Normaliz: A package for linear diophantine systems, polyhedra and lattices

W. Bruns
(Universität Osnabrück, Institut für Mathematik, Germany)

wbruns@uos.de

The very first version of Normaliz [5] was meant to compute normalizations of affine monoids (or semigroups), hence the name. Over the years it has been extended to a powerful package for discrete convex geometry. We explain its main computation goals by a simple example, sketch the mathematical background and discuss the basic steps in the Normaliz primal algorithm. Some remarks on the technical aspects and the history conclude this overview.

The main computation goals

Suppose we are interested in the following type of 3×3 matrices

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

and the problem is to find nonnegative integer values for $x_1, \dots, x_9$ such that the 3 numbers in all rows, all columns, and both diagonals sum to the same constant $\mathcal{M}$. Sometimes such matrices are called *magic squares* and $\mathcal{M}$ is the *magic constant*. This leads to a system of 7 diophantine linear equations:

$$\begin{aligned} x_1 + x_2 + x_3 &= x_4 + x_5 + x_6; \\ \dots &= \dots \\ x_1 + x_2 + x_3 &= x_3 + x_5 + x_7. \end{aligned} \tag{6}$$

Our goal is to understand the set of solutions in nonnegative integers.

Since the equations are linear homogeneous, and the positive orthant is closed under addition, the sum of two solutions is again a solution (the magic constants add up). It follows that the set M of all solutions is a monoid (or semigroup). In a systematic framework we can describe it as follows:

$$M = C \cap L$$

where C is a pointed cone in $\mathbb{R}^d$ and L is a sublattice of $\mathbb{Z}^d$. In our case $d = 9$, C is the positive orthant, and L is the lattice of integer solutions to the system (6). There are other ways to describe M, but this choice of C and L is exactly Normaliz' approach to the problem. Two natural questions suggest themselves:

> (*Generation*) Is the monoid of magic squares finitely generated, and if so, what is a system of generators?
>
> (*Enumeration*) Given $\mathcal{M}$, how many magic squares of magic constant $\mathcal{M}$ are there?

The answers to *Generation* and *Enumeration* are the main computation goals of Normaliz.

Some background

The introductory example is a rather special case of the objects for which Normaliz solves *Generation* and *Enumeration*: since version 3.0 it can be applied to arbitrary intersections of rational polyhedra and affine lattices. In other words, Normaliz is a solver for systems of affine-linear diophantine inequalities, equations and congruences.

Nevertheless, for the sake of simplicity, in this overview we stick to the special case $M = C \cap L$, and specialize it even further, following Normaliz' course of computation. In our example we can first introduce coordinates in the solution space of the system (6), and then intersect the positive orthant with it. After this reduction step, we can assume that we want to compute the integer points in a d-dimensional pointed cone $C \subset \mathbb{R}^d$. This is the core case to which Normaliz reduces the given input data by preliminary transformations. Let us first solve *Generation* for cones:

Theorem 1 (Minkowski-Weyl) Let $C \subset \mathbb{R}^d$. Then the following are equivalent:

1. There exist (integer) vectors $x_1, \dots, x_n \in \mathbb{R}^d$ such that

$$C = \{x \in \mathbb{R}^d : x = \alpha_1 x_1 + \cdots + \alpha_n x_n, \\ \alpha_1 \dots, \alpha_n \geq 0\}.$$

2. There exist linear forms $\lambda_1, \dots, \lambda_s$ (with integer coefficients) on $\mathbb{R}^d$ such that

$$C = \{x \in \mathbb{R}^d : \lambda_i(x) \geq 0,\ i = 1, \dots, s\}.$$

If the equivalent conditions of the theorem are satisfied, C is called a (*rational*) *cone*. (Sometimes the attribute *polyhedral* is added.) The cone C is *pointed* if it does not contain a linear subspace of positive dimension. In this case the elements of a minimal set $x_1, \dots, x_n$ of generators are unique up to permutation and multiplication by positive scalars. We call them *extreme rays*. The scalar is 1 in the integer case if we require that the coefficients are coprime.

The dimension of a cone is the dimension of the vector subspace it generates. If $\dim C = d$, then the linear forms λ_i are unique up to permutation and positive scalar multiples if the system is minimal, and again the scalar must be 1 if we require that their coordinates are coprime integers. The linear forms $\lambda_1, \dots \lambda_s$ then define the (relevant) *support hyperplanes* $H_i = \{x : \lambda_i(x) = 0\}$, and Theorem 1(2) represents C as an intersection of the (positive) *halfspaces* $H^+_{\lambda_i} = \{x : \lambda_i(x) \geq 0\}$.

The conversion from generators to support hyperplanes is usually called *convex hull computation* and the converse is called *vertex enumeration*. Both directions are completely equivalent since they amount to the dualization of a cone.

That *Generation* makes sense also for lattice points is guaranteed by

Theorem 2 (Gordan's lemma) Let $C \subset \mathbb{R}^d$ be a rational pointed cone. Then the monid $M = C \cap \mathbb{Z}^d$ is finitely generated. More precisely, it has a unique minimal system E of generators.

A set $\{z_1, \dots, z_m\} \subset M$ generates the monoid M if every element of M is a linear combination of $z_1, \dots, z_m$ with nonnegative integer coefficients. The unique minimal generating set $\mathrm{Hilb}(C)$ (or $\mathrm{Hilb}(M)$) in Theorem 2 is called the *Hilbert basis*

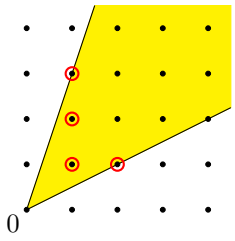

Figure 1: *A Hilbert basis*

of C (or M). We do not see a good historical reason for this nomenclature – the term "Gordan basis" would be much more appropriate. In the restricted setting that we have reached, we can now reformulate *Generation* as follows:

Compute the Hilbert basis of a rational cone C.

The Hilbert basis is the set of *irreducible* elements in M, i.e., elements x that can not be written in the form $x = y + z$ with $y, z \in M$, $y, z \neq 0$. This proves uniqueness, and is important for the reduction of a generating set to the Hilbert basis.

Let us turn to *Enumeration*. For it we need a *grading*, a $\mathbb{Z}$-linear form $\deg : \mathbb{Z}^d \to \mathbb{Z}$. For the next theorem we restrict the generality even further.

Theorem 3 (Hilbert; Ehrhart) Let $d \geq 1$. Suppose that the extreme rays of the d-dimensional cone C have degree 1 and set $M = C \cap \mathbb{Z}^d$. Then the lattice point enumerator

$$H(M, k) = \#\{x \in M : \deg x = 1\}$$

is given by a polynomial q_M with rational coefficients for all $k \geq 0$. Equivalently, the generating function $\sum_{k=0}^{\infty} H(M, k) t^k$ defines a rational function of type

$$H_M(t) = \frac{1 + h_1 t + \cdots + h_u t^u}{(1-t)^d}, \\ h_1, \dots, h_u \in \mathbb{Z},\ u < d.$$

The polynomial q_M is called the *Hilbert polynomial* and $H_M(t)$ the *Hilbert series* of M – for very good reason since the theorem can be derived from the theory of graded algebras, a key observation of Stanley. In the combinatorial context it was independently proved by Ehrhart, and therefore one speaks of the *Ehrhart polynomial* and *Ehrhart series* as well. The h_i are nonnegative; this follows from Hochster's theorem by which the monoid algebra $K[M]$ is Cohen-Macaulay for any field K, but it can also be shown combinatorially.

Without the hypothesis on the degree of the extreme rays, the theorem must be reformulated: the polynomial is only a quasipolynomial in general, i.e., a "polynomial" with coefficients that are periodic in k, and the denominator takes a more complicated form.

The leading coefficient of the Hilbert/Ehrhart polynomial q_M has the form $e(M)/(d-1)!$ with a positive integer $e(M)$ that is called the *multiplicity* of M. It is the lattice normalized volume of the polytope spanned by the extreme rays. Moreover, $e(M) = 1 + h_1 + \cdots + h_u$. Clearly

Compute the Hilbert series of M

is the right formulation of *Enumeration* now.

The magic squares continued

The input to Normaliz for the computation of the magic squares is encoded as follows:

```
amb_space 9
equations 7
1 1 1 -1 -1 -1  0  0  0
...
1 1 0  0 -1  0 -1  0  0
grading
1 1 1 0 0 0 0 0 0
```

The first equation reads

$$x_1 + x_2 + x_3 - x_4 - x_5 - x_6 = 0,$$

as desired, and the other equations are encoded analogously. The magic constant is the grading. If only equations are given, but no inequalities, Normaliz assumes that the nonnegative solutions should be computed. We look at some data in the output file:

```
5 Hilbert basis elements
5 Hilbert basis elements of degree 1
4 extreme rays
4 support hyperplanes
embedding dimension = 9
rank = 3
...
grading:
1 1 1 0 0 0 0 0 0
with denominator = 3
```

The input grading is the magic constant. However, as the denominator 3 shows, the magic constant is always divisible by 3, and therefore the effective degree is $\mathcal{M}/3$. This degree is used for the multiplicity and the Hilbert series.

```
multiplicity = 4
Hilbert series:
1 2 1
denominator with 3 factors:
1: 3
degree of Hilbert Series ... = -1
Hilbert polynomial:
1 2 2
with common denominator = 1
```

The Hilbert series and the Hilbert polynomial are

$$H_M(t) = \frac{1 + 2t + t^2}{(1-t)^3} \quad \text{and} \quad q_M(k) = 1 + 2k + 2k^2,$$

and after substituting $\mathcal{M}/3$ for k we obtain the number of magic squares of magic constant $\mathcal{M}$, provided 3 divides $\mathcal{M}$.

```
5 Hilbert basis elements of degree 1:
0 2 1 2 1 0 1 0 2
1 0 2 2 1 0 0 2 1
1 1 1 1 1 1 1 1 1
1 2 0 0 1 2 2 0 1
2 0 1 0 1 2 1 2 0
```

The 5 elements of the Hilbert basis represent magic squares. We show the first, third and fifth:

0	2	1
2	1	0
1	0	2

1	1	1
1	1	1
1	1	1

2	0	1
0	1	2
1	2	0

All other solutions are linear combinations of these squares with nonnegative integer coefficients.

Actually we were lucky: if we increase the format of the squares to 4×4 or higher, the extreme rays are no longer of degree 1. Normaliz computes the Hilbert basis and Hilbert series quickly for 5×5 squares, and the Hilbert basis (of $522,347$ vectors) for 6×6 squares in reasonable time, but the Hilbert series for 6×6 is out of reach.

The primal algorithm

For the computation of Hilbert bases Normaliz provides two algorithms at the user's disposal. Here we restrict ourselves to the *primal algorithm* that also computes the Hilbert series if this is desired. For the *dual algorithm* we refer the reader to Bruns and Ichim [3].

The primal algorithm is based on triangulations. We assume that the cone C is pointed and defined by a generating set. (If it is defined by inequalities, Normaliz first computes the extreme rays.) After some preliminary transformations we can further assume that C has dimension d and that $\mathbb{Z}^d$ is the lattice to be used. Then the primal algorithm proceeds in the following steps:

1. Fourier-Motzkin elimination computing the support hyperplanes of C;
2. pyramid decomposition and computation of the lexicographic triangulation Δ;
3. evaluation of the simplicial cones in the triangulation:
   (a) enumeration of the set of lattice points E_σ in the fundamental domain of a simplicial subcone σ,
   (b) reduction of E_σ to the Hilbert basis $\mathrm{Hilb}(\sigma)$,
   (c) Stanley decomposition for the Hilbert series of $\sigma \cap L$;
4. Collection of the local data:
   (a) reduction of $\bigcup_{\sigma\in\Delta} \mathrm{Hilb}(\sigma)$ to $\mathrm{Hilb}(C \cap L)$,
   (b) accumulation of the Hilbert series of the monoids $\sigma \cap L$.

This is the true chronological order only for small examples. Typically the steps are interleaved in a complicated way. We now explain some steps in more detail.

Convex hulls and triangulation

Fourier-Motzkin elimination allows us to compute the support hyperplanes incrementally by extending the cone $C' = \mathbb{R}_+x_1 + \cdots + \mathbb{R}_+x_{n-1}$ to $C = \mathbb{R}_+x_1 + \cdots + \mathbb{R}_+x_n$. Suppose that

$$C' = H^+_{\lambda_1} \cap \cdots \cap H^+_{\lambda_r}$$

for linear forms $\lambda_1, \ldots, \lambda_r$. We may assume that

$$\lambda_i(x_n) \begin{cases} = 0 & i = 1, \ldots, p, \\ > 0 & i = p+1, \ldots, q, \\ < 0 & i = q+1, \ldots, r. \end{cases}$$

Set

$$\mu_{ij} = \lambda_i(x_n)\lambda_j - \lambda_j(x_n)\lambda_i, \quad \begin{matrix} i = p+1, \ldots, q, \\ j = q+1, \ldots, r. \end{matrix}$$

Theorem 4 The cone $C = \mathbb{R}_+x_1 + \cdots + \mathbb{R}_+x_n$ is the intersection of the halfspaces defined by the linear forms λ_i, $i = 1, \ldots, q$, and μ_{ij}, $i = p+1, \ldots, q$, $j = q+1, \ldots, r$.

The theorem leads to an extremely simple algorithm. However, one must discard the superfluous ones among the μ_{ij}, and this needs some care; see Bruns and Ichim [3] for the strategies of Normaliz.

The extension of a triangulation of C' to a triangulation of C is easy as well. A triangulation is a face-to-face decomposition of C into simplicial cones, i.e., cones whose extreme rays are linearly independent.

Theorem 5 Let Δ' be a triangulation of C'. Then we obtain a triangulation Δ of C by adding to Δ' all simplicial cones $(\sigma \cap H) + \mathbb{R}_+x_n$ where σ runs through Δ and H is a support hyperplane of C' such that x_n belongs to the negative halfspace defined by H.

The triangulations computed by Theorem 5 are called *lexicographic* or *pushing*.

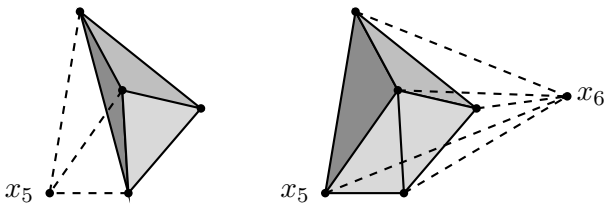

Figure 2: *A lexicographic triangulation of a cone in dimension* 4

Applied algorithmically, both theorems require a double loop, and especially the rapid growth of triangulations in high dimensions forbids a direct application of Theorem 5. We replace it by a hybrid approach based on *pyramid decomposition*: instead of computing all the intersections $\sigma \cap H$, $\sigma \in \Delta'$, we triangulate $(H \cap C') + \mathbb{R}_+x_n$ directly. Pyramid decomposition can be applied both recursively and in parallel for several H.

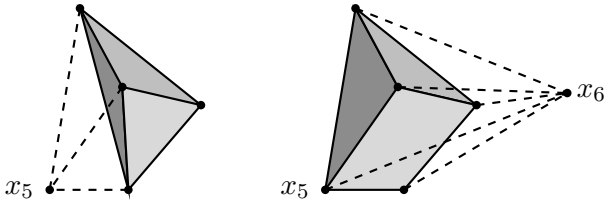

Figure 3: *Pyramid decomposition of Figure 2*

Also the computation of support hyperplanes profits from it. We refer the reader to Bruns, Ichim and Söger [6] for a detailed description of the strategy applied by Normaliz.

Reduction, simplicial cones and Hilbert bases

It is characteristic for Hilbert basis algorithms that they first compute a superset of the Hilbert basis and then reduce it by discarding reducible elements:

Theorem 6 Let $C \subset \mathbb{R}^d$ be a pointed rational cone, and let G be a system of generators of the monoid $M = C \cap \mathbb{Z}^d$. Then

$$\mathrm{Hilb}(C) = \{x \in G : x - y \notin C \text{ for all } y \in G, x \neq y\}.$$

Fortunately the condition $x - y \notin C$ can be tested quickly if the linear forms λ_i defining the support hyperplanes of C are known (and there are not too many of them): $x - y \notin C \iff \lambda_i(x) < \lambda_i(y)$ for at least one i. Nevertheless the reduction algorithm wants to be well-organized; see [3].

Suppose Σ is a triangulation of C. Then the union of the Hilbert bases $\mathrm{Hilb}(\sigma)$, $\sigma \in \Sigma$, is evidently a system of generators for M, and it must "only" be reduced. So it remains to explain how $\mathrm{Hilb}(C)$ is computed if C is a simplicial cone.

Let $v_1, \ldots, v_d \in \mathbb{Z}^d$ be linearly independent, generating the simplicial cone C. Set

$$\mathrm{par}(C) = \{x : x = \alpha_1v_1 + \cdots + \alpha_dv_d, \\ 0 \leq \alpha_i < 1,\ i = 1, \ldots, d\}.$$

The set $\mathrm{par}(C)$ is a semiopen parallelotope. It is a fundamental domain for the action of $U = \mathbb{Z}v_1 + \cdots + \mathbb{Z}v_d$ on $\mathbb{R}^d$ by parallel translation.

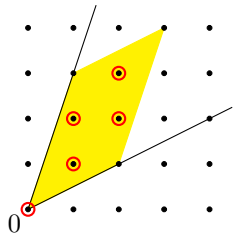

Figure 4: *The parallelotope* $\mathrm{par}(C)$ *and the set* E'

Theorem 7

1. The set $E' = \mathrm{par}(C) \cap \mathbb{Z}^d$ represents the residue classes of $\mathbb{Z}^d$ modulo the subgroup U.
2. $\#E' = |\det(v_1, \ldots, v_d)|$.
3. $E = \{v_1, \ldots, v_d\} \cup E'$ generates the monoid $C \cap \mathbb{Z}^d$.

The theorem shows that one can compute the elements of E' by enumerating the residue classes of $\mathbb{Z}^d$ and division with remainder. We have taken great care to make this process as efficient as possible.

The sets E' are reduced "locally" for all simplicial cones in the triangulation, and then their union is reduced "globally". This concludes the computation of the Hilbert basis by the primal algorithm.

Normaliz applies a strategy of *partial triangulation* which is based on pyramid decomposition. It tries to avoid the triangulation of those pyramids that can only yield previously known Hilbert basis elements. See Bruns et al. [2]. Often partial triangulation has an overwhelming effect.

With notation as above, let L be the monoid generated by $v_1, \dots, v_d$. Then $\bigcup_{x \in E'} x + L$ is a disjoint (!) decomposition of the monoid $M = C \cap \mathbb{Z}^d$, and this allows one to write down the Hilbert series of M immediately:

$$H_M(t) = \frac{\sum_{x \in E'} : t^{\deg x}}{(1-t^{g_1}) \cdots (1-t^{g_d})}, \qquad \begin{matrix} g_i = \deg v_i, \\ i = 1, \dots, d. \end{matrix}$$

The next step is not so easy, since a general cone is not the disjoint union of the simplicial cones in a triangulation. Fortunately there exist a disjoint decomposition into semiopen simplicial cones, and their Hilbert series are almost as easy to compute as those of closed simplicial cones. For the details we refer the reader to [6].

Use of Normaliz

Normaliz is implemented as a front end, called `normaliz`, and the kernel `libnormaliz`. The latter is a C++ class library and serves as an API. It uses GMP for infinite precision arithmetic and a small part of the Boost library. Parallelization is based on OpenMP. We provide binaries for Linux, MacOS and MS Windows.

The communication with the frontend `normaliz` is via an input file. Normaliz then returns the computation results in one or several output files. There is a multitude of input types for generators, equations, inequalities, congruences and gradings.

The computation goals and several other options can be set on the command line or in the input file. In particular, the computation goals can be restricted, for example to the support hyperplanes, to the lattice points in a polytope or the multiplicity. Normaliz tries to do its computation in 64 bit arithmetic, and if this precision fails, it automatically switches to infinite precision.

Normaliz has interfaces to CoCoA, GAP, Macaulay 2, polymake and Singular. Especially the GAP interface is very extensive, and at present GAP is the best environment for interactive access to Normaliz. The GUI jNormaliz (by Vicinius Almendra and Bogdan Ichim) is also very helpful.

Of course Normaliz has found many applications in the areas for which it has been made, commutative algebra, toric geometry, polytope theory and integer optimization. But since it is a solver for linear diophantine systems, it can be applied everywhere where such systems must be solved. A prime example is Burton's topological package Regina. The very efficient triangulation algorithm is applied in the package SecDec by Borowka et al. for the computation of multiscale integrals.

Normaliz has an offspring NmzIntegrate that computes weighted Ehrhart series and integrals of polynomials over rational polytopes [8].

Performance data of Normaliz can be found in [6].

History

The first version of Normaliz was developed by the author and his PhD student Robert Koch as a C program in 1998–2001 (see [7]). As the implementation did not allow any extensions, Normaliz was transferred to C++ by Bogdan Ichim in 2007–2008. On this basis the present code has been written by Christof Söger and the author since 2009. In 2014 Richard Sieg joined the team and made some contributions. The team is also supported by Tim Römer. The present published version is 3.1.1.

The first interface to Normaliz was the Singular library written by the author in 2002. The Macaulay 2 package was developed by Gesa Kämpf, and we owe the GAP interface to Sebastian Gutsche, Max Horn and Christof Söger. The CoCoA interface is due to John Abbott, Anna Bigatti and Christof Söger.

The list of references below has been restricted to primary references to Normaliz and its algorithms. We trust that the reader will be able to locate all mentioned software packages. The sources [1] and [6] contain extensive lists of references.

Acknowledgement. In 2013–2016 the development of Normaliz was supported by the DFG SPP 1489 "Algorithmische und experimentelle Methoden in Algebra, Geometrie und Zahlentheorie".

References

[1] W. Bruns and J. Gubeladze, *Polytopes, rings and K-theory*, Springer, 2009.

[2] W. Bruns, R. Hemmecke, B. Ichim, M. Köppe, and C. Söger, *Challenging computations of Hilbert bases of cones associated with algebraic statistics*. Exp. Math. **20** (2011), 25–33.

[3] W. Bruns and B. Ichim, *Normaliz: Algorithms for affine monoids and rational cones.* J. Algebra **324** (2010), 1098–1113.

[4] W. Bruns, B. Ichim, T. Römer, R. Sieg and C. Söger: Normaliz. Algorithms for rational cones and affine monoids. Available at `http://normaliz.uos.de`.

[5] W. Bruns, B. Ichim, T. Römer, R. Sieg and C. Söger: Normaliz. Algorithms for rational cones and affine monoids. Available at `http://normaliz.uos.de`.

[6] W. Bruns, B. Ichim and C. Söger. *The power of pyramid decompositions in Normaliz*. J. Symb. Comp. **74** (2016), 513–536.

[7] W. Bruns and R. Koch, *Computing the integral closure of an affine semigroup.* Univ. Iagel. Acta Math. **39** (2001), 59–70.

[8] W. Bruns and C. Söger, *Generalized Ehrhart series and integration in Normaliz*. J. Symb. Comp. **68** (2015), 75–86.