

Normaliz 3.10.0

Winfried Bruns Max Horn Ulrich von der Ohe

Former Normaliz 3 team members: Tim Römer, Richard Sieg and Christof Söger

Normaliz 2 team member: Bogdan Ichim

<https://normaliz.uos.de>

<https://github.com/Normaliz>

<mailto:normaliz@uos.de>

<https://hub.docker.com/u/normaliz/dashboard/>

<https://mybinder.org/v2/gh/Normaliz/NormalizJupyter/master>

Short reference: NmzShortRef.pdf

Contents

1	Introduction	10
1.1	The objectives of Normaliz	10
1.2	Platforms, implementation and access from other systems	11
1.3	Major changes relative to version 3.8.0	12
1.4	Future extensions	14
1.5	Download and installation	14
1.6	Exploring Normaliz online	15
2	Discrete convex geometry by examples	16
2.1	Terminology	16
2.2	Practical preparations	16
2.3	A cone in dimension 2	18
2.3.1	The Hilbert basis	19
2.3.2	The cone by inequalities	20
2.3.3	The interior	21

2.4	A lattice polytope	23
2.4.1	Only the lattice points	26
2.5	A rational polytope	26
2.5.1	The series with vertices?	29
2.5.2	The rational polytope by inequalities	29
2.6	Magic squares	30
2.6.1	Blocking the grading denominator	34
2.6.2	With even corners	35
2.6.3	The lattice as input	37
2.7	Decomposition in a numerical semigroup	38
2.8	A job for the dual algorithm	39
2.9	A dull polyhedron	40
2.9.1	Defining it by generators	42
2.10	The Condorcet paradox	42
2.10.1	Excluding ties	44
2.10.2	At least one vote for every preference order	45
2.10.3	The f-vector with codimension bound	46
2.11	Testing normality	46
2.11.1	Computing just a witness	47
2.12	Convex hull computation/vertex enumeration	48
2.13	Lattice points in a polytope and its Euclidean volume	50
2.14	The integer hull	53
2.15	Inhomogeneous congruences	55
2.15.1	Lattice and offset	56
2.15.2	Variation of the signs	57
2.16	Integral closure and Rees algebra of a monomial ideal	57
2.16.1	Only the integral closure of the ideal	58
3	Affine monoids and binomial ideals by examples	59
3.0.1	Input and default computation goals	59
3.0.2	Markov and Gröbner bases, Representations	60
3.0.3	Hilbert series and multiplicity	63
3.0.4	Binomial ideals from cone input	63
3.1	Monoids from binomials	65
3.1.1	Affine monoids from binomial ideals	65
3.1.2	Normalization of monoids from binomials	67
3.2	Lattice ideals	68
4	The input file	69
4.1	Input items	69
4.1.1	The ambient space and lattice	69
4.1.2	Plain vectors	70
4.1.3	Formatted vectors	70

4.1.4	Plain matrices	71
4.1.5	Formatted matrices	72
4.1.6	Constraints in tabular format	72
4.1.7	Constraints in symbolic format	73
4.1.8	Polynomials	74
4.1.9	Rational numbers	74
4.1.10	Decimal fractions and floating point numbers	74
4.1.11	Numbers in algebraic extensions of \mathbb{Q}	75
4.1.12	Numerical parameters	75
4.1.13	Computation goals and algorithmic variants	75
4.1.14	Comments	75
4.1.15	Restrictions	75
4.1.16	Homogeneous and inhomogeneous input	76
4.1.17	Default values	76
4.1.18	Normaliz takes intersections	77
4.2	Homogeneous generators	77
4.2.1	Cones	77
4.2.2	Lattices	78
4.2.3	Affine monoids	78
4.3	Homogeneous Constraints	78
4.3.1	Cones	78
4.3.2	Lattices	79
4.4	Inhomogeneous generators	79
4.4.1	Polyhedra	79
4.4.2	Affine lattices	80
4.5	Inhomogeneous constraints	80
4.5.1	Polyhedra	80
4.5.2	Affine lattices	81
4.6	Tabular constraints	81
4.6.1	Forced homogeneity	82
4.7	Symbolic constraints	82
4.8	Converting equations to inequalities	82
4.9	Polynomial constraints	82
4.10	Binomial ideals	83
4.11	Unit vectors and unit matrix	83
4.12	Grading	83
4.12.1	With binomial ideal input	84
4.13	Dehomogenization	84
4.14	Open facets	85
4.15	Coordinates for projection	85
4.16	Numerical parameters	85
4.16.1	Degree bound for series expansion	86
4.16.2	Number of significant coefficients of the quasipolynomial	86
4.16.3	Codimension bound for the face lattice	86

4.16.4	Degree bounds for Markov and Gröbner bases	86
4.16.5	Number of digits for fixed precision	86
4.16.6	Block size for distributed computation	86
4.17	Pointedness	86
4.18	The zero cone	87
5	Computation goals and algorithmic variants	88
5.1	Default choices and basic rules	88
5.2	Computation goals	89
5.2.1	Lattice data	89
5.2.2	Support hyperplanes and extreme rays	89
5.2.3	Hilbert basis and lattice points	89
5.2.4	Enumerative data	90
5.2.5	Combined computation goals	90
5.2.6	The class group	90
5.2.7	Integer hull	90
5.2.8	Triangulation and Stanley decomposition	91
5.2.9	Face structure	91
5.2.10	Semiopen polyhedra	92
5.2.11	Automorphism groups	92
5.2.12	Weighted Ehrhart series and integrals	92
5.2.13	Markov and Gröbner bases	92
5.2.14	Local structure	93
5.2.15	Boolean valued computation goals	93
5.3	Integer type	94
5.4	The choice of algorithmic variants	94
5.4.1	Primal vs. dual	94
5.4.2	Lattice points in polytopes	95
5.4.3	Bottom decomposition and order	95
5.4.4	Multiplicity, volume and integrals	96
5.4.5	Symmetrization	97
5.4.6	Subdivision of simplicial cones	97
5.4.7	Options for the grading	97
5.5	Control of computations and communication with interfaces	97
5.6	Rational and integer solutions in the inhomogeneous case	98
6	Running Normaliz	99
6.1	Basic rules	100
6.2	Info about Normaliz	100
6.3	Control of execution	100
6.4	Interruption	101
6.5	Control of output files	101
6.6	Ignoring the options in the input file	102

7	Advanced topics	103
7.1	Computations with a polytope	103
7.1.1	Lattice normalized and Euclidean volume	104
7.1.2	Developer's choice: homogeneous input	104
7.2	Lattice points in polytopes once more	104
7.2.1	Project-and-lift	105
7.2.2	Project-and-lift with floating point arithmetic	107
7.2.3	LLL reduced coordinates and relaxation	107
7.2.4	Positive systems, coarse project-and-lift and patching	108
7.2.5	Polynomial constraints for lattice points	109
7.2.6	The triangulation based primal algorithm	109
7.2.7	Lattice points by approximation	110
7.2.8	Lattice points by the dual algorithm	111
7.2.9	Counting lattice points	111
7.3	The bottom decomposition	111
7.4	Subdivision of large simplicial cones	113
7.5	Primal vs. dual – division of labor	114
7.6	Various volume versions	114
7.6.1	The primal volume algorithm	115
7.6.2	Volume by descent in the face lattice	116
7.6.3	Descent exploiting isomorphism classes of faces	117
7.6.4	Volume by signed decomposition	117
7.6.5	Fixed precision for signed decomposition	120
7.6.6	Comparing the algorithms	120
7.7	Checking the Gorenstein property	121
7.8	Symmetrization	122
7.9	Computations with a polynomial weight	124
7.9.1	A weighted Ehrhart series	125
7.9.2	Virtual multiplicity	127
7.9.3	An integral	127
7.10	Expansion of the Hilbert or weighted Ehrhart series	128
7.10.1	Series expansion	128
7.10.2	Counting lattice points by degree	129
7.10.3	Significant coefficients of the quasipolynomial	130
7.11	Explicit dehomogenization	131
7.12	Projection of cones and polyhedra	132
7.13	Nonpointed cones	133
7.13.1	A nonpointed cone	134
7.13.2	A polyhedron without vertices	135
7.13.3	Checking pointedness first	137
7.13.4	Input of a subspace	137
7.13.5	Data relative to the original monoid	138
7.14	Exporting the triangulation	139
7.14.1	Nested triangulations	141

7.14.2	Disjoint decomposition	141
7.15	Terrific triangulations	142
7.15.1	Just Triangulation	143
7.15.2	All generators triangulation	144
7.15.3	Lattice point triangulation	144
7.15.4	Unimodular triangulation	144
7.15.5	Placing triangulation	145
7.15.6	Pulling triangulation	145
7.16	Exporting the Stanley decomposition	146
7.17	Face lattice, f-vector and incidence matrix	147
7.17.1	Dual face lattice, f-vector and incidence matrix	149
7.18	Module generators over the original monoid	150
7.18.1	An inhomogeneous example	151
7.19	Lattice points in the fundamental parallelepiped	153
7.20	Semiopen polyhedra	155
7.21	Rational lattices	157
7.22	Automorphism groups	159
7.22.1	Euclidean automorphisms	160
7.22.2	Rational automorphisms	163
7.22.3	Integral automorphisms	163
7.22.4	Combinatorial automorphisms	164
7.22.5	Ambient automorphisms	165
7.22.6	Automorphisms from input	166
7.23	Precomputed data	168
7.23.1	Precomputed cones and coordinate transformations	168
7.23.2	An inhomogeneous example	169
7.23.3	Precomputed Hilbert basis of the recession cone	171
7.24	Singular locus	171
7.25	Packed format in the output of binomials	172
8	Algebraic polyhedra	173
8.1	An example	173
8.2	Input	175
8.3	Computations	175
9	Optional output files	177
9.1	The homogeneous case	177
9.2	Modifications in the inhomogeneous case	178
9.3	Algebraic polyhedra	179
9.4	Precomputed data for future input	179
9.5	Overview: Output files forced by computation goals	179

10	Performance	179
10.1	Parallelization	179
10.2	Running large computations	180
11	Distribution and installation	182
11.1	Docker image	182
11.2	Binary release	182
11.3	Source package	183
11.4	Conda	183
11.5	Cloning the GitHub repository	184
12	Building Normaliz yourself	184
12.1	Prerequisites	184
12.1.1	Linux	185
12.1.2	Mac OS X	185
12.2	Normaliz at a stroke	185
12.3	Packages for rational polyhedra	186
12.3.1	CoCoALib	186
12.3.2	nauty	187
12.3.3	Hash library	187
12.3.4	Flint	187
12.4	Packages for algebraic polyhedra	188
12.5	MS Windows	188
13	Copyright and how to cite	188
A	Mathematical background and terminology	190
A.1	Polyhedra, polytopes and cones	190
A.2	Cones	191
A.3	Polyhedra	191
A.4	Affine monoids	193
A.5	Lattice points in polyhedra	194
A.6	Hilbert series and multiplicity	195
A.7	The class group	197
A.8	Affine monoid algebras and their defining ideals	197
A.9	Affine monoid algebras from binomial ideals	198
A.10	Local properties of affine monoid algebras	199
B	Annotated console output	199
B.1	Primal mode	199
B.2	Dual mode	201
C	Normaliz 2 input syntax	203

D	libnormaliz	204
D.1	The master header file	204
D.2	Optional packages and configuration	204
D.3	Integer type as a template parameter	204
D.3.1	Alternative integer types	205
D.3.2	Decimal fractions and floating point numbers	205
D.4	Construction of a cone	205
D.4.1	Construction from an input file	208
D.5	Setting and changing additional data	209
D.5.1	Polynomials	209
D.5.2	Grading	209
D.5.3	Projection coordinates	209
D.5.4	Numerical parameters	209
D.6	Modifying a cone after construction	210
D.7	Computations in a cone	211
D.8	Retrieving results	217
D.8.1	Checking computations	217
D.8.2	Rank, index and dimension	217
D.8.3	Support hyperplanes and constraints	218
D.8.4	Extreme rays and vertices	218
D.8.5	Generators	218
D.8.6	Lattice points in polytopes and elements of degree 1	219
D.8.7	Hilbert basis	219
D.8.8	Module generators over original monoid	219
D.8.9	Generator of the interior	220
D.8.10	Grading and dehomogenization	220
D.8.11	Enumerative data	220
D.8.12	Weighted Ehrhart series and integrals	221
D.8.13	Triangulation and disjoint decomposition	222
D.8.14	Stanley decomposition	223
D.8.15	Scaling of axes	224
D.8.16	Coordinate transformation	224
D.8.17	Coordinate transformations for precomputed data	225
D.8.18	Automorphism groups	225
D.8.19	Class group	227
D.8.20	Face lattice and f-vector	227
D.8.21	Local properties	228
D.8.22	Markov and Grobner bases, representations	228
D.8.23	Integer hull	228
D.8.24	Projection of the cone	228
D.8.25	Excluded faces	229
D.8.26	Boolean valued results	229
D.8.27	Results by type	230
D.9	Algebraic polyhedra	230

D.10	Reusing previous computation results	231
D.11	Control of execution	232
D.11.1	Exceptions	232
D.11.2	Interruption	232
D.11.3	Inner parallelization	233
D.11.4	Outer parallelization	233
D.11.5	Control of terminal output	233
D.11.6	Printing the cone	234
D.12	A simple program	234
E	Normaliz interactive: PyNormaliz	239
E.1	Installation	239
E.2	The high level interface by examples	239
E.2.1	Creating a cone	240
E.2.2	Matrices, vectors and numbers	241
E.2.3	Triangulations, automorphisms and face lattice	242
E.2.4	Hilbert and other series	245
E.2.5	Multiplicity, volume and integral	247
E.2.6	Integer hull and other cones as values	247
E.2.7	Boolean values	248
E.2.8	Algebraic polyhedra	249
E.2.9	The collective compute command and algorithmic variants	250
E.2.10	Miscellaneous functions	250
E.3	The low level interface	252
E.3.1	The main functions	252
E.3.2	Additional input and modification of existing cones	253
E.3.3	Additional data access	253
E.3.4	Miscellaneous functions	254
E.3.5	Raw formats of numbers	254
F	Distributed computation for volume via signed decomposition	255
	References	257
	Index of keywords	259

3 Affine monoids and binomial ideals by examples

The role of binomials in the computation of affine monoids and their algebras is briefly explained in Sections A.8 and A.9. We assume that the user is familiar with them.

3.0.1 Input and default computation goals

Affine monoids are given to Normaliz by the input type

monoid

as in `monoid.in`:

```
amb_space 3
monoid 6
1 0 0
2 3 5
0 0 1
1 1 2
0 1 3
3 1 0
/* grading 1 -2 1*/
/*HilbertSeries*/
/*GroebnerBasis*/
/*Lex*/
/*MarkovBasis*/
/*gb_degree_bound 11*/
/*gb_min_degree 9*/
/*Multiplicity*/
/*SingularLocus*/
/*CodimSingularLocus*/
/*IsSerreR1*/
```

Positivity of the monoid does *not* mean that all components of the input vectors are nonnegative. It only means that $x = 0$ if both x and $-x$ belong to it.

Let us translate this example into multiplicative notation. We have binomials in $K[X_1, X_2, X_3]$, namely

$$M_1 = X_1, \quad M_2 = X_1^2 X_2^3 X_3^5, \dots, \quad M_6 = X_1^3 X_2.$$

In the output file we see

```
...
original monoid is not integrally closed in chosen lattice
...
6 Hilbert basis elements:
0 0 1
1 0 0
0 1 3
```

```
1 1 2
3 1 0
2 3 5
```

5 support hyperplanes:

```
0 0 1
0 1 0
1 0 0
2 -3 1
5 -15 7
```

The support hyperplanes are those of the cone generated by the monoid. They are used in auxiliary computations, for example in finding the Hilbert basis, i.e., the unique minimal system of generators of our monoid. In this case the input vectors are all in the Hilbert basis, but this need not be the case. The Hilbert basis is ordered by degree and lexicographically within each degree. In fact, we have a grading

```
grading:
1 1 1
```

For the default choice of the grading we start from the standard grading on the ambient lattice. Then the grading, whether the default choice or an explicit grading in the input, is divided by the greatest common divisor of the degrees of the generators. In the context of monomial algebras it is the most natural choice. The division by the gcd can be suppressed by `NoGradingDenom`. In our example the gcd is 1.

Our monoid actually has another grading, in which all generators have degree 1: grading 1 -2 1 in the input file. Activate it and study the changes.

Note: The input type `monoid` is close to `cone_and_lattice` if the monoid is normal. But there are two differences in the default choices: (1) The default computation goals and (2) the default grading. In fact, for `monoid` it is derived from the standard grading on the ambient lattice, whereas for `cone_and_lattice` it gives degree 1 to the extreme integral generators, provided this is possible.

3.0.2 Markov and Gröbner bases, Representations

The purpose of the computations in this section is to understand the defining ideal of the subalgebra A of $K[X_1, X_2, X_3]$ generated by our binomials M_1, \dots, M_6 introduced above. To this end we activate

MarkovBasis

in `monoid.in`, the Markov basis is computed and returned in the file with suffix

mrk file containing the Markov basis

In our case `monoid.mrk`:

```
7
```

```

6
1 0 -1 -1 1 0
-2 0 2 -1 0 1
-1 0 1 -2 1 1
2 1 0 -1 -1 -1
0 0 0 -3 2 1
1 1 1 -3 0 0
0 1 2 -2 -1 0

```

Each column corresponds to an input vector, and the rows are indeed relations: the scalar product of a row listed in the Markov basis and a column of the matrix monoid is 0. The binomials in $P = K[Y_1, \dots, Y_6]$ corresponding to the rows in the Markov basis form a system of generators of the binomial ideal defining our monoid algebra as a residue class ring of P . The binomials are

$$b_1 = Y_1 Y_5 - Y_3 Y_4, \quad b_2 = Y_3^2 Y_6 - Y_1^2 Y_3, \dots, \quad b_7 = Y_2 Y_3^2 - Y_4^2 Y_5.$$

and indeed the binomials vanish if we substitute M_i for Y_i , $i = 1, \dots, 7$. That they generate the defining ideal is claimed by Normaliz.

For easier reference the input matrix is mirrored in the file with suffix

ogn file with the original generators

in our case monoid.ogn:

```

6
3
1 0 0
2 3 5
0 0 1
1 1 2
0 1 3
3 1 0

```

In order to compute a Gröbner basis of our binomial ideal, we activate

GroebnerBasis

and get the output file with suffix

grb containing the Gröbner basis.

For the Gröbner basis one has to choose a monomial order. The default choice is ‘degree reverse lexicographic’ In our case it yields

```

8
6
-1 0 1 1 -1 0
0 0 0 3 -2 -1
1 0 -1 2 -1 -1
2 0 -2 1 0 -1
0 1 2 -2 -1 0

```

```

1 1 1 0 -2 -1
2 1 0 -1 -1 -1
3 0 -3 0 1 -1

```

More precisely: the indeterminates in the polynomial ring housing the binomials are ordered $Y_1 > \dots Y_6$ and we take the degree reverse lexicographic extension, where ‘degree’ means the total standard degree on the polynomial ring P . The file with suffix `ogn` is also created for the Gröbner basis. There is no output of the (minimal) Markov basis, unless you ask for it explicitly.

Despite of being the default choice, the degree reverse lexicographic order is in the list of pertaining computation goals:

RevLex degree reverse lexicographic order

Lex lexicographic order

DegLex degree lexicographic order

Activate also Lex in our example and see what changes. DegLex is taken with respect to the total standard degree as well, and makes no difference in our case, since the generating binomials are homogeneous in this grading.

A grading of the monomial algebra induces a grading on the binomials in its defining ideal such that the latter are homogeneous polynomials. With respect to this grading the output of Markov and Gröbner bases can be restricted:

gb_degree_bound `<n>` sets upper degree bound `<n>` for binomials,

gb_min_degree `<n>` sets lower degree bound `<n>` for binomials.

There is one more computation goal for monoids that complements HilbertBasis (switched on by default):

Representations representations of reducible elements in monoid in terms of the Hilbert basis

The output is a list of binomials in the file with suffix

rep representations of reducible elements in terms of the Hilbert basis.

Also the file with suffix `ogn` is written.

As a simple example we consider `representations.in`

```

amb_space 3
monoid 8
1 0 0
2 3 5
0 1 1
0 0 1
1 1 2
0 1 3
3 1 0
1 2 5
Representations

```

```
/*BinomialsPacked*/
```

with

```
4 Hilbert basis elements:
0 0 1
1 0 0
0 1 1
3 1 0
```

and representations of the other 4 elements in the input:

```
4
8
-1 0 -1 -1 1 0 0 0
0 0 -1 -2 0 1 0 0
-1 0 -2 -3 0 0 0 1
-2 1 -3 -2 0 0 0 0
```

The entries 1 in each row mark the reducible elements and the row should be read as a binomial vanishing on the input vectors (or monomials).

If you want to see computations that take longer than our toy example so far, run `A443monoid.in` and `Kwak80.in`.

3.0.3 Hilbert series and multiplicity

If we activate both (!) `HilbertSeries` and `Multiplicity` in `monoid.in`, the result is

```
multiplicity = 19/40
multiplicity (float) = 0.475

Hilbert series:
1 1 0 0 3 2 -2 -1 6 2 -4 0 6 1 -3 1 4 0 0 1 1
denominator with 3 factors:
1:1 2:1 20:1
...
```

followed by the representation with cyclotomic denominator and the Hilbert quasipolynomial. Activate grading `1 -2 1` and observe the changes.

3.0.4 Binomial ideals from cone input

Defining binomial ideals can be computed not only for monoids defined by the input type `monoid`, but also for the monoids that defined by other input types as intersections of cones and lattices, for example `cone`, `cone_and_lattice`, `equations`, `inequalities` etc. In the case of generator input there are actually two monoids, the “original monoid” as discussed in Section 7.18, and its integral closure in the lattice defined by `tghe` input. So, if we ask for the Markov

basis of the defining ideal, which monoid is taken? Answer: always the integral closure generated by its Hilbert basis, unless the property makes only sense for the original monoid: if we ask `IsIntegrallyClosed`, the answer is always ‘yes’ for the integral closure.

As an example we take `cone_latt_markov.in` (`monoid.in` with a different input type):

```
amb_space 3
cone_and_lattice 6
1 0 0
2 3 5
0 0 1
1 1 2
0 1 3
3 1 0
MarkovBasis
SingularLocus
```

The output file contains

```
original monoid is not integrally closed in chosen lattice
...
codim singular locus = 2
18 Markov basis elements

*****

9 lattice points in polytope (Hilbert basis elements of degree 1):
0 0 1
0 1 3
1 0 0
1 1 2
1 2 4
2 1 1
2 2 3
2 3 5
3 1 0

0 further Hilbert basis elements of higher degree:
```

The Markov basis is contained in `cone_latt_markov.mrk`:

```
18
9
-1 0 1 0 0 1 0 0 -1
0 0 0 -1 0 2 0 0 -1
...
0 -1 0 0 2 0 0 -1 0
```

The columns correspond to the Hilbert basis elements in the order they are listed above. Fpr

completeness the file with suffix `ogn` is written also in this case. It contains the Hilbert basis as listed in the out file.

The singular locus has codimension 2, the minimum for a normal monoid (algebra). The singular locus is stored in `cone_latt_markov.sng`.

We could equally well start from the inequalities defining the integral closure (the generators above generate the lattice \mathbb{R}^3) in `cone_latt_markov_supp.in`,

```
amb_space 3
inequalities 5
0 0 1
0 1 0
1 0 0
2 -3 1
5 -15 7
MarkovBasis
SingularLocus
```

with the same result as above, except that there is no originl monoid.

3.1 Monoids from binomials

As an example, we consider the binomial ideal generated by

$$Y_1^2 Y_2 - Y_4 Y_5 Y_6, \quad Y_1 Y_4^2 - Y_3 Y_5 Y_6, \quad Y_1 Y_2 Y_3 - Y_5^2 Y_6.$$

in the polynomial ring $P = K[Y_1, \dots, Y_6]$. We want to find an embedding of the toric ring it defines. When we say “defines”, then we do not claim that the residue ring P/I is a toric ring. But there is a unique smallest binomial ideal $J \supset I$ with this property, and `Normaliz` finds the monoid and, if wanted, also a Markov (or Gröbner) basis of J . A priori $R = P/J$ is only defined as a residue class ring. It doesn’t have a ‘canonical’ embedding into another polynomial ring, but `Normaliz` computes such an embedding if the monoid underlying R/J is positive. As pointed out already, non-positive affine monoids can only be computed by `Normaliz` if they are normal.

3.1.1 Affine monoids from binomial ideals

The input type that asks for a toric ring from binomial input is

`toric_ideal`

The input vectors are obtained as the differences of the two exponent vectors in the binomials. So the input ideal `toric_ideal.in` for our 3 binomials is

```
amb_space 6
toric_ideal 3
2 1 0 -1 -1 -1
```



```

1 0 -1 2 -1 -1
1 1 1 0 -2 -1
/* total_ddegree */

```

In order to avoid special input rules for this case in which our object is not defined as a subset of an ambient space, but as a quotient of type *generators/relations*, we abuse the name `amb_space`: it determines the space in which the input vectors live.

It is possible to define a grading. It must give positive degree to the unit vectors of the ambient space and degree 0 to the vectors representing the binomials so that the latter become homogeneous polynomials with respect to this grading.

In the output we get

```

6 original generators:
1 0 0
2 3 5
0 0 1
1 1 2
0 1 3
3 1 0

```

namely the residue classes of the indeterminates realized in an embedding. Test the binomials on the original generators! We know this monoid already from `monid.in`, and you can try the other computationn goals discussed for the latter.

We see

```

grading:
1 1 1

```

So `Normaliz` uses the standard grading on the ambient polynomoal ring into which R/J has been embedded. This is the default choice, as it is for the input type `monolid`. Our toric ring actually has its own standard tgrading: activate `total_degree` in the input file and look at the output. In fact, the binomials above are homogeneous in the standard garing on P , and `total_degree` sets this grading.

The generators are repeated (in this case) in a different order,m as wem know already:

```

6 Hilbert basis elements:
0 0 1
1 0 0
0 1 3
1 1 2
3 1 0
2 3 5

```

Now they are sorted by degree and then lexicographically, as we always sort Hilbert bases.

As a trivial example in which the Hilbert basis does not simply repeat the original generators in a different order, compute `lin_bin.in`:

```
amb_space 2
toric_ideal 1
1 -1
```

The output contains

```
2 original generators:
1
1

1 Hilbert basis elements:
1
```

3.1.2 Normalization of monoids from binomials

One go can a step further, using the input type

normal_toric_ideal

It asks for the *normalization* of the toric ring defined by the binomials. In `normal_toric_ideal.in` we take the same binomials as above:

```
amb_space 6
normal_toric_ideal 3
2 1 0 -1 -1 -1
1 0 -1 2 -1 -1
1 1 1 0 -2 -1
```

In the output file we find

```
6 original generators:
1 0 0
2 3 5
0 0 1
1 1 2
0 1 3
3 1 0

2 lattice points in polytope (Hilbert basis elements of degree 1):
0 0 1
1 0 0

7 further Hilbert basis elements of higher degree:
0 1 3
1 1 2
2 1 1
3 1 0
1 2 4
```

```
2 2 3
2 3 5
```

The “original generators” are the same as above, as they should be. Also the default grading is the same, and the default computation goals are identical as well. But the Hilbert series, Markov basis, Gröbner basis etc. are computed for the normalization, as the user can see by playing with the commented out computation goals.

Note: Until version 3.9.4 the input type `normal_toric_ideal` was called `lattice_ideal`, which has a different meaning now and is discussed in the next subsection.

3.2 Lattice ideals

A lattice ideal I in a polynomial ring P is a binomial ideal modulo which all monomials are nonzerodivisors. This implies that P/J is a monoid ring whose underlying monoid is the natural image of the monoid of monomials in P . Moreover, it is a cancellative monoid, but not necessarily affine—the latter property requires torsion freeness additionally. The input type is

`lattice_ideal`

`Normaliz` tests whether the lattice ideal is toric and indicates it in the input file, but does not automatically treat the input like `toric_ideal` in the positive case.

A simple example of a non-toric lattice ideal is `non_toric.in`

```
amb_space 4
lattice_ideal 4
2 -2 0 0
1 1 -1 -1
2 -1 1 -2
-1 -1 1 -1
/*GroebnerBasis
DegLex*/
```

The default computation goal is `MarkovBasis`. In our case the result is

```
4
4
-4 -1 1 0
-3 0 0 1
-2 2 0 0
6 0 0 0
```

Attention: the last binomial is $x_1^6 - 1$ so that the residue class of x_1 is a torsion element in the monoid of residue classes.

For internal reasons and the exchange of data with external programs we can ask

```
IsLatticeIdealToric
```

There is only one more allowed computation goal, namely `GroebnerfBasis`. There are no monoid generators for `lattice_ideal`.

Note: In versions until 3.9.4 `lattice_ideal` had the meaning of `normal_toric_ideal`.

References

- [1] 4ti2 team. 4ti2-A software package for algebraic, geometric and combinatorial problems on linear spaces. Available at <https://github.com/4ti2/4ti2>.
- [2] J. Abbott, A. M. Bigatti and G. Lagorio, *CoCoA-5: a system for doing Computations in Commutative Algebra*. Available at <http://cocoa.dima.unige.it>.
- [3] V. Almendra and B. Ichim, *jNormaliz 1.7*. Available at <https://normaliz.uos.de>.
- [4] V. Baldoni, N. Berline, J. A. De Loera, B. Dutra, M. Köppe, S. Moreinis, G. Pinto, M. Vergne and J. Wu, *A User's Guide for LattE integrale v1.7.2, 2013*. Software package LattE is available at <https://www.math.ucdavis.edu/~latte/>.
- [5] D. Bremner, M. D. Sikirić, D. V. Pasechnik, Th. Rehn and A. Schürmann, *Computing symmetry groups of polyhedra*. LMS J. Comp. Math. 17 (2014), 565–581.
- [6] St. Brumme, *Hash library*. Package available at <https://create.stephan-brumme.com/>.
- [7] W. Bruns, *Automorphism groups and normal forms in Normaliz*. Res. Math. Sci. 9 (2022), no. 2, Paper No. 20, 15 pp.
- [8] W. Bruns, *Polytope volume in Normaliz*. São Paulo J. Math. Sci. <https://doi.org/10.1007/s40863-022-00317-9>
- [9] W. Bruns, P. Garcia-Sanchez, C. O'Neill and D. Wilburne, *Wilf's conjecture in fixed multiplicity*. Int. J. Algebra Comp. 30 (2020), 861–882.
- [10] W. Bruns and J. Gubeladze, *Polytopes, rings, and K-theory*. Springer, 2009.
- [11] 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.
- [12] W. Bruns and B. Ichim, *Normaliz: algorithms for rational cones and affine monoids*. J. Algebra 324 (2010) 1098–1113.
- [13] W. Bruns and B. Ichim, *Polytope volume by descent in the face lattice and applications in social choice*. Math. Prog. Comp. 113 (2020), 415–442.
- [14] W. Bruns, B. Ichim and C. Söger, *The power of pyramid decomposition in Normaliz*. J. Symb. Comp. 74 (2016), 513–536.
- [15] W. Bruns, B. Ichim and C. Söger, *Computations of volumes and Ehrhart series in four candidates elections*. Ann. Oper. Res. 280 (2019), 241–265.
- [16] W. Bruns and R. Koch, *Computing the integral closure of an affine semigroup*. Univ. Iagell. Acta Math. 39 (2001), 59–70.
- [17] W. Bruns, R. Sieg and C. Söger, *Normaliz 2013–2016*. In G. Böckle, W. Decker and G. Malle, editors, *Algorithmic and Experimental Methods in Algebra, Geometry, and Number Theory*, pages 123–146. Springer, 2018.
- [18] W. Bruns and C. Söger, *The computation of weighted Ehrhart series in Normaliz*. J. Symb. Comp. 68 (2015), 75–86.
- [19] B. Büeler and A. Enge, *Vinci*. Package available from <https://www.math.u-bordeaux.fr/~aenge/>
- [20] B. Büeler, A. Enge, K. Fukuda, *Exact volume computation for polytopes: a practical study*. In: *Polytopes - combinatorics and computation* (Oberwolfach, 1997), pp. 131 – 154, DMV Sem. 29,

Birkhäuser, Basel, 2000.

- [21] J. A. De Loera, R. Hemmecke and M. Köppe. Algebraic and geometric ideas in the theory of discrete optimization. MOS-SIAM Series on Optimization, 14. Society for Industrial and Applied Mathematics (SIAM), Philadelphia 2013.
- [22] V. Delecroix, *embedded algebraic number fields (on top of antic)*, package available at <https://github.com/flatsurf/e-antic>.
- [23] P. Filliman, *The volume of duals and sections of polytopes*. Mathematika 37 (1992), 67–80.
- [24] S. Gutsche, M. Horn and C. Söger, *NormalizInterface for GAP*. Available at <https://github.com/gap-packages/NormalizInterface>.
- [25] S. Gutsche and R. Sieg, *PyNormaliz - an interface to Normaliz from python*. Available at <https://github.com/Normaliz/PyNormaliz>.
- [26] W. B. Hart, *Algebraic Number Theory In C*. Package available at <https://github.com/wbhart/antic>.
- [27] W. B. Hart, F. Johansson and S. Pancratz, *FLINT: Fast Library for Number Theory*. Available at <https://flintlib.org>.
- [28] Hemmecke and P. N. Malkin. Computing generating sets of lattice ideals and Markov bases of lattices. J. Symb. Comp. 44, 1463–1476 (2009).
- [29] F. Johansson, *Arb - a C library for arbitrary-precision ball arithmetic*. Available at <https://arblib.org/>.
- [30] J. Lawrence, *Polytope volume computation*. Math. Comp. 57 (1991), 259–271.
- [31] M. Köppe and S. Verdoolaege, *Computing parametric rational generating functions with a primal Barvinok algorithm*. Electron. J. Comb. 15, No. 1, Research Paper R16, 19 p. (2008).
- [32] B. D. McKay and A. Piperno, *Practical graph isomorphism, II*. J. Symbolic Comput. 60 (2014), 94–112.
- [33] L. Pottier, *The Euclidean algorithm in dimension n*. Research report, ISSAC 96, ACM Press 1996.
- [34] A. Schürmann, *Exploiting polyhedral symmetries in social choice*. Social Choice and Welfare 40 (2013), 1097–1110.
- [35] B. Sturmfels, *Gröbner bases and convex polytopes*. American Mathematical Society 1996.

Index of keywords

-h, 90
-n, 90
-p, 90
-x=<T>, 100
--help, -?, 100
--ignore, -i, 88
--version, 100
<project>.basic.data, 254
<project>.hollow_tri.<n>.gz, 254
--OutputDir=<outdir>, 101
--<suffix>, 101
--all-files, -a, 101
--files, -f, 101
--ignore, -i, 102
--verbose, -c, 100

AffineDim, 98
AllGeneratorsTriangulation, 91, 144
AmbientAutomorphisms, 92, 165
Approximate, -r, 95, 110
aut, 179
Automorphisms, 92

BigInt, -B, 94
BinomialsPacked, 102, 172
block_size_hollow_tri <n>, 86, 254
BottomDecomposition, -b, 95

ClassGroup, -C, 90
CodimSingularLocus, 93, 172
CombinatorialAutomorphisms, 92
cone, 77
cone_and_lattice, 77, 78
ConeDecomposition, -D, 91
Congruences, 98
congruences, 79
constraints <n>, 81
constraints <n> symbolic, 82
convert_equations, 82
CoveringFace, 98
cst, 177

dec, 179
decimal_digits <n>, 86, 120
DefaultMode, 88
Deg1Elements, -1, 89
DegLex, 62, 93
Dehomogenization, 98
dehomogenization, 84
Descent ExploitIsosMult, 96, 117
Descent, -F, 96, 116
DistributedComp, 96, 254
DualFaceLattice, 92
DualFVector, 92
DualIncidence, 92
DualMode Deg1Elements, -d1, 99
DualMode HilbertBasis, -dN, 98
DualMode LatticePoints, 99
DualMode ModuleGenerators, 99
DualMode, -d, 95

egn, esp, 178
EhrhartQuasiPolynomial, 98
EhrhartSeries, 90
EmbeddingDim, 97
Equations, 98
equations, 78
EuclideanAutomorphisms, 92
EuclideanVolume, 98
excluded_faces, 79
ExcludedFaces, 98
expansion_degree <n>, 86
ext, 177
ExternalIndex, 98
extreme_rays, 77
ExtremeRays, 97, 99
ExtremeRaysFloat, 89

fac, 179
face_codim_bound <n>, 86
FaceLattice, 91
FixedPrecision, 96, 120
FVector, 91

gb_degree_bound <n>, 62, 86
gb_min_degree <n>, 62, 86
gen, 177
generated_lattice, 78
GeneratorOfInterior, 98
Grading, 98
grading, 84
GradingDenom, 98
GradingIsPositive, 97
grb, 61, 179
GroebnerBasis, 61, 92

hilbert_basis_rec_cone, 78
HilbertBasis, -N, 89
HilbertQuasiPolynomial, 98
HilbertSeries, -q, 90
hom_constraints, 82
HSOP, 90
ht1, 177

IdsSerreR1, 93
inc, 179
Incidence, 91
InclusionExclusionData, 98
inequalities, 78
inhom_congruences, 81
inhom_equations, 80
inhom_excluded_faces, 80
inhom_inequalities, 80
InputAutomorphisms, 92, 166
IntegerHull, -H, 90
Integral, -I, 92
InternalIndex, 98
inthull.out, 179
inv, 177
IsDeg1ExtremeRays, 93
IsDeg1HilbertBasis, 93
IsEmptySemiopen, 92
IsGorenstein, -G, 93
IsInhomogeneous, 98
IsIntegrallyClosed, 93
IsPointed, 93
IsReesPrimary, 93
IsSerreR1, 172

IsTriangulationNested, 98
IsTriangulationPartial, 98

KeepOrder, -k, 95

lat, 178
lattice, 78
lattice_ideal, 67, 83
LatticePoints, 89
LatticePointTriangulation, 91, 144
Lex, 61, 93
LongLong, 94

MarkovBasis, 60, 92
maximal_subspace, 77
MaximalSubspace, 97
mod, 178
ModuleGenerators, 97
ModuleGeneratorsOverOriginalMonoid, -M, 89

ModuleRank, 98
monoid, 59, 78
mrk, 60, 179
msp, 178
Multiplicity, -v, 90

NoBottomDec, -o, 95
NoCoarseProjection, 95, 108
NoDescent, 96, 116
NoExtRaysOutput, 52, 101
NoGradingDenom, 97
NoHilbertBasisOutput, 102
NoLLL, 95, 107
NoMatricesOutput, 102
nonnegative, 79
NoPatching, 95, 109
NoPeriodBound, 90
NoProjection, 95
NoRelax, 95, 108
normal_toric_ideal, 66, 83
NoSignedDec, 96, 119
NoSubdivision, 97
NoSuppHypsOutput, 52, 102
NoSymmetrization, -X, 97
nr_coeff_quasipol <n>, 86

NumberLatticePoints, 90, 111
offset, 80
ogn, 61, 179
open_facets, 85
OriginalMonoidGenerators, 97
PlacingTriangulation, 91, 145
polynomial_equations <n>, 82
polynomial_inequalities <n>, 82
polytope, 77
PrimalMode, -P, 95
proj.out, 179
ProjectCone, 89, 132
Projection, -j, 95, 106
projection_coordinates, 85
ProjectionFloat, -J, 95, 107
PullingTriangulation, 91, 145
Rank, 98
rational_lattice, 78
rational_offset, 80
RationalAutomorphisms, 92
RecessionRank, 98
rees_algebra, 77
ReesPrimaryMultiplicity, 98
rep, 62, 179
Representations, 62, 93
RevLex, 61, 93
saturation, 78
SignedDec, 96, 118
signs, 78
SingularLocus, 93, 171
sng, 171, 179
StanleyDec, -y, 91
strict_inequalities, 80
strict_signs, 80
StrictIsoTypes, 117
StrictTypeChecking, 96
Sublattice, -S, 89
subspace, 77
SuppHypsFloat, 52, 89
support_hyperplanes, 79
SupportHyperplanes, -s, 89, 99
symm.out, 179
Symmetrize, -Y, 97
tgn, 139, 179
toric_ideal, 65, 83
total_degree, 84
tri, 139, 179
Triangulation, -T, 91
TriangulationDetSum, 91
TriangulationSize, -t, 91
UnimodularTriangulation, 91, 144
unit_matrix, 83
unit_vector <n>, 83
UnitGroupIndex, 98
vertices, 79
VerticesFloat, 52, 89
VerticesOfPolyhedron, 97, 99
VirtualMultiplicity, -L, 92
Volume, -V, 53, 90
WeightedEhrhartQuasiPolynomial, 98
WeightedEhrhartSeries, -E, 92
WitnessNotIntegrallyClosed, -w, 89
WritePreComp, 179