NmzIntegrate 1.3.3

January 13, 2017

Winfried Bruns and Christof Söger

mailto:normaliz@uos.de

https://normaliz.uos.de

Contents

1 The objectives of NmzIntegrate

We assume in the following that the reader is familiar with Normaliz, in particular with its treatment of Ehrhart series and quasipolynomials. NmzIntegrate 1.3.3requires Normaliz 3.2.0.

Normaliz computes certain data for a monoid

$$M = C \cap L$$

where $C \subset \mathbb{R}^n$ is a rational, polyhedral and pointed cone, and $L \subset \mathbb{Z}^n$ is a sublattice. These data are defined by the input to Normaliz. NmzIntegrate requires that M has been endowed with a grading deg (see the manual of Normaliz 3.0).

For such graded monoids Normaliz can compute the volume of the rational polytope

$$P = \{x \in \mathbb{R}_+M : \deg x = 1\},$$

the Ehrhart series of P, and the quasipolynomial representing the Ehrhart function. (Here $\mathbb{R}_+M$ is the cone generated by the elements of M; it may be smaller than C if L has rank $< n$.)

These computations can be understood as integrals of the constant polynomial $f = 1$, namely with respect to the counting measure defined by L for the Ehrhart function, and with respect to the (suitably normed) Lebesgue measure for the volume. NmzIntegrate generalizes these computations to arbitrary polynomials f in n variables with rational coefficients. (Mathematically, there is no need to restrict oneself to rational coefficients for f.)

More precisely, set

$$E(f,k) = \sum_{x \in M, \deg x = k} f(x),$$

and call $E(f,\_)$ the *generalized Ehrhart function* for f. (With $f = 1$ we simply count lattice points.) The *generalized Ehrhart series* is the ordinary generating function

$$E_f(t) = \sum_{k=0}^{\infty} E(f,k)t^k.$$

It turns out that $E_f(t)$ is the power series expansion of a rational function at the origin, and can always be written in the form

$$E_f(t) = \frac{Q(t)}{(1-t^\ell)^{\text{totdeg}\, f + \text{rank}\, M}}, \qquad Q(t) \in \mathbb{Q}[t],\ \deg Q < \text{totdeg}\, f + \text{rank}\, M.$$

Here totdeg f is the total degree of the polynomial f, and ℓ is the least common multiple of the degrees of the extreme integral generators of M. See [2] for an elementary account and the algorithm used by NmzIntegrate.

NmzIntegrate 1.3.3, like Normaliz 3.2.0can compute Ehrhart series for *semiopen* cones. For them the monoid M is replaced by the set

$$M' = C' \cap L$$

where $C' = C \setminus \mathscr{F}$ and $\mathscr{F}$ is the union of a set of faces (not necessarily facets) of C. What has been said above about the structure of the generalized Ehrhart series remains true. We discuss an example in Section 5.

It follows from the general theory of rational generating functions that there exists a quasipolynomial $q(k)$ with rational coefficients and of degree $\leq \operatorname{totdeg} f + \operatorname{rank} M - 1$ that evaluates to $E(f,k)$ for all $k \geq 0$. A quasipolynomial is a "polynomial" with periodic coefficients: there exist a *period* $\pi \in \mathbb{N}$ and true polynomials $q^j \in \mathbb{Q}[X]$, $j = 0, \ldots, \pi - 1$, such that

$$q(k) = q^{(j)}(k) \qquad \text{if} \quad k \equiv j \quad (\pi).$$

Each of the polynomials $q^{(j)}$ is given as

$$q^{(j)}(k) = q_0^{(j)} + q_1^{(j)} X + \cdots + q_{\operatorname{totdeg} f + \operatorname{rank} M - 1}^{(j)} X^{\operatorname{totdeg} f + \operatorname{rank} M - 1}$$

with constant coefficients in $\mathbb{Q}$. The period π divides ℓ.

Let $m = \operatorname{totdeg} f$ and f_m be the degree m homogeneous component of f. By letting k go to infinity and approximating f_m by a step function that is constant on the meshes of $\frac{1}{k}L$ (with respect to a fixed basis), one sees

$$q_{\operatorname{totdeg} f + \operatorname{rank} M - 1}^{(j)} = \int_P f_m \, d\lambda$$

where $d\lambda$ is the Lebesgue measure that takes value 1 on a basic mesh of $L \cap \mathbb{R}M$ in the hyperplane of degree 1 elements in $\mathbb{R}M$. In particular, the *virtual leading coefficient* $q_{\operatorname{totdeg} f + \operatorname{rank} M - 1}^{(j)}$ is constant and depends only on f_m. If the integral vanishes, the quasipolynomial q has smaller degree, and the true leading coefficient need not be constant. Following the terminology of commutative algebra and algebraic geometry, we call

$$(\operatorname{totdeg} f + \operatorname{rank} M - 1)! \cdot q_{\operatorname{totdeg} f + \operatorname{rank} M - 1}$$

the *virtual multiplicity* of M and f. It is an integer if f_m has integral coefficients and P is a lattice polytope.

Since a semiopen cone C' differs from its closure in a set of measure 0, the passage to C' does not change the Lebesgue integrals just mentioned, and is therefore irrelevant for their computation.

NmzIntegrate computes

(ES) the generalized Ehrhart series and its quasipolynomial,
(Int) the Lebesgue integral of f over P, or
(LC) the virtual leading coefficient and the virtual multiplicity.

The user controls the type of computation by a command line option. (ES) contains (LC), and (LC) is just the evaluation of (Int) on the highest homogeneous component of f. It is presently not possible to compute the Ehrhart series and the integral together if f is not homogeneous, but the two computations can be combined in one run of NmzIntegrate.

Acknowledgement. We gratefully acknowledge the support we received from John Abbott and Anna Bigatti in using CoCoALib, on which the multivariate polynomial algebra in NmzIntegrate is based.

The development of Normaliz is currently supported by the DFG SPP 1489 "Algorithmische und experimentelle Methoden in Algebra, Geometrie und Zahlentheorie".

2 Major changes in this version

None.

3 Input files

NmzIntegrate can be used in two ways:

1. by direct call from the command line,
2. by call from within Normaliz with the appropriate options.

If NmzIntegrate misses an input file that should have been produced by Normaliz, it calls Normaliz and makes it produce the missing file(s). Normaliz is also called if a file produced by it is older than the Normaliz input file, provided the latter is accessible.

For mutual calls it is necessary that the executables of Normaliz and NmzIntegrate reside in the same directory.

Note: NmzIntegrate can only use the *homogeneous* input types of Normaliz (including `excluded_faces`).

3.1 Basic input files

The files `<project>.in` and `<polynomial>.pnm` must be provided by the user ()or are made by Normaliz if it calls NmzIntegtate on after symmetrization). Normaliz needs `<project>.in` in order to produce the files read by NmzIntegrate. The file `<polynomial>.pnm` contains the polynomial to be integrated.

Unless the user defines `<polynomial>` explicitly (see below), NmzIntegrate sets `<polymomial>=` `<project>`. The explicit choice of the `<polynomial>` is only possible if NmzIntegrate is called

from the command line.

3.2 Auxiliary files produced by Normaliz

One runs Normaliz with the option

`-T` (or `-y`) for (Int) and (LC),
`-y` for (ES).

(It is allowed to combine `-T` and `-y`.) If NmzIntegrate calls Normaliz, then it chooses these options automatically.

This will produce the files with the following suffixes (in addition to `<project>.out` and possibly further output files determined by the Normaliz options `-f` and `-a`):

`-T` `inv`, `tgn`, `tri`
`-y` `inv`, `tgn`, `dec`.

NmzIntegrate reads

- the grading and the rank from `<project>.inv`,
- the rays of the triangulation from `<project>.tgn`,
- the triangulation from `<project>.tri` (for (Int) and (LC)) and
- the Stanley decomposition from `<project>.dec` (for (ES)).

If `<project>.tri` does not exist for one of the tasks (Int) or (LC), NmzIntegrate checks for the existence of `<project>.dec` and reads the triangulation from it.

NmzIntegrate itself does not read `<project>.in` nor any other output file of Normaliz than those just mentioned.

3.3 The polynomial

The polynomial is read from the file `<polynomial>.pnm`. The polynomial can be defined by a usual polynomial expression using rational coefficients, addition, subtraction, multiplication and exponentiation, following the standard precedence rules for the evaluation of such expressions.

Note:

1. The names of the variables are fixed: `x[1]`,...,`x[<n>]` where `<n>` represents the number n.
2. An explicit multiplication sign $*$ is necessary for all multiplications, in particular between a coefficient and an indeterminate or between indeterminates.

Examples:

```
1/120*(x[1]+x[2]^2)*(-2*x[3]*x[4])^2+x[3]
```

is a well formed input polynomial, but

```
1/120(x[1]+x[2]^2)*(-2x[3]*x[4])^2+x[3]
```

is not allowed.

NmzIntegrate is now using the CoCoALib input function for polynomials. In the previous version some multiplication signs that are now necessary had to be omitted.

4 Running NmzIntegrate

There are five ways to run NmzIntegrate:

1. direct call from the command line,
2. call from Normaliz, (i) via one of the options `-E,-I,-L` or (ii) implicitly via the option `--Symmetrize, -Y` (see Normaliz manual),
3. from jNormaliz via Normaliz (with the same two variants as in 2).

Note that the implicit call from Normaliz via `--Symmetrize` deletes the output file of NmzIntegrate, after the results have been transferred to Normaliz. The other three ways preserve the output file (and also the files used for the transfer of data from Normaliz to NmzIntegrate).

The shortest possible command to start NmzIntegrate is

```
nmzIntegrate <project>
```

This will run the default computation (ES) on the `<project>`. The full input syntax is

```
nmzIntegrate [-cEIL] [-x=<T>] [--help] [--version]
                          [-F=<polynomial>] [--OutputDir=<outdir>] <project>
```

where `-c` and `-x=<T>` have the same meaning as for Normaliz:

`-c` activates the verbose mode in which control information is written to the terminal,

`-x=<T>` limits the number of parallel threads to `<T>`.

The following options control the type of computation:

`-E` activates the computation (ES) (the default mode, can be omitted),

`-I` activates the computation (Int),

`-L` activates the computation (LC).

These three options can be accumulated. If at least two options are set, the computations are carried out according to the following rules:

- If `-E` is present, `-L` will be suppressed since its result is contained in that of `-E`.
- If `-I` is present, then it will be suppressed if one of `-E` or `-L` is set and the polynomial is homogeneous since `-L` and `-I` are identical for homogeneous polynomials.

If two different computations are carried out, then their output will appear consecutively in the output file.

If `-F=<polynomial>` appears, then the polynomial is read from the file `<polynomial>.pnm`. Note that `<polynomial>.pnm` must reside in the directory defined by `<project>`. It is not possible to prefix `<polynomial>` by a path name (which may be necessary for `<project>`).

If the option `-F=<polynomial>` is omitted, the polynomial is read from `<project>.pnm`.

The options `-c` and `-x=<T>` are passed form Normaliz to NmzIntegrate and vice versa. There is no need to worry about the integer precision of Normaliz or NmzIntegrate: Normaliz chooses it automatically and NmzIntegrate does always work with infinite precision.

It is not possible (presently) to use the option `-F=<polynomial>.pnm` if NmzIntegrate is called from Normaliz.

Note that NmzIntegrate may need much more memory than Normaliz, especially with a high number of parallel threads, due to the fact that it may have to cope with very long polynomials.

One can explicitly set the output directory. This information is passed from NmzIntegrate to Normaliz and vice versa. The syntax is

`--OutputDir=<outdir>` . The path `<outdir<` is an absolute path or a path relative to the current directory (which is not necessarily the directory of `<project>.in`.)

Note that all output files, also those of Normaliz, will be written to the chosen directory. It must be created before Normaliz or NmzIntegrate is started.

As usual, `--help` displays a short help screen and `--version` prints the version number to the standard poutput.

5 The output file

If the option `-F=<polynomial>` is not set, the output is written to the file `<project>.intOut` (so that it is clearly distinguished from the Normaliz output file). If `-F=<polynomial>` appears, the output is written to `<project>.<polynomial>.intOut`.

NmzIntegrate factors the polynomial, and the factorization is written to the output file. For the computation (LC) the polynomial is first replaced by its leading form, and the output file then contains the factorization of the leading form.

The output file is essentially self explanatory. Nevertheless we have added two examples below. In addition you can have a look at the files

`rationalES.intOut`, `rationalInt.intOut` and `rationalLC.intOut`.

They were all produced from the example file `rational.in` in the Normaliz distribution and the file `rational.pnm`, and `rational.intOut` was suitably renamed.

5.1 A generalized Ehrhart series

We choose an example from combinatorial voting theory which is discussed in more detail in [2]. The file `CondorcetSymm.in` from the directory `examples` contains the following:

```
amb_space 8
inequalities 3
-1 -1 -1 -1  1  1  1 1
-1 -1  1  1 -1 -1  1 1
-1  1 -1  1 -1  1 -1 1
nonnegative
total_degree
```

By `nonnegative` we choose the nonnegative orthant in $\mathbb{R}^8$ and the linear forms $\lambda_1, \lambda_2, \lambda_3$ specified by the `inequalities` cut out a cone from it by the conditions $\lambda_i(x) \geq 0$, $i = 1, \ldots, 3$. The `total_degree` gives degree 1 to every coordinate. The polynomial (counting the preimages of x under a projection $\mathbb{R}^{24}_+ \to \mathbb{R}^8_+$) is

$$f(x) = \binom{x_1+5}{5}(x_2+1)(x_3+1)(x_4+1)(x_5+1)(x_6+1)(x_7+1)\binom{x_8+5}{5}.$$

It is given in `CondorcetSymm.pnm` by

```
(x[1]+1)*(x[1]+2)*(x[1]+3)*(x[1]+4)*(x[1]+5)*
(x[2]+1)*(x[3]+1)*(x[4]+1)*(x[5]+1)*(x[6]+1)*(x[7]+1)*
(x[8]+1)*(x[8]+2)*(x[8]+3)*(x[8]+4)*(x[8]+5)*1/14400;
```

From the Normaliz directory we invoke NmzIntegrate by

```
nmzIntegrate -c example/CondorcetSymm
```

(replace the slash by a backslash in MS Windows, and similarly below). The file `CondorcetSymm.intOut` starts with the factorization:

```
Factorization of polynomial:
x[8] +5  mult 1
x[8] +4  mult 1
...
x[1] +1  mult 1
Remaining factor 1/14400
```

Next we find the information on the Hilbert series:

```
Generalized Ehrhart series:
1 5 133 363 ... 481 15 6
Common denominator of coefficients: 1
Series denominator with 24 factors:
1: 1  2: 14  4: 9
```

It is to be read as follows:

$$H_{M,f}(t) = \frac{1 + 5t^1 + 133t^2 + 363t^3 + \cdots + 481t^{38} + 15t^{39} + 6t^{40}}{(1-t^1)(1-t^2)^{14}(1-t^4)^9}$$

Next we find the presentation of $H_{M,f}(t)$ as a rational function with coprime numerator and denominator (which in this case is the same as the previous one, except that the denominator is factored differently):

```
Generalized Ehrhart series with cyclotomic denominator:
1 5 133 363 ... 481 15 6
Common denominator of coefficients: 1
Series cyclotomic denominator:
1: 24  2: 23  4: 9
```

This means

$$H_{M,f}(t) = \frac{1+t+5t^1+133t^2+363t^3+\cdots+481t^{38}+15t^{39}+6t^{40}}{\zeta_1^{24}\zeta_2^{23}\zeta_4^9}$$

where ζ_i is the i-th cyclotomic polynomial. Now the quasipolynomial:

```
Generalized Ehrhart quasi-polynomial of period 4:
 0:   693959790182222163590774784000 20899225...000000 ... 56262656
 1:   203475031022335179700809216000  7092764...648000 ... 56262656
 2:   693308184929915219977568256000 20892455...168000 ... 56262656
 3:   203475031022335179700809216000  7092764...648000 ... 56262656
with common denominator: 693959790182222163590774784000
```

The left most column indicates the residue class modulo the period, and the numbers in line k are the coefficients of the k-th polynomial after division by the common denominator. The list starts with $q_0^{(k)}$ and ends with (the constant) $q_{23}^{(k)}$. The interpretation of the remaining data is obvious:

```
Degree of (quasi)polynomial: 23
Expected degree: 23
Virtual multiplicity: 1717/8192
```

Now suppose we want to work with the strict inequalities $\lambda_i(x) > 0$, as customary in voting theory (in order to exclude draws). Then we replace `inequalities` by `excluded_faces` to obtain the file `CondorcetSymmSemi.in`. the polynomial hasn't changed, and so NmzIntegrate is called by

```
nmzIntegrate -c -F=CondorcetSymm example/CondorcetSymmSemi
```

The output is now in `CondorcetSymmSemi.CondorcetSymm.intOut`.

5.2 An integral

The paper [3] asks for the computation of the integral

$$\int\limits_{\substack{[0,1]^m \\ \sum x=t}} (x_1\cdots x_m)^{n-m} \prod_{1\le i<j\le m} (x_j-x_i)^2 \mathrm{d}\mu$$

taken over the intersection of the unit cube in $\mathbb{R}^m$ and the hyperplane of constant coordinate sum t. It is supposed that $t \leq m \leq n$. We compute the integral for $t = 2$, $m = 4$ and $n = 6$.

The polytope is specified in the input file `j462.in` (typeset in 2 columns):

```
amb_space 5          -1 0 0 0 1
inequalities 8       0 -1 0 0 1
1 0 0 0 0            0 0 -1 0 1
0 1 0 0 0            0 0 0 -1 1
0 0 1 0 0            equations 1
0 0 0 1 0            -1 -1 -1 -1 2
```

The 8 inequalities describe the unit cube in $\mathbb{R}^4$ by the inequalities $0 \leq z_i \leq 1$ and the equation gives the hyperplane $z_1 + \cdots + z_4 = 2$ (we must use homogenized coordinates!). There is no need to specify the grading since Normaliz finds it because the polytope is a lattice polytope. If one doesn't know this in advance, it is better to give the grading explicitly by

```
grading
unit_vector 5
```

See the Normaliz documentation, Section 3.2.5 how to define rational polytopes by inequalities and equations.

The polynomial does not depend on t so that we can use the same polynomial for various t. It is contained in `j46.pnm`:

```
(x[1]*x[2]*x[3]*x[4])^2*(x[1]-x[2])^2*(x[1]-x[3])^2*
(x[1]-x[4])^2*(x[2]-x[3])^2*(x[2]-x[4])^2*(x[3]-x[4])^2
```

NmzIntegrate is called by

```
nmzIntegrate -cI -F=j46 example/j462
```

It produces the output in `j462.j46.intOut`:

```
Factorization of polynomial:          x[1]  mult 2
x[4]  mult 2                          x[1] -x[2]  mult 2
x[3]  mult 2                          x[1] -x[3]  mult 2
x[3] -x[4]  mult 2                    x[1] -x[4]  mult 2
x[2]  mult 2                          Remaining factor 1
x[2] -x[3]  mult 2
x[2] -x[4]  mult 2                    Integral: 27773/29515186701000
```

6 Distribution and installation

The basic package (source, documentation, examples) for NmzIntegrate is contained in the basic package of Normaliz that you can download from

The installation is described in the Normaliz documentation.

Likewise the executable of NmzIntegrate is contained in the Normaliz executable package for your system. Therefore NmzIntegrate does not need a separate installation.

7 Restrictions under MS Windows

Unfortunately we have not been able to recompile NmzIntegrate under MS Windows. The newest available binary file is of version 1.3. It has the following restrictions:

(1) Due to a bug it is possible that a segmentation fault occurs if excluded faces are used.
(2) The option `OutputDir` is not available.
(3) One cannot use the option `ConeDecomposition` of Normaliz. (Not a real problem in this context, of course.)

8 Compilation

Before the compilation of NmzIntegrate you must install CoCoALib 0.99540 [1] (not contained in the Normaliz distribution). See the Normaliz manual for the details.

9 Copyright and how to cite

Please refer to Normaliz in any publication for which NmzIntegrate it has been used:

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

The corresponding `\bibitem`:

```
\bibitem W. Bruns, B. Ichim, R. Sieg, T. R\"omer and C. S\"oger:
Normaliz. Algorithms for rational cones and affine monoids.
Available at \url{https://normaliz.uos.de}.
```

A BibTeX entry:

```
@Misc{Normaliz,
author = {W. Bruns and B. Ichim and R. Sieg and T. R\"omer and C. S\"oger},
title = Normaliz. Algorithms for rational cones and affine monoids,
howpublished ={Available at \url{https://normaliz.uos.de}.}
}
```

You can add a reference to [2] in order to indicate that NmzIntegrate has been used.

10 History

10.1 1.0 → 1.1

1. NmzIntegrate can now be used on objects that do not have maximal dimension in their surrounding space.
2. NmzIntegrate calls Normaliz if input files are missing.
3. The input syntax for polynomials has been improved: white space is neglected.
4. The efficiency has been improved significantly by using integral arithmetic internally instead of rational arithmetic.

10.2 1.1 → 1.2

1. Use of the (now existing) CoCoALib function for input of polynomials and other small changes reflecting the development of CoCoALib.
2. Extension to semiopen cones.
3. Name of file with suffix `pnm` can be specified independently of the name of the project.

10.3 1.2 → 1.3

1. Adaptation to CoCoALib 0.99538.
2. Adaptation (of this manual) to Normaliz 3.0.

10.4 1.3 → 1.3.1

1. Adaptation to CoCoALib 0.99540.

2. Changes implied by new features of Normaliz 3.1.1.

10.5 1.3.1 → 1.3.2

1. OutputDir added.

10.6 1.3.2 → 1.3.3

1. Manual adapted to the introduction of option `--Symmetrize` in Normaliz.
2. Bugfix

References

[1] J. Abbott and A. Bigatti, *CoCoALib*. A GPL C++ library for doing Computations in Commutative Algebra. Available from `http://cocoa.dima.unige.it/cocoalib/`

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

[3] J. Jeffries, J. Montaño and M. Varbaro, *Multiplicities of classical varieties*. Proc. Lond. Math. Soc. (3) 110 (2015), 1033–1055.