TUW-12-10
IPMU12-0094

PALP – a User Manual

Andreas P. Braun[1], Johanna Knapp[2], Emanuel Scheidegger[3],
Harald Skarke[1] and Nils-Ole Walliser[1]

[1] Institute for Theoretical Physics, Vienna University of Technology,
Wiedner Hauptstrasse 8-10/136, 1040 Vienna, Austria
abraun, skarke, walliser@hep.itp.tuwien.ac.at

[2] Kavli IPMU (WPI), The University of Tokyo,
5-1-5 Kashiwanoha, Kashiwa, Chiba 277-8583, Japan
johanna.knapp@ipmu.jp

[3] Institute for Mathematics, University of Freiburg,
Eckerstrasse 1, 79104 Freiburg, Germany
emanuel.scheidegger@math.uni-freiburg.de

Abstract

This article provides a complete user's guide to version 2.1 of the toric geometry package PALP by Maximilian Kreuzer and others. In particular, previously undocumented applications such as the program `nef.x` are discussed in detail. New features of PALP 2.1 include an extension of the program `mori.x` which can now compute Mori cones and intersection rings of arbitrary dimension and can also take specific triangulations of reflexive polytopes as input. Furthermore, the program `nef.x` is enhanced by an option that allows the user to enter reflexive Gorenstein cones as input. The present documentation is complemented by a Wiki which is available online.

Contents

1 Introduction

1.1 A brief history of PALP

The first lines of code that would eventually become a part of PALP were probably written in 1992. At that time Max Kreuzer worked together with one of us (HS) on certain quasi-homogeneous functions relevant to the description of Landau–Ginzburg models that also had interpretations in terms of Calabi–Yau hypersurfaces in weighted projective spaces. This culminated in the classification (also found, independently, by Klemm and Schimmrigk [1]) of all such functions relevant to standard string compactifications. As the title of the paper [2], 'No mirror symmetry in Landau–Ginzburg spectra!', suggests, mirror symmetry was incomplete in that class of models and it was necessary to look for more general scenarios. These were indeed provided by Batyrev's elegant construction of mirror pairs of Calabi–Yau spaces via dual pairs of reflexive polytopes [3].

After the proposal [4, 5] of an algorithm for the classification of reflexive polytopes, work on the implementation of the required routines commenced. The expertise gained in this project and parts of the code could be used to consider questions like the manifestation of fibration structures in the toric context [6, 7] or the connectedness of the moduli space of Calabi–Yau hypersurfaces described by reflexive polytopes [8], and these projects in turn enhanced the stock of available C routines. A first implementation of the whole algorithm led to the generation of the complete list of reflexive 3-polytopes [9], but only a thoroughly revised and optimized version of the code could generate all 473,800,776 reflexive polytopes in four dimensions [10].

By that time Max was also working with his graduate student Erwin Riegler on an extension to include nef partitions (leading eventually to [11, 12, 13] and to `nef.x`). The collection of available routines had reached a number, a level of complexity and a lack of documentation that would have rendered them useless within a very short time without any efforts at preservation. Besides, it was clear that the programs might be useful to other people as well. So it was decided to work on polishing and documenting the existing routines with the aim of combining them into a publicly available package. After some time and several candidates (among them 'lpoly') the name of the package became PALP, containing `poly.x`, `class.x`, `cws.x`. This is an acronym for 'Package for Analysing Lattice Polytopes', but we find it quite appropriate that it shares this name with rather obscure body parts of arachnidae [14].

During the period when Max and Erwin were starting to compile nef partitions, one of us (ES) joined Max' group in Vienna as a postdoc. This led to a shift of the focus from the classification of nef partitions more towards applications in mirror symmetry and led to a number of new options in `nef.x`.

A refinement of polytope data by triangulations and the corresponding Mori cones was made desirable by the following well known facts. Different triangulations of a polytope, hence different intersection rings, may lead to topologically distinct Calabi–Yau manifolds, while non–isomorphic polytopes can give rise to equivalent Calabi–Yau manifolds; the intersection ring is an essential ingredient in Wall's theorem on the classification of 6-manifolds [15]. From the point of view of mirror symmetry, the intersection ring and the Mori cone are important as they enter the GKZ hypergeometric system of differential equations governing the periods of the mirror hypersurface.

This was enough motivation to extend the existing routines to the computation of the Mori cone which can be defined entirely in terms of combinatorial data. At that time the triangulation was viewed as an external input determined by some other specialized program such as TOPCOM [16]. After the initial success one of Max' graduate students (JK) started to develop a code in SINGULAR [17] that computes from this combinatorial data the intersection rings of the toric variety and the Calabi–Yau hypersurface. This spawned what later would become `mori.x`.

A couple of years later Max, together with another graduate student (NW) took this up with the goal of creating a routine which determines all the unimodular coherent star triangulations within PALP without having to rely on any external input.

Despite the fact that PALP was originally designed for the specific purposes mentioned above it has become a versatile tool for both mathematics and physics applications. One indicator for the success of PALP is that it has been included into the Sage package [18] and the Debian repositories.

1.2 How to use this manual

One of the biggest drawbacks of PALP is the combination of complicated syntax and lack of concise documentation. While we decided to keep the syntax and its oddities (cf. section 2) for the sake of continuity, we would like to overcome the documentation issue with this article and a PALP Wiki which is available at [19]. Some parts of PALP have already been discussed previously. The original paper accompanying the first version of PALP is [20]. It contains documentation on the programs `poly.x`, `cws.x` and `class.x`. The program `mori.x` has been presented in [21]. The program `nef.x` for analyzing complete intersections in toric ambient spaces has been written by Erwin Riegler as part of his PhD thesis [13] but has never been documented. In writing the present manual we have tried to cover all applications, i.e. there should be no need to read the older papers as well, except for few passages that we cite at the appropriate points.

In general we do not explain concepts from the theory of polytopes or from toric geometry, except where this serves to fix notation or where we use non–standard terminology. The reader is referred to the standard textbooks [22, 23, 24] or any of a number of reviews (those written by PALP programmers [25, 26] are probably closest to the style of this manual).

We recommend that everyone interested in using PALP read section 2 on general aspects of the package, which may hold some surprises even for reasonably experienced users. The next step is to choose some application of PALP (consulting the following paragraph should help to decide which program provides this application). Then one can jump to the section describing that program and read the general part of that section. Finally one should consult the subsections where the required options are described.

This article is organized as follows. In section 2 we give a general overview of the PALP package and discuss generic properties such as the input of polytope data and error handling. Furthermore we point out some peculiarities of PALP. The remaining sections each correspond to one of the executable programs, with a brief general introduction followed by descriptions for all the available options. Section 3 is devoted to the program `poly.x` which contains mainly general purpose routines for analyzing lattice polytopes but also some specialized routines related to applications in string theory and algebraic geometry that do not fit into other parts of the package. In sections 4 and 5 we describe the programs `cws.x` and `class.x` which have been essential for the classification of reflexive polytopes. Section 6 contains the documentation of the program `nef.x` which provides routines to analyze nef partitions of reflexive polytopes. In section 7 we discuss PALP's most recent application `mori.x` which computes the Mori cone of a toric variety and, with the help of the program SINGULAR, topological data such as intersection numbers of (not necessarily Calabi-Yau) hypersurfaces in those ambient spaces.

2 General aspects of using PALP

In this section we treat aspects of PALP that are common to most or all of its applications. The first step is to download the package from the website [27] and follow the compilation instructions given there, which should result in the existence of a directory 'palp' containing the program as well as the executable files.

2.1 Polytope input

The majority of applications requires input in the form of a list of polytopes. There are essentially two ways of entering the data of a polytope. Matrix input starts with a line containing two numbers n_{lines} and n_{columns} (which may be followed by text which is simply ignored by the program) and proceeds with a matrix with the corresponding numbers of lines and columns. PALP requires $n_{\text{lines}} \neq n_{\text{columns}}$ and interprets the smaller of the two numbers as the dimension of the polytope and the other one as the number of polytope points entered as lines or columns of input.

```
palp$ poly.x
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 2  This text is ignored by PALP
Type the 6 coordinates as #pts=3 lines with dim=2 columns:
2 0
0 2
0 0
M:6 3 F:3
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
2 3 The same example with transposed input
Type the 6 coordinates as dim=2 lines with #pts=3 columns:
2 0 0
0 2 0
M:6 3 F:3
```

In both cases the input specifies the polygon (2-polytope) that is the convex hull of the 3 points $\{(2,0),(0,2),(0,0)\}$ in $M = \mathbb{Z}^2$. The output just means that this polygon has 6 lattice points, 3 vertices and 3 facets (here, edges). The possibility of ignored text in the input is useful because PALP's output can often be used as input for further applications; thereby extra information can be displayed without destroying the permissible format.

For applications in the context of toric geometry one should be aware of the fact that there are two relevant, mutually dual lattices M and N whose toric interpretations are quite different. By default PALP interprets the input polytope as $\Delta \subset M_{\mathbb{R}}$. Note that PALP refers to this polytope as P; in this paper we shall use both notations. If Δ $(= P)$ is reflexive, it is very natural (and, for some applications, more natural) to consider its dual $\Delta^* \subset N_{\mathbb{R}}$ as well. If PALP should interpret the input as Δ^*, it must be instructed to do that by an option (`-D` for `poly.x` and `mori.x`, `-N` for `nef.x`). In fact, in the case of `mori.x` it would be extremely unnatural to use Δ as input; therefore matrix input is allowed only with `-D` to avoid errors.

A second input format uses the fact that many polytopes (in particular the ones related to the toric description of weighted projective spaces) afford a description as the convex hull of all points X that lie in the $(n-1)$-dimensional sublattice $M \subset \mathbb{Z}^n$ determined by $\sum_{i=1}^n w_i X_i = 0$ and satisfy the inequalities $X_i \geq -1$ for $i \in \{1,\ldots,n\}$. Given such a weight system in the format `d w1 w2 ...  wn` where the w_i must be positive integers and $d = \sum_{i=1}^n w_i$, PALP computes the corresponding list of points and makes a transformation to $M \simeq \mathbb{Z}^{n-1}$. The following example corresponds to the Newton polytope of the quintic threefold in $\mathbb{P}^4$.

```
palp$ poly.x -v
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
```

```
  or ‘#lines #columns’ (= ‘PolyDim #Points’ or ‘#Points PolyDim’):
5 1 1 1 1 1
4 5  Vertices of P
   -1    4   -1   -1   -1
   -1   -1    4   -1   -1
   -1   -1   -1    4   -1
   -1   -1   -1   -1    4
```

As the first line of the prompt indicates, this format can be generalized to the case of k weight systems describing a polytope in $M \simeq \mathbb{Z}^{n-k}$. We call the corresponding data, which should satisfy $w_{ij} \geq 0$ and $(w_{1j}, \ldots, w_{kj}) \neq (0, \ldots, 0)$, a CWS ('combined weight system').

It is also possible to specify a sublattice of finite index corresponding to the condition $\sum_{i=1}^{n} l_i x_i = 0 \bmod r$ by writing `/Zr:  l1 ...ln` after the specification of the (C)WS.

In the following example, `21100 20011` describes a square whose edges have lattice length 2, whereas the condition indicated by `Z2:  1 0 1 0` eliminates the interior points of the edges. The particular output arises because PALP transforms the original and the reduced lattice to $\mathbb{Z}^2$ in different ways.

```
palp$ poly.x -v
Degrees and weights  ‘d1 w11 w12 ... d2 w21 w22 ...’
  or ‘#lines #columns’ (= ‘PolyDim #Points’ or ‘#Points PolyDim’):
2 1 1 0 0  2 0 0 1 1
2 4  Vertices of P
   -1    1   -1    1
   -1   -1    1    1
Degrees and weights  ‘d1 w11 w12 ... d2 w21 w22 ...’
  or ‘#lines #columns’ (= ‘PolyDim #Points’ or ‘#Points PolyDim’):
2 1 1 0 0  2 0 0 1 1  /Z2: 1 0 1 0
2 4  Vertices of P
   -1    0    0    1
    1   -1    1   -1
```

For a reconstruction of the CWS given a polytope as matrix input see the option `cws.x -N` in Section 4.2.6.

If PALP is used interactively, it can be terminated by entering an empty line instead of the data of a polytope. In the case of file input the end of the file results in the termination.

2.2 Error handling

PALP is designed in such a way that it should exit with an error message rather than crash or display wrong results. The main sources for problems are inappropriately set parameters, lack of memory and numerical overflows. The most important settings of parameters all occur at the beginning of `Global.h`, which is probably the only file that a user may want to modify.

Here are some typical error messages. If we want to analyze the Calabi–Yau sixfold that is a hypersurface in $\mathbb{P}^7$ with `poly.x`, the following will happen if PALP has been compiled with the default settings.

```
8  1 1 1 1 1 1 1 1
Please increase POLY_Dmax to at least 7
```

In this case one should edit `Global.h` (see also section 2.3.4), setting

```
#define          POLY_Dmax       7       /* max dim of polytope    */
```

and compile again. Similarly the program may ask for changes of other basic parameters, all of which are defined within the first 52 lines of `Global.h`.

In many cases we have implemented checks with the help of the 'assert' routine, leading to error messages such as the following.

```
poly.x: Vertex.c:613: int Finish_IP_Check(PolyPointList *, ...
EqList *, CEqList *, INCI *, INCI *): Assertion '_V->nv<32' failed.
Abort
```

In this case one should look up line 613 of `Vertex.c`,

```
assert(_V->nv<VERT_Nmax);
```

This indicates that the value of `_V->nv` has risen above the value 32 assigned to `VERT_Nmax` in `Global.h` and that the value of `VERT_Nmax` should be changed correspondingly. At this point it is important to note that the setting of parameters in `Global.h` depends on the setting of `POLY_Dmax`:

```
#define         POLY_Dmax        6      /* max dim of polytope     */
...
#if (POLY_Dmax <= 3)
#define         POINT_Nmax      40      /* max number of points    */
#define         VERT_Nmax       16      /* max number of vertices  */
#define         FACE_Nmax       30      /* max number of faces     */
#define         SYM_Nmax        88      /* cube: 2^D*D! plus extra */

#elif (POLY_Dmax == 4)
#define         POINT_Nmax     700      /* max number of points    */
#define         VERT_Nmax       64      /* max number of vertices  */
...
```

Of course one should then modify a parameter such as `VERT_Nmax` or `SYM_Nmax` at the position corresponding to the chosen value of `POLY_Dmax`. For `POLY_Dmax` taking values up to 4, the default parameters in `Global.h` are chosen in such a way that they work for any reflexive polytope.

While the error messages mentioned above are related to parameter values that are too low, excessively high values may also lead to problems such as slowing down the calculation. In particular, computation time depends very sensitively upon whether `VERT_Nmax` is larger than 64. Very high parameter values may also lead to troubles with memory which manifest themselves in error messages such as `Unable to alloc space for ...` or `Allocation failure in ...` or even `Segmentation fault`. In such a case one can only check whether there are possibilities for making parameters smaller, or use a computer with more RAM.

An assertion failure that does not refer to an inequality involving a parameter or an allocation failure, such as

```
NFX_Limit in GL -> 1074575416 !!
```

is very likely to point to a numerical overflow. In such a case it might help to change line 12 of `Global.h` from

```
#define Long long
```

to

```
#define Long long long
```

These issues are particularly relevant to the analysis of high-dimensional polytopes, e.g. in the case of `nef.x` with nef partitions of large length. In this case, it may happen that certain parameters in the header file `Nef.h` may also need to be modified. Here we give a particularly nasty example:

```
palp$ nef.x -Lp -N -c6 -P
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
7 9
Please increase POLY_Dmax to at least 12 = 7 + 6 - 1
(nef.x requires POLY_Dmax >= dim N + codim - 1)
```

This means that in `Global.h` we need to set `POLY_Dmax` to at least 12:

```
#define         POLY_Dmax       12       /* max dim of polytope  */
```

After recompiling PALP we get further but not far enough:

```
palp$ nef.x -Lp -N -c6 -P
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
7 9
Type the 63 coordinates as dim=7 lines with #pts=9 columns:
 1  0  0  0  0 -1  0  0 -1
 0  1  0  0  0 -1  0  0 -1
 0  0  1  0  0 -1  0  0 -1
 0  0  0  1  0 -1  0  0  0
 0  0  0  0  1 -1  0  0  0
 0  0  0  0  0  0  1  0 -1
 0  0  0  0  0  0  0  1 -1
M:5214 12 N:10 9  codim=6 #part=1
7 10  Points of Poly in N-Lattice:
    1    0    0    0    0   -1    0    0   -1    0
    0    1    0    0    0   -1    0    0   -1    0
    0    0    1    0    0   -1    0    0   -1    0
    0    0    0    1    0   -1    0    0    0    0
    0    0    0    0    1   -1    0    0    0    0
    0    0    0    0    0    0    1    0   -1    0
    0    0    0    0    0    0    0    1   -1    0
----------------------------------------------------
    1    1    1    1    1    1    0    0    0  d=6  codim=2
    1    1    1    0    0    0    1    1    1  d=6  codim=2
nef.x: Vertex.c:613: Finish_Find_Equations:
                     Assertion '_V->nv<64' failed.
Aborted
```

This can be remedied by adjusting the global variable `VERT_Nmax` in `Global.h` as follows (it should not be too large):

```
#define           VERT_Nmax    128    /* !! use optimal value !!  */
```

After recompilation it works for a while. Then the following error occurs

```
Unable to alloc space for _BL
```

This means that the program has run out of memory.

2.3 Some peculiarities of PALP

Much of PALP's code was written originally with a very specific aim (the classification problem) in mind, and not with the intention of designing a package that would be immediately accessible to many users. Other applications were added later by different people. This has resulted in several peculiar features whose comprehension might help to avoid errors. In the following we list a few of them.

2.3.1 Option names

PALP does not have any clear convention on how options are named. In fact, it can happen that options with the same effects have different names in different parts of the package: for instance, in order to make PALP interpret an input polytope as the (dual) N lattice polytope, one has to use `-D` with `poly.x` and `mori.x` but `-N` with `nef.x`. Ironically, `poly.x` also has an option `-N` whose effect is completely different. It is also worth noting that `poly.x` and `mori.x` admit concatenating several options into one string, e.g. `poly.x -gve`, `mori.x -Hb` whereas other programs require them to be separate, e.g. `nef.x -Lp -N -c6 -P`.

2.3.2 Indexing conventions

Most of PALP, being programmed in C, follows the convention of using $\{0, 1, \ldots, n-1\}$ as the standard n element set. So lists of vertices take the form $v_0, v_1, \ldots, v_{n_v-1}$, a list of points is given as $p_0, p_1, \ldots, p_{n_p-1}$, and so on. The exception is `mori.x` which uses $\{1, \ldots, n\}$ as the standard set.

2.3.3 Binary representation of incidences

PALP represents incidences as bit patterns both internally and in its output; in the case of `mori.x -M` even input is required in that format. This works in the following manner. To any face ϕ and vertex v_i of P a bit b_i is assigned via $b_i = 1$ if $v_i \in \phi$ and $b_i = 0$ otherwise; $v_0, \ldots, v_{n_v-1}$ are ordered as in the output of `poly.x -v`. This results in a bit pattern $B_v(\phi)$ which can be written as if it represented a binary number, $B_v = b_{n_v-1} b_{n_v-2} \ldots b_0$. This is the convention implemented in `poly.x`, whereas `mori.x` writes it as a sequence from left to right, $B_v = b_1 \ldots b_{n_v-1} b_{n_v}$ (remember the last subsection about indexing conventions!). Furthermore, a bit pattern $B_f = \{\tilde{b}_j\}$ related to P's facets f_j (ordered as in the output of `poly.x -e`) can be assigned via $\tilde{b}_j = 1$ if $\phi \subseteq f_j$ and $\tilde{b}_j = 0$ otherwise.

2.3.4 The parameter `POLY_Dmax`

`POLY_Dmax` (line 22 of `Global.h`) is the most important parameter to set before compilation; for most applications it is probably the only parameter one has to care about. While it usually suffices to have `POLY_Dmax` not smaller than the dimension of any polytope that one wants to analyze, there are the following important exceptions.
`nef.x` normally requires `POLY_Dmax` $\geq \dim(N) + \mathrm{codim} - 1$, which is the dimension of the support polytope of the corresponding Gorenstein cone which `nef.x` analyzes;
`nef.x -G` requires `POLY_Dmax` $\geq \dim(\text{input-polytope}) + 1$ because the input-polytope is interpreted as the support of the cone and the full dimension of the cone is required for technical reasons;

`mori.x -M` requires `POLY_Dmax` $\geq n_p - \dim(N) - 1$, the dimension of the Mori cone; n_p is the total number of input points including the lattice origin, hence the subtraction of 1.

2.3.5 IP property and IP simplices

In PALP's help screens and output messages every now and then the abbreviation IP occurs. A priori this is just a shortcut for writing 'interior point', as in 'the generic CY hypersurface does not intersect the divisors corresponding to IPs of facets'. However, when we say that a polytope has the IP property, we mean that the polytope has the lattice origin in its interior. IP simplices are simplices with this property, usually with vertices that are points or vertices of some given polytope. This concept played an important role in the classification scheme of [4, 5, 28]. The lattice vectors corresponding to the vertices of an IP simplex define a linear relation with positive coefficients which is unique up to scaling. Conversely any positive linear relation among lattice points that cannot be written as the sum of two other such relations defines an IP simplex. Of course, the set of coefficients is just the weight system defined by the IP simplex.

3 `poly.x`

3.1 General description of `poly.x`

`poly.x` is the program that provides an interface for PALP's general purpose routines as well as certain specialized applications that do not belong to any of the other executables. In other words, `poly.x` deals with all applications that are not related to nef partitions, Mori cones or the classification of reflexive polytopes. As for all of PALP's programs, a rough guide can be obtained with the help option:

```
palp$ poly.x -h

This is 'poly.x':  computing data of a polytope P
Usage:   poly.x [-<Option-string>] [in-file [out-file]]

Options (concatenate any number of them into <Option-string>):
h  print this information
f  use as filter
g  general output:
   P reflexive: numbers of (dual) points/vertices, Hodge numbers
   P not reflexive: numbers of points, vertices, equations
p  points of P
v  vertices of P
e  equations of P/vertices of P-dual
m  pairing matrix between vertices and equations
d  points of P-dual (only if P reflexive)
a  all of the above except h,f
l  LG-'Hodge numbers' from single weight input
r  ignore non-reflexive input
D  dual polytope as input (ref only)
n  do not complete polytope or calculate Hodge numbers
i  incidence information
s  check for span property (only if P from CWS)
```

```
I  check for IP property
S  number of symmetries
T  upper triangular form
N  normal form
t  traced normal form computation
V  IP simplices among vertices of P*
P  IP simplices among points of P* (with 1<=codim<=# when # is set)
Z  lattice quotients for IP simplices
#  #=1,2,3  fibers spanned by IP simplices with codim<=#
## ##=11,22,33,(12,23): all (fibered) fibers with specified codim(s)
   when combined: ### = (##)#
A  affine normal form
B  Barycenter and lattice volume [# ... points at deg #]
F  print all facets
G  Gorenstein: divisible by I>1
L  like 'l' with Hodge data for twisted sectors
U  simplicial facets in N-lattice
U1 Fano (simplicial and unimodular facets in N-lattice)
U5 5d fano from reflexive 4d projections (M lattice)
C1 conifold CY (unimodular or square 2-faces)
C2 conifold FANO (divisible by 2 & basic 2 faces)
E  symmetries related to Einstein-Kaehler Metrics

Input:    degrees and weights 'd1 w11 w12 ... d2 w21 w22 ...'
          or 'd np' or 'np d' (d=Dimension, np=#[points]) and
              (after newline) np*d coordinates
Output:   as specified by options
```

If an input file is indicated, `poly.x` reads its input data from there; otherwise it asks for input interactively. The output is displayed to the screen unless an output file is specified. The following subsection will explain all of the possible options, in the order in which they appear in the help screen. Here is a rough guide in terms of specific topics:

- General polytope analysis without reference to toric geometry or string theory: `-g, -p, -v, -e, -m, -d, -a, -i, -s, -I, -T, -N, -t, -A, -B, -F, -G, -U, -U1, -E`;
- Conifold singularities: `-C1, -C2`;
- Fano varieties: `-U1, -U5, -C2`;
- Fibration structures: `-[number], -V, -P, -Z`;
- IP property (see section 2.3.5): `-I`;
- IP simplices (see section 2.3.5): `-V, -P, -Z, -[number]`;
- Landau-Ginzburg type superconformal field theories: `-l, -L`;
- Normal forms: `-N, -A`;
- Sublattices and quotient actions: `-S, -Z, -G`;
- Symmetries of a polytope: `-S, -t`.

3.2 Options of `poly.x`

As many of the following options are very simple to use, we do not always provide examples. In such cases we highly recommend to try using the option with simple input, e.g. the weight system `5 1 1 1 1 1` corresponding to the quintic, and a non-reflexive example such as

```
3 2
Type the 6 coordinates as #pts=3 lines with dim=2 columns:
2 0
0 2
0 0
```

3.2.1 no option set

In this case the program behaves as if the `-g` option (see below) were set. This is also the case if no option other than `-r`, `-D`, `-n`, `-Z`, `-U` or `-U1`, which do not generate any output per se, is applied.

3.2.2 -h

The help screen is displayed (see above).

3.2.3 -f

The filter flag switches off the prompt for input data. This is useful for building pipelines.

3.2.4 -g

The following output is generated. First, the input is repeated if it is of weight/CWS type, but not otherwise. Then the numbers $\#p$ and $\#v$ of lattice points and vertices, respectively, are displayed in the format 'M: $\#p$ $\#v$'. The remaining output depends on whether P is reflexive. In this case the numbers $\#d$ and $\#e$ of dual lattice points and vertices are displayed in the format 'N: $\#d$ $\#e$', followed by information on the Hodge numbers of the corresponding Calabi–Yau manifold if $\dim(P) \geq 4$; in the case of a three dimensional polytope corresponding to a K3 surface, where the Hodge numbers are determined anyway, information on the Picard number and the 'correction term' is given instead (the latter is the non-linear term in Batyrev's formula for the Picard number [3]; the Picard numbers of a K3 and its mirror add up to $20 + Cor$). For non-reflexive P the number $\#e$ of facets is shown as 'F: $\#e$'.

Using this option implies the completion of the set of lattice points in the convex hull ('points' in the help screen always means 'lattice points'). In the reflexive case it also leads to the completion of the dual polytope and the computation of the complete incidence structure which is required for the calculation of the Hodge numbers. For large dimensions these tasks may result in a long response time or in a crash of the program. In such a case one should use other options, e.g. `-nve`, if information on the number of lattice points or Hodge numbers is not required.

3.2.5 -p

The lattice points of the polytope are displayed.

3.2.6 -v

The vertices of the polytope are displayed.

3.2.7 -e

The equations of the hyperplanes bounding the polytope are displayed. If the polytope is not reflexive, these facet equations are given as lines 'a_1 ... a_n c' normalized such that the a_i have no common divisor and the inequalities $\vec{a} \cdot \vec{x} + c \geq 0$ are satisfied for all points of P.

Reflexivity of a lattice polytope P is equivalent to P^* being a lattice polytope, i.e. to $c = 1$ for all facet equations. In that situation the lines $\vec{a}$ can be interpreted as vertices of P^* and `poly.x` omits the final column of 1's, indicating that the resulting matrix can be interpreted as the list of vertices of the dual polytope. This has the advantage that the output can be used as input for further computations.

```
palp$ poly.x -e
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 2
Type the 6 coordinates as #pts=3 lines with dim=2 columns:
1 0
0 1
0 0
3 2  Equations of P
   1   0     0
   0   1     0
  -1  -1     1
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 2
Type the 6 coordinates as #pts=3 lines with dim=2 columns:
1 0
0 1
-1 -1
3 2  Vertices of P-dual <-> Equations of P
   2  -1
  -1   2
  -1  -1
```

In the first case the output indicates that P can be described by $x_1 \geq 0$, $x_2 \geq 0$, $-x_1 - x_2 + 1 \geq 0$; in the second case, where a last output column of 1's only is implicit, P corresponds to $2x_1 - x_2 + 1 \geq 0$, $-x_1 + 2x_2 + 1 \geq 0$, $-x_1 - x_2 + 1 \geq 0$.

3.2.8 -m

One gets the $n_v \times n_e$ matrix with entries $\vec{a}_j \cdot \vec{v}_i + c_j$, $1 \leq i \leq n_v$, $1 \leq j \leq n_e$, where n_v, n_e are the numbers of vertices and equations, respectively. The elements of this 'pairing matrix' represent the lattice distances between the respective vertices and facets. The orders of vertices and facets are the same as for `-v` and `-e`, so it is useful to combine `-m` with these options to see precisely which vertex and facet an entry of the pairing matrix corresponds to.

3.2.9 -d

If the polytope is reflexive the lattice points of the dual polytope are displayed.

3.2.10 -a

This is a shortcut for `-gpvemd`; it can be combined with any other options.

3.2.11 -l

This option is relevant to applications in the context of Landau-Ginzburg models. Together with its close relative `-L`, it is the only option of `poly.x` that requires non-standard input. Rather than indicate a polytope via matrix or CWS input, one specifies a single weight system which need not satisfy $\sum n_i = d$, which is interpreted as data for a superconformal field theory. If the central charge of this SCFT is a multiple of 3 (which is required by PALP 2.1), the analogue of the Hodge numbers [29, 30] is computed.

```
palp$ poly.x -l
type degree and weights  [d  w1 w2 ...]: 5 1 1 1 1 1
5 1 1 1 1 1 M:126 5 N:6 5 V:1,101 [-200]
type degree and weights  [d  w1 w2 ...]: 3 1 1 1 1 1 1
3 1 1 1 1 1 1 M:56 6 F:6 LG: H0:1,0,1 H1:0,20 H2:1 RefI2
type degree and weights  [d  w1 w2 ...]: 3 1 1 1 1 1 1 /Z3:
0 1 2 0 1 2 3 1 1 1 1 1 1 /Z3: 0 1 2 0 1 2
M:20 6 F:6 LG: H0:1,0,1 H1:0,20 H2:1 RefI2
```

Here the first example is the familiar quintic treated as the Gepner model $(3)^5$, the second example is the Gepner model $(1)^6$ and the third example an orbifold of the second one. More information can be found in section 4.1 of [20].

3.2.12 -r

Any input that does not correspond to a reflexive polytope will be ignored. This is useful for filtering out reflexive polytopes from a larger list and saves calculation time if one is interested only in reflexive polytopes.

3.2.13 -D

The input is regarded as the dual polytope $P^* \subset N_{\mathbb{R}}$. As this makes sense only in the reflexive case there is an error message (but no exit from the program) for non-reflexive input. This option is useful, in particular, if one wants to have control over the order of the points in the N lattice.

3.2.14 -n

The completion of the set of lattice points is suppressed. Hence the Hodge numbers cannot be calculated and the output will look like the one for non-reflexive polytopes even in the reflexive case. In particular, if the input is not of the (C)WS type, the number of points may be displayed wrongly. If $\dim(M)$ is large this option saves a lot of calculation time.

3.2.15 -i

Information on the incidence structure is displayed in the following manner. Remember from section 2.3.3 that any face ϕ can be assigned a bit pattern $B_v(\phi)$ encoding which vertices lie on ϕ and a bit pattern $B_f(\phi)$ encoding to which facets ϕ belongs. `poly.x -i` displays both types of bit patterns as binary numbers for all faces of P, with '`v[i]:`' starting a line of B_v's corresponding to i-faces and '`f[j]:`' starting a line of B_f's corresponding to j-faces. Bit patterns at the same positions in the '`v[i]:`' and '`f[i]:`' lines correspond to the same i-faces. The orders of faces within the lines are a consequence of the way PALP computes them; they conform to the output of other options in the case of $i = n - 1$ (facets) but not for $i = 0$ (vertices).

3.2.16 -s

This option refers to a property of (combined) weight systems that was considered in the early stages of the classification program [4]. In the higher dimensional embedding defined by a weight system, the polytope is bounded by the inequalities $X_i \geq -1$. We say that the polytope has the span property if the pullbacks of the equations $X_i = -1$ to the subspace carrying the polytope are spanned by vertices of the polytope, i.e. if these equations correspond to facets. With `-s` a message is given if the (combined) weight systems does not have this property (try, for example, the weight system '8 3 3 2').

3.2.17 -I

There is a message if the polytope does not have the origin of the coordinate system in its interior.

3.2.18 -S

The output contains the following two numbers. The first is the number of lattice automorphisms (elements of $GL(n, \mathbb{Z})$) that leave the polytope invariant; each such automorphism acts as a permutation on the set of vertices. The second one is the number of permutations of the set of vertices that leave the vertex pairing matrix (see section 3.2.8) invariant (after taking into account the induced permutations of facets). This number can also be interpreted as the number of symmetries of the polytope in $\mathbb{R}^n$; it may be larger than the number of symmetries in the given lattice.

```
palp$ poly.x -S
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
5 1 1 1 1 1
#GL(Z,4)-Symmetries=120, #VPM-Symmetries=120
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
5 1 1 1 1 1 /Z5: 0 1 2 3 4
#GL(Z,4)-Symmetries=20, #VPM-Symmetries=120
```

In the first case the symmetry group of the lattice polytope and that of the vertex pairing matrix are just the permutation group of the 5 vertices, of order 120. In the second case the invariance under the $\mathbb{Z}_5$ group fixes any permutation once the permutations of two of the five

vertices have been chosen, reducing the number of group elements to 20. The vertex pairing matrix remains the same, namely diag(5, 5, 5, 5, 5), and therefore keeps all 120 symmetries.

3.2.19 -T

A coordinate change is performed that makes the matrix of coordinates of the points specified in the input upper triangular, with minimal entries above the diagonal. This may be useful for representing the polytope in a specific lattice basis consisting of points of P, or for finding the volume of a specific cone (if the generators of the cone are the first input points, the volume will be the product of the entries in the diagonal after the transformation). Using this option only makes sense if its results are displayed. Therefore it exits with an error if it is not combined with an output generating option (`-v` is a natural choice).

3.2.20 -N

This option leads to the computation of a normal form of the polytope, i.e. a matrix containing the vertices in a specific order in a particular coordinate system, such that this output is the same for any two polytopes related by a lattice automorphism (see section 3.4 of [9] for the actual algorithm). This is useful, for example, if two polytopes are suspected to be isomorphic because they are isomorphic if and only if their normal forms are identical.

Example: the weight systems '3402 40 41 486 1134 1701' and '3486 41 42 498 1162 1743' give rise to the same pair of Hodge numbers (491, 11). The suspicion that they correspond to the same polytope is confirmed by

```
palp$ poly.x -N
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3402 40 41 486 1134 1701
4 5  Normal form of vertices of P    perm=43210
   1   0   0   0 -42
   0   1   0   0 -28
   0   0   1   0 -12
   0   0   0   1  -1
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3486 41 42 498 1162 1743
4 5  Normal form of vertices of P    perm=43210
   1   0   0   0 -42
   0   1   0   0 -28
   0   0   1   0 -12
   0   0   0   1  -1
```

The `perm=43210` part indicates how the vertices of the polytope had to be permuted to arrive at the normal form; this is only interesting if the input is of matrix type.

3.2.21 -t

The calculation of the normal form involves determining the pairing matrix, a normal form for the pairing matrix, an analysis of which symmetries leave this normal form invariant,

a preferred ordering of the vertices and a conversion of the resulting vertex coordinate matrix to upper triangular form. The results of these steps, which may provide further useful information on the structure of P, are displayed.

3.2.22 -V

The IP simplices (see section 2.3.5) whose vertices are also vertices of the dual polytope are displayed. For illustrating examples look at the next options (`-P`, `-Z`) which are closely related.

3.2.23 -P

The IP simplices (see section 2.3.5) whose vertices are lattice points of the dual polytope P^* are displayed. This option should only be used if it is fairly clear that P^* does not have too many lattice points.

```
palp$ poly.x -P
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
6 1 2 3
2 7  points of P-dual and IP-simplices
    1    0   -2   -1    0   -1    0
    0    1   -3   -2   -1   -1    0
-----------------------------    #IP-simp=4
    2    3    1    0    0    0   6=d  codim=0
    1    2    0    1    0    0   4=d  codim=0
    1    1    0    0    0    1   3=d  codim=0
    0    1    0    0    1    0   2=d  codim=1
```

This example shows that the N lattice polytope for $\mathbb{P}^2_{(1,2,3)}$ contains 3 lattice triangles with the origin in their respective interiors, as well as a lattice line segment with that property. The weight systems corresponding to these simplices are indicated.

3.2.24 -Z

This option only has an effect if combined with `-V` or `-P`. Any IP simplex S of dimension d occurring there defines two potentially distinct d dimensional sublattices of N: the lattice generated by the vertices of S, and the sublattice of N lying in the linear subspace that is spanned by S. Then the program computes the corresponding quotient action. The following example illustrates this for the case of a bipyramid over a triangle whose vertices generate an index 3 sublattice of $\mathbb{Z}^2$.

```
palp$ poly.x -VZD
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 5
Type the 15 coordinates as dim=3 lines with #pts=5 columns:
   -1   -1    2    0    0
   -1    2   -1    0    0
    0    0    0    1   -1
3 5  vertices of P-dual and IP-simplices
```

```
   -1   -1    2    0    0
   -1    2   -1    0    0
    0    0    0    1   -1
------------------------   #IP-simp=2 I=3 /Z3: 2 1 0 0 0
    1    1    1    0    0    3=d  codim=1 /Z3: 2 1 0 0 0
    0    0    0    1    1    2=d  codim=2
```

3.2.25 -[numbers]

If one of the options `-B`, `-U` or `-C` is set, any number is assumed to refer to that option (see below for descriptions). Otherwise information on fibration structures of the Calabi–Yau manifold corresponding to a reflexive polytope is displayed. These structures correspond to reflexive subpolytopes of the N lattice polytope P^* that are intersections of the dual polytope with a linear subspace of $N_{\mathbb{R}}$; see, e.g., [6, 7, 28]. As this is a time consuming task and the desired output format may vary, there exist two different versions.

If a single number $\# \in \{1, 2, 3\}$ is specified, the intersections of P^* with all linear subspaces spanned by IP simplices (see section 2.3.5) are checked for reflexivity. This is much faster than a complete search for fibration structure but misses fibrations whose corresponding subspaces are only spanned by a combination of two or more IP simplices. The codimension of the IP simplices is restricted to $1 \leq \text{codim} \leq \#$ (in contrast to our usual policy this option has a side effect on `-P` if combined with that option). The output has the same structure as the output of `nef.x` $-$ `F`$*$. For further details see Section 6.4.12.

If two numbers are specified, all fibration structures of the given type are computed. For `-11`, `-22` and `-33` all reflexive sections of codimension 1, 2 and 3, respectively, are constructed. For `-12` and `-23` all reflexive subpolytopes of codimension 1 and 2 that themselves contain a reflexive subpolytope with relative codimension 1 are constructed. The output is a polytope in the N lattice whose choice of bases and whose order of points reflects the fibration structure. For example, CY threefolds that are both K3 and elliptically fibered are found by applying `poly.x -12` to reflexive 4-polytopes, and fourfolds of that type are found by applying `poly.x -23` to reflexive 5-polytopes.

```
palp$ poly.x -12
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
84 1 1 12 28 42
4 25  Em:7 3 n:7 3  Km:24 4 n:24 4  M:680 5 N:26 5  p=13bgjn256789...
 1  0 -2 -1  0 -1  0 -14 -12 -10 -8 -6 -4 -9 -7 -5 -3 -4 -2 -7 -5 ...
 0  1 -3 -2 -1 -1  0 -21 -18 -15 -12 -9 -6 -14 -11 -8 -5 -7 -4 -10...
 0  0  0  0  0  0  1 -6 -5 -4 -3 -2 -1 -4 -3 -2 -1 -2 -1 -3 -2 -1 ...
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ...
```

Here `Em:7 3 n:7 3` refers to data of the elliptic fiber E lying within the K3 fiber indicated by `Km:24 4 n:24 4`. The 25 non-zero points of the N lattice polytope are displayed in an order (first points from the subspace corresponding to E, then points in the K3 but not in E, finally all other points) and a coordinate system (only the first 2 coordinates non-vanishing for E, last coordinate vanishing for the K3) that reflect the corresponding nesting of polytopes. The sequence after `p=` indicates what permutation with respect to the original order of these points led to this representation (this is only interesting if the points were entered directly with the `-D` option). Further examples on how to use the fibration options can be found in section 4.2 of [20].

3.2.26 -A

Given an arbitrary lattice polytope P, `poly.x -A` computes its 'affine normal form', i.e. a polytope in $\mathbb{Z}^n$ affinely isomorphic to P, such that the normal forms of any two polytopes P and Q coincide if and only if P and Q are related by an affine lattice isomorphism (cf. `-N`, which performs the same task w.r.t. linear rather than affine transformations).

3.2.27 -B

For a given polytope its volume (normalized such that the standard simplex has volume 1) and the coordinates of its barycentre are displayed.

If an integer n is specified after `-B`, the polytope is interpreted as the origin and the first level of a Gorenstein cone (for a discussion see Section 6.2. The points of the cone up to level n are computed and displayed together with information on the type of face of the cone they represent (w.r.t. codimension, i.e. the origin has maximal codimension and points interior to the cone have `cd=0`). As example, we consider the Gorenstein cone with support polytope given by the dual of the Newton polytope of the quintic hypersurface in $\mathbb{P}^4$.

```
palp$ poly.x -B2
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
5 6
Type the 30 coordinates as dim=5 lines with #pts=6 columns:
1  1  1  1  1  0
1  0  0  0 -1  0
0  1  0  0 -1  0
0  0  1  0 -1  0
0  0  0  1 -1  0
vol=5, baricent=(5,0,0,0,0)/6
IPs:
 2 -2 -2 -2 -2  cd=4
 2 -1 -1 -1 0  cd=3
 2 0 0 0 2  cd=4
 2 -1 0 -1 -1  cd=3
 2 0 1 0 1  cd=3
 2 0 2 0 0  cd=4
 2 -1 -1 0 -1  cd=3
 2 0 0 1 1  cd=3
 2 0 1 1 0  cd=3
 2 0 0 2 0  cd=4
 2 -1 -1 -1 -1  cd=0
 2 0 0 0 1  cd=0
 2 0 1 0 0  cd=0
 2 0 0 1 0  cd=0
 2 0 0 0 0  cd=0
 2 0 -1 -1 -1  cd=3
 2 1 0 0 1  cd=3
 2 1 1 0 0  cd=3
 2 1 0 1 0  cd=3
 2 1 0 0 0  cd=0
 2 2 0 0 0  cd=4
 1 -1 -1 -1 -1  cd=4
```

```
1 0 0 0 1  cd=4
1 0 1 0 0  cd=4
1 0 0 1 0  cd=4
1 0 0 0 0  cd=0
1 1 0 0 0  cd=4
0 0 0 0 0  cd=5
```

3.2.28 -F

Each facet of a polytope is displayed by listing its vertices in some basis for the sublattice carrying the facet. The result is not the affine normal form of the facet, however.

3.2.29 -G

If the input polytope P is a non-trivial multiple $P = gQ$ of another lattice polytope Q, the maximal proportionality factor g and P are displayed.

3.2.30 -L

This option results in the same calculations as the `-l` option (see above), but gives a more detailed output including the Witten index.

3.2.31 -U

If[1] `-U` is specified without a number following (otherwise see below), it is computed whether the N lattice polytope has only simplicial facets; if this is not the case no further computations are performed on the polytope and no output results from it.

3.2.32 -U1

Like `-U` without number, but the facets now have to be unimodular (i.e. of volume 1). This corresponds to the case of Fano varieties.

3.2.33 -U5

This option is related to the classification of Fano polytopes (unimodular and simplicial reflexive polytopes) up to dimension five [31]. The corresponding data supplement [32] contains examples of the usage of `-U5`.

3.2.34 -C1

This option was implemented for a search of Calabi-Yau threefolds which are related to previously known ones via conifold transitions [33]. The use of PALP to produce these results is described at [34]. The input should be a 4-dimensional reflexive polytope P such that P^* has only basic triangles or squares as two-dimensional faces. In this case the associated generic CY-hypersurface has isolated conifold singularities. The option `poly.x -C1` checks whether this three-dimensional CY is smoothable.

[1] We are very grateful to Benjamin Nill for providing infomation for the `-U`, `-C` and `-E` options.

3.2.35 -C2

The option `-C2` was used to generate three-dimensional Fano hypersurfaces with conifold singularities [35]. For the corresponding data see [36]. The input must be a reflexive 4-polytope that is divisible by 2, in which case the generic hypersurface associated to $P/2$ is a three-dimensional Fano variety. A list of (hopefully all) such polytopes can be found at Max Kreuzer's site [37]. It is not clear how he obtained this list. Possibly he piped an extraction of the complete database of reflexive 4-polytopes through `poly.x -G` and then used some script to eliminate polytopes that are odd multiples of other polytopes. The program requires the input to be in matrix (rather than weight) format.

Example: `poly.x -C2` with input '8 1 1 1 1 4' crashes, but with the corresponding matrix input one gets

```
palp$ poly.x -C2
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
4 5
Type the 20 coordinates as dim=4 lines with #pts=5 columns:
   -1    7   -1   -1   -1
   -1   -1    7   -1   -1
   -1   -1   -1    7   -1
   -1   -1   -1   -1    1
pic=1  deg=64  h12= 0  rk=0 #sq=0 #dp=0 py=1  F=5 10 10 5 #Fano=1
4 5  Vertices of P* (N-lattice)    M:201 5 N:7 5
 1  0  0  0 -1
 0  0  1  0 -1
 0  1  0  0 -1
 0  0  0  1 -4
P/2: 36 points (5 vertices) of P'=P/2 (M-lattice):
P/2:  0  4  0  0  0   0  0  0  0  0  0  0  0  0  0  0  1  1  1 ...
P/2:  0  0  4  0  0   1  2  3  0  1  2  3  0  1  2  0  1  0  1  2 ...
P/2:  0  0  0  4  0   0  0  0  1  1  1  1  2  2  2  3  3  0  0  0 ...
P/2:  0  0  0  0  1   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ...
```

3.2.36 -E

`poly.x -E` checks several symmetry properties of a reflexive polytope related to the set of roots of the associated toric variety. These are of interest with respect to the existence of Einstein-Kaehler metrics. Here, a root is a lattice point in the interior of a facet of the reflexive polytope (in the M lattice). Centrally-symmetric roots are called semisimple. If all roots are semisimple, then `ssroot=1`. Only in this case PALP proceeds to check and display the following conditions: if the barycenter is 0 (`bary=1`), the sum of lattice points is 0 (#`Psum=1`), the sum of lattice points in each multiple is 0 (#`kPsum=1`), the group of lattice automorphisms has only the origin as a fixed point (#`symm=1`). These conditions can be found explained in Chapter 5 (in particular, sections 5.5 and 5.6) of the dissertation of Benjamin Nill [38]. This option cannot be combined with others; if `-E` is specified, all other options are ignored.

4 cws.x

4.1 General description of cws.x

cws.x is concerned mainly with the first steps of the algorithm of [4, 5, 28] for the classification of reflexive polytopes in a given dimension. In particular it contains an implementation of the algorithm [5] for the classification of weight systems, and routines for combining these weight systems into CWS. As always, rough information can be obtained with the help screen.

```
palp$ cws.x -h
This is 'cws.x': create weight systems and combined weight systems.
Usage:    cws.x -<options>;
          the first option must be 'w', 'c', 'i', 'd' or 'h'.
Options:
-h        print this information
-f        use as filter; otherwise parameters denote I/O files
-w# [L H] make IP weight systems for #-dimensional polytopes.
          For #>4 the lowest and highest degrees L<=H are required.
    -r/-t make reflexive/transversal weight systems (optional).
-c#       make combined weight systems for #-dimensional polytopes.
          For #<=4 all relevant combinations are made by default,
          otherwise the following option is required:
    -n[#] followed by the names wf_1 ... wf_# of weight files
          currently #=2,3 are implemented.
     [-t] followed by # numbers n_i specifies the CWS-type, i.e.
          the numbers n_i of weights to be selected from wf_i.
          Currently all cases with n_i<=2 are implemented.
 -i       compute the polytope data M:p v [F:f] N:p [v] for all IP
          CWS, where p and v denote the numbers of lattice points
          and vertices of a dual pair of IP polytopes; an entry
          F:f and no v for N indicates a non-reflexive 'dual pair'.
 -d#      compute basic IP weight systems for #-dimensional reflexive
          Gorenstein cones;
     -r#  specifies the index as #/2.
 -2       adjoin a weight of 1/2 to the input of the weight system.
 -N       make CWS for PPL in N lattice.
```

There is also a second help screen that can be called with the option -x (see below).

4.2 Options of cws.x

4.2.1 -w[number]

The behaviour of cws.x -w# depends crucially on #.

If # ≤ 4 all weight systems corresponding to #-dimensional IP-simplices are determined by executing the algorithm of [5]:

```
palp$ cws.x -w2
3  1 1 1  rt
4  1 1 2  rt
6  1 2 3  rt  #=3  #cand=3
```

The algorithm determines candidates for weight systems and prints them if they lead to polytopes with the IP property (see section 2.3.5); this holds for all 3 candidates, as `#=3 #cand=3` indicates. If such a weight system gives rise to a reflexive polytope (which is always the case in dimension ≤ 4 [5]) this is indicated by an `r`; if the (possibly singular) weighted projective space corresponding to the weight system obeys the 'transversality condition' that the Calabi–Yau hypersurface equation introduces no additional singularities, this is indicated by a `t`.

If $\# > 4$, one has to enter a lower and an upper bound for the degrees of the weight systems. `cws.x -w#` then examines all possible such systems and displays the ones that define polytopes with the IP property.

If an extra option of `-r` or `-t` is specified, the output contains only the reflexive or transverse weight systems, respectively. Just try `cws.x -w5 5 8`, `cws.x -w5 5 8 -r` and `cws.x -w5 5 8 -t` to see how this works.

4.2.2 -c[number]

Now the output contains combined weight systems (CWS). Again all of them are created if the number after `-c` is ≤ 4 (try `cws.x -c3`). Otherwise weight systems that are read from files are combined. We apologize for not being able to give information beyond the one given in the help screen.

4.2.3 -i

In this case polytope input is required. The output is like that of `poly.x -g`, but suppressed for polytopes without the IP property. This can be useful to filter a list of CWS for the IP property.

4.2.4 -d[number] [-r[number]]

The so-called basic weight systems for reflexive Gorenstein cones of a given dimension (the number after `-d`) and a given index are computed; if `-r` is used, the index is *half* the number after `-r`, otherwise the index is 1 by default. See [39, 40] for more details.

4.2.5 -2

`cws.x -2 < infile > outfile` writes the list of weight systems in `infile` to `outfile`, but with a weight of 1/2 adjoined to each input weight system; this is useful because the `-d`–option produces only weight systems without weights of 1/2.

4.2.6 -N

Given a polytope as matrix input, this option reconstructs the CWS of a reflexive pair (P, P^*) with P^* isomorphic to the input polytope. The option only accepts matrix input, otherwise it returns the message `Only PPL-input in Npoly2cws!`. It is useful to verify whether a polytope is actually defined on a sublattice of finite index. As a simple example consider the polytope consisting of only the vertices of the Newton polytope of the quintic hypersurface in $\mathbb{P}^4$

```
palp$ cws.x -N
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
4 5
Type the 20 coordinates as dim=4 lines with #pts=5 columns:
-1 -1 -1 -1  4
-1 -1 -1  4 -1
-1 -1  4 -1 -1
-1  4 -1 -1 -1
5 1 1 1 1 1 /Z5: 4 1 0 0 0 /Z5: 4 0 1 0 0 /Z5: 4 0 0 1 0
```

This option is in some sense inverse to `poly.x` without any options, see Section 3, and hence should viewed as part of `poly.x` which for historical reasons ended up in `cws.x`.

4.2.7 `-x`

A further help screen with additional options is displayed:

```
palp$ cws.x -x
This is 'cws.x': -x gives undocumented extensions:
              -ip    printf PolyPointList
              -id    printf dual PolyPointList
              -p#    [infile1] [infile2] makes cartesian product
                     of Vertices. # dimensions are identified.
              -S     count simplex points for weight system
              -L     count using LattE (-> count redcheck cdd)
```

As `-x` refers to 'experimental' and none of the authors is familiar with them, we leave it to the reader to play with them and perhaps find useful applications. It would be greatly appreciated if any insights gained in this way were communicated via the PALP Wiki [19].

5 `class.x`

5.1 General description of `class.x`

`class.x` implements the actual creation of the complete lists of reflexive polytopes in dimensions up to 4 (see [9, 10]). It contains routines for determining all subpolytopes of a given polytope in an efficient manner, for finding all lattices on which a given polytope is reflexive, and for several other tasks relevant to the classification. In particular, as the resulting numbers of polytopes tend to get very large, `class.x` can encode a corresponding list in various binary formats. This can take the form of a single binary file or a collection of binary files to which we refer as a database (even though it is not a relational database in the usual sense). Actually there are two types of database: the first type is used in the classification procedure and contains only information on the polytopes themselves, whereas the second type also contains Hodge number information; the latter is accessed by the website [40]. `class.x` contains routines for converting the various formats and for creating set theoretic unions or differences of the corresponding lists.

It is very important to use `class.x` with the appropriate parameter–setting of `POLY_Dmax = 4`. To get a good feeling for how `class.x` functions, we recommend to use it for rederiving the classification of reflexive 3-polytopes following section 3.2 of [20]. The available options are listed in the help screen:

```
palp$ class.x -h
This is  'class.x', a program for classifying reflexive polytopes
Usage:     class.x  [options] [ascii-input-file [ascii-output-file]]
Options:
-h          print this information
-f or -     use as filter; otherwise parameters denote I/O files
-m*         various types of minimality checks (* ... lvra)
-p* NAME    specification of a binary I/O file (* ... ioas)
-d* NAME    specification of a binary I/O database (DB) (* ... ios)
-r          recover: file=po-file.aux, use same pi-file
-o[#]       original lattice [omit up to # points] only
-s*         subpolytopes on various sublattices (* ... vphmbq)
-k          keep some of the vertices
-c, -C      check consistency of binary file or DB
-M[M]       print missing mirrors to ascii-output
-a[2b], -A  create binary file from ascii-input
-b[2a], -B  ascii-output from binary file or DB
-H*         applications related to Hodge number DBs (* ...cstfe)

Type one of [m,p,d,r,o,s,c,M,a,b,H] for help on options,
'g' for general help, 'I' for general information on I/O or 'e'
to exit:
```

As the last lines show, this help screen is interactive, i.e. by typing one of the option letters within the help program one can obtain further information. For this reason, and also because there are probably not too many potential users of this program, we shall be rather brief in the following descriptions.

5.2 Options of `class.x`

In the following we assume that the reader also consults the help text of `class.x -h` followed by the letter for the corresponding option; otherwise our explanations may not make much sense or important details can be missed.

5.2.1 -h

The interactive help screen is displayed (see above).

5.2.2 -f

The filter flag turns off the prompt for input from the screen.

5.2.3 -ml, -mv, -mr, -ma

In [4, 9, 10] several versions of the concept of a minimal polytope were defined (see [28] for a summary). Given polytope input, it is determined whether these definitions are satisfied. This is the only option except `-h` for which both input and output are ascii.

5.2.4 -pi, -pa, -ps, -po

`-p*` specifies the name of a binary file. Depending on the second letter, this may be an input file, possibly such that the list it encodes should be added to or subtracted from some other list, or an output file.

5.2.5 -di, -ds, -do

Like `-p*`, but now the name of a database is specified.

5.2.6 -r

This option was used in the classification of reflexive 4-polytopes to recover intermediate results after computer crashes.

5.2.7 -o, -o[number], -oc

`-o` instructs `class.x` to ignore polytopes that are reflexive only on a sublattice. With a number following, only up to that number of points are omitted in the search for subpolytopes (for an example see section 3.1 of [20]). `-oc` modifies the behaviour of the previous option `-r`.

5.2.8 -sh, -sp, -sv, -sm, -sb, -sq

A given polytope Δ may be reflexive w.r.t. several distinct lattices if the lattice M_{coarsest} generated by the vertices of Δ is not dual to the lattice N_{coarsest} generated by the vertices of Δ^* (see e.g. [28]). `-s*` finds such cases in the database; various versions called by different letters after `-s` correspond to lattices generated by different types of subsets of the set of lattice points of Δ. For example Calabi-Yau hypersurfaces that are free quotients result from sublattices generated by all points except those interior to facets.

5.2.9 -k

With `-k` (for 'keep') one can specify some vertices of an input polytope Δ such that `class.x` finds all reflexive subpolytopes of Δ that still contain these vertices.

5.2.10 -c, -C

These options perform consistency checks on a given file or database. This is very useful if one suspects that something may have gone wrong with the data.

5.2.11 -M

A list given in binary format is checked for mirror symmetry. Those polytopes that would be required to make the list mirror symmetric are displayed.

5.2.12 -a, -A

Ascii input is converted to binary file format. `-a` corresponds to the standard normal form (cf. section 3.2.20) and `-A` to the affine normal form (cf. section 3.2.26).

5.2.13 -b, -B

Binary input is converted to ascii. `-b` is the standard version and `-B` should be used for lists created with `-A`.

5.2.14 -Hc, -Hs, -Ht, -Hf, -He

Options of this type are related to the second type of data base which contains data on Hodge numbers. They work only in dimension 4.

6 nef.x

6.1 General Description of nef.x

`nef.x` is a package to analyze nef partitions of a reflexive polytope. Such nef partitions determine complete intersections of Calabi-Yau type in toric varieties of, in principle, arbitrary codimension. Given a reflexive polytope in terms of a combined weight system (cf. Section 2.1) or a list of points the main objective of the program is to determine the nef partitions and the Hodge numbers of the corresponding Calabi-Yau varieties. Further features include the calculation of the corresponding reflexive Gorenstein cones as well as information about the fibration structure. A short summary of the available options can be obtained from the help screen:

```
palp$ nef.x -h
This is './nef.x':  calculate Hodge numbers of nef-partitions
Usage:   ./nef.x <Options>
Options:
-h        prints this information
-f or -   use as filter; otherwise parameters denote I/O files
-N        input is in N-lattice (default is M)
-H        gives full list of Hodge numbers
-Lv       prints L vector of Vertices (in N-lattice)
-Lp       prints L vector of Points (in N-lattice)
-p        prints only partitions, no Hodge numbers
-D        calculates also direct products
-P        calculates also projections
-t        full time info
-cCODIM   codimension (default = 2)
-Fcodim   fibrations up to codim (default = 2)
-y        prints poly/CWS in M lattice if it has nef-partitions
-S        information about #points calculated in S-Poly
-T        checks Serre-duality
-s        don't remove symmetric nef-partitions
-n        prints polytope only if it has nef-partitions
-v        prints vertices and #points of input polytope in one
          line; with -u, -l the output is limited by #points:
    -uPOINTS  ... upper limit of #points (default = POINT_Nmax)
    -lPOINTS  ... lower limit of #points (default = 0)
-m        starts with [d  w1 w2 ... wk d=d_1 d_2 (Minkowski sum)
-R        prints vertices of input if not reflexive
-V        prints vertices of N-lattice polytope
```

```
-Q        only direct products (up to lattice Quotient)
-gNUMBER  prints points of Gorenstein polytope in N-lattice
-dNUMBER  prints points of Gorenstein polytope in M-lattice
      if NUMBER = 0 ... no            0/1 info
      if NUMBER = 1 ... no redundant  0/1 info (=default)
      if NUMBER = 2 ... full          0/1 info
-G        Gorenstein cone: input <-> support polytope
```

Note that for examples of codimension greater than two calculation times can become very long and parameters specified in the files `Global.h` and `Nef.h` may have to be suitably chosen upon compilation (cf. Section 2.2).

In this section we will begin with a brief reminder of the notion of a nef partition. Then we will describe in detail the standard output of `nef.x` when called without any options. Finally we will discuss each option in detail and demonstrate its functionality in examples. Further examples and details can be found at the PALP Wiki [19].

6.2 Nef partitions and reflexive Gorenstein cones

In this subsection, we give a brief summary of nef ('nef' stands for numerically effective) partitions and related notions. For details, see [41, 42, 11, 43].

Consider a dual pair of d-dimensional reflexive polytopes $\Delta \subset M_{\mathbb{R}}, \Delta^* \subset N_{\mathbb{R}}$. A partition $V = V_0 \cup \cdots \cup V_{r-1}$ of the set of vertices of Δ^* into disjoint subsets $V_0, \ldots, V_{r-1}$ is called a nef partition of length r if there exist r integral upper convex $\Sigma(\Delta^*)$-piecewise linear support functions $\phi_l : N_{\mathbb{R}} \to \mathbb{R}$, $l = 0, \ldots, r-1$ such that

$$\phi_l(v) = \begin{cases} 1 & \text{if } v \in V_l, \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

Each ϕ_l corresponds to a divisor $D_{0,l} = \sum_{v \in V_l} D_v$ on the toric variety $\mathbb{P}_{\Delta^*}$ associated to Δ^*, and the intersection of all these divisors

$$X = D_{0,0} \cap \cdots \cap D_{0,r-1} \tag{6.2}$$

defines a family $X \subset \mathbb{P}_{\Delta^*}$ of Calabi-Yau complete intersections of codimension r.

Moreover, each ϕ_l corresponds to a lattice polytop Δ_l defined as

$$\Delta_l = \left\{ x \in M_{\mathbb{R}} \,\middle|\, (x,y) \geq -\phi_l(y) \quad \forall y \in N_{\mathbb{R}} \right\}.$$

The sum of the functions ϕ_l is equal to the support function of the anticanonical divisor $K^{-1}_{\mathbb{P}_{\Delta^*}}$ and, therefore, the corresponding Minkowski sum is $\Delta_0 + \cdots + \Delta_{r-1} = \Delta$. Moreover, the knowledge of the decomposition $V = V_0 \cup \cdots \cup V_{r-1}$ is equivalent to that of the set of supporting polytopes $\Pi(\Delta) = \{\Delta_0, \ldots, \Delta_{r-1}\}$, and therefore this data is often also called a nef partition.

For a given nef partition $\Pi(\Delta)$ the polytopes[2]

$$\nabla_{l'} = \left\langle \{0\} \cup V_{l'} \right\rangle \subset N_{\mathbb{R}}$$

define again a nef partition $\Pi^*(\nabla) = \{\nabla_0, \ldots, \nabla_{r-1}\}$ such that the Minkowski sum $\nabla = \nabla_0 + \cdots + \nabla_{r-1}$ is a reflexive polytope. Then, its dual ∇^* is also reflexive, and $\Pi^*(\nabla)$ is called

[2]The brackets $\langle \cdots \rangle$ denote the convex hull.

the dual nef partition. This is the combinatorial manifestation of mirror symmetry in terms of dual pairs of nef partitions of Δ^* and ∇^*, which we summarize in the following diagram

$$\begin{array}{ccc} M_{\mathbb{R}} & & N_{\mathbb{R}} \\ \Delta = \Delta_0 + \ldots + \Delta_{r-1} & & \Delta^* = \langle \nabla_0, \ldots, \nabla_{r-1} \rangle \\ & (\Delta_l, \nabla_{l'}) \geq -\delta_{l\,l'} & \\ \nabla^* = \langle \Delta_0, \ldots, \Delta_{r-1} \rangle & & \nabla = \nabla_0 + \ldots + \nabla_{r-1} \end{array} \tag{6.3}$$

In the horizontal direction, we have the duality between the lattices M and N and mirror symmetry goes from the upper right to the lower left. The complete intersections $X \subset \mathbb{P}_{\Delta^*}$ and $\check{X} \subset \mathbb{P}_{\nabla^*}$ associated to the dual nef partitions are then mirror Calabi-Yau varieties.

There are two constructions to build new nef partitions from old ones: projections and direct products. Given a nef partition $V = V_0 \cup \cdots \cup V_{r-1}$ where one of the subsets V_l, say V_0, consists of a single vertex v, the nef condition implies that the projection Δ^*_v of Δ^* along v is reflexive. Moreover, by (6.2) the Calabi–Yau complete intersection X is given by $D \cap X'$ with $X' = \bigcap_{l=1}^{r-1} D_{0,l}$. Since D can only intersect the toric divisors that correspond to points bounding the reflexive projection along v, the variety X is isomorphic to the variety $X' \subset \mathbb{P}_{\Delta^*_v}$, where $\mathbb{P}_{\Delta^*_v}$ is obtained from the projection Δ^*_v. In [12] such nef partitions were called trivial. In `nef.x` they are labeled by `P` for projection, see Section 6.4.9.

Suppose we are given two lattices $M^{(1)}, M^{(2)}$ and two reflexive polytopes $\Delta^{(1)} \subset M^{(1)}_{\mathbb{R}}$, $\Delta^{(2)} \subset M^{(2)}_{\mathbb{R}}$ such that $(\Delta^{(1)})^*$ and $(\Delta^{(2)})^*$ admit nef partitions $V^{(1)} = \bigcup_l V^{(1)}_l$ and $V^{(2)} = \bigcup_l V^{(2)}_l$, respectively. Then $\Delta = \Delta^{(1)} \times \Delta^{(2)}$ is reflexive with respect to $M = M^{(1)} \oplus M^{(2)}$ and dual to Δ^* whose set of vertices V is $\{(v^{(1)}, 0) \,|\, v^{(1)} \in V^{(1)}\} \cup \{(0, v^{(2)}) \,|\, v^{(2)} \in V^{(2)}\}$. V admits a nef partition induced from the nef partitions $V^{(1)}$ and $V^{(2)}$. Such a nef partition is called a direct product since the corresponding Calabi–Yau complete intersection X is a direct product $X = X^{(1)} \times X^{(2)}$ in $\mathbb{P}_{\Delta^*} = \mathbb{P}_{(\Delta^{(1)})^*} \times \mathbb{P}_{(\Delta^{(2)})^*}$.

One can reformulate the duality of nef partitions in terms of reflexive Gorenstein cones as follows. We extend the lattices M and N to $\widetilde{M} = \mathbb{Z}^r \oplus M$ and $\widetilde{N} = \mathbb{Z}^r \oplus N$ and set $\widetilde{d} = d + r$.

A $\widetilde{d}$–dimensional rational polyhedral cone C in $\widetilde{M}_{\mathbb{R}}$ is called Gorenstein if $C \cap (-C) = \{0\}$, there exists an element $n_C \in \widetilde{N}_{\mathbb{R}}$ such that $\langle x, n_C \rangle > 0$ for any nonzero $x \in C$, and all vertices of the $(\widetilde{d} - 1)$–dimensional convex polytope

$$\Delta(C) = \{x \in C \,|\, \langle x, n_C \rangle = 1\}$$

belong to $\widetilde{M}$. The polytope $\Delta(C)$ is called the support of C. Conversely, any $(\widetilde{d} - 1)$–dimensional lattice polytope Λ determines a $\widetilde{d}$–dimensional Gorenstein cone $C(\Lambda)$ as the cone over Λ with apex at lattice distance 1 from the hyperplane carrying Λ; obviously $\Delta(C(\Lambda)) = \Lambda$. For any $m \in C \cap \widetilde{M}$, we define the degree of m as $\deg m = \langle m, n_C \rangle$.

A Gorenstein cone C is called reflexive if the dual cone

$$\check{C} = \{y \in \widetilde{N}_{\mathbb{R}} \,|\, \langle x, y \rangle \geq 0 \; \forall\, x \in C\}$$

is also Gorenstein, i.e., there exists $m_{\check{C}} \in \widetilde{M}$ such that $\langle m_{\check{C}}, y \rangle > 0$ for all $y \in \check{C} \setminus \{0\}$, and all vertices of the support $\Delta(\check{C}) = \{y \in \check{C} \,|\, \langle m_{\check{C}}, y \rangle = 1\}$ belong to $\widetilde{N}$. We will call the integer $r = \langle m_{\check{C}}, n_C \rangle$ the index of C (or $\check{C}$).

Any nef partition $\Pi(\Delta) = \{\Delta_0, \ldots, \Delta_{r-1}\}$ of length r of a reflexive polytope Δ determines a $\widetilde{d}$–dimensional dual pair of reflexive Gorenstein cones $C = C(\Delta_1, \ldots, \Delta_r) \subset \widetilde{M}_{\mathbb{R}}$, $\check{C} = \check{C}(\nabla_1, \ldots, \nabla_r) \subset \widetilde{N}_{\mathbb{R}}$ of index r by

$$C = \{(\lambda_1, \ldots, \lambda_r, \lambda_1 x_1 + \cdots + \lambda_r x_r) \in \widetilde{M}_{\mathbb{R}} \,|\, \lambda_i \geq 0, x_i \in \Delta_i, i = 1, \ldots, r\},$$
$$\check{C} = \{(\mu_1, \ldots, \mu_r, \mu_1 x_1 + \cdots + \mu_r x_r) \in \widetilde{N}_{\mathbb{R}} \,|\, \mu_i \geq 0, x_i \in \nabla_i, i = 1, \ldots, r\}.$$

There are, however, reflexive Gorenstein cones that do not come from nef partitions.

A reflexive Gorenstein cone admits a representation in terms of the points of the underlying reflexive polytope as follows. Given a point $p \in \nabla_l$, the corresponding point $\widetilde{p} \in \check{C}(\nabla_1, \ldots, \nabla_r)$ is given as

$$\widetilde{p} = (\phi_0(p), \ldots, \phi_{r-1}(p), p). \tag{6.4}$$

where ϕ_l is the support function (6.1). To see that the two descriptions of $\check{C}$ are equivalent, note that both correspond to a cone whose support has vertices

$$(e_{i(1)}, v_1), \ldots, (e_{i(n)}, v_n), (e_1, 0_N), \ldots, (e_r, 0_N), \tag{6.5}$$

where $\{e_i\}$ is the standard basis of $\mathbb{Z}^r$, $i(k)$ is the number such that $v_k \in V^{(i(k))}$ and 0_N is the origin in the N–lattice.

The Hodge numbers of a Calabi–Yau manifold X defined by means of a nef partition as in (6.2) depend only on the structure of the corresponding reflexive Gorenstein cone in a manner described in [44, 11]. The corresponding formulas rely heavily on the counting of lattice points. For any lattice polytope Λ let us denote by $\ell(\Lambda)$ the number of lattice points of Λ and by $\ell^*(\Lambda)$ the number of lattice points in the interior of Λ. It can be shown that

$$S_\Lambda(t) = (1-t)^{\dim \Lambda + 1} \sum_{k \geq 0} \ell(k\Lambda) t^k \tag{6.6}$$

is a polynomial of degree $d \leq \dim \Lambda + 1$; $S_\Lambda(t)$ is called the Ehrhart polynomial of Λ. Similarly one can define a polynomial

$$T_\Lambda(t) = (1-t)^{\dim \Lambda + 1} \sum_{k \geq 0} \ell^*(k\Lambda) t^k. \tag{6.7}$$

In terms of a Gorenstein cone C over Λ, with underlying lattice M_C, S and T can be written as

$$S(C,t) = (1-t)^{\dim C} \sum_{m \in C \cap M_C} t^{\deg m}, \tag{6.8}$$

$$T(C,t) = (1-t)^{\dim C} \sum_{m \in \mathrm{int}(C) \cap M_C} t^{\deg m}. \tag{6.9}$$

The two polynomials satisfy a relation which is a consequence of Serre duality,

$$S(C,t) = t^{\dim C}\, T(C, t^{-1}), \tag{6.10}$$

which provides a stringent test on any results involving lattice point counting. For the computation of Hodge numbers, the S– and T– polynomials for all the faces of $C(\Delta)$ as well as a polynomial called B, which is related to the poset structure of $C(\Delta)$, are required.

6.3 Standard output

In this subsection we will explain in detail how to interpret the output of `nef.x` when called without any options.

The standard output slightly depends on whether the reflexive polytope is input as a combined weight system or as a collection of points. If the polytope was entered as a collection of points, the first line of the output takes the following form:

```
M:# # N:# # codim=# #part=#
```

Note that the input polytope is interpreted as $\Delta \subset M_{\mathbb{R}}$ unless the option `-N` (cf. Section 6.4.3) is used, while any output of a polytope in matrix format refers to its dual $\Delta^* \subset N_{\mathbb{R}}$ except for the option `-y` (cf. Section 6.4.13. If the input is a CWS, the line starts with the CWS repeated before the letter `M`.

```
# M:# # N:# # codim=# #part=#
```

where the first `#` stands for the sequence of numbers describing the CWS. The two numbers `#` after `M` correspond to the numbers of lattice points and vertices of $\Delta \subset M_{\mathbb{R}}$ and the numbers `#` after `N` correspond to the numbers of lattice points and vertices of $\Delta^* \subset N_{\mathbb{R}}$, respectively. The number r in `codim=r` is the length of the nef partition, i.e. the codimension of the corresponding Calabi–Yau complete intersection. The default value is 2, otherwise it is specified by the option `-c*` described in Section 6.4.11. The number n in `#part=n` is the number of all the nef partitions that `nef.x` has found, up to symmetries of the underlying lattice. If the symmetries of the underlying lattice should not be taken into account, use the option `-s` (cf. Section 6.4.16).

The subsequent lines contain the information about the various nef partitions. Note that the standard output suppresses the output of nef partitions which are equivalent under symmetries of the CWS. If the codimension is 2 the output line containing the information on a particular nef partition takes the following form:

```
H:# [#] P:# V:#  #      #sec #cpu
```

The numbers `#` after `H:` are the Hodge numbers $h^{1,i}(X)$, $i = 1, \ldots, d-1$, where d is the dimension of the Calabi-Yau manifold X obtained via (6.2).

The number `#` in the square brackets `[#]` is the Euler number of X. If $h^{0,i}(X) \neq 0$ for some $i = 1, \ldots, d-1$ the Calabi-Yau manifold factorizes. See the option `-D` (Section 6.4.8) for this case. In any case, the full Hodge diamond is displayed with the option `-H` (Section 6.4.4).

The number `#` after `P:` is a counter specifying the nef partition. It runs from 0 to $n-1$. Note that nef partitions corresponding to direct products and projections to nef partitions of lower length are omitted by default. To display them use the options `-D` (cf. Section 6.4.8), `-Q` (cf. Section 6.4.22) for direct products and `-P` (cf. Section 6.4.9) for projections.

The sequence of numbers `#` separated by a single space after `V:` corresponds to the vertices that belong to the first part V_0 of the nef partition. Note that the vertices are counted starting from 0. These numbers only make sense if the options `-n` (cf. Section 6.4.17), `-Lv` (cf. Section 6.4.5) or `-Lp` (cf. Section 6.4.6) are used. The vertices that belong to the second part V_1 of the nef partition are not displayed, since they are simply the remaining ones. If the polytope entered also has points that are not vertices and if the option `-Lp` is used, then the second sequence of numbers `#` that is separated from the first sequence by two spaces corresponds to

the non-vertex points that belong to the first part V_0. For representations of the nef partition in terms of the Gorenstein cone see the option `-g*` (cf. Section 6.4.23).

The number `#` before `sec` indicates the time that was needed to compute this partition. The number `#` before `cpu` indicates the number of CPU seconds that were needed to compute the Hodge numbers. For determining the nef partitions without computing the Hodge numbers see the option `-p` (cf. Section 6.4.7).

If the length r is bigger than 2 the lines containing the information about the various nef partitions take the following form:

```
H:# [#] P:# V0:#  # V1:#  # ... V(r-2):#  #      #sec #cpu
```

Now, there are $r-1$ expressions of the form `Vi:# #`, where i runs from 0 to $r-2$ each representing a part V_i of the nef partition. The points and vertices in each V_i are listed in the same order as in the codimension two case.

The final line of the output always takes the following form:

```
np=# d:# p:#    #sec     #cpu
```

The numbers `#` after `d:`, `p:`, `np=` are the numbers of nef partitions which are direct products, projections, and neither of the two, respectively. The total of the three numbers adds up to n, the total number of nef partitions as indicated in the first line after `#part=`.

The following example illustrates the standard output of `nef.x`. We consider complete intersections of codimension 2 in $\mathbb{P}^2 \times \mathbb{P}^1 \times \mathbb{P}^2$ discussed in [45]. Let $e_1, \ldots, e_5$ be the standard basis of $\mathbb{R}^5$. We define the polytope $\Delta^* \subset N$ by the vertices $v_0, \ldots, v_7$ given by

$$\begin{aligned} &v_0 = e_1, \quad v_1 = e_2, \quad v_2 = -e_1 - e_2, \quad v_3 = e_3, \\ &v_4 = -e_3, \quad v_5 = e_4, \quad v_6 = e_5, \qquad\quad v_7 = -e_4 - e_5. \end{aligned}$$

By elementary toric geometry, we see that $\mathbb{P}_{\Delta^*} = \mathbb{P}^2 \times \mathbb{P}^1 \times \mathbb{P}^2$ and the combined weight system can be read off from the linear relations

$$v_0 + v_1 + v_2 = 0, \quad v_3 + v_4 = 0, \quad v_5 + v_6 + v_7 = 0.$$

First, we enter the polytope by giving this combined weight system

```
palp$ nef.x
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #colums' (= 'PolyDim #Points' or '#Points PolyDim'):
3 1 1 1 0 0 0 0 0  2 0 0 0 1 1 0 0 0  3 0 0 0 0 0 1 1 1
3 1 1 1 0 0 0 0 0  2 0 0 0 1 1 0 0 0  3 0 0 0 0 0 1 1 1
M:300 18 N:9 8  codim=2 #part=15
H:19 19 [0] P:0 V:2 4 6 7       1sec  0cpu
H:9 27 [-36] P:2 V:3 4 6 7       1sec  0cpu
H:3 51 [-96] P:3 V:3 5 6 7       1sec  1cpu
H:3 75 [-144] P:4 V:3 6 7       1sec  0cpu
H:3 51 [-96] P:6 V:4 5 6 7       2sec  1cpu
H:3 51 [-96] P:7 V:4 5 6       1sec  1cpu
H:6 51 [-90] P:8 V:4 6 7       1sec  1cpu
H:3 75 [-144] P:9 V:4 6       1sec  1cpu
H:3 60 [-114] P:10 V:5 6 7       2sec  1cpu
H:3 69 [-132] P:11 V:5 6       1sec  1cpu
H:3 75 [-144] P:12 V:6 7       1sec  0cpu
np=11 d:2 p:2    0sec     0cpu
```

Equivalently, we can use the option `-N` and enter the points of the polytope Δ^* of the normal fan of $\mathbb{P}^2 \times \mathbb{P}^1 \times \mathbb{P}^2$:

```
palp$ nef.x -N
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #colums' (= 'PolyDim #Points' or '#Points PolyDim'):
5 8
Type the 40 coordinates as dim=5 lines with #pts=8 colums:
 1  0 -1  0  0  0  0  0
 0  1 -1  0  0  0  0  0
 0  0  0  1 -1  0  0  0
 0  0  0  0  0  1  0 -1
 0  0  0  0  0  0  1 -1
M:300 18 N:9 8  codim=2 #part=15
H:3 51 [-96] P:0 V:2 3 4 7       1sec  1cpu
H:3 51 [-96] P:1 V:2 4 6 7       2sec  1cpu
H:3 60 [-114] P:2 V:2 4 7       2sec  1cpu
H:3 51 [-96] P:3 V:2 6 7       1sec  1cpu
H:3 69 [-132] P:4 V:2 7       1sec  1cpu
H:9 27 [-36] P:5 V:3 4 6 7       1sec  0cpu
H:3 75 [-144] P:6 V:3 4 7       0sec  0cpu
H:19 19 [0] P:8 V:4 5 6 7       1sec  0cpu
H:6 51 [-90] P:9 V:4 6 7       1sec  1cpu
H:3 75 [-144] P:10 V:4 7       1sec  0cpu
H:3 75 [-144] P:13 V:6 7       1sec  1cpu
np=11 d:2 p:2    0sec     0cpu
```

Note that both the points and the nef partitions are given in different orders. The polytope $\Delta^* \subset N_\mathbb{R}$ has 9 points, 8 vertices and the interior point, while the dual polytope $\Delta \subset M_\mathbb{R}$ has 300 points, 18 of which are vertices. The codimension is 2 and there are 15 nef partitions. There are 11 nef partitions listed, furthermore there are 2 nef partitions which are direct products, and 2 which are projections. According to the output the nef partitions e.g. 0 and 8 are given as follows (with the Hodge numbers and the Euler number of the corresponding Calabi-Yau 3-fold X):

$$
\begin{aligned}
&0: V_0 = \langle v_2, v_3, v_4, v_7 \rangle, \quad V_1 = \langle v_0, v_1, v_5, v_6 \rangle \\
&\quad h^{1,1}(X) = 3,\ h^{2,1}(X) = 51,\ \chi(X) = -96. \\
&\ldots \\
&8: V_0 = \langle v_4, v_5, v_6, v_7 \rangle, \quad V_1 = \langle v_0, v_1, v_2, v_3 \rangle, \\
&\quad h^{1,1}(X) = 19,\ h^{2,1}(X) = 19,\ \chi(X) = 0. \\
&\ldots
\end{aligned}
$$

6.4 Options of `nef.x`

In this subsection we will explain all the options of `nef.x` in the order of their appearance in the help screen. Here is a rough guide in terms of specific topics:

- Polytope structure: `-N`, `-Lv`, `-Lp`, `-v`, `-R`, `-V`
- Input control: `-N`, `-c*`, `-m`

- Structure of nef partitions: `-D`, `-p`, `-P`, `-s`, `-m`
- Hodge numbers: `-H`, `-t`, `-S`, `-T`
- CWS: `-N`, `-Lv`, `-Lp`, `-m`
- Fibrations: `-F*`
- Gorenstein cone: `-g*`, `-d*`, `-S`, `-T`, `-G`
- Diagnostics: `-t`, `-S`, `-T`
- Polytope statistics: `-y`, `-n`, `-v`, `-R`

6.4.1 -h

This option prints the help screen.

6.4.2 -f or -

This option switches off the prompt for the input data. This is useful for building pipelines.

6.4.3 -N

The option `-N` interprets the input polytope in the N-lattice instead of the M-lattice. The following example of a complete intersection of degree $(2, 2)$ in $\mathbb{P}^3$ illustrates the difference. In order to point out the difference we display the points in the two lattices with the option `-Lv`.

```
palp$ nef.x -Lv
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #colums' (= 'PolyDim #Points' or '#Points PolyDim'):
3 4
Type the 12 coordinates as dim=3 lines with #pts=4 colums:
   -1    0    0    1
   -1    0    1    0
   -1    1    0    0
M:5 4 N:35 4  codim=2 #part=0
3 4 Vertices in N-lattice:
   -1   -1   -1    3
   -1   -1    3   -1
   -1    3   -1   -1
--------------------
    1    1    1    1  d=4  codim=0
np=0 d:0 p:0    0sec     0cpu
```

Without the option `-N`, the output polytope with 35 points and no nef partition is the dual of the input polytope.

```
palp$ nef.x -Lv -N
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #colums' (= 'PolyDim #Points' or '#Points PolyDim'):
3 4
```

```
Type the 12 coordinates as dim=3 lines with #pts=4 colums:
   -1    0    0    1
   -1    0    1    0
   -1    1    0    0
M:35 4 N:5 4  codim=2 #part=2
3 4 Vertices in N-lattice:
   -1    0    0    1
   -1    0    1    0
   -1    1    0    0
--------------------
    1    1    1    1  d=4  codim=0
H:[0] P:0 V:2 3   (2 2)     0sec  0cpu
np=1 d:0 p:1    0sec     0cpu
```

With the option `-N`, the output polytope is the same as input polytope with 4 points and the expected nef partition corresponding to the complete intersection of degree $(2, 2)$ in $\mathbb{P}^3$. Note that the order of the points in the output is the same as in the input. This last feature is the main advantage of the option `-N`. The reason is that the basis chosen does not respect the order given by the combined weight system that was entered. This can be extremely inconvenient at times. The option `-N` provides a way to work around this issue: first use the option `-Lv` to obtain the vertices for a given CWS. Then reorder them so that the basis of linear relations complies with the input and enter the reshuffled vertices into `nef.x` using the option `-N`. This will force the linear relations chosen by `nef.x` to be the same as the CWS.

6.4.4 -H

The option `-H` replaces the output lines starting with `H:` with the full Hodge diamond of the corresponding partition. Note that the information about the nef partitions is omitted. The following example of codimension 2 complete intersections in $\mathbb{P}^7$ illustrates this option (increase `POLY_Dmax` to 7):

```
palp$ nef.x -H
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
7 1 1 1 1 1 1 1
7 1 1 1 1 1 1 1 M:1716 7 N:8 7  codim=2 #part=3


                          h 0 0
                      h 1 0     h 0 1
                  h 2 0     h 1 1     h 0 2
              h 3 0     h 2 1     h 1 2     h 0 3
          h 4 0     h 3 1     h 2 2     h 1 3     h 0 4
              h 4 1     h 3 2     h 2 3     h 1 4
                  h 4 2     h 3 3     h 2 4
                      h 4 3     h 3 4
                          h 4 4


                            1
                        0       0
                    0       1       0
```

```
                0         0         0         0
          1       237       996       237         1
                0         0         0         0
                     0         1         0
                          0         0
                               1
     16sec  15cpu
[analogous output for a second nef partition]
```

6.4.5 -Lv

The option `-Lv` prints the vertices of $\Delta^* \subset N_{\mathbb{R}}$ and the non-negative linear relations among them in addition to the standard output. If only the vertices should be printed use the option `-V` in Section 6.4.21. The output takes the following form: The part before the dashed line reads:

```
 D n Vertices in N-lattice:
    #    #      ...      #    #
    .    .      ...      .    .
    .    .      ...      .    .
    #    #      ...      #    #
```

The first line means that Δ^* has dimension D and is given by n vertices which are the columns of the subsequent $D \times n$ array of numbers `#`.

Below the dashed line the non-negative linear relations among these vertices are indicated as follows: Let $v_0, \ldots, v_{n-1}$ denote the n vertices corresponding to the n columns above the dashed line. Any IP simplex (cf. Section 2.3.5) with vertices in $\{v_0, \ldots, v_{n-1}\}$ determines a linear relation $\sum_{i=0}^{n-1} l_i\, v_i = 0$, with l_i that are positive for the vertices of the IP simplex and 0 otherwise. It results in an output line of the form

```
l_0 l_1 ... l_{n-1} d=l codim=c
```

where $l = \sum_{i=0}^{n-1} l_i$ is the degree of the linear relation and c is the codimension of the IP simplex. In other words, these lines give the set of generators of the cone of non-negative linear relations within the $(n - D)$–dimensional vector space of linear relations among the vertices. This set is completely fixed by the order of the vertices, and the conditions that each vector, i.e. each linear relation, is positive and primitive.

The information contained in these lines can be very useful in conjunction with the option `-F*` (cf. Section 6.4.12). To suppress them see the option `-V` (cf. Section 6.4.21).

Besides the standard output the degrees of the nef partition with respect to the linear relations are inserted in the output lines containing the information about the nef partitions as follows. Consider a nef partition of length r defined by r collections of vertices $V_0, \ldots, V_{r-1}$ such that every vertex is a member of some collection V_j. The (multi)degree of the nef partition $\{V_0, \ldots, V_{r-1}\}$ with respect to the linear relation $\sum_{i=0}^{n-1} l_i\, v_i = 0$ is the vector $(d_0, \ldots, d_{r-1})$ where $d_j = \sum_{i:v_i \in V_j} l_i$. Note that $\sum_{j=1}^{r} d_j = l$, the degree of the linear relation. The degrees $(d_0, \ldots, d_{r-1})$ are the degrees of the polynomials defining the complete intersection. If the codimension is 2 the output lines describing the nef partitions take the following form

```
H:# [#] P:# V:#  #   (d10 d11) ... (dn0 dn1)    #sec #cpu
```

or if the codimension r is bigger than 2

```
H:# [#] P:# V0:#  # V1:#  # ... V(r-2):#  #
      (d10 ... d1(r-1)) ... (dn0 ... dn(r-1)) #sec #cpu
```

The additional data is `(d10 d11) ... (dn0 dn1)` and `(d10 ... d1(r-1)) ... (dn0 ... dn(r-1))`, respectively, where n is the number of linear relations. If $d_i = (d_{i0}, \ldots, d_{i,r-1})$ are the degrees with respect to the i-th linear relation, then `di0` $= d_{i0}, \ldots,$ `di(r-1)` $= d_{i,r-1}$. The following example of a codimension 2 complete intersection taken from [12] illustrates this option:

```
palp$ nef.x -Lv
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #colums' (= 'PolyDim #Points' or '#Points PolyDim'):
5 1 1 1 1 1 0 0  4 0 0 0 1 1 1 1
5 1 1 1 1 1 0 0  4 0 0 0 1 1 1 1 M:378 12 N:8 7  codim=2 #part=8
5 7 Vertices in N-lattice:
    0   -1    0    1    0    0    0
    0   -1    1    0    0    0    0
   -1    0    0    0    0    0    1
   -1    1    0    0    1    0    0
   -1    1    0    0    0    1    0
-------------------------------------
    1    1    1    1    0    0    1  d=5  codim=1
    1    0    0    0    1    1    1  d=4  codim=2
H:2 64 [-124] P:0 V:0 6   (2 3) (2 2)     1sec  0cpu
[standard output for the remaining Hodge data and nef partitions]
```

From the output we deduce that the 7 vertices of the 5-dimensional polytope satisfy the following linear relations:

$$v_0 + v_1 + v_2 + v_3 + v_6 = 0, \quad v_0 + v_4 + v_5 + v_6 = 0.$$

The first of these linear relations has degree 5, the second has degree 4. The corresponding IP simplices have codimension 1 and 2, respectively.

6.4.6 -Lp

The option `-Lp` prints all the points of the N-lattice polytope and the linear relations among them, including those that are not vertices. The output has the same structure as for the option `-Lv`. The points are ordered such that first the vertices $\{v_0, \ldots, v_k\}$ are listed, then the points $\{p_{k+1}, \ldots, p_{N-2}\}$ which are not vertices and finally the origin p_{N-1}. Note that there will be additional linear relations including the points which are neither vertices nor the origin. The following example of a codimension 2 complete intersection taken from [12] illustrates this option:

```
palp$ nef.x -Lp
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
5 1 1 1 1 1 0 0  10 2 2 2 2 2 0 1 1
5 1 1 1 1 1 0 0  10 2 2 2 2 2 0 1 1 M:378 6 N:8 6  codim=2 #part=4
5 8  Points of Poly in N-Lattice:
```

```
   -1    0    0    0    1    0    0    0
   -1    0    1    0    0    0    0    0
   -1    0    0    1    0    0    0    0
   -1    2    0    0    0    0    1    0
   -1    1    0    0    0    1    1    0
----------------------------------------
    2    1    2    2    2    1    0  d=10  codim=0
    1    0    1    1    1    0    1  d=5  codim=1
H:2 86 [-168] P:0 V:1 5  6   (2 8) (1 4)     2sec  2cpu
H:2 68 [-132] P:1 V:2 3 4   (6 4) (3 2)     1sec  0cpu
H:2 68 [-132] P:2 V:3 4   (4 6) (2 3)     1sec  0cpu
np=3 d:0 p:1    0sec     0cpu
```

The last two points are not vertices. There is one more linear relation including the point p_6.

6.4.7 -p

The option `-p` computes the nef partitions without the (time-consuming) calculation of Hodge numbers. As an example we consider the codimension 4 (cf. Section 6.4.11) complete intersections in $\mathbb{P}^7$. Note that one must set `POLY_Dmax` in `Global.h` to at least 10.

```
palp$ nef.x -c4 -p
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
8 1 1 1 1 1 1 1
8 1 1 1 1 1 1 1 1 M:6435 8 N:9 8  codim=4 #part=5
 P:0 V0:2 3  V1:4 5  V2:6 7       0sec  0cpu
np=1 d:0 p:4    0sec     0cpu
```

The Hodge data in the line containing the partition information is omitted, and the computation time is 0. Without the option `-p` this line would look like this:

```
H:1 65 [-128] P:0 V0:2 3  V1:4 5  V2:6 7       13127sec  13120cpu
```

Note the computation time.

6.4.8 -D

The option `-D` also prints those nef partitions which are direct products of lower-dimensional nef partitions. If only direct products are to be printed use the option `-Q` described in Section 6.4.22. As an example we consider a codimension 2 complete intersection in $\mathbb{P}^2 \times \mathbb{P}^2$:

```
palp$ nef.x -D
3 1 1 1 0 0 0 3 0 0 0 1 1 1
3 1 1 1 0 0 0  3 0 0 0 1 1 1 M:100 9 N:7 6  codim=2 #part=5
H:4 [0] h1=2 P:0 V:2 3 5   D     0sec  0cpu
H:20 [24] P:1 V:3 4 5       0sec  0cpu
H:20 [24] P:2 V:3 5       0sec  0cpu
H:20 [24] P:3 V:4 5       0sec  0cpu
np=3 d:1 p:1    0sec     0cpu
```

The last three nef partitions each describe a K3 surface. The first one is a $T^4 = T^2 \times T^2$. The extra output triggered by `-D` is:

```
H:4 [0] h1=2 P:0 V:2 3 5   D     0sec  0cpu
```

`h1=2` indicates that the Hodge number $h^{1,0}(T^4) = 2$. Furthermore the letter `D` indicates that the nef partition is a direct product.

6.4.9 -P

The option `-P` also prints nef partitions corresponding to projections. Consider for example a complete intersection of codimension 2 in $\mathbb{P}^3$:

```
palp$ nef.x -P
4 1 1 1 1
4 1 1 1 1 M:35 4 N:5 4  codim=2 #part=2
H:[0] P:0 V:2 3       0sec  0cpu
H:[0] P:1 V:3       0sec  0cpu
np=1 d:0 p:1    0sec     0cpu
```

Compared to the output without `-P` there is one additional line:

```
H:[0] P:1 V:3       0sec  0cpu
```

Let $v_0, \ldots, v_3$ denote the vertices of the polytope. The nef partition `P:0` is then as follows:

$$0: \; V_0 = \langle v_3 \rangle, \quad V_1 = \langle v_0, v_1, v_2 \rangle. \tag{6.11}$$

The part V_0 only contains the vertex v_3. Therefore the equation of the corresponding divisor $D_{0,0}$ in (6.2) reads $x_3 = 0$. The projection π of Δ^* along v_3 yields a reflexive polytope $\Delta^*_{v_3} = \langle \pi v_0, \pi v_1, \pi v_2 \rangle$. Thus, we are left with a hypersurface $X' = D_{0,1} \subset \mathbb{P}^2 = \mathbb{P}^3 \cap D_{0,0}$. If there is a nef partition such that the dual nef partition in the M-lattice has a summand with only one vertex, then `DP` is displayed in the output[3].

6.4.10 -t

The option `-t` gives detailed information about the calculation times of the Hodge numbers. The Hodge numbers of a Calabi–Yau complete intersection are generated by the so called stringy E-function introduced by Batyrev and Borisov in [44]. The combinatorial construction of the E-function involves the construction of a B-polynomial and an S-polynomial defined in [44]. The option `-t` returns the accumulated computing times of the respective polynomials. We illustrate this option with the example of complete intersections of codimension 4 in $\mathbb{P}^7$ (cf. Section 6.4.7).

```
palp$ nef.x -t -c4
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
8 1 1 1 1 1 1 1 1
8 1 1 1 1 1 1 1 1 M:6435 8 N:9 8  codim=4 #part=5
   BEGIN S-Poly     0sec  0cpu
   BEGIN B-Poly     11564sec  11558cpu
   BEGIN E-Poly     13126sec  13119cpu
H:1 65 [-128] P:0 V0:2 3  V1:4 5  V2:6 7       13126sec  13119cpu
np=1 d:0 p:4    0sec     0cpu
```

This option can be useful for finding at which point in the calculation of the Hodge numbers the program crashes.

[3] We thank Benjamin Nill for pointing this out to us.

6.4.11 -c*

The option `-c*` where `*` is a positive integer r allows to specify the length of the nef partition and hence the codimension of the Calabi-Yau complete intersection. The default value for the codimension is 2. Note that the computation time can take several hours for $r = 4$ or even days for $r > 4$ and PALP may crash because the limits such as the number of vertices etc. set in `Global.h` may be exceeded, cf. Section 2.2. We illustrate this option with complete intersections of codimension 3 in $\mathbb{P}^2 \times \mathbb{P}^2$:

```
palp$ nef.x -c3
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 1 1 1 0 0 0 3 0 0 0 1 1 1
3 1 1 1 0 0 0  3 0 0 0 1 1 1 M:100 9 N:7 6  codim=3 #part=7
H:[0] P:0 V0:1 3  V1:4 5       1sec  1cpu
H:[0] P:1 V0:2 3  V1:4 5       1sec  0cpu
np=1 d:1 p:5    0sec     0cpu
```

Also hypersurfaces can be analyzed with `nef.x`. As an example we consider the quintic hypersurface in $\mathbb{P}^4$:

```
palp$ nef.x -c1
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
5 1 1 1 1 1
5 1 1 1 1 1 M:126 5 N:6 5  codim=1 #part=1
H:1 101 [-200] P:0      0sec  0cpu
np=1 d:0 p:0    0sec     0cpu
```

Compare this to the output of `poly.x`:

```
palp$ poly.x
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
5 1 1 1 1 1
5 1 1 1 1 1 M:126 5 N:6 5 H:1,101 [-200]
```

6.4.12 -F*

The option `-F*` yields information about possible toric fibrations of the toric variety associated to the given reflexive lattice polytope. The polytopes associated to the toric fibers are restricted to be reflexive. By considering nef partitions for the given lattice polytope this option also computes possible fibrations of the corresponding complete intersection Calabi-Yau manifolds by lower-dimensional complete intersection Calabi-Yau manifolds. For more details see [28, 12]. In practice one should always use the option `-F*` in conjunction with either `-Lv` or `-Lp`. Here `*` is a non-negative integer s that specifies the maximal codimension s of the fiber polytope. The default value for s is 2. Note that this codimension does not need to coincide with the codimension of the corresponding complete intersection Calabi-Yau fiber. Besides the standard output and the output from the options `-Lv` or `-Lp`, the full information about fibration structures is listed below a second dashed line. The output takes the following form:

```
------------------------------------------------ #fibrations=#
    _    _    v    v    ...    p    p    p    v  cd=#  m: # # n: # #

    .    .    .    .    ...    .    .    .    .    .      . .    . .
    .    .    .    .    ...    .    .    .    .    .      . .    . .
    .    .    .    .    ...    .    .    .    .    .      . .    . .

    v    p    _    v    ...    v    _    _    p  cd=#  m: # # n: # #
```

The number `#` in `#fibrations=#` specifies the number of fibrations by reflexive polytopes up to symmetry that have been found. Then each of the subsequent lines corresponds to one of these fibrations. The points of the given polytope are labeled by either `v`, `p` or `_`. This label indicates whether the corresponding point is a vertex (`v`), a non-vertex point (`p`) or not a point at all (`_`) of the fiber polytope. The latter correspond to the directions of the toric base. The non-negative integer `#` in `cd=#` specifies the codimension of the fiber polytope. The two positive integers `# #` after `m`: specify the numbers of points and vertices of the dual of the fiber polytope, respectively. The two positive integers `# #` after `n`: specify the numbers of points and vertices of the fiber polytope, respectively.

We illustrate this option with a complete intersection of codimension 2 with several fibrations. In order to find all fibrations the argument of `-F` must be set to 3. This is an example where the interpretation of the fibration information depends on the choice of the nef partition.

```
palp$ echo "12 4 2 2 2 1 1 0  8 4 0 0 0 1 1 2" | nef.x -f -Lp -F3
12 4 2 2 2 1 1 0  8 4 0 0 0 1 1 2 M:371 12 N:10 7  codim=2 #part=5
5 10  Points of Poly in N-Lattice:
    0    0    0    1    0   -1    0    0    0    0
    0    0    1    0    0   -1    0    0    0    0
   -1    4    0    0    0    0    0    1    2    0
    0   -1    0    0    1    0    0    0    0    0
   -1    2    0    0    0    1    1    1    1    0
--------------------------------------------------
    4    1    2    2    1    2    0    0    0  d=12  codim=0
    4    1    0    0    1    0    2    0    0  d=8  codim=2
    2    0    1    1    0    1    0    0    1  d=6  codim=1
    2    0    0    0    0    0    1    0    1  d=4  codim=3
    1    0    0    0    0    0    0    1    0  d=2  codim=4
------------------------------------------------ #fibrations=3
    v    v    _    _    v    _    v    p    p  cd=2  m: 35  4 n: 7 4
    v    _    v    v    _    v    v    p    v  cd=1  m:117  9 n: 8 6
    v    _    _    _    _    _    v    p    v  cd=3  m:  9  3 n: 5 3
H:4 58 [-108] P:1 V:0 2   (6 6) (4 4) (3 3) (2 2) (1 1)   1sec  0cpu
H:3 65 [-124] P:2 V:0 2 3   (8 4) (4 4) (4 2) (2 2) (1 1) 1sec  0cpu
H:3 83 [-160] P:3 V:3 5   (4 8) (0 8) (2 4) (0 4) (0 2)   1sec  1cpu
np=3 d:0 p:2    0sec     0cpu
```

There are three fibrations. The fiber polytope of the second fibration is of codimension 1, hence has dimension $5-1=4$. As usual, we label the vertices and points by $v_0,\ldots,v_6,p_7,p_8,p_9$. The vertices labeled with `_` are v_1 and v_4, which are all in V_1 for all the three nef partitions. Since we are considering a complete intersection of codimension 2, the corresponding Calabi–Yau threefold admits a fibration by K3 surfaces since the fiber has dimension $4-2=2$. The

linear relation of codimension 1 and degree 6 does not involve v_1 and v_4, hence it describes the fiber polytope. The degrees of the nef partitions with respect to this linear relation are given in the third parentheses in the lines containing the information of the nef partitions. Hence, the K3 fibers are $\mathbb{P}(2,1,1,1,1)[3,3]$, $\mathbb{P}(2,1,1,1,1)[4,2]$, and $\mathbb{P}(2,1,1,1,1)[2,4]$, respectively. Note that the second fibration is an instance of the situation that a non-vertex point of the polytope becomes a vertex of the fiber polytope. Here, this is the point p_8.

The fiber polytope of the first fibration is of codimension 2, hence has dimension $5-2=3$. Naively, one would expect that the corresponding Calabi–Yau threefolds admit elliptic fibrations. This is indeed true for the first two nef partitions where both V_0 and V_1 contain vertices belonging to the fiber polytope. Repeating the steps of the second fibration above in this case yields the complete intersection $\mathbb{P}(4,1,1,2)[4,4]$ for both nef partitions. After discarding the trivial projection to the first coordinate, they become the hypersurfaces $\mathbb{P}(1,1,2)[4]$.

For the third nef partition, however, the vertices and points of the fiber polytope only lie in the part V_1 of the nef partition. Hence, the part V_0 reduces the dimension of the base. The fiber of the corresponding Calabi-Yau threefold is only of codimension 1 in the 3-dimensional toric fiber, i.e. it is a K3 surface. In fact, the linear relation of codimension 2 and degree 8 involves all points of V_1, hence it describes the fiber polytope. The degrees of the third nef partition with respect to this linear relation are the second parentheses in the line with `P:3`. Hence, the K3 fiber is $\mathbb{P}(4,1,1,2)[8]$. This phenomenon is further described in [12].

Finally, the fiber polytope of the third fibration is of codimension 3, and hence has dimension $5-3=2$. Naively, one would expect that the corresponding Calabi-Yau threefolds do not admit any fibrations since the codimension is also 2 and hence the fibers would be points. This is indeed the case for the first two nef partitions. For the third nef partition, the fiber polytope consists of the points v_0, v_6, p_7, and p_8, all of which lie in V_1. Hence, the fiber of the corresponding Calabi-Yau threefold is only of codimension 1 in the 2-dimensional toric fiber, i.e. it is an elliptic curve. The degrees of the third nef partition with respect to the linear relation of codimension 3 are the fourth parentheses in the line with `P:3`. Hence, the elliptic curve is $\mathbb{P}(2,1,1)[4]$.

6.4.13 -y

Depending on the input the option `-y` returns the CWS or the vertices of the M-lattice polytope if there is at least one nef partition. In order to trigger the output this nef partition may also be a projection. If there is no nef partition there is no output. Depending on the input the following output is given:

- if there is a nef partition:
  - If the input is a CWS, the CWS is returned along with the polytope data.
  - If the input is a polytope in the M-lattice or N-lattice (cf. option `-N` in Section 6.4.3) the M-lattice polytope is returned.
- if there is no nef partition
  - If the input is a CWS, the CWS is returned without further information about the polytope.
  - If the input is a polytope there is no output.

As an example consider the codimension 2 complete intersection in $\mathbb{P}^3$ from Section 6.4.3. If we enter the N-lattice polytope we get the following output:

```
palp$ nef.x -y -N
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 4
Type the 12 coordinates as dim=3 lines with #pts=4 columns:
-1    0    0    1
-1    0    1    0
-1    1    0    0
3 4 Vertices of Poly in M-lattice:  M:35 4 N:5 4  codim=2 #part=2
   -1   -1   -1    3
   -1   -1    3   -1
   -1    3   -1   -1
```

6.4.14 -S

The option `-S` gives information about the number of points in the reflexive Gorenstein cone and its dual (cf. options `-g*` and `-d*` discussed in Sections 6.4.23 and 6.4.24) for each nef partition which is not a direct product or a projection. It displays the numbers ℓ of lattice points and ℓ^* of interior lattice points in degrees $k \leq (\widetilde{d}+1)/2$, where $\widetilde{d}$ is the dimension of the Gorenstein cone C, and the analogous data for the dual cone $\check{C}$. These data enter the calculation of the (stringy) Hodge numbers via the S-polynomial (hence the name `-S`) as described in Section 6.2. The output takes the following form. After the first line of the standard output, there is a part referring to the polytope $\Delta(\check{C})$:

```
#points in largest cone:
layer:  1 #p:        l1 #ip:        l*1
  ...   . ...        .  ...         .
layer:  . #p:        .  #ip:        .
  ...   . ...        .  ...         .
layer:  k #p:        lk #ip:        l*k
```

where `l1` $= \ell(\Delta(\check{C}))$,..., `lk` $= \ell(k\Delta(\check{C}))$, `l*1` $= \ell^*(\Delta(\check{C}))$,..., `l*k` $= \ell^*(k\Delta(\check{C}))$. Subsequently there is a second part referring to the polytope $\Delta(C)$.

```
#points in largest cone:
layer:  1 #p:        l1 #ip:        l*1
  ...   . ...        .  ...         .
layer:  . #p:        .  #ip:        .
  ...   . ...        .  ...         .
layer:  k #p:        lk #ip:        l*k
```

where `l1` $= \ell(\Delta(C))$,..., `lk` $= \ell(\Delta(C))$, `l*1` $= \ell^*(\Delta(C))$,..., `l*k` $= \ell^*(k\Delta(C))$. Then the rest of the standard output concerning the nef partitions follows.

The following example illustrates this option. We consider a complete intersection of codimension 2 in $\mathbb{P}^4$:

```
palp$ nef.x -S
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
4 1 1 1 1
```

```
4 1 1 1 1 M:35 4 N:5 4  codim=2 #part=2


#points in largest cone:
layer:  1 #p:        6 #ip:        0
layer:  2 #p:       21 #ip:        1
layer:  3 #p:       56 #ip:        6


#points in largest cone:
layer:  1 #p:       20 #ip:        0
layer:  2 #p:      105 #ip:        1
layer:  3 #p:      336 #ip:       20
H:[0] P:0 V:2 3       0sec  0cpu
np=1 d:0 p:1    0sec     0cpu
```

One of the two nef partitions is a projection and is not analyzed. The output for the remaining nef partition has two blocks: The first block counts the numbers of points (after `#p:`) and points in the relative interior (after `#ip:`) of the Gorenstein cone $\check{C} \subset \widetilde{N}$ at degrees $k = 1, 2, 3$. Hence

$$\ell(\Delta(\check{C})) = 6, \quad \ell(2\Delta(\check{C})) = 21, \quad \ell(3\Delta(\check{C})) = 56,$$
$$\ell^*(\Delta(\check{C})) = 0, \quad \ell^*(2\Delta(\check{C})) = 1, \quad \ell^*(3\Delta(\check{C})) = 6.$$

One can check that the number of points at degree $k = 1$ indeed coincides with the number of points in the output of the option `-g2`.

The second block gives the same information for the dual Gorenstein cone $C \subset \widetilde{M}$. Hence

$$\ell(\Delta(C)) = 20, \quad \ell(2\Delta(C)) = 105, \quad \ell(3\Delta(C)) = 336,$$
$$\ell^*(\Delta(C)) = 0, \quad \ell^*(2\Delta(C)) = 1, \quad \ell^*(2\Delta(C)) = 20.$$

The output of the option `-d2` coincides with the number of points at degree $k = 1$.

6.4.15 -T

The option `-T` turns on an explicit check of the relation (6.10) relating the S– and T–polynomials. Normally the program actually uses that relation to avoid point counting beyond degree $(\widetilde{d}+1)/2$, but with `-T` the counting goes up to degree $\widetilde{d}$ and an error message is given if (6.10) is violated. This can be useful if one suspects that the program gives wrong Hodge numbers, for example because of numerical overflows. If nothing goes wrong, the only effect is a significantly increased computation time. The best way to illustrate this option is by combining it with `-S`. We consider the same example as in the previous subsection.

```
palp$ nef.x -S -T
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #colums' (= 'PolyDim #Points' or '#Points PolyDim'):
4 1 1 1 1
4 1 1 1 1 M:35 4 N:5 4  codim=2 #part=2


#points in largest cone:
```

```
layer:  1 #p:        6 #ip:        0
layer:  2 #p:       21 #ip:        1
layer:  3 #p:       56 #ip:        6
layer:  4 #p:      125 #ip:       21
layer:  5 #p:      246 #ip:       56


#points in largest cone:
layer:  1 #p:       20 #ip:        0
layer:  2 #p:      105 #ip:        1
layer:  3 #p:      336 #ip:       20
layer:  4 #p:      825 #ip:      105
layer:  5 #p:     1716 #ip:      336
H:[0] P:0 V:2 3       0sec  0cpu
np=1 d:0 p:1    0sec     0cpu
```

Note how now the point counting proceeds up to degree 5. With these data we can compute the Ehrhart polynomial

$$S_{\Delta(\check{C})}(t) = (1-t)^5(1 + 6t + 21t^2 + 56t^3 + 125t^4 + 246t^5 + \dots)$$

Since it has degree at most $\widetilde{d} = 5$, we find

$$S_{\Delta(\check{C})} = 1 + t + t^2 + t^3.$$

Similarly

$$T_{\Delta(\check{C})}(t) = (1-t)^5(t^2 + 6t^3 + 21t^4 + 56t^5 + \dots) = t^2 + t^3 + t^4 + t^5,$$

and it is clear that (6.10) is satisfied. A similar check can be performed for $C \cap \widetilde{M}$ with the data from the second block.

6.4.16 -s

The option `-s` includes all nef partitions in the output, not just one representative for each class of nef partitions that are equivalent under symmetries of the CWS. Note that this option does not print all possible nef partitions as those corresponding to projections (cf. option `-P` in Section 6.4.9) or direct products (cf. option `-D` in Section 6.4.8) are omitted. The example we consider is a complete intersection of codimension 2 in $\mathbb{P}^2 \times \mathbb{P}^2$. We add the option `-Lv` in order to print the vertices and the CWS.

```
palp$ nef.x -s -Lv
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 1 1 1 0 0 0 3 0 0 0 1 1 1
3 1 1 1 0 0 0  3 0 0 0 1 1 1 M:100 9 N:7 6  codim=2 #part=31
4 6 Vertices in N-lattice:
    0    0    0    1    0   -1
    0    0    1    0    0   -1
   -1    0    0    0    1    0
   -1    1    0    0    0    0
------------------------------
```

```
    1    1    0    0    1    0  d=3  codim=2
    0    0    1    1    0    1  d=3  codim=2
H:20 [24] P:2 V:4 5   (1 2) (1 2)     0sec  0cpu
H:20 [24] P:4 V:0 5   (1 2) (1 2)     0sec  0cpu
H:20 [24] P:5 V:0 4   (2 1) (0 3)     0sec  0cpu
H:20 [24] P:6 V:0 4 5   (2 1) (1 2)     0sec  0cpu
H:20 [24] P:8 V:1 5   (1 2) (1 2)     1sec  0cpu
H:20 [24] P:9 V:1 4   (2 1) (0 3)     0sec  0cpu
H:20 [24] P:10 V:1 4 5   (2 1) (1 2)     0sec  0cpu
[further Hodge data and nef partitions]
```

Note that the CWS is symmetric under permutations of the vertices labeled by $0, 1, 4$ and those labeled by $2, 3, 5$. Furthermore there only exist three pairs of degrees of the complete intersection (up to exchange within a pair): $\{(1,2),(1,2)\}, \{(0,3),(2,1)\}, \{(1,2),(2,1)\}$. Therefore we conclude that there are only three inequivalent nef partitions. This is indeed confirmed by calling `nef.x` without the option `-s`.

6.4.17 -n

The option `-n` prints the points of the polytope in the N-lattice only if there is at least one nef partition which does not correspond to a projection or a direct product. In addition, the first line of the standard output is printed while the other lines are suppressed. As an example we consider a codimension 2 complete intersection in $\mathbb{P}^3$

```
palp$ nef.x -n
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
4 1 1 1 1
4 1 1 1 1 M:35 4 N:5 4  codim=2 #part=2
3 5  Points of Poly in N-Lattice:
   -1    0    0    1    0
   -1    0    1    0    0
   -1    1    0    0    0
```

6.4.18 -v

The option `-v` prints the size of the matrix of vertices, the number of points and the vertices of the polytope that has been entered (M-lattice or N-lattice, depending on the input). If the input is the CWS the M-lattice polytope is analyzed. The output is printed in a single line with the character `E` as separator. Furthermore one can limit the output to polytopes whose number of points is constrained by a lower and an upper bound:

- `-v -u#`, where `#` is an integer ≥ 0, only gives output if the polytope has at most # points. The default value is the parameter `POINT_Nmax` which fixes the maximal number of points of a polytope at compilation.
- `-v -l#`, where `#` is an integer ≥ 0, only gives output if the polytope has at least # points. The default value is 0.

After closing the program a summary is printed. It contains information on the number of the examined polytopes which satisfy the bounds and the number of polytopes with `#` of points.

As an example we consider complete intersections of codimension 2 in $\mathbb{P}^3$ and $\mathbb{P}^2 \times \mathbb{P}^2$ with the weight matrices as input and without bounds.

```
palp$ nef.x -v
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
4 1 1 1 1
3 4 P:35 E   -1    3   -1   -1E   -1   -1    3   -1E   -1   -1 ...
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 1 1 1 0 0 0 3 0 0 0 1 1 1
4 9 P:100 E   -1    2   -1   -1    2   -1   -1    2   -1E
   -1   -1    2   -1   -1    2   -1   -1    2E
   -1   -1   -1    2    2    2   -1   -1   -1E
   -1   -1   -1   -1   -1   -1    2    2    2
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):

2  of  2

  35#    1
 100#    1
```

Since we have entered a CWS the M-lattice polytope is analyzed. Let us discuss the first line of output:

```
3 4 P:35 E   -1    3   -1   -1E   -1   -1    3   -1E   -1   -1 ...
```

The first two numbers indicate the number of rows and columns of the matrix of vertices in the M-lattice polytope. `P:35` indicates that the M-lattice polytope has 35 points. The vertices of the M-lattice polytope are then written in one line with the separator `E`. The output of the second example is analogous. After we quit PALP by hitting enter without input the following output is given:

```
2  of  2

  35#    1
 100#    1
```

This means that 2 out of the 2 polytopes analyzed satisfy the bounds and that there is one polytope with 35 points and one with 100.

Next, we consider the same example as above but with the upper bound for the number of points set to 50:

```
palp$ nef.x -v -u50
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
4 1 1 1 1
3 4 P:35 E   -1    3   -1   -1E   -1   -1    3   -1E   -1   -1 ...
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 1 1 1 0 0 0 3 0 0 0 1 1 1
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
```

```
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):

1  of  2

  35#    1
```

Now the second polytope exceeds the upper bound for the points as it has 100 points (cf. previous example). There is no output for the second polytope and the summary indicates that only one of the two polytopes analyzed satisfies the bounds.

6.4.19 -m

The option `-m` returns a nef partition of length 2 resulting from a partition $d = d_1 + d_2$ of the degree of a weight system. More precisely, the input data is a single weight system w and two positive integers d_1, d_2 such that $\sum_i w_i = d_1 + d_2$. The input format is `# d=# #`, where the first `#` is the usual CWS, while the `#` after `d=` refer to d_1 and d_2, respectively. As always, w specifies $\Delta^{(d)} \subset M$ as the Newton polytope of degree d. Furthermore, the degrees d_1, d_2 specify Newton polytopes $\Delta^{(d_1)}, \Delta^{(d_2)}$ from which one obtains the Minkowski sum $\Delta^{(d_1,d_2)} = \Delta^{(d_1)} + \Delta^{(d_2)} \subseteq \Delta^{(d)}$. If $\Delta_1 = \Delta^{(d_1)}, \Delta_2 = \Delta^{(d_2)}$ define a nef partition (∇_1, ∇_2) of the vertices of $(\Delta^{(d_1,d_2)})^*$, then the data of this nef partition are given in the standard output.

The following example taken from [12] illustrates this option. We consider the weighted projective space $\mathbb{P}(1,1,1,1,4,6)$ specified by the weight vector 14 1 1 1 1 4 6 of degree $d = 14$. The polytope $\Delta = \Delta^{(14)}$ is the Newton polytope of degree 14 monomials in this space. We first analyze the toric variety determined by $\Delta^{(14)}$:

```
palp$ nef.x -Lv
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #colums' (= 'PolyDim #Points' or '#Points PolyDim'):
14 1 1 1 1 4 6
14 1 1 1 1 4 6 M:1271 13 N:10 8  codim=2 #part=2
5 8 Vertices in N-lattice:
    0   -1    0    0    0    1    0    0
    0   -1    1    0    0    0    0    0
    0   -1    0    1    0    0    0    0
    0   -4    0    0    1    0   -1   -1
    1   -6    0    0    0    0   -1   -2
----------------------------------------
    6    1    1    1    4    1    0    0  d=14  codim=0
    1    0    0    0    1    0    1    0  d=3  codim=3
    2    0    0    0    1    0    0    1  d=4  codim=3
H:1 149 [-296] P:1 V:3 4 5 7  8   (6 8) (1 2) (2 2)     2sec  1cpu
np=1 d:0 p:1    2sec     1cpu
```

So $\Delta^{(14)}$ has 1271 lattice points and 13 vertices, and $(\Delta^{(14)})^*$ is the convex hull of the eight vertices shown in the output. By considering the weight systems below the dashed line one sees that $\mathbb{P}_{(\Delta^{(14)})^*}$ is the blowup of $\mathbb{P}(1,1,1,1,4,6)$ along the divisors corresponding to the last two vertices of $(\Delta^{(14)})^*$. Now we want to use the option `-m` to see whether the partition $14 = 2 + 12$ determines a nef partition via the Minkowski sum $\Delta^{(2,12)} = \Delta^{(2)} + \Delta^{(12)}$.

```
palp$ nef.x -Lv -m
```

```
type degrees and weights [d  w1 w2 ... wk d=d_1 d_2]:
14 1 1 1 1 4 6  d=2 12
14 1 1 1 1 4 6 d=2 12 M:1270 12 N:11 7  codim=2 #part=2
5 7 Vertices in N-lattice:
    0   -1    0    0    0    1    0
    0   -1    1    0    0    0    0
    0   -1    0    1    0    0    0
    0   -4    0    0    1    0   -2
    1   -6    0    0    0    0   -3
-----------------------------------
    6    1    1    1    4    1    0  d=14  codim=0
    3    0    0    0    2    0    1  d=6  codim=3
 d=12 2H:3 243 [-480] P:0 V:3 5   (2 12) (0 6)     7sec  6cpu
np=1 d:0 p:1    0sec     0cpu
```

The output indeed yields such a nef partition. Since not every monomial of degree 14 is a product of monomials of degree 2 and 12, the polytope $\Delta^{(2,12)}$ is only a proper subpolytope of $\Delta^{(14)}$. Consequently $\mathbb{P}_{(\Delta^{(2,12)})^*}$ is obtained from $\mathbb{P}_{(\Delta^{(14)})^*}$ by a further blowup along the vertex $(0,0,0,-2,-3)^T$.

By using the option `-m` in the same way one can find that $\Delta^{(6,8)} = \Delta^{(14)}$ and that $14 = 3 + 11$ does not give rise to a nef partition.

6.4.20 -R

The option `-R` prints the vertices of the input polytope if it is not reflexive. To illustrate this we enter the CWS of a polytope which is not reflexive:

```
palp$ nef.x -R
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
6 3 2 1 0 0 5 0 0 1 1 3
3 7  Vertices of input polytope:
   -1    1    0    0    1    0   -1
    0    1   -1    1    4    4    1
   -1    0    0    0   -1   -1   -1
```

The same output is given if we enter the polytope itself. Without the option `-R` there is no output if the polytope is not reflexive.

6.4.21 -V

The option `-V` prints the vertices of the polytope in the N-lattice together with the standard output. In contrast to the option `-Lv` (cf. Section 6.4.5) the information about the linear relations is not given. Furthermore, in the lines containing the nef partitions the additional information about the degrees is omitted. The option `-V` also works for non-reflexive polytopes. As an example we consider a complete intersection of codimension 2 in $\mathbb{P}^3$:

```
palp$ nef.x -V
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
4 1 1 1 1
4 1 1 1 1 M:35 4 N:5 4  codim=2 #part=2
```

```
3 4  Vertices of P:
   -1    0    0    1
   -1    0    1    0
   -1    1    0    0
H:[0] P:0 V:2 3       0sec  0cpu
np=1 d:0 p:1    0sec     0cpu
```

We can also enter the M-lattice polytope to get the same result. If the polytope is non-reflexive the output is the same as for the option `-R` (cf. Section 6.4.20).

6.4.22 -Q

The option `-Q` prints the information about the nef partitions and the Hodge numbers only if the corresponding complete intersection is a direct product (cf. option `-D` in Section 6.4.8) up to lattice quotients. If none of the nef partitions is a direct product only the numbers of points and vertices in the M- and N-lattice, together with the codimension and the number of nef partitions is given.

Consider the complete intersection of codimension 2 in $\mathbb{P}^2 \times \mathbb{P}^2$. As one can check using the option `-D` one of the nef partitions corresponds to a direct product:

```
palp$ nef.x -Q
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 1 1 1 0 0 0 3 0 0 0 1 1 1
3 1 1 1 0 0 0  3 0 0 0 1 1 1 M:100 9 N:7 6  codim=2 #part=5
H:4 [0] h1=2 P:0 V:2 3 5   D     0sec  0cpu
np=4 d:1 p:0    0sec     0cpu
```

The N-lattice polytope of $\mathbb{P}^3$ has no nef partition corresponding to a direct product. Then the output looks as follows:

```
palp$ nef.x -Q
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
4 1 1 1 1
4 1 1 1 1 M:35 4 N:5 4  codim=2 #part=2
np=2 d:0 p:0    0sec     0cpu
```

6.4.23 -g*

The option `-g*`, where `*` is an integer $m = 0, 1, 2$, returns the points of the supports $\Delta(\check{C})$ of the Gorenstein cones $\check{C} \subset \widetilde{N}_{\mathbb{R}}$ associated to the nef partitions of length r of the input polytope $\Delta^* \subset N_{\mathbb{R}}$. For the notation on Gorenstein cones see Section 6.2. The default value is $m = 1$. The standard output is changed as follows. The lines containing the information about the nef partition including the Hodge numbers, the parts of the nef partition etc. are suppressed. Instead, for each nef partition the points of $\Delta(\check{C})$ are printed in the following form:

```
D n Points of PG: (nv=#)
    #    #      ...      #    #
    .    .      ...      .    .
    .    .      ...      .    .
    #    #      ...      #    #
```

The interpretation depends on the integer m. For $m = 2$ the output is the list of points $\widetilde{p} \in \check{C}$ as given in (6.4). Note that since the origin 0_N belongs to every part of the nef partition, it appears r times, each time another of the r support functions being equal to 1. For $m = 1$ the redundant coordinate $\phi_0(p)$ is omitted in $\widetilde{p}$ and we obtain vectors $\widetilde{p}'$. For $m = 0$ all $\phi_i(p)$ are omitted and the resulting r-fold occurrence of 0_N is reduced to just a single occurrence; information on the nef partition is lost and the output becomes just the list of lattice points of Δ^*. The values of `D`, `n` and the `#` columns are summarized in Table 6.1, where n is the

m	0	1	2
`D`	d	$\widetilde{d}-1$	$\widetilde{d}$
`n`	n	$n+r-1$	$n+r-1$
`#` column	p	$\widetilde{p}'$	$\widetilde{p}$

Table 6.1: The meaning of the output of the options `-g*` and `-d*`

number of lattice points in Δ^* and $d, r, \widetilde{d}$ are as in Section 6.2. The number `#` in `nv=#` denotes the number of vertices of the cone $\check{C}$. The order of the points is first the vertices, then the non-vertex points with the origin at the end.

The following example illustrates this option. We consider complete intersections of codimension 2 in $\mathbb{P}^2 \times \mathbb{P}^1 \times \mathbb{P}^2$ discussed in [45]. The nef partitions for this example were discussed in Section 6.3. With the choice of `m=2` we obtain the information about the partition in terms of the Gorenstein cone. Let $v_0, \ldots, v_7$ denote the vertices of the polytope in the N-lattice.

```
palp$ nef.x -N -g2
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #colums' (= 'PolyDim #Points' or '#Points PolyDim'):
5 8
Type the 40 coordinates as dim=5 lines with #pts=8 colums:
 1  0 -1  0  0  0  0  0
 0  1 -1  0  0  0  0  0
 0  0  0  1 -1  0  0  0
 0  0  0  0  0  1  0 -1
 0  0  0  0  0  0  1 -1
M:300 18 N:9 8  codim=2 #part=15
7 10 Points of PG: (nv=8)
    1    1    0    0    0    1    1    0    0    1
    0    0    1    1    1    0    0    1    1    0
    1    0   -1    0    0    0    0    0    0    0
    0    1   -1    0    0    0    0    0    0    0
    0    0    0    1   -1    0    0    0    0    0
    0    0    0    0    0    1    0   -1    0    0
    0    0    0    0    0    0    1   -1    0    0
7 10 Points of PG: (nv=8)
    1    1    0    1    0    1    0    0    0    1
    0    0    1    0    1    0    1    1    1    0
    1    0   -1    0    0    0    0    0    0    0
    0    1   -1    0    0    0    0    0    0    0
    0    0    0    1   -1    0    0    0    0    0
    0    0    0    0    0    1    0   -1    0    0
    0    0    0    0    0    0    1   -1    0    0
```

```
[output of further nef partitions]
```

Let us consider the nef partition `P:8` as produced for instance by the option `-N -Lp`:

```
H:19 19 [0] P:8 V:4 5 6 7   (0 3) (1 1) (3 0)     0sec  0cpu
```

This example was the focus of [45]. The output of `-g2` for this nef partition is:

```
7 10 Points of PG: (nv=8)
    1    1    1    1    0    0    0    0    0    1
    0    0    0    0    1    1    1    1    1    0
    1    0   -1    0    0    0    0    0    0    0
    0    1   -1    0    0    0    0    0    0    0
    0    0    0    1   -1    0    0    0    0    0
    0    0    0    0    0    1    0   -1    0    0
    0    0    0    0    0    0    1   -1    0    0
```

Since the first four vertices v_0, v_1, v_2 and v_3 are in V_0, we have $\phi_0(v_i) = 1$ and $\phi_1(v_i) = 0$, hence the corresponding points of the Gorenstein cone take the form $(1, 0, v_i)$ for $i = 0, \ldots, 3$. The next four vertices v_4, v_5, v_6 and v_7 are in V_1, we have $\phi_0(v_i) = 0$ and $\phi_1(v_i) = 1$, hence the corresponding points of the Gorenstein cone take the form $(0, 1, v_i)$ for $i = 4, \ldots, 7$. Finally, the origin always belongs to every part of the nef partition, hence it appears as often as the codimension which here is $r = 2$. So $p_8 = 0$ and $p_9 = 0$. Once with $\phi_0(p_8) = 0$ and $\phi_1(p_8) = 1$ and once with $\phi_0(p_9) = 1$ and $\phi_1(p_9) = 0$.

6.4.24 -d*

The option `-d*`, where `*` is an integer $m = 0, 1, 2$, returns the points of the Gorenstein cones $C \subset \widetilde{M}_{\mathbb{R}}$ associated to the nef partitions of length r of the polytope $\nabla^* \subset M_{\mathbb{R}}$. For the notation on Gorenstein cones see Section 6.2, in particular (6.3) for the polytope ∇^*. This option can be used to determine the polytope ∇^* for each of the nef partitions of the given polytope Δ^*. The polytope ∇^* can then be further analyzed with `poly.x`.

The integer m triggers the same output format as for the option `-g*` in Section 6.4.23, The default value is $m = 1$. The option `-d2` automatically sets the flag `-p`.

The following example illustrates this option. We consider complete intersections of codimension 2 in $\mathbb{P}^2 \times \mathbb{P}^1 \times \mathbb{P}^2$ discussed in [45]. For more details on the nef partitions see the example in Section 6.3 and Section 6.4.23.

```
palp$ nef.x -N -d2
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #colums' (= 'PolyDim #Points' or '#Points PolyDim'):
5 8
Type the 40 coordinates as dim=5 lines with #pts=8 colums:
 1  0 -1  0  0  0  0  0
 0  1 -1  0  0  0  0  0
 0  0  0  1 -1  0  0  0
 0  0  0  0  0  1  0 -1
 0  0  0  0  0  0  1 -1
M:300 18 N:9 8  codim=2 #part=15
7 63 Points of dual PG: (nv=27)
   1   0   1   0   0   1   1   1 ...
   0   1   0   1   1   0   0   0 ...
```

```
  -1   0   1   1   0  -1  -1  -1 ...
  -1   1  -1   0   1   1   1  -1 ...   [63-8=55 more points]
   0   1   0   1  -1   0   0   0 ...
  -1   0  -1   0   0   1  -1  -1 ...
  -1   1   1   1   1  -1  -1   1 ...
[...]
```

For each of the 11 nef partitions of the input polytope Δ^* we get a 7-dimensional dual Gorenstein cone C from which the points of the polytope ∇^* can be read off by omitting the first two entries of each column, cf (6.4). The numbers of points and vertices of ∇^* depend on which of the nef partitions is considered. The nef partition of interest in [45] was `P:8`. The corresponding output of `-d2` is

```
7 40 Points of dual PG: (nv=12)
   0   0   1   1   1   0   0   0 ...
   1   1   0   0   0   1   1   1 ...
   0   0  -1   2  -1   0   0   0 ...
   0   0   2  -1  -1   0   0   0 ...   [40-8=32 more points]
   0   1   0   0   0   1   1   0 ...
  -1  -1   0   0   0   2  -1  -1 ...
  -1   2   0   0   0  -1  -1   2 ...
```

We see that the polytope ∇^* has 39 points (the interior point appears twice) and 12 vertices. Let $e_1, \ldots, e_5$ be the standard basis of $\mathbb{R}^5$. Let $\check{v}_0, \ldots, \check{v}_{11}$ denote the vertices of the polytope ∇^*. with

$$\begin{aligned}
&\check{v}_0 = -e_4 - e_5, && \check{v}_1 = e_3 - e_4 + 2e_5, && \check{v}_2 = -e_1 + 2e_2,\\
&\check{v}_3 = 2e_1 - e_2, && \check{v}_4 = -e_1 - e_2, && \check{v}_5 = e_3 + 2e_4 - e_5,\\
&\check{v}_6 = e_3 - e_4 - e_5, && \check{v}_7 = -e_4 + 2e_5, && \check{v}_8 = -e_1 + 2e_2 - e_3,\\
&\check{v}_9 = 2e_1 - e_2 - e_3, && \check{v}_{10} = -e_1 - e_2 - e_3, && \check{v}_{11} = 2e_4 - e_5.
\end{aligned}$$

From the first two rows of the above output we can read off the nef partition $\check{V} = \check{V}_0 \cup \check{V}_1$ of ∇^*:

$$\check{V}_0 = \langle \check{v}_2, \check{v}_3, \check{v}_4, \check{v}_8, \check{v}_9, \check{v}_{10} \rangle, \quad \check{V}_1 = \langle \check{v}_0, \check{v}_1, \check{v}_5, \check{v}_6, \check{v}_7, \check{v}_{11} \rangle.$$

We can check this by feeding the vertices back into `nef.x` with the options `-N` and `-Lv`.

```
palp$ nef.x -N -Lv
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
 or '#lines #colums' (= 'PolyDim #Points' or '#Points PolyDim'):
5 12
Type the 60 coordinates as dim=5 lines with #pts=12 colums:
  0   0  -1   2  -1   0   0   0  -1   2  -1   0
  0   0   2  -1  -1   0   0   0   2  -1  -1   0
  0   1   0   0   0   1   1   0  -1  -1  -1   0
 -1  -1   0   0   0   2  -1  -1   0   0   0   2
 -1   2   0   0   0  -1  -1   2   0   0   0  -1
M:24 15 N:39 12  codim=2 #part=2
5 12 Vertices in N-lattice:
   0    0   -1    2   -1    0    0    0   -1    2   -1    0
   0    0    2   -1   -1    0    0    0    2   -1   -1    0
   0    1    0    0    0    1    1    0   -1   -1   -1    0
```

```
  -1   -1    0    0    0    2   -1   -1    0    0    0    2
  -1    2    0    0    0   -1   -1    2    0    0    0   -1
-----------------------------------------------------------
[linear relations]
H:19 19 [0] P:0 V:1 2 3 4 5 6  13 15 17 ...   [degrees]
H:19 19 [0] P:1 V:2 3 4 8 9 10  16 17 18 ...   [degrees]
np=2 d:0 p:0    0sec     0cpu
```

We see that the nef partition `P:1` agrees with $\check{V} = \check{V}_0 \cup \check{V}_1$.

6.4.25 -G

The option `-G` works directly with Gorenstein cones which need not correspond to nef partitions. The input polytope is interpreted as the support polytope $\Delta(C)$ of a reflexive Gorenstein cone C, cf. Section 6.2. The index r of the cone is 2 by default and can be set to different values with the `-c` option, cf. Section 6.4.11. The standard output contains information on the support polytopes of the cone and the dual cone and the string–theoretic Hodge numbers h^{ij}, $0 \le i, j \le \dim C - 2r$, see [44]. If the input does not correspond to a reflexive Gorenstein cone of index r, no Hodge numbers and no N lattice data can be computed; as usual, the number of facets is displayed instead. If the input corresponds to a reflexive Gorenstein cone of an index different from r, this is treated like a non-reflexive case but with a warning message.

```
palp$ nef.x -G
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
4 2
Type the 8 coordinates as #pts=4 lines with dim=2 columns:
0 0
0 1
1 0
1 1
M:4 4 N:4 4 H:[0] h0=0
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 1 1 1 1 1
3 1 1 1 1 1 1 M:56 6 N:6 6 H:20 [24]
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
7 1 1 1 2 3 3 3
7 1 1 1 2 3 3 3 M:154 18 F:9
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
7 1 1 2 2 2 3 3
7 1 1 2 2 2 3 3 M:116 18 N:9 9 H:2 70 [-136]
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
3 2
Type the 6 coordinates as #pts=3 lines with dim=2 columns:
0 0
1 0
0 1
```

```
Warning: Input has index 3, should be 2!   M:3 3 F:3
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
  or '#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
1 1 1 0 0 0 0  2 0 0 1 1 1 1
1 1 1 0 0 0 0  2 0 0 1 1 1 1 M:20 8 N:6 6 H:[0]
```

As the examples show, weight input `d w1 ... wn` requires $w_1 + \ldots w_n = rd$; in other words the weights $q_i = w_i/d$ add up to r rather than to 1 as in the standard case. See [39] for more information on how weight systems determine Gorenstein cones.

`nef.x -G` cannot be combined with all of the other options. Nevertheless `-N` swaps the lattices M and N as usual; `-H`, `-S`, `-T` work as expected; `-t` works (without it no time information is given); `-c*` determines the index; `-R` displays the vertices of the input polytope if the cone is not reflexive of index r; `-V` displays the vertices of the support polytope of the dual (N lattice) cone; `-g`, `-d` display the full sets of points of the support polytopes in the lattices N or M, respectively (here no numbers can be specified with these options).

7 mori.x

The main purpose of `mori.x` is the computation of the Mori cone of toric varieties given by star triangulations of reflexive polytopes, which correspond to crepant subdivisions of the associated fans. The program is able to perform such triangulations for four–dimensional polytopes with up to three non–vertex points if the secondary fan is at most three–dimensional. The program can also be used with a known triangulation as its input starting from PALP release 2.1. This option, which was not contained in PALP 2.0 as described in [21], works for arbitrary dimensions.

7.1 General aspects of `mori.x`

We distinguish two types of functionalities of `mori.x`. The first kind yields information about the appropriately resolved ambient space (see options `-g, -I, -m, -P, -K` below). This includes the Stanley-Reisner (SR) ideal (with `-g`) as well as specific information on the geometry of the lattice polytope that determines the ambient toric variety: incidence structure of the facets (`-I`), IP-simplices (`-P`) and subdivisions of the fan (`-g`); furthermore, the Oda-Park algorithm [46, 47] is used to find the Mori cone of the ambient space (`-m`). The second kind of functionalities deals with the intersection ring (`-i, -t`) and topological quantities (`-b, -c, -d`) of the embedded hypersurface. They are determined with the help of SINGULAR [17], a computer algebra system for polynomial computations. Correspondingly, the options `-b, -i, -c, -t, -d` (as well as `-a, -H`, see below) need SINGULAR to be installed.

The generators of the Mori cone are given in terms of their intersections with the toric divisors. For singular toric varieties, the Picard group of Cartier divisors is a non-trivial subgroup of the Chow group, which contains the Weil divisors. Hence one can consider the Kähler cone, which is dual to the Mori cone, as a cone in the vector space spanned by the elements of either the Picard or the Chow group. The program `mori.x` only deals with simplicial toric varieties, for which the Picard group is always a finite index subgroup of the Chow group [22, 23]. Hence the Cartier divisors are integer multiples of the Weil divisors and this ambiguity does not arise.

Starting with PALP 2.1, `mori.x` affords two distinct modes of operation. If used with the option `-M` arbitrary reflexive polytopes of any dimension can serve as input (at least in principle), see section 7.2.16 for details. Without `-M` the program only works if the input polytope can be triangulated by `mori.x` or if it does not require triangulation, as we will outline in the following sections.

As described in [21], `mori.x` can perform star triangulations of certain four–dimensional reflexive polytopes. This operation was designed for the CY hypersurface case. Generic CY hypersurfaces avoid point-like singularities of the ambient space as well as divisors that correspond to interior points of facets. Consequently, the algorithm performs star triangulations only up to such interior points.

Polytopes can be triangulated by subdividing the secondary fans of its non-simplicial facets [48, 49]. This triangulation algorithm is implemented in `mori.x` for polytopes with up to three points that are neither vertices nor interior to the polytope or one of its facets; this implies that the secondary fan of any facet can be at most three–dimensional. The program exits with a warning message if the subdivision is not properly completed.

As the dimension of the secondary fan corresponding to a facet grows with the number of points in the facet, this limitation tends to become relevant for toric varieties for which $h^{1,1}$ is large: $h^{1,1}$ increases with the number of points on the polytope and polytopes with many points are more likely to have facets containing many points.

Complete triangulations of arbitrary polytopes can be performed programs such as TOPCOM [16], which is also included in the open source mathematics software system Sage [18]. Sage also contains various tools for handling toric varieties. The triangulations performed in `mori.x` are attuned to the case of three-dimensional CY hypersurfaces. This means, in particular, that interior points of facets are ignored: one must use `-M` to avoid this. For small Picard numbers, `mori.x` is hence faster than programs which perform a complete triangulation.

If a polytope of arbitrary dimension has only simplicial facets whose only lattice points are its vertices and possibly interior points, it does not require any triangulation. Hence `mori.x` can also handle such cases without `-M`.

With the option `-H` the program can also analyze arbitrary hypersurfaces embedded in the ambient toric varieties. It is capable of computing the intersection ring and certain characteristic classes. Here the omission of interior points of facets, which happens as a consequence of `mori.x`'s triangulation algorithm, may introduce severe singularities which often result in non–integer intersection numbers. There is a warning if there are indeed points interior to facets; in such a case it is probably better to repeat the computation with the combination `-HM`.

The help screen provides essential information about all the functionalities of the program:

```
palp$ mori.x -h
This is ‘‘mori.x’’:
                star triangulations of a polytope P* in N
                Mori cone of the corresponding toric ambient spaces
                intersection rings of embedded (CY) hypersurfaces
Usage:   mori.x [-<Option-string>] [in-file [out-file]]
Options (concatenate any number of them into <Option-string>):
 -h    print this information
 -f    use as filter
 -g    general output: triangulation and Stanley-Reisner ideal
 -I    incidence information of the facets (ignoring IPs of facets)
```

```
 -m    Mori generators of the ambient space
 -P    IP-simplices among points of P* (ignoring IPs of facets)
 -K    points of P* in Kreuzer polynomial form
 -b    arithmetic genera and Euler number
 -i    intersection ring
 -c    Chern classes of the (CY) hypersurface
 -t    triple intersection numbers
 -d    topological information on toric divisors &
       del Pezzo conditions
 -a    all of the above except h, f, I and K
 -D    lattice polytope points of P* as input (default CWS)
 -H    arbitrary (also non-CY) hypersurface
       'H = c1*D1 + c2*D2 + ...' input: coefficients 'c1 c2 ...'
 -M    Stanley-Reisner ideal and Mori generators with an
       arbitrary triangulation as input; must be combined with -D
Input: 1) standard: degrees and weights
          'd1 w11 w12 ... d2 w21 w22 ...'
       2) alternative (use -D): 'd np' or 'np d'
          (d=Dimension, np=#[points]) and (after newline) np*d
          coordinates
Output:   as specified by options
```

Following PALP's notation we refer to the M lattice polytope which determines the CY hypersurface as P; consequently its dual, which gives rise to the fan of the ambient toric variety, is P^*.

As PALP always interprets the input as $P \subset M_{\mathbb{R}}$ unless some option modifies this behavior, matrix input of $P^* \subset N_{\mathbb{R}}$ requires the option `-D`. In order to avoid errors, matrix input is not allowed unless this option is set. If only P but not P^* is known one can use `poly.x -e` to obtain the latter.

7.2 Options of `mori.x`

This section contains a detailed description of the options listed in the help screen. If no flag is specified, the program starts with the parameter `-g`. By default, the program considers a CY hypersurface embedded in the ambient toric variety. The option `-H` has to be used in order to consider non-CY hypersurfaces. Note that the options `-b, -i, -c, -t, -d, -a, -H` need SINGULAR [17] to be installed.

Most options of `mori.x` produce output that is related to the points of P^* in a specific order which can be determined by combining the desired functionality with the option `-P` (see sec. 7.2.6 below). In order to avoid repeating this information for every option, we now present an example that will be used for many of the options below:

```
palp$ mori.x -P
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...':
8 4 1 1 1 1 0  6 3 1 0 1 0 1
4 8  points of P* and IP-simplices
   -1    0    0    0    1    3    1    0
    0    0    0    1    0   -1    0    0
   -1    1    0    0    0    3    1    0
    1    0    1    0    0   -4   -1    0
------------------------------   #IP-simp=2
```

```
4    1    0    1    1    1   8=d  codim=0
3    0    1    1    0    1   6=d  codim=1
```

The output above the dashed line just means that P^* has the lattice points[4]

$$p_1 = \begin{pmatrix} -1 \\ 0 \\ -1 \\ 1 \end{pmatrix},\ p_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix},\ \cdots,\ p_7 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ -1 \end{pmatrix},\ p_8 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{7.1}$$

and the last two lines encode the facts $4p_1 + p_2 + p_4 + p_5 + p_6 = 0$, $3p_1 + p_3 + p_4 + p_6 = 0$. Note how the dashed line proceeds only up to p_6. This is because `mori.x` always ignores the origin, and, if used without `-M`, also ignores points that are interior to facets: $p_7 = (p_2 + p_4 + p_5 + p_6)/4 = (p_3 + p_4 + p_6)/3$ lies inside the facet with vertices p_2, p_3, p_4, p_5, p_6. The reader is invited to check that the same example with `mori.x -PM` results in a dashed line below all points except the origin.

7.2.1 -h

This option prints the help screen.

7.2.2 -f

This parameter suppresses the prompt of the command line. This is useful if one wants to build pipelines or shorten the input; e.g. our standard example (7.1) can be entered as

```
palp$  echo  '8 4 1 1 1 1 0  6 3 1 0 1 0 1' | mori.x -fP
4 8  points of P* and IP-simplices
...
```

7.2.3 -g

This triggers the general output. First, the triangulation data of the facets is displayed. The number of triangulated simplices is followed by the incidence structure of the simplices. The incidence information for each simplex is encoded in terms of a bit sequence (cf. sec. 2.3.3): there is a digit for each relevant polytope point; a 1 denotes that the point belongs to the simplex. Second, the SR ideal is displayed: the number of elements of the ideal is followed by its elements. Each element is denoted by a bit sequence as above.

```
palp$ echo  '8 4 1 1 1 1 0  6 3 1 0 1 0 1' | mori.x -fg
8 Triangulation
110101 111100 101011 101110 100111 111001 001111 011101
2 SR-ideal
010010 101101
9 Triangulation
110101 111100 101011 101110 100111 111001 010111 011011 011110
2 SR-ideal
110010 001101
```

[4]Here the index starts at 1 instead of 0 as it is standard in PALP. This shift is needed to match the counting of toric divisor classes displayed in certain outputs of `mori.x` and hence avoids confusion.

The program performs the two possible triangulations of the facet $\langle 23456\rangle$, which is the only non–simplicial one (see section 7.2.4). The last two bit sequences of the first result describe the simplices $\langle\widehat{25}346\rangle$, whereas the second triangulation gives the three simplices $\langle\widehat{2346}5\rangle$ (in this notation the hat indicates that one of the points is dropped). Nevertheless, the two resolutions give the same CY intersection polynomial.[5]

7.2.4 -I

The incidence structure of the facets of the polytope P^* is displayed. Interior points of the facets are neglected.

```
palp$  echo  '8 4 1 1 1 1 0  6 3 1 0 1 0 1' | mori.x -fI
Incidence: 110101 111100 011111 101011 101110 100111 111001
```

The incidence data show the intersections of $p_1,\ldots,p_6$ (ignoring p_7, p_8!) with the seven facets. The third facet contains the five points $p_2,\ldots,p_6$, hence it is not simplicial and needs to be triangulated. See section 2.3.3 for more details on the representation of incidences as bit sequences.

7.2.5 -m

The Mori cone generators of the ambient space are displayed in the form of a matrix.[6] Each row corresponds to a generator. The entries of each row are the intersections of the generator with the toric divisor classes. The Oda-Park algorithm is used to compute the generators. Furthermore, the incidence structure between the generators of the Mori cone and its facets is displayed. For the standard example this takes the following form.

```
palp$ echo  '8 4 1 1 1 1 0  6 3 1 0 1 0 1' | mori.x -fm
2 MORI GENERATORS / dim(cone)=2
  3  0  1  1  0  1   I:10
  0  3 -4 -1  3 -1   I:01
2 MORI GENERATORS / dim(cone)=2
  1  1 -1  0  1  0   I:10
  0 -3  4  1 -3  1   I:01
```

The Mori cone is two-dimensional, so that its facets can be identified with the generators. This explains the trivial incidence structure.

Let us consider another simple example, a hypersurface in $\mathbb{P}^2\times\mathbb{P}^1\times\mathbb{P}^1$.

```
palp$ mori.x -m
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...':
3 1 1 1 0 0 0 0  2 0 0 0 1 1 0 0  2 0 0 0 0 0 1 1
3 MORI GENERATORS / dim(cone)=3
   0  0  0  1  1  0  0   I:110
   1  1  0  0  0  0  1   I:101
   0  0  1  0  0  1  0   I:011
```

[5] This fact suggests that the two resolutions give rise to the same CY hypersurface. Indeed, simply connected CY threefolds are completely determined up to diffeomorphisms by their Hodge numbers, intersection rings and second Chern classes [15, 33].

[6] As divisors corresponding to interior points of facets do not intersect a CY hypersurface, such divisors are neglected in the computation of the Mori cone of the ambient space.

The Mori cone generators can easily be seen to be dual to the hyperplane sections. Now, the Mori cone is three-dimensional, so that each of its facets contains two generators. Let us, for instance, consider the incidence structure between the first generator and the three facets of the Mori cone. Here, the string `I:110` tells that the vector lies on the first and second facets but does not intersect the third one.

For an example with a more complicated structure of the Mori cone see section 7.2.16.

7.2.6 -P

First a list of lattice points of P^* is displayed in the following manner. If `-P` is combined with both `-M` and `-D`, the list is just the input provided by the user, in the same order except for the fact that the lattice origin comes at the end of the list. In all other cases the complete list of lattice points of P^* is given in the following order:

1. vertices (with `-D` in the order provided by the user),
2. points not interior to the polytope or its facets,
3. points interior to facets,
4. the lattice origin.

Then a dashed line indicates which points are 'relevant': all points except for the origin in the case of `-M`, but not points interior to facets otherwise. Finally the IP-simplices with vertices among these relevant points are displayed.

The output for the standard example can be found above equation (7.1). The following example features all types of lattice points:

```
palp$ echo '16 8 4 2 1 1' | mori.x -fP
4 9  points of P* and IP-simplices
   -1    0    0    2    0    0    0    1    0
   -1    0    0    1    2    0    1    1    0
    0    0    2   -1    1    1    1    0    0
    0    1    1    0   -1    1    0    0    0
-----------------------------------   #IP-simp=3
    8    1    1    4    2    0    0  16=d  codim=0
    4    0    0    2    1    1    0   8=d  codim=1
    2    0    0    1    0    0    1   4=d  codim=2
```

$p_1, \ldots, p_5$ are vertices, p_6, p_7 further relevant points, but $p_8 = -p_1 = (p_2 + p_3 + 4p_4 + 2p_5)/8$ is interior to the facet spanned by $p_2, \ldots, p_5$. Note that the ordering of the CWS input is not obeyed by the output of lattice points. Once the order is displayed, however, it is fixed and determines the labeling of toric divisors in any further output.

7.2.7 -K

The Kreuzer polynomial[7] of PALP's representation of P^* is displayed. It encodes lattice polytope points in a compact form. The number of variables equals the dimension of the polytope. Each lattice point gives rise to a Laurent monomial in which the exponents of the variables are the coordinates. Vertices and non-vertices are distinguished by coefficients '+'

[7] We named this output format after Maximilian Kreuzer, who designed it. This is an example of his proverbial ability to eliminate unnecessary data redundancies and recast essential information in condensed form.

and '−' respectively. Points in the interior of facets are ignored. As this is closely connected with the way `mori.x` works when used without `-M`, the combination `-MK` is not allowed.

```
palp$ echo  '8 4 1 1 1 1 0  6 3 1 0 1 0 1' | mori.x -fK
KreuzerPoly=t_4/(t_1t_3)+t_3+t_4+t_2+t_1+t_1^3t_3^3/(t_2t_4^4);
intpts=1;  Pic=2
```

A comparison with the output for `-P`, which can be found above equation (7.1), might help for a better understanding of the present option.

Negative coordinates are always displayed by putting the variables in the denominator. The number of points in the interior of facets is shown as `intpts`. The multiplicities of the toric divisors are displayed as `multd` if they are greater than one. Furthermore, the Picard number of the CY hypersurface is computed and printed as `Pic`.

7.2.8 -b

The zeroth and first arithmetic genera of the hypersurface are determined according to the following formulas [50]:

$$\chi_q(X) = \sum_p (-1)^p h^{p,q}(X) = \int_X \mathrm{ch}(\Omega^q(X))\mathrm{Td}(X), \quad q = 0, 1. \tag{7.2}$$

where $\Omega^0(X) = \mathcal{O}_X$ is the trivial bundle and $\Omega^1(X) = T^*X$ is the bundle of 1-forms, ch is the corresponding Chern character, and $\mathrm{Td}(X)$ is the Todd class of X.

Furthermore the Euler characteristic is displayed. Here we compute it by means of the intersection polynomial:

$$\chi = \int_X \mathrm{c}_n \,. \tag{7.3}$$

where c_n is the top Chern class, $n = \dim X$.

These formulas hold for arbitrary smooth hypersurfaces; in particular, they do not need to be CY. Indeed, if X is CY, its Euler characteristic can also be computed by `poly.x` in terms of polytope combinatorics. Compare the two Euler characteristics for a consistency check.

Consider the $K3$ surface as a simple example:

```
palp$ echo '4 1 1 1 1' | mori.x -bf
SINGULAR  -> Arithmetic genera and Euler number of the CY:
chi_0:  2 , chi_1: -20  [ 24 ]
```

As expected, the Euler characteristic is 24 and $h^{1,1} = 20$. Using the example discussed before we find

```
palp$ echo '8 4 1 1 1 1 0 6 3 1 0 1 0 1' | mori.x -bf
SINGULAR  -> Arithmetic genera and Euler number of the CY:
chi_0: 0 , chi_1: 126  [ -252 ]
SINGULAR  -> Arithmetic genera and Euler number of the CY:
chi_0: 0 , chi_1: 126  [ -252 ]
```

Special care is needed in the interpretation of the results for non-CY hypersurfaces: the triangulation algorithm might fail to make these varieties smooth, in which case the formulas above do not hold and hence the output is misleading; see the description of the option `-H` for more details.

7.2.9 -i

This option displays the intersection polynomial in terms of an integral basis of the toric divisors. The coefficients of the monomials are the triple intersection numbers in this basis. This option can also be used together with `-H` to perform this task for non-CY hypersurfaces.

```
palp$ echo  '8 4 1 1 1 1 0  6 3 1 0 1 0 1' | mori.x -fi
SINGULAR -> divisor classes (integral basis J1 ... J2):
d1=J1+3*J2, d2=J1, d3=-J1+J2, d4=J2, d5=J1, d6=J2
SINGULAR -> intersection polynomial:
2*J1*J2^2+2*J2^3
SINGULAR -> divisor classes (integral basis J1 ... J2):
d1=J1+3*J2, d2=J1, d3=-J1+J2, d4=J2, d5=J1, d6=J2
SINGULAR -> intersection polynomial:
2*J1*J2^2+2*J2^3
```

`d1`, ..., `d6` denote the toric divisors corresponding to the lattice points $p_1, \ldots, p_6$, cf. eq. (7.1). There are two independent divisor classes. Indeed, `mori.x` expresses the intersection polynomial in terms of the integral basis $J_1 = D_2 = D_5$ and $J_2 = D_4 = D_6$.

7.2.10 -c

The Chern classes of the hypersurface (CY or non-CY) are displayed in terms of an integral basis of the toric divisors:

```
palp$ echo '8 4 1 1 1 1 0 6 3 1 0 1 0 1' | mori.x -fc
SINGULAR -> divisor classes (integral basis J1 ... J2):
d1=J1+3*J2, d2=J1, d3=-J1+J2, d4=J2, d5=J1, d6=J2
SINGULAR  -> Chern classes of the CY-hypersurface:
c1(CY)=  0
c2(CY)=  10*J1*J2+12*J2^2
c3(CY)=  -252 *[pt]
SINGULAR -> divisor classes (integral basis J1 ... J2):
d1=J1+3*J2, d2=J1, d3=-J1+J2, d4=J2, d5=J1, d6=J2
SINGULAR  -> Chern classes of the CY-hypersurface:
c1(CY)=  0
c2(CY)=  10*J1*J2+12*J2^2
c3(CY)=  -252 *[pt]
```

7.2.11 -t

The triple intersection numbers of the toric divisors are displayed. The form of this output[8] is designed for further use in Mathematica [51]. Before computing the intersection ring for our standard example (7.1), let us state some expectations. Inspection of the data of the polytope reveals that it describes a K3 fibration with the fiber determined by the weight system 6 3 1 1 1. There are only the two points p_2, p_5 outside the corresponding 3–plane, so each of them must represent the generic fiber with self–intersection 0. In other words, the self–intersections of d_2 and d_5 as well as $d_2 \cdot d_5$ must all vanish. This is confirmed by the following excerpt from the output:

[8]The pre-compiler command `DijkEQ` in the C file `SingularInput.c` controls the symbol '`->`' in option `-t`.

```
echo '8 4 1 1 1 1 0  6 3 1 0 1 0 1' | mori.x -ft
SINGULAR -> triple intersection numbers:
d6^3->2,
d5*d6^2->2,
d4*d6^2->2,
d3*d6^2->0,
d2*d6^2->2,
d1*d6^2->8,
d5^2*d6->0,
d4*d5*d6->2,
d3*d5*d6->2,
d2*d5*d6->0,
d1*d5*d6->6,
d4^2*d6->2,
d3*d4*d6->0,
d2*d4*d6->2,
d1*d4*d6->8,
d3^2*d6->-2,
d2*d3*d6->2,
d1*d3*d6->2,
d2^2*d6->0,
d1*d2*d6->6,
d1^2*d6->30,
d5^3->0,
d4*d5^2->0,
d3*d5^2->0,
d2*d5^2->0,
d1*d5^2->0,
d4^2*d5->2,
d3*d4*d5->2,
d2*d4*d5->0,
d1*d4*d5->6,
d3^2*d5->2,
d2*d3*d5->0,
d1*d3*d5->6,
d2^2*d5->0,
d1*d2*d5->0,
d1^2*d5->18,
[...]
```

7.2.12 -d

This option displays topological data of the toric divisors restricted to the (CY or non-CY) hypersurface. The Euler characteristics of the toric divisor classes and their arithmetic genera are shown.

Furthermore, in the case of a three-dimensional hypersurface, the program checks the del Pezzo property against two necessary conditions and analyses the mutual intersections of the del Pezzo candididates. The number of del Pezzo candidates is displayed followed by their type in parenthesis; furthermore, those among them that do not intersect other del Pezzo candidates are listed.

For a del Pezzo divisor S of type n, the following equations should hold:

$$\int_S c_1(S)^2 = 9 - n\,, \qquad \int_S c_2(S) = n + 3 \quad \Longrightarrow \quad \chi_0(S) = \int_S \mathrm{Td}(S) = 1\,. \tag{7.4}$$

Here, Td(S) denotes the Todd class of S, which gives the zeroth arithmetic genus of S upon integration. This test also allows to determine the type of the del Pezzo surface in question. A second necessary condition comes from the fact that a del Pezzo surface is a two-dimensional Fano manifold. Hence, the first Chern class of S integrated over all curves on S has to be positive:

$$D_i \cap S \cap c_1(S) > 0 \qquad \forall D_i : \; D_i \neq S\,, \quad D_i \cap S \neq 0\,. \tag{7.5}$$

This condition would be sufficient if we were able to access *all* curves of the hypersurface. In our construction, however, we can only check for curves induced by toric divisors. This functionality was added to carry out the analysis of base manifolds for elliptic fibrations in [52].

Consider the following example: it is well-known that the del Pezzo surface dP_6 can be realized as a homogeneous polynomial of degree 3 in $\mathbb{CP}^3$. Hence a Calabi-Yau hypersurface in a toric variety with CWS

$$\begin{array}{c} 5\,1\,1\,1\,1\,1\,0 \\ 2\,0\,0\,0\,0\,1\,1 \end{array}$$

i.e. a $\mathbb{CP}^1$ fibration over $\mathbb{CP}^3$ contains a dP_6: setting the last coordinate z_6 to zero forces all terms to be of the form $z_5^2 P_3(z_1, \ldots, z_4)$, where $P_3(z_1, \ldots, z_4)$ is a homogeneous polynomial of degree 3 in $z_1, \ldots, z_4$. We may set z_5 to 1 by using the second $\mathbb{C}^*$ action, so that the divisor D_6 corresponds to a homogenous polynomial of degree 3 in $\mathbb{CP}^3$, i.e. a dP_6.

This is confirmed by

```
palp$ mori.x -d
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...':
5 1 1 1 1 1 0 2 0 0 0 0 1 1
SINGULAR -> topological quantities of the toric divisors:
Euler characteristics: 46 46 46 9 46 55
Arithmetic genera: 4 4 4 1 4 5
dPs: 1 ; d4(6)  nonint: 1 ; d4
```

Note that PALP has exchanged the ordering of the divisors, so that the dP_6 is now given by D_4. This divisor does not intersect any other del Pezzo as it is the only del Pezzo candidate in this example.

7.2.13 -a

This is a shortcut for `-gmPbictd`.

7.2.14 -D

An alternative way to provide the input is to type lattice polytope points of P^* directly. In this case, one has to use the parameter `-D`. Let us reconsider the example of sec. 7.2.6:

```
palp$ mori.x -DP
'#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
4 5
Type the 20 coordinates as dim=4 lines with #pts=5 columns:
-1  2  0  0  0
-1  1  2  0  0
 0 -1  1  0  2
 0  0 -1  1  1
4 9  points of P* and IP-simplices
   -1    2    0    0    0    0    0    1    0
   -1    1    2    0    0    0    1    1    0
    0   -1    1    0    2    1    1    0    0
    0    0   -1    1    1    1    0    0    0
-----------------------------------   #IP-simp=3
    8    4    2    1    1    0    0  16=d  codim=0
    4    2    1    0    0    1    0   8=d  codim=1
    2    1    0    0    0    0    1   4=d  codim=2
```

Note how the order of the vertices corresponds to that of the input (cf. section 7.2.6).

7.2.15 -H

Using this option, one can specify a (non-CY) hypersurface. The user determines the hypersurface divisor class $H = \sum_i c_i D_i$ in terms of the toric divisor classes D_i by typing its coefficients c_i. The hypersurface can then be analyzed by combining `-H` with other options, as described above. Just using `-H`, the program runs `-Hb`.

The reader is warned: smoothness is not guaranteed anymore, so that the intersection numbers can become fractional. Some choices of the hypersurface equation may intersect singularities not resolved by the triangulation. Consider e.g. the hypersurface determined by the divisor class $H = D_1 + D_6$ in our example (7.1). Remember that the order in which `mori.x` expects the coefficients of the hypersurface divisor class is fixed by the polytope matrix and not by the CWS input. Hence, the correct input for H is the string `1 0 0 0 0 1`.

```
palp$ mori.x -H
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...'
8 4 1 1 1 1 0  6 3 1 0 1 0 1
WARNING: there is 1 facet-IP ignored in the triangulation.
This may lead to unresolved singularities in the hypersurface.
Type the 6 (integer) entries for the hypersurface class:
1 0 0 0 0 1
Hypersurface degrees: ( 5  4 )
Hypersurface class: 1*d1 1*d6
SINGULAR  -> Arithmetic genera and Euler number of H:
chi_0: 29/27 , chi_1: 128/27  [ -22/3 ]
```

To calculate these quantities, the program determines the characteristic classes of the divisors using adjunction. It then performs the appropriate integration with the help of the triple intersection numbers. The fractional results of the arithmetic genera and the Euler number in our example indicate that the intersection polynomial has fractional entries. This happens because the program introduces a singularity into the ambient toric variety which descends to the hypersurface H. It is therefore much better to combine `-H` with `-M`:

```
palp$ mori.x -HM
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...':
8 4 1 1 1 1 0  6 3 1 0 1 0 1
4 8
   -1    0    0    0    1    3    1    0
    0    0    0    1    0   -1    0    0
   -1    1    0    0    0    3    1    0
    1    0    1    0    0   -4   -1    0
'#triangulations':
1
1 triangulations:
12 1111000 1110010 1101010  0111001 0110011 0101011
1011100 1010110 1001110  0011101 0010111 0001111
Type the 7 (integer) entries for the hypersurface class:
1 0 0 0 0 1 0
Hypersurface degrees: ( 5  4  1 )
Hypersurface class: 1*d1 1*d6
SINGULAR  -> Arithmetic genera and Euler number of H:
chi_0:  1 , chi_1: 3  [ -4 ]
```

As a second example, consider the quadric in $\mathbb{CP}^3$:

```
palp$ mori.x -H
Degrees and weights  'd1 w11 w12 ... d2 w21 w22 ...':
4 1 1 1 1
Type the 4 (integer) entries for the hypersurface class:
2 0 0 0
Hypersurface degrees: ( 2 )
Hypersurface class: 2*d1
SINGULAR  -> Arithmetic genera and Euler number of H:
chi_0:  1 , chi_1: -2  [ 4 ]
```

The hypersurface is smooth in this case, so that the arithmetic genera and the Euler number are those of $\mathbb{CP}^1 \times \mathbb{CP}^1$. Of course, one needs to independently check smoothness in order to rely on the output, as the integrality of the arithmetic genera alone is not sufficient to conclude that the hypersurface is non-singular.

7.2.16 -M

Option `-M` allows polytopes of (in principle) arbitrary dimensions, but expects the triangulations to be provided by the user; it can be combined with any other option except for `-K`. As the Mori cone is analysed with PALP's routines, the parameter `POLY_Dmax` must possibly be adjusted for this; see sec. 2.3.4.

This functionality is useful, for instance, when `mori.x` fails to triangulate the polytope by itself. This happens whenever the dimension of the polytope is different from four, or when the polytope contains more than three lattice points that are neither vertices nor (facet–)IPs. Fortunately, there are programs capable of efficiently performing complete triangulations of arbitrary polytopes [16, 18]; the user can redirect their output as an input for `mori.x` to determine the Mori generators of the ambient space. Other situations where `-M` is useful arise whenever we prefer to keep control over the lattice points involved in the triangulation, rather than accept `mori.x`'s convention of omitting precisely the interior points of facets from a completed list; this is particularly relevant if `-M` is combined with `-H`.

After the usual polytope input, P^* is displayed in the following manner. If the input is of CWS type, all points of P^* are given in `mori.x`'s standard order (see section 7.2.6). If matrix input is used via `-D`, the points entered by the user are displayed again in the same order, but with the origin appended (if the origin is accidentally entered somewhere in the point list, it is swapped with the last point in the list); possible further polytope points are ignored by the program, hence singularities can be introduced if desired. Then the user is asked for the number of triangulations to be analysed, and afterwards each triangulation should be entered as a line starting with the number of simplices involved in the triangulation, followed by bit sequences encoding these simplices. The number of bits in each sequence should be the number of non–zero lattice points in the displayed list, with 1's indicating that the point belongs to the simplex, and 0's otherwise.

An application to our standard example (7.1) was already demonstrated in section 7.2.15. Consider also the following two-dimensional polytope:

```
palp$ mori.x -MDgm
'#lines #columns' (= 'PolyDim #Points' or '#Points PolyDim'):
2 7
Type the 14 coordinates as dim=2 lines with #pts=7 columns:
    1    0   -1   -1   -1   -1    0
    0    1    2    1    0   -1   -1
2 8
    1    0   -1   -1   -1   -1    0    0
    0    1    2    1    0   -1   -1    0
'#triangulations':
1
1 triangulations:
7 1100000 0110000 0011000 0001100 0000110 0000011 1000001
14 SR-ideal
0000101 0001001 0001010 0010001 0010010 0010100 0100001 0100010
0100100 0101000 1000010 1000100 1001000 1010000
6 MORI GENERATORS / dim(cone)=5
  1 -2  1  0  0  0  0   I:0111011
  0  1 -1  1  0  0  0   I:1101101
  0  0  1 -2  1  0  0   I:1110110
  0  0  0  1 -2  1  0   I:1011111
  0  0  0  0  1 -1  1   I:1100011
  1  0  0  0  0  1 -1   I:0111100
```

Since P^* is just a polygon there is only one maximal triangulation of its boundary. For this reason we have typed `1` after `'#triangulations':`. The triangulation has seven simplices, which are just the line segments along the circumference of the polygon. For instance, the string `0011000` denotes the third simplex containing the points denoted by the third and fourth columns of the polytope matrix, i.e. the points $p_3 = (-1, 2)$ and $p_4 = (-1, 1)$.

As the Mori cone encodes linear relations among seven points in $d = 2$, it is five-dimensional and has four-dimensional facets; there are seven of them, as the lengths of the bit sequences after `I:` indicate. There are six generators. Consider the matrix of incidences whose rows are preceded by `I:`. The second column reads `111011`, i.e. on the second facet of the Mori cone lie five generators. It is easily checked that they satisfy $m_1 + 2m_2 + m_3 = m_5 + m_6$. All other facets contain instead four generators and are hence simplicial.

Acknowledgments

This work is based on the legacy of Maximilian Kreuzer who has been an inspiration for all of us. We are grateful to Benjamin Nill for providing information on several PALP options and for useful remarks. J. Knapp thanks the Vienna University of Technology for hospitality. N.-O. Walliser thanks the École Polytechnique, Paris for hospitality. The work of A.P.Braun was supported by the FWF under grant I192. The work of J. Knapp was supported by World Premier International Research Center Initiative (WPI Initiative), MEXT, Japan.

References

[1] A. Klemm and R. Schimmrigk, "Landau-Ginzburg string vacua," *Nucl.Phys.* **B411** (1994) 559–583, arXiv:hep-th/9204060 [hep-th].

[2] M. Kreuzer and H. Skarke, "No mirror symmetry in Landau-Ginzburg spectra!," *Nucl.Phys.* **B388** (1992) 113–130, arXiv:hep-th/9205004 [hep-th].

[3] V. V. Batyrev, "Dual polyhedra and mirror symmetry for Calabi-Yau hypersurfaces in toric varieties," *J. Algebraic Geom.* **3** no. 3, (1994) 493–535, arXiv:alg-geom/9310003 [alg-geom].

[4] M. Kreuzer and H. Skarke, "On the classification of reflexive polyhedra," *Commun.Math.Phys.* **185** (1997) 495–508, arXiv:hep-th/9512204 [hep-th].

[5] H. Skarke, "Weight systems for toric Calabi-Yau varieties and reflexivity of Newton polyhedra," *Mod.Phys.Lett.* **A11** (1996) 1637–1652, arXiv:alg-geom/9603007 [alg-geom].

[6] A. Avram, M. Kreuzer, M. Mandelberg, and H. Skarke, "Searching for K3 fibrations," *Nucl.Phys.* **B494** (1997) 567–589, arXiv:hep-th/9610154 [hep-th].

[7] M. Kreuzer and H. Skarke, "Calabi-Yau four folds and toric fibrations," *J.Geom.Phys.* **26** (1998) 272–290, arXiv:hep-th/9701175 [hep-th].

[8] A. Avram, M. Kreuzer, M. Mandelberg, and H. Skarke, "The Web of Calabi-Yau hypersurfaces in toric varieties," *Nucl.Phys.* **B505** (1997) 625–640, arXiv:hep-th/9703003 [hep-th].

[9] M. Kreuzer and H. Skarke, "Classification of reflexive polyhedra in three-dimensions," *Adv.Theor.Math.Phys.* **2** (1998) 847–864, arXiv:hep-th/9805190 [hep-th].

[10] M. Kreuzer and H. Skarke, "Complete classification of reflexive polyhedra in four-dimensions," *Adv.Theor.Math.Phys.* **4** (2002) 1209–1230, arXiv:hep-th/0002240 [hep-th].

[11] M. Kreuzer, E. Riegler, and D. A. Sahakyan, "Toric complete intersections and weighted projective space," *J.Geom.Phys.* **46** (2003) 159–173, arXiv:math/0103214 [math-ag].

[12] A. Klemm, M. Kreuzer, E. Riegler, and E. Scheidegger, "Topological string amplitudes, complete intersection Calabi-Yau spaces and threshold corrections," *JHEP* **0505** (2005) 023, arXiv:hep-th/0410018 [hep-th].

[13] E. Riegler, *Toric Geometry and Mirror Symmetry in String Theory*. PhD thesis, Technische Universität Wien, 2004. http://hep.itp.tuwien.ac.at/~kreuzer/pra/RieglerPhD.pdf.

[14] http://en.wikipedia.org/wiki/Palp.

[15] C. T. C. Wall, "Classification problems in differential topology. V. On certain 6-manifolds," *Invent. Math. 1 (1966), 355-374; corrigendum, ibid* **2** (1966) 306.

[16] J. Rambau, "TOPCOM: triangulations of point configurations and oriented matroids," in *Mathematical software (Beijing, 2002)*, A. M. Cohen, X.-S. Gao, and N. Takayama, eds., pp. 330–340. World Sci. Publ., River Edge, NJ, 2002. http://www.zib.de/PaperWeb/abstracts/ZR-02-17.

[17] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, "SINGULAR 3-1-4 — A computer algebra system for polynomial computations,". http://www.singular.uni-kl.de.

[18] W. Stein *et al.*, *Sage Mathematics Software (Version 4.8)*. The Sage Development Team, 2012. http://www.sagemath.org.

[19] various authors, *PALP wiki*. http://palp.itp.tuwien.ac.at/wiki/index.php/Main_Page.

[20] M. Kreuzer and H. Skarke, "PALP: A Package for analyzing lattice polytopes with applications to toric geometry," *Comput.Phys.Commun.* **157** (2004) 87–106, arXiv:math/0204356 [math-sc].

[21] A. P. Braun and N.-O. Walliser, "A New offspring of PALP," arXiv:1106.4529 [math.AG].

[22] W. Fulton, *Introduction to toric varieties*, vol. 131 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ, 1993.

[23] T. Oda, *Convex bodies and algebraic geometry*, vol. 15 of *Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)]*. Springer-Verlag, Berlin, 1988.

[24] D. A. Cox, J. B. Little, and H. K. Schenck, *Toric varieties*, vol. 124 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2011.

[25] H. Skarke, "String dualities and toric geometry: an introduction," *Chaos Solitons Fractals* **10** no. 2-3, (1999) 543–554, arXiv:hep-th/9806059 [hep-th].

[26] M. Kreuzer, "Toric geometry and Calabi-Yau compactifications," *Ukr.J.Phys.* **55** (2010) 613, arXiv:hep-th/0612307 [hep-th].

[27] M. Kreuzer and H. Skarke, *PALP: a Package for Analyzing Lattice Polytopes*. http://hep.itp.tuwien.ac.at/~kreuzer/CY/CYpalp.html.

[28] M. Kreuzer and H. Skarke, "Reflexive polyhedra, weights and toric Calabi-Yau fibrations," *Rev.Math.Phys.* **14** (2002) 343–374, arXiv:math/0001106 [math-ag].

[29] C. Vafa, "String Vacua and Orbifoldized L-G Models," *Mod.Phys.Lett.* **A4** (1989) 1169.

[30] K. A. Intriligator and C. Vafa, "Landau-Ginzburg Orbifolds," *Nucl.Phys.* **B339** (1990) 95–120.

[31] M. Kreuzer and B. Nill, "Classification of toric Fano 5-folds," *Adv. Geom.* **9** no. 1, (2009) 85–97, arXiv:math/0702890 [math].

[32] M. Kreuzer and B. Nill, *Classification of toric Fano 5-folds (data supplement)*, 2007. http://hep.itp.tuwien.ac.at/~kreuzer/CY/math/0702890/.

[33] V. Batyrev and M. Kreuzer, "Constructing new Calabi-Yau 3-folds and their mirrors via conifold transitions," *Adv. Theor. Math. Phys.* **14** no. 3, (2010) 879–898, arXiv:0802.3376 [math.AG]. http://projecteuclid.org/getRecord?id=euclid.atmp/1309526468.

[34] V. Batyrev and M. Kreuzer, *Constructing new Calabi-Yau 3-folds and their mirrors via conifold transitions (data supplement)*, 2008. http://hep.itp.tuwien.ac.at/~kreuzer/CY/math/0802/.

[35] V. Batyrev and M. Kreuzer, "Conifold degenerations of fano 3-folds as hypersurfaces in toric varieties," (2012) , arXiv:1203.6058 [math.AG].

[36] V. Batyrev, B. Nill, and M. Kreuzer, *Fano hypersurfaces with conifold singularities (data supplement)*, 2002. http://hep.itp.tuwien.ac.at/~kreuzer/CY/math/Fano/Fano.html.

[37] M. Kreuzer, *Reflexive2x.4d.gz (data file)*, 2002. http://hep.itp.tuwien.ac.at/~kreuzer/CY/math/Fano/Fano.html.

[38] B. Nill, *Gorenstein toric Fano varieties*. PhD thesis, Universität Tübingen, 2005. http://tobias-lib.uni-tuebingen.de/dbt/volltexte/2005/1888/.

[39] H. Skarke, "How to Classify Reflexive Gorenstein Cones," arXiv:1204.1181 [hep-th].

[40] M. Kreuzer and H. Skarke, *Calabi Yau data*. http://hep.itp.tuwien.ac.at/~kreuzer/CY/.

[41] L. Borisov, "Towards the Mirror Symmetry for Calabi-Yau Complete intersections in Gorenstein Toric Fano Varieties ," arXiv:alg-geom/9310001 [alg-geom].

[42] V. V. Batyrev and L. A. Borisov, "On Calabi-Yau complete intersections in toric varieties," arXiv:alg-geom/9412017 [alg-geom].

[43] V. Batyrev and B. Nill, "Combinatorial aspects of mirror symmetry," in *Integer points in polyhedra—geometry, number theory, representation theory, algebra, optimization, statistics*, vol. 452 of *Contemp. Math.*, pp. 35–66. Amer. Math. Soc., Providence, RI, 2008. arXiv:math/0703456 [math.CO].

[44] V. V. Batyrev and L. A. Borisov, "Mirror duality and string-theoretic Hodge numbers," *Invent. Math.* **126** no. 1, (1996) 183–203, arXiv:alg-geom/9509009 [alg-geom].

[45] V. Braun, M. Kreuzer, B. A. Ovrut, and E. Scheidegger, "Worldsheet Instantons and Torsion Curves, Part B: Mirror Symmetry," *JHEP* **0710** (2007) 023, arXiv:0704.0449 [hep-th].

[46] T. Oda and H. S. Park, "Linear Gale transforms and Gel′fand-Kapranov-Zelevinskij decompositions," *Tohoku Math. J. (2)* **43** no. 3, (1991) 375–399.

[47] P. Berglund, S. H. Katz, and A. Klemm, "Mirror symmetry and the moduli space for generic hypersurfaces in toric varieties," *Nucl.Phys.* **B456** (1995) 153–204, arXiv:hep-th/9506091 [hep-th].

[48] L. J. Billera, P. Filliman, and B. Sturmfels, "Constructions and complexity of secondary polytopes," *Adv. Math.* **83** no. 2, (1990) 155–179.

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

[50] F. Hirzebruch, *Topological methods in algebraic geometry*. Classics in Mathematics. Springer-Verlag, Berlin, 1995.

[51] Wolfram Research, Inc., "Mathematica edition: Version 8.0,". http://www.wolfram.com/.

[52] J. Knapp, M. Kreuzer, C. Mayrhofer, and N.-O. Walliser, "Toric Construction of Global F-Theory GUTs," *JHEP* **1103** (2011) 138, arXiv:1101.4908 [hep-th].