubeladze, J.: Polytopes, Rings, and K-Theory. Springer, Dordrecht (2009)
3. Bruns, W., Ichim, B.: Polytope volume by descent in the face lattice and applications in social choice. Preprint arXiv:1807.02835
4. Bruns, W., Ichim, B., Römer, T., Sieg, R., Söger, C.: Normaliz. Algorithms for rational cones and affine monoids. http://normaliz.uos.de
5. Bruns, W., Ichim, B., Söger, C.: The power of pyramid decomposition in Normaliz. J. Symb. Comp. **74**, 513–536 (2016)

6. Bruns, W., Sieg, R., Söger, C.: Normaliz 2013–2016. In: Böckle, G., Decker, W., Malle, G. (eds.) Algorithmic and Experimental Methods in Algebra, Geometry, and Number Theory, pp. 123–146. Springer (2018)
7. Delecroix, V.: Embedded algebraic number fields (on top of antic). Package available at https://github.com/videlec/e-antic
8. Delecroix, V., Page, A.: Dirichlet fundamental domains for real hyperbolic spaces. Work in progress
9. Gutsche, S., Horn, M., Söger, C.: NormalizInterface for GAP. https://github.com/gap-packages/NormalizInterface
10. Gutsche, S., Sieg, R.: PyNormaliz - an interface to Normaliz from python. https://github.com/Normaliz/PyNormaliz
11. Hart, W., Johansson, F.: Algebraic Number Theory In C. Package available at https://github.com/wbhart/antic
12. Hart, W., Johansson, F., Pancratz, S.: FLINT: Fast Library for Number Theory. http://flintlib.org
13. Johansson, F.: Arb - a C library for arbitrary-precision ball arithmetic. http://arblib.org/
14. Köppe, M., Zhou, Y.: New computer-based search strategies for extreme functions of the Gomory-Johnson infinite group problem. Math. Program. Comput. **9**, 419–469 (2017)
15. McKay, B.D., Piperno, A.: Practical graph isomorphism. II. J. Symbolic Comput. **60**, 94–112 (2014)
16. Ohsugi, H., Hibi, T.: Toric ideals arising from contingency tables. In: Bruns, W. (ed.) Commutative Algebra and Combinatorics. Ramanujan Mathematical Society Lecture Note Series 4, pp. 87–111 (2006)
17. Rote, G.: The maximum number of minimal dominating sets in a tree. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1201–1214. SIAM, Philadelphia (2019)