NORMAL FORMS OF CONVEX LATTICE POLYTOPES

ROLAND GRINIS AND ALEXANDER M. KASPRZYK

ABSTRACT. We describe an algorithm for determining whether two convex polytopes P and Q, embedded in a lattice, are isomorphic with respect to a lattice automorphism. We extend this to a method for determining if P and Q are equivalent, i.e. whether there exists an affine lattice automorphism that sends P to Q. Methods for calculating the automorphism group and affine automorphism group of P are also described.

An alternative strategy is to determine a normal form such that P and Q are isomorphic if and only if their normal forms are equal. This is the approach adopted by Kreuzer and Skarke in their PALP software. We describe the Kreuzer–Skarke method in detail, and give an improved algorithm when P has many symmetries. Numerous examples, plus two appendices containing detailed pseudo-code, should help with any future reimplementations of these techniques. We conclude by explaining how to define and calculate the normal form of a Laurent polynomial.

1. INTRODUCTION

Determining whether two convex polytopes P and Q, embedded in a lattice Λ, are isomorphic with respect to a lattice automorphism is a fundamental computational problem. For example, in toric geometry lattice polytopes form one of the key constructions of projective toric varieties, and any classification must somehow address the issue of whether there exists an automorphism of the underlying lattice sending P to Q. In general, any isomorphism problem can be solved in one of two ways: on a case-by-case basis by constructing an explicit isomorphism between the two objects, or by determining a normal form for each isomorphism class.

The first approach – dynamically constructing a lattice-preserving isomorphism in $\mathrm{GL}_n(\mathbb{Z})$ between the two polytopes – is discussed in §2. We describe one possible way to determine isomorphism of polytopes via the labelled face graph $\mathscr{G}(P)$ (see §2.1). This has the advantage that it works equally well for rational polytopes and for polytopes of non-zero codimension. By reducing the problem to a graph isomorphism question, well-developed tools such as Brendan McKay's NAUTY software [McK81, McK] can then be applied.

Because our approach to isomorphism testing works equally well for rational polytopes, we are able to answer when two polytopes are equivalent, i.e. when there exists an isomorphism $B \in \mathrm{GL}_n(\mathbb{Z})$ and lattice translation $c \in \Lambda$ such that $PB + c = Q$. This is discussed in §2.4. We can also calculate the automorphism group $\mathrm{Aut}(P) \leq \mathrm{GL}_n(\mathbb{Z})$ of P: this is a subgroup of the automorphism group of $\mathscr{G}(P)$, as explained in §2.5. Since our methods make no assumptions on the codimension of P, by considering the automorphism group of $P \times \{1\}$ in $\Lambda \times \mathbb{Z}$ we are able to calculate the group of affine automorphisms $\mathrm{AffAut}(P) \leq \mathrm{GL}_n(\mathbb{Z}) \ltimes \Lambda$. As an illustration of our methods, we calculate the order of the automorphism group for each of the 473,800,776 four-dimensional reflexive polytopes [KS00]: see Table 1.

The second approach – to compute a normal form $NF(P)$ for each isomorphism class – is discussed in §3. This is the approach adopted by Kreuzer and Skarke in their PALP software [KS04], and was used to construct the classification of three- and four-dimensional reflexive polytopes [KS98, KS00]. Briefly, row and column permutations are applied to the vertex–facet pairing matrix PM of P, placing it in a form $PM^{\max}$ that is maximal with respect to a certain ordering. This in turn defines an order in which to list the vertices of P; the choice of basis is fixed by taking the Hermite normal form. In §3.3 we address how this can be modified to give an affine normal form for P, and in §3.4 we describe how PALP applies an additional reordering of the columns of $PM^{\max}$ before computing the normal form. The PALP source code for computing $NF(P)$ is analyzed in detail in Appendix A.

In §4 we address the problem of calculating $PM^{\max}$. We describe an inductive algorithm which attempts to exploit automorphisms of the matrix in order to simplify the calculation; pseudo-code is given in Appendix B. Applying our algorithm to smooth Fano polytopes [Øbr07], which often have large numbers of symmetries, illustrates the advantage of this approach: see §4.1 and Table 2. We end by giving, in §5, an application of normal form to Laurent polynomials.

A note on implementation. The algorithms described in §2 were implemented using MAGMA in 2008 and officially released as part of MAGMA V2.16 [BCP97, BBK09]; PALP normal form was introduced by Kreuzer and Skarke in their PALP software [KS04] and reimplemented natively in MAGMA V2.18 by the authors. The MAGMA algorithms[1], including the reimplementation of PALP normal form, have recently been ported to the SAGE project [S+] by Samuel Gonshaw[2], assisted by Tom Coates and the second author, and should appear in the 5.6.0 release.

2. ISOMORPHISM TESTING VIA THE FACE GRAPH

Conventions. Throughout this section we work with very general convex polytopes; we assume only that $P \subset \Lambda_{\mathbb{Q}} := \Lambda \otimes \mathbb{Q}$ is a (non-empty) rational convex polytope, not necessarily of maximum dimension in the ambient lattice Λ. The dual lattice $\mathrm{Hom}(\Lambda, \mathbb{Z})$ is denoted by Λ^*.

Given two polytopes P and P', how can we decide whether they are isomorphic and, if they are, how can we construct an isomorphism between them? There are, of course, some obvious checks that can quickly provide a negative answer. We give a few examples, although this list is far from comprehensive.

- Do the dimensions of the polytopes agree?
- Does P contain the origin in its relative interior? Is the same true for P'?

---

[1]Users of MAGMA can freely view and edit the package code. The relevant files are contained in the subdirectory `package/Geometry/ToricGeom/polyhedron/`.

[2]Gonshaw's implementation is available from http://trac.sagemath.org/sage_trac/ticket/13525.

- Are P and P' both lattice polytopes?
- Are the f-vectors of P and P' equal?
- Do P and P' have the same number of primitive vertices?
- Are P and P' simplicial? Are they simple?
- If P is of codimension one then there exists a unique hyperplane $H \subset \Lambda_\mathbb{Q}$ containing P, where $H = \{v \in \Lambda_\mathbb{Q} \mid \langle v, u \rangle = k\}$ for some non-negative rational value k and primitive dual lattice point $u \in \Lambda^*$. In particular, k is invariant under change of basis. Does k agree for both P and P'?
- If P is of maximum dimension, any facet F can be expressed in the form $F = \{v \in P \mid \langle v, u_F \rangle = -c_F\}$, where $u_F \in \Lambda^*$ is a primitive inward-pointing vector normal to F, and $c_F \in \mathbb{Q}$ is the lattice height of F over the origin. The value of c_F is invariant under change of basis. Do the facet heights of P and P' agree, up to permutation?
- If P is a rational polytope, let r_P be the smallest positive integer such that the dilation $r_P P$ is a lattice polytope. Do r_P and $r_{P'}$ agree?

Remark 2.1. From a computational point of view, the intention with the above list is to suggest tests that are easy to perform. We assume that data such as the vertices and supporting hyperplanes of P have already been calculated. Some computations, such as finding the f-vector, are more involved, but since the calculations will be required in what follows it seems sensible to use them at this stage.

In practice a number of other invariants may already be cached and could also be used: the volume $\mathrm{Vol}(P)$ or boundary volume $\mathrm{Vol}(\partial P)$; the number of lattice points $|P \cap \Lambda|$ or boundary lattice points $|\partial P \cap \Lambda|$; the Ehrhart δ-vector; information about the polar polyhedron P^*. In particular cases some of this additional data may be easy to calculate; in general they are usually more time-consuming to compute than the isomorphism test described below.

Remark 2.2. There are a few potential catches for the unwary when considering rational polytopes with $\dim P < \dim \Lambda$. For example, care needs to be taken when defining the supporting hyperplanes. Also, the notion of (normalised) volume $\mathrm{Vol}(P)$ requires some attention: the affine sublattice $\mathrm{aff}(P) \cap \Lambda$ may be empty, forcing us to either accept that $\mathrm{Vol}(P)$ can be undefined, or to employ interpolation. There is a natural dichotomy between those polytopes whose affine span contains the origin and those where $0 \notin \mathrm{aff}(P)$. In the latter case, it is often better to consider the cone $C_P := \mathrm{cone}(P)$ equipped with an appropriate grading such that dilations of P can be realised by taking successive slices through C_P.

2.1. **The labelled face graph.** In order to determine isomorphism we make use of the face graph $G(P)$ of P.

Definition 2.3. Let P be an n-dimensional polytope with f-vector $(f_{-1}, f_0, \ldots, f_n)$, where f_k denotes the number of k-faces of P. By convention we set $f_{-1} = f_n = 1$, representing, respectively, the empty set $\varnothing$ and the polytope P. The *face graph* $G(P)$ is the graph consisting of $f_{-1} + f_0 + \ldots + f_n$ vertices, where each vertex v corresponds to a face F_v. Two vertices v and v' are connected by an edge if and only if $F_{v'} \subset F_v$ and $\dim F_{v'} = \dim F_v + 1$. Here the dimension of the empty face $\varnothing$ is taken to be -1.

The face graph of a polytope is completely determined by the vertex–facet relations, and is the standard tool for determining combinatorial isomorphism of polytopes. We augment $G(P)$ by assigning labels to the vertices determined by some invariants of the corresponding

face. Reducing a symmetry problem to the study of a (labelled) graph is a well-established computational technique: see, for example, [KS03, Pug05, MdlBW09, BSP$^+$12]. The intention is to decorate the graph with data capturing how P lies in the underlying lattice Λ. To that end, we make the following definition.

Definition 2.4. For a point $v \in \Lambda_{\mathbb{Q}}$, let $u \in \Lambda$ be the unique primitive lattice point such that $v = \lambda u$ for some non-negative value λ (set $u = 0$, $\lambda = 0$ if $v = 0$). We define $\tilde{v}$ to be given by $\lceil \lambda \rceil u$, i.e. $\tilde{v}$ is the first lattice point after or equal to v on the ray defined by v. Let $P \subset \Lambda_{\mathbb{Q}}$ be a polytope with vertices $\mathcal{V}(P)$. Then the *index* $|P : \Lambda|$ of P is the index of the sublattice generated by $\{\tilde{u} \mid u \in \mathcal{V}(P)\}$ in $\mathrm{span}(P) \cap \Lambda$.

Definition 2.5. Let P be an n-dimensional polytope with face graph $G(P)$. To each vertex v of $G(P)$ we assign the label

$$\begin{cases} (\dim F_v), & \text{if } F_v = \varnothing \text{ or } F_v = P; \\ (\dim F_v, |F_v : \Lambda|), & \text{otherwise.} \end{cases}$$

We denote this labelled graph by $\mathscr{G}(P)$.

Remark 2.6. In place of the index $|F_v : \Lambda|$, it is tempting to use the volume $\mathrm{Vol}(F_v)$. However, computing the index is basic linear algebra, whereas computing the volume is generally difficult.

2.2. **Additional labels.** When P contains the origin strictly in its interior, we can make use of the *special facets*. Recall from [Øbr07, §3.1] that a facet F is said to be special if $u \in \mathrm{cone}(F)$, where $u := \sum_{v \in \mathcal{V}(P)} v$ is the sum of the vertices of P. Since P contains the origin, there exists at least one special facet; we can extend the labelling to indicate which vertices of $\mathscr{G}(P)$ correspond to a special facet.

Example 2.7. The polytope $P := \mathrm{conv}\{(1,0), (0,1), (-2,-3)\}$ and its labelled face graph $\mathscr{G}(P)$ are depicted below. In the graph, the top-most vertex represents P and the bottom vertex $\varnothing$. The sum of the vertices is $(-1,-2)$, so there is a unique special facet: the edge joining vertices $(1,0)$ and $(-2,-3)$ of index three, labelled $(1,3,1)$ in $\mathscr{G}(P)$. The edge joining vertices $(1,0)$ and $(0,1)$ is of index one and labelled $(1,1,0)$; the remaining edge is of index two and labelled $(1,2,0)$. The final entry of each facet label is used to indicate whether this is a special facet.
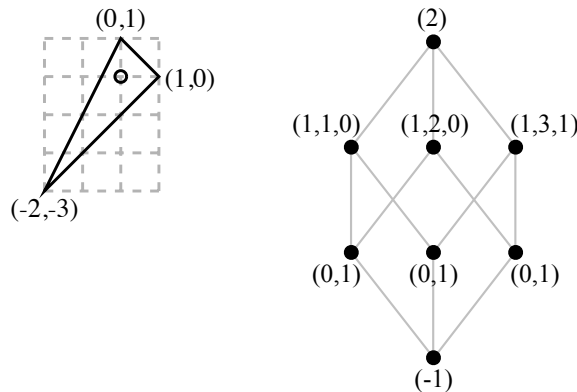
If P is a rational polytope, the vertices of P provide an augmentation to the labelling of $\mathscr{G}(P)$. For any vertex $v \in \mathcal{V}(P)$ there exists a primitive lattice point $u \in \Lambda$ and a non-negative rational value λ such that $v = \lambda u$. Since λ is invariant under change of basis, the corresponding labels

can be extended with this information. (Note that $\lambda = |v : \Lambda|$ when v is a lattice point, so this only provides additional information in the rational case.)

We do not claim that these are the only easily-computed invariants that can be associated with $\mathscr{G}(P)$. Other possibilities include encoding the linear relations between the vertices $\mathcal{V}(P)$ of P in the graph labelling, and, in the maximum dimensional case, adding information about the lattice height of the supporting hyperplanes for each face.

2.3. **Recovering the isomorphism.** We now describe our algorithm for computing an isomorphism between two polytopes P and P'. The initial step is to normalise the polytopes. If P and P' are not of maximum dimension in the ambient lattice Λ, then we first restrict to the sublattice $\mathrm{span}(P) \cap \Lambda$ (and, respectively, $\mathrm{span}(P') \cap \Lambda$). It is possible that, even after restriction, P and P' are of codimension one. In that case, we work with the convex hull $\mathrm{conv}(P \cup \{0\})$ (and similarly for P'). The important observations are that, after normalisation, P is of maximum dimension, and that there exists at least one facet F_0 of P such that $0 \notin \mathrm{aff}(F_0)$.

Now we calculate an arbitrary graph isomorphism $\phi : \mathscr{G}(P) \to \mathscr{G}(P')$. By restricting to the vertices of $\mathscr{G}(P)$ corresponding to the vertices $\mathcal{V}(P)$ of P, ϕ induces a map from the vertices of P to the vertices of P'. The two polytopes P and P' are isomorphic only if ϕ exists, and any isomorphism $\Phi : \Lambda \to \Lambda$ mapping P to P' can be factored as $\phi \circ \chi$, where $\chi \in \mathrm{Aut}(\mathscr{G}(P))$.

It remains to decide whether a particular choice of $\chi \in \mathrm{Aut}(\mathscr{G}(P))$ determines a lattice isomorphism $\phi \circ \chi : \Lambda \to \Lambda$ sending P to P'. For this we make use of the facet F_0. By construction F_0 is of codimension one, and does not lie in a hyperplane containing the origin. Hence there exists a choice of vertices $v_1, \ldots, v_n$ of F_0 which generate $\Lambda_{\mathbb{Q}}$ (over $\mathbb{Q}$). Denote the image of v_i in P' by v_i', and consider the $n \times n$ matrices V and V' whose rows are given by, respectively, the v_i and the v_i'. In order for $\phi \circ \chi$ to be a lattice map we require that $B := V^{-1}V' \in \mathrm{GL}_n(\mathbb{Z})$. In order for this to be an isomorphism from P to P' we require that $\{vB \mid v \in \mathcal{V}(P)\} = \mathcal{V}(P')$.

Remark 2.8. We make two brief observations. First, in practice the automorphism group $\mathrm{Aut}(\mathscr{G}(P))$ is often small. Second, it is an easy exercise in linear algebra to undo our normalisation process, lifting B back to act on the original polytope.

2.4. **Testing for equivalence.** Recall that two polytopes $P, P' \subset \Lambda_{\mathbb{Q}}$ are said to be *equivalent* if there exists an isomorphism $B \in \mathrm{GL}_n(\mathbb{Z})$ and a translation $c \in \Lambda$ such that $PB + c = P'$.

Definition 2.9. Let $\mathcal{V}(P)$ be the set of vertices of a polytope P. Then the *vertex average* of P is the point

$$b_P := \frac{1}{|\mathcal{V}(P)|} \sum_{v \in \mathcal{V}(P)} v \in \Lambda_{\mathbb{Q}}.$$

Two polytopes P and P' are equivalent if and only if $b_P - b_{P'} \in \Lambda$ and $P - b_P$ is isomorphic to $P' - b_{P'}$.

Example 2.10. Consider the simplices

$$P := \mathrm{conv}\{(0,0,0), (2,1,1), (1,2,1), (1,1,2)\},$$
$$P' := \mathrm{conv}\{(0,1,2), (1,0,0), (3,1,4), (4,2,6)\}.$$

The vertex averages are $b_P = (1, 1, 1)$ and $b_{P'} = (2, 1, 3)$, and $(P - b_P)B = P' - b_{P'}$, where

$$B := \begin{pmatrix} 2 & 1 & 3 \\ -2 & 0 & -1 \\ 1 & 0 & 1 \end{pmatrix}$$

Hence P and P' are equivalent.

2.5. **Determining the automorphism group of a polytope.** We can use the labelled face graph $\mathscr{G}(P)$ to compute the automorphism group $\mathrm{Aut}(P)$. We simply use the elements χ of $\mathrm{Aut}(\mathscr{G}(P))$ to construct $\mathrm{Aut}(P) \leq \mathrm{GL}_n(\mathbb{Z})$. Notice that there is no requirement that P is of maximum dimension in the ambient lattice Λ. Given this, we can also compute the affine automorphism group $\mathrm{AffAut}(P)$. Begin by embedding P at height one in the lattice $\Lambda \times \mathbb{Z}$ (equivalently, consider the cone C_P spanned by P with appropriate grading). We refer to this embedded image of P as $\tilde{P}$. The action of the automorphism group $\mathrm{Aut}(\tilde{P})$ on $\tilde{P}$ restricts to an action on P, realising the full group of affine lattice automorphisms of P. A detailed discussion of polyhedral symmetry groups and their applications can be found in [BEK84, BDSS09, BSP$^+$12].

Example 2.11. Let P be the three-dimensional simplicial polytope with seven vertices given by $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$, $(0, 0, 1)$, $(1, 1, 0)$, $(0, -1, -1)$. This is sketched below; the f-vector is $(1, 7, 15, 10, 1)$. The index $|F : \Lambda|$ of each face F is one (in fact P is a smooth Fano polytope[3]), and P has four special facets (the four facets incident to the vertex $(1, 0, 0)$). The resulting labelled graph $\mathscr{G}(P)$ has automorphism group of order four, however $\mathrm{Aut}(P)$ has order two, and is generated by the involution $(0, 0, 1) \mapsto (0, -1, -1)$.
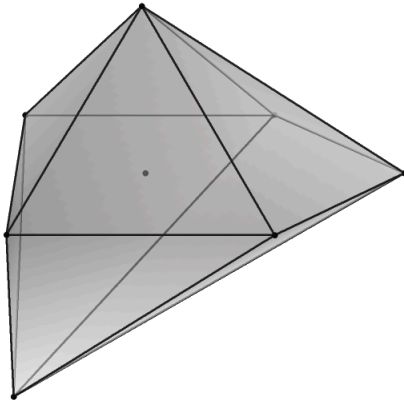
Example 2.12. The four-dimensional centrally symmetric polytope P with vertices

$$\pm (1, 0, 0, 0), \pm(0, 1, 0, 0), \pm(0, 0, 1, 0), \pm(0, 0, 0, 1),$$
$$\pm (1, -1, 0, 0), \pm(1, 0, -1, 0), \pm(1, 0, 0, -1), \pm(0, 1, -1, 0), \pm(0, 1, 0, -1),$$
$$\pm (1, 0, -1, -1), \pm(0, 1, -1, -1), \pm(1, 1, -1, -1)$$

is the reflexive realisation of the 24-cell, with f-vector $(1, 24, 96, 96, 24, 1)$. It is unique amongst all 473,800,776 reflexive polytopes in having $|\mathrm{Aut}(P)| = 1152$; in fact $\mathrm{Aut}(P)$ is isomorphic to the Weyl group $W(F_4)$. In particular, P must be self-dual. The number of four-dimensional reflexive polytopes with $|\mathrm{Aut}(P)|$ of given size are recorded in Table 1.

---

[3]Smooth Fano polytope number 13 in the Graded Ring Database [BK].

$|\mathrm{Aut}(P)|$	$\#P$	$|\mathrm{Aut}(P)|$	$\#P$
1	467705246	36	11
2	5925190	48	79
3	1080	64	5
4	151416	72	10
6	8218	96	22
8	6935	120	2
10	4	128	2
12	1509	144	2
16	756	240	4
18	2	288	2
20	4	384	6
24	247	1152	1
32	23		

TABLE 1. The number $\#P$ of four-dimensional reflexive polytopes with automorphism group of size $|\mathrm{Aut}(P)|$.

Example 2.13. Let $P = \mathrm{conv}\{(0,0),(1,0),(0,1)\}$ be the empty simplex in $\mathbb{Z}^2$. Then $\mathrm{Aut}(P)$ is of order two, corresponding to reflection in the line $x = y$. To compute the affine automorphism group $\mathrm{AffAut}(P)$ of P, we consider $\tilde{P} = \mathrm{conv}\{(0,0,1),(1,0,1),(0,1,1)\}$. The group $\mathrm{Aut}(\tilde{P})$ is of order six, generated by

$$\begin{pmatrix} -1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

The first generator corresponds to the involution exchanging the vertices $(0,0)$ and $(1,0)$ of P, whilst the second generator corresponds to rotation of P about its barycentre $(1/3, 1/3)$, given by

$$(x,y) \mapsto (x - 1/3, y - 1/3)\begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix} + (1/3, 1/3) = (x,y)\begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix} + (0,1).$$

3. NORMAL FORMS

The method for determining isomorphism adopted by Kreuzer and Skarke in the software package PALP [KS04] is to generate a *normal form* for the polytope P. We shall briefly sketch their approach. Their algorithm is described in detail in Appendix A.

Throughout we require that the polytope $P \subset \Lambda_{\mathbb{Q}}$ is a lattice polytope of maximum dimension. It is essential to the algorithm that the vertices are lattice points; one could dilate a rational polytope by a sufficiently large factor to overcome this restriction, but in practice the resulting large vertex coefficients can cause computational problems of their own. Let n denote the dimension of P, and n_v be the number of vertices $\mathcal{V}(P)$. We can represent P by an $n \times n_v$ matrix V whose columns are given by the vertices. Obviously V is uniquely defined only up to permutations $\sigma \in S_{n_v}$ of the columns.

Given any matrix V with integer entries, we can compute its Hermite normal form $H(V)$. This has the property that, for all $B \in \mathrm{GL}_n(\mathbb{Z})$, $H(V) = H(V \cdot B)$, however permuting the

columns of V will result in different Hermite normal forms. Naïvely one could define the normal form $\mathrm{NF}(P)$ of P to be

$$\min \left\{ H(\sigma V) \mid \sigma \in S_{n_v} \right\},$$

where σV denotes the matrix obtained by permuting the columns of V by σ, and the minimum is taken with respect to some ordering of the set of $n \times n_v$ integer matrices (say, lexicographic ordering). Unfortunately the size of S_{n_v} is too large for this to be a practical algorithm.

3.1. **The pairing matrix.** The key to making this approach tractable is the vertex–facet pairing matrix.

Definition 3.1. Let P be a lattice polytope with vertices v_j, and let $(w_i, c_i) \in \Lambda^* \times \mathbb{Z}$ define the supporting hyperplanes of P; each w_i is a primitive inward-pointing vector normal to the facet F_i of P, such that $\langle w_i, v \rangle = -c_i$ for all $v \in F_i$. The *vertex–facet pairing matrix* PM is the $n_f \times n_v$ matrix with integer coefficients

$$PM_{ij} := \langle w_i, v_j \rangle + c_i.$$

In other words, the ij-th entry of PM correspond to the lattice height of v_j above the facet F_i. This is clearly invariant under the action of $\mathrm{GL}_n(\mathbb{Z})$. It is also invariant under (lattice) translation of P. Permuting the vertices of P corresponds to permuting the columns of PM, and permuting the facets of P corresponds to permuting the rows of PM. Thus there is an action of $S_{n_f} \times S_{n_v}$ on PM: given $\sigma = (\sigma_f, \sigma_v) \in S_{n_f} \times S_{n_v}$,

$$(\sigma PM)_{ij} := PM_{\sigma_f(i), \sigma_v(j)}.$$

There is a corresponding action on V given by restriction:

$$\sigma V := \sigma_v V.$$

Let $PM^{\max}$ denote the maximal matrix (ordered lexicographically) obtained from PM by the action of $S_{n_f} \times S_{n_v}$, realised by some element $\sigma_{\max}$. Let $\mathrm{Aut}(PM^{\max}) \leq S_{n_f} \times S_{n_v}$ be the automorphism group of $PM^{\max}$. Then:

Definition 3.2. The *normal form* of P is

$$\mathrm{NF}(P) = \min \left\{ H(\sigma \circ \sigma_{\max} V) \mid \sigma \in \mathrm{Aut}(PM^{\max}) \right\}.$$

Remark 3.3. Let G be the group generated by the action of $\mathrm{Aut}(PM)$ on the columns of PM. Then $\mathrm{Aut}(P) \leq G$. Hence we have an alternative method for constructing the automorphism group when P is a lattice polytope of maximum dimension.

Example 3.4. Consider the three-dimensional polytope P with vertices $(1,0,0)$, $(0,1,0)$, $(0,0,1)$, $(-1,0,1)$, $(0,1,-1)$, $(0,-1,0)$, $(0,0,-1)$; P is isomorphic to the polytope in Example 2.11 via the change of basis

$$\begin{pmatrix} 0 & -1 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}.$$

With the vertices in the order written above, and some choice of order for the facets, the vertex–facet pairing matrix is given by

$$
PM = \begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 2 & 2 \\
0 & 0 & 0 & 1 & 1 & 2 & 2 \\
2 & 0 & 1 & 0 & 0 & 2 & 1 \\
0 & 0 & 1 & 2 & 0 & 2 & 1 \\
0 & 2 & 0 & 1 & 3 & 0 & 2 \\
1 & 2 & 0 & 0 & 3 & 0 & 2 \\
0 & 1 & 2 & 3 & 0 & 1 & 0 \\
0 & 2 & 2 & 3 & 1 & 0 & 0 \\
3 & 2 & 2 & 0 & 1 & 0 & 0 \\
3 & 1 & 2 & 0 & 0 & 1 & 0
\end{pmatrix}.
$$

The maximum vertex–facet pairing matrix is

$$
PM^{\max} = \begin{pmatrix}
3 & 2 & 2 & 1 & 0 & 0 & 0 \\
3 & 2 & 2 & 0 & 1 & 0 & 0 \\
1 & 2 & 0 & 3 & 0 & 2 & 0 \\
1 & 2 & 0 & 0 & 3 & 2 & 0 \\
1 & 0 & 2 & 1 & 0 & 0 & 2 \\
1 & 0 & 2 & 0 & 1 & 0 & 2 \\
0 & 1 & 0 & 3 & 0 & 2 & 1 \\
0 & 1 & 0 & 0 & 3 & 2 & 1 \\
0 & 0 & 1 & 2 & 0 & 1 & 2 \\
0 & 0 & 1 & 0 & 2 & 1 & 2
\end{pmatrix},
$$

realised by, for example, the permutation $((1\ 5\ 2\ 6)(3\ 9)(4\ 10\ 7\ 8), (1\ 4\ 5)(3\ 6\ 7))$ of PM. The automorphism group of $PM^{\max}$ is of order two, generated by

$$((1\ 2)(3\ 4)(5\ 6)(7\ 8)(9\ 10), (4\ 5)).$$

We see that $\mathrm{NF}(P)$ is equal to

$$
\begin{pmatrix}
1 & 0 & 1 & 0 & -1 & -1 & 0 \\
0 & 1 & -1 & 0 & 1 & 1 & -1 \\
0 & 0 & 0 & 1 & -1 & 0 & 0
\end{pmatrix},
$$

corresponding to the sequence of vertices $(1,0,0)$, $(0,1,0)$, $(1,-1,0)$, $(0,0,1)$, $(-1,1,-1)$, $(-1,1,0)$, and $(0,-1,0)$. In this example $\mathrm{Aut}(PM^{\max}) \cong \mathrm{Aut}(\mathrm{NF}(P))$, and acts by exchanging the vertices $(0,0,1)$ and $(-1,1,-1)$.

Example 3.5. Let $P := \mathrm{conv}\{(-1,-2,-2),(1,0,0),(0,2,1),(0,0,1)\}$ be a three-dimensional reflexive polytope. This has

$$
PM^{\max} = \begin{pmatrix}
4 & 0 & 0 & 0 \\
0 & 4 & 0 & 0 \\
0 & 0 & 4 & 0 \\
0 & 0 & 0 & 4
\end{pmatrix},
$$

with $\mathrm{Aut}(PM^{\max}) \cong S_4$ of order 24. However, $|\mathrm{Aut}(P)| = 8$; with ordering as above, the action on the vertices is given by the permutation group with generators $(1\ 4\ 2\ 3)$ and $(3\ 4)$.

3.2. Lattice polytopes of non-zero codimension.

Suppose that P is a lattice polytope such that $\dim P < \dim \Lambda$. We can still define a normal form: how we proceed depends on whether $0 \in \operatorname{aff}(P)$.

First suppose that $0 \in \operatorname{aff}(P)$, so that $\operatorname{aff}(P) = \operatorname{span}(P)$. Set $d = \dim P$. We restrict P to the sublattice $\operatorname{span}(P) \cap \Lambda \cong \mathbb{Z}^d$ and calculate the normal form there. The result can be embeded back into Λ via

$$(a_1, \ldots, a_d) \mapsto (0, \ldots, 0, a_1, \ldots, a_d).$$

Now suppose that $0 \notin \operatorname{aff}(P)$. In this case we consider the polytope $P_0 := \operatorname{conv}(P \cup \{0\})$. The normal form $\operatorname{NF}(P_0)$ can be calculated and then the origin discarded.

Example 3.6. Let $P := \operatorname{conv}\{(-1,1,1,0),(1,1,1,1),(0,0,0,-1)\}$ be a lattice polygon of codimension two. The three-dimensional sublattice $\operatorname{span}(P_0) \cap \mathbb{Z}^4$ has generators $(1,0,0,0)$, $(0,1,1,0)$, and $(0,0,0,1)$. Let $\varphi : \mathbb{Z}^3 \to \mathbb{Z}^4$ be the embedding given by right multiplication by the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Then $\varphi^* P_0$ has vertices $(-1,1,0)$, $(1,1,1)$, $(0,0,-1)$, and $(0,0,0)$, with normal form given by $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(1,1,2)$. Hence $\operatorname{NF}(P)$ corresponds to the vertices $(0,1,0,0)$, $(0,0,1,0)$, and $(0,1,1,2)$. In fact P is isomorphic to $\operatorname{NF}(P)$ via the change of basis

$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & -2 \end{pmatrix} \in \operatorname{GL}_4(\mathbb{Z}).$$

3.3. Affine normal form.

The normal form can be adapted to give an *affine normal form* $\operatorname{AffNF}(P)$ such that $\operatorname{AffNF}(P) = \operatorname{AffNF}(P')$ if and only if polytopes P and P' are equivalent. One could simply define

$$\operatorname{AffNF}(P) := \min \{\operatorname{NF}(P - v) \mid v \in \mathcal{V}(P)\}.$$

However, since the relative height of a vertex over a facet is unchanged by lattice translation, we have that $PM^{\max}$ is invariant. Hence

$$\operatorname{AffNF}(P) = \min \{H(\sigma \circ \sigma_{\max}(V - v)) \mid \sigma \in \operatorname{Aut}(PM^{\max}), v \in \mathcal{V}(P)\}.$$

Example 3.7. Returning to the polytope in Example 3.4 we obtain

$$\operatorname{AffNF}(P) = \begin{pmatrix} 0 & 1 & 0 & 0 & 3 & 2 & 1 \\ 0 & 0 & 1 & 0 & 2 & 1 & 2 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \end{pmatrix}.$$

3.4. The Palp normal form.

Kreuzer and Skarke's PALP normal form applies an additional modification to the order of the columns of the maximum vertex–facet pairing matrix $PM^{\max}$. For any $n_f \times n_v$ matrix M, let $c_M(j) := \max \{M_{ij} \mid 1 \le i \le n_f\}$, and $s_M(j) := \sum_{i=1}^{n_f} M_{ij}$, where $1 \le j \le n_v$. The following pseudo-code describes how the columns of $PM^{\max}$ (or, equivalently, the vertices of P) are rearranged.

$M \leftarrow PM^{\max}$
for $i = 1$ **to** n_v **do**
    $k \leftarrow i$
    for $j = i + 1$ **to** n_v **do**

$\qquad$ **if** $c_M(j) < c_M(k) \vee (c_M(j) = c_M(k) \wedge s_M(j) < s_M(k))$ **then**

$\qquad\qquad k \leftarrow j$

$\qquad$ **end if**

$\quad$ **end for**

$\quad M \leftarrow \mathrm{SwapColumn}(M, i, k)$

end for

Example 3.8. We revisit Example 3.4. In this case, $PM^{\max}$ is modified by applying the permutation $(1\ 6\ 3\ 2)(4\ 7)$ to the columns, giving

$$
\begin{pmatrix}
2 & 2 & 0 & 0 & 0 & 3 & 1 \\
2 & 2 & 0 & 0 & 1 & 3 & 0 \\
2 & 0 & 2 & 0 & 0 & 1 & 3 \\
2 & 0 & 2 & 0 & 3 & 1 & 0 \\
0 & 2 & 0 & 2 & 0 & 1 & 1 \\
0 & 2 & 0 & 2 & 1 & 1 & 0 \\
1 & 0 & 2 & 1 & 0 & 0 & 3 \\
1 & 0 & 2 & 1 & 3 & 0 & 0 \\
0 & 1 & 1 & 2 & 0 & 0 & 2 \\
0 & 1 & 1 & 2 & 2 & 0 & 0
\end{pmatrix}.
$$

The resulting PALP normal form corresponds to the sequence of vertices $(1,0,0)$, $(0,1,0)$, $(0,-1,0)$, $(-1,0,0)$, $(0,0,1)$, $(1,1,0)$, and $(0,-1,-1)$.

Example 3.9. The affine normal form for the polytope in Example 3.4 with modified $PM^{\max}$ is given by

$$
\mathrm{AffNF}(P) = \begin{pmatrix}
0 & 1 & 1 & 2 & 0 & 0 & 2 \\
0 & 0 & 2 & 2 & 0 & -1 & 3 \\
0 & 0 & 0 & 0 & 1 & 0 & -1
\end{pmatrix}.
$$

4. Exploiting the automorphism group of the pairing matrix

A crucial part of the normal form algorithm described in §3 is the ability to efficiently calculate the maximum vertex–facet pairing matrix $PM^{\max}$. One also needs to know a permutation σ such that $\sigma PM = PM^{\max}$, and to be able to calculate $\mathrm{Aut}(PM^{\max})$. These data can be constructed as $PM^{\max}$ is calculated – this is the approach taken by the PALP source code described in Appendix A – or recovered later. This section focuses on this second approach. A detailed algorithm is given in Appendix B.

Consider a case when PM is very symmetric, so that the order of $\mathrm{Aut}(PM)$ is large (for example, the vertex–facet pairing matrix for the n-dimensional polytope associated with projective space $\mathbb{P}^n$ has $|\mathrm{Aut}(PM)| = (n+1)!$). In such situations, the PALP algorithm is highly inefficient. Whilst computing $PM^{\max}$ the symmetries are not taken into account, so the algorithm needlessly explores equivalent permutations. Intuitively, one should be able to improve on the PALP algorithm by exploiting the automorphism group of PM.

Given an $n_r \times n_c$ matrix PM and a group of possible column permutations S (initially set to S_{n_c}), one can inductively convert this into $PM^{\max}$ as follows:

(1) If $n_r = 1$ then $PM^{\max} = \max\{\sigma PM \mid \sigma \in S\}$.

(2) If $|S| = 1$ then no permutations of the columns of PM are possible, and $PM^{\max}$ is given by sorting the rows of PM in decreasing order.

(3) Suppose now that $n_r > 1$ and $|S| > 1$.

(a) Let $R^{\max} := \max \{\sigma PM_i \mid \sigma \in S, 1 \leq i \leq n_r\}$ be the largest row in PM, up to the action of S.

(b) Set $S' := \{\sigma \in S \mid \sigma R^{\max} = R^{\max}\}$.

(c) For each row $1 \leq i \leq n_r$ such that there exists a permutation $\sigma \in S$ with $\sigma PM_i = R^{\max}$, consider the matrix $M_{(i)}$ obtained from σPM by deleting the i-th row. If $M_{(i)} \cong M_{(j)}$ for some $j < i$, then skip this case. Otherwise let $M_{(i)}^{\max}$ be the $(n_r - 1) \times n_c$ matrix obtained by inductively applying this process with $PM \leftarrow M_{(i)}$ and $S \leftarrow S'$.

(d) Set $M^{\max}$ to be the maximum of all such $M_{(i)}^{\max}$. Then

$$PM^{\max} = \begin{pmatrix} R^{\max} \\ \hline M^{\max} \end{pmatrix}.$$

4.1. **Test case: the database of smooth Fano polytopes.** The algorithm described in Appendix B, which we shall hereafter refer to as SYMM, was implemented by the authors and compared against the PALP algorithm. As Examples 4.1 and 4.2 illustrate, the difference in run-time between the two approaches can be considerable.

Example 4.1. Let P be the six-dimensional polytope[4] with 14 vertices

$$\pm (1,0,0,0,0,0), \pm(0,1,0,0,0,0), \pm(0,0,1,0,0,0), \pm(0,0,0,1,0,0),$$
$$\pm (0,0,0,0,1,0), \pm(0,0,0,0,0,1), \pm(1,1,1,1,1,1).$$

The automorphism group $\mathrm{Aut}(PM)$ is of order 10,080. On our test machine the PALP algorithm took 512.88 seconds, whereas the SYMM algorithm took only 5.83 seconds.

Example 4.2. Let P be the six-dimensional polytope[5] with 12 vertices

$$(1,0,0,0,0,0), (0,1,0,0,0,0), (0,0,1,0,0,0), (0,0,0,1,0,0), (0,0,0,0,1,0),$$
$$(0,0,0,0,0,1), (-1,-1,-1,1,1,1), (0,0,1,-1,0,0), (0,0,-1,0,0,0),$$
$$(0,1,1,-1,-1,-1), (0,-1,-1,0,0,0), (0,0,0,0,-1,-1).$$

The automorphism group $\mathrm{Aut}(PM)$ is of order 16; the PALP algorithm took 0.55 seconds whilst the SYMM algorithm took 4.30 seconds.

Table 2 contains timing data comparing the PALP algorithm with the SYMM algorithm. This data was collected by sampling polytopes from Øbro's classification of smooth Fano polytopes [Øbr07]. For each smooth polytope P selected, the calculation was performed for both P and P^*. In small dimensions the number of polytopes, and the time required for the computations, is small enough that the entire classification can be used. It is important to emphasise that the smooth Fano polytopes are atypical in that they can be expected to have a large number of symmetries, and so favour SYMM. Experimental evidence suggests that the ratio $r := |\mathrm{Aut}(PM)| / n_v$ is a good proxy for deciding between the two choices. When $r < 1$ the PALP algorithm often performs better, whereas larger values indicate SYMM should be used.

---

[4]Smooth Fano polytope number 1930 in the Graded Ring Database [BK].

[5]Smooth Fano polytope number 1854 in the Graded Ring Database [BK].

Dim.	#P	PALP		SYMM		BEST	
		Total	Average	Total	Average	Total	Average
4	248	6.28	0.03	4.48	0.02	3.41	0.01
5	1732	98.30	0.06	59.53	0.03	46.17	0.03
6	15244	6148.45	0.40	1510.32	0.10	1214.25	0.08
7	150892	152279.91	1.01	45230.55	0.30	34818.32	0.23
8	281629	611795.13	2.17	152902.73	0.54	111426.70	0.40

TABLE 2. Timing data, in seconds, for the PALP algorithm and for the SYMM algorithm. The best possible time if one could infallibly choose the faster of the two algorithms is recorded by BEST.

5. APPLICATIONS TO LAURENT POLYNOMIALS

Let $f \in \mathbb{C}[x_1^{\pm 1}, \ldots, x_n^{\pm 1}]$ be a Laurent polynomial in n variables, and let $P := \mathrm{Newt}(f)$ denote the Newton polytope of f. We require throughout that $\dim P = n$, i.e. that P is of maximum dimension in the ambient lattice. An element $B \in \mathrm{GL}_n(\mathbb{Z})$ corresponds to the invertible monomial transformation

$$(5.1) \qquad \begin{aligned} \varphi_B : (\mathbb{C}^*)^n &\to (\mathbb{C}^*)^n \\ x_j &\mapsto x_1^{B_{1j}} \cdots x_n^{B_{nj}}, \end{aligned}$$

and $g = \varphi_B^* f$ is also a Laurent polynomial. In particular, $\mathrm{Newt}(g) = P \cdot B$.

As when working with lattice polytopes, it can be advantageous to be able to present f in a normal form with respect to transformations of type (5.1).

Definition 5.1. Given two Laurent polynomials f and g such that $\mathrm{Newt}(f) = \mathrm{Newt}(g)$, we define an order $\preceq$ on f and g as follows. Let $v_1 < v_2 < \ldots < v_k$ be the lattice points in $\mathrm{Newt}(f)$, listed in lexicographic order. To each point v_i there exists a (possibly zero) coefficient c_i of x^{v_i} in f, and coefficient d_i in g. Define $\mathrm{coeffs}(f) := (c_1, c_2, \ldots, c_k)$. We write $f \preceq g$ if and only if $\mathrm{coeffs}(f) \leq \mathrm{coeffs}(g)$.

Remark 5.2. Any Laurent polynomial f determines a pair $(\mathrm{coeffs}(f), \mathrm{Newt}(f))$. Conversely, given any pair (c, P), where $c \in \mathbb{C}^k$ and $P \subset \Lambda_{\mathbb{Q}}$ is a maximum-dimensional lattice polytope such that $k = |P \cap \Lambda|$, we can associate a Laurent polynomial. If we insist that the c_i associated with the vertices $\mathcal{V}(P)$ are non-zero then we have a one-to-one correspondence.

Definition 5.3. Let f be a Laurent polynomial, and set $P := \mathrm{Newt}(f)$. Let $B \in \mathrm{GL}_n(\mathbb{Z})$ be such that $P \cdot B = \mathrm{NF}(P)$. The *normal form* for f is

$$\mathrm{NF}(f) := \min_{\preceq} \{\varphi_A \circ \varphi_B(f) \mid A \in \mathrm{Aut}(\mathrm{NF}(P))\}.$$

Example 5.4. Consider the Laurent polynomial

$$f = 2x^2 y + \frac{1}{x} + \frac{3}{xy}.$$

Then $\mathrm{NF}(P)$ has vertices $(1, 0)$, $(0, 1)$, and $(-1, -1)$, with corresponding transformation matrix

$$B = \begin{pmatrix} 0 & -1 \\ -1 & 1 \end{pmatrix} \in \mathrm{GL}_2(\mathbb{Z}).$$

Under this transformation,

$$\varphi_B^* f = 3x + y + \frac{2}{xy}$$

and $\mathrm{coeffs}(\varphi_B^* f) = (2, 0, 1, 3)$. The automorphism group $\mathrm{Aut}(\mathrm{NF}(P)) \cong S_3$ acts by permuting the non-zero elements in the coefficient vector, hence

$$\mathrm{NF}(f) = 3x + 2y + \frac{1}{xy}.$$

A naïve implementation of Laurent normal form faces a serious problem: listing the points in a polytope is computationally expensive, and will often be the slowest part of the algorithm by many orders of magnitude. With a little care this can be avoided. What is really needed in Definition 5.3 is not the entire coefficient vector, but the closure of the non-zero coefficients under the action of $\mathrm{Aut}(\mathrm{NF}(P))$. We illustrate this observation with an example.

Example 5.5. Consider the Laurent polynomial

$$f = x^{50}y^{50}z^{50} + x^{50}y^{30} + \frac{x^{30}z^{30}}{y^{40}} + \frac{x^{10}}{y^{40}z^{20}} + xyz + \frac{y^{40}z^{20}}{x^{10}} + \frac{y^{40}}{x^{30}z^{30}} + \frac{1}{x^{50}y^{30}} + \frac{1}{x^{50}y^{50}z^{50}}.$$

Set $P = \mathrm{Newt}(f)$. Notice that $|P \cap \Lambda| = 285241$; enumerating the points in P is clearly not the correct approach. The normal form $\mathrm{NF}(P)$ is given by change of basis

$$B = \begin{pmatrix} -3 & -4 & -6 \\ 5 & 7 & 10 \\ -12 & -16 & -23 \end{pmatrix} \in \mathrm{GL}_3(\mathbb{Z}),$$

with

$$g := \varphi_B^* f = x^{650}y^{880}z^{1270} + x^{500}y^{650}z^{950} + x^{10} + y^{10} + \frac{1}{y^{10}} +$$
$$\frac{1}{x^{10}} + \frac{1}{x^{10}y^{13}z^{19}} + \frac{1}{x^{500}y^{650}z^{950}} + \frac{1}{x^{650}y^{880}z^{1270}}.$$

The automorphism group $G := \mathrm{Aut}(\mathrm{NF}(P))$ is of order two, generated by the involution $u \mapsto -u$. We consider the closure of the nine lattice points corresponding to the exponents of g under the action of G. The only additional point is $(10, 13, 19)$. Thus we can express $\mathrm{coeffs}(g)$ with respect to these ten points:

$$\mathrm{coeffs}(g) = (1, 1, 1, 1, 1, 1, 1, 0, 1, 1).$$

The key observation is that the action of G on g will not introduce any additional points, hence the lexicographically smallest coefficient sequence with respect to these points will also be the smallest coefficient sequence with respect to all the points of $\mathrm{NF}(P)$. By applying the involution we obtain the smaller coefficient sequence $(1, 1, 0, 1, 1, 1, 1, 1, 1, 1)$, hence

$$\mathrm{NF}(f) = x^{650}y^{880}z^{1270} + x^{500}y^{650}z^{950} + x^{10}y^{13}z^{19} + x^{10} + y^{10} +$$
$$\frac{1}{y^{10}} + \frac{1}{x^{10}} + \frac{1}{x^{500}y^{650}z^{950}} + \frac{1}{x^{650}y^{880}z^{1270}}.$$

We conclude this section by remarking that the automorphism group $\mathrm{Aut}(f) \leq \mathrm{GL}_n(\mathbb{Z})$ of a Laurent polynomial f can easily be constructed from $\mathrm{Aut}(\mathrm{Newt}(f))$ by restricting to the subgroup that leaves $\mathrm{coeffs}(f)$ invariant.

Appendix A. The Kreuzer–Skarke algorithm

We describe in detail the algorithm used by Kreuzer and Skarke in Palp [KS04] to compute the normal form of a lattice polytope P of maximum dimension n. Any such polytope can be represented by a $n \times n_v$ matrix V whose columns correspond to the vertices of P. This matrix is unique up to permutation of columns and the action of $\mathrm{GL}_n(\mathbb{Z})$; i.e. one can change the order of the vertices and the underlying basis for the lattice to obtain a different matrix V'.

The Palp normal form is a unique representation of the polytope P such that if Q is any other maximum dimensional lattice polytope, then P and Q are isomorphic if and only if their normal forms are equal. For any matrix V with integer entries, and any $G \in \mathrm{GL}_n(\mathbb{Z})$, the Hermite normal form of $G \cdot V$ is uniquely defined. The question is how to define a canonical order for the vertices, since permuting the vertices will lead to a different Hermite normal form.

In what follows, the line numbers refer to the Palp source file `Polnf.c`[6]. We have chosen our notation to correspond as closely as possible to the source code. The algorithm will be described in eight stages:

A.1. The pairing matrix;
A.2. The maximal pairing matrix;
A.3. Constructing the first row;
A.4. Computing the restricted automorphism group, step I;
A.5. Constructing the k-th row;
A.6. Updating the set of permutations;
A.7. Computing the restricted automorphism group, step II;
A.8. Computing the normal form of the polytope.

A.1. **The pairing matrix.** We start by constructing the pairing matrix PM.

Line:	197 (`Init_rVM_VPM`)
Input:	A list of vertices and a list of equations for the supporting hyperplanes.
Output:	The pairing matrix PM.

Let $\{v_i\}_{i=1}^{n_v}$ be the vertices of P, in some order, and $\sum_{j=1}^{n} w_{ij}x_j + c_i = 0$, $i = 1, \ldots, n_f$, be the equations of the supporting hyperplanes of P. Here n_v is equal to the number of vertices of P, and n_f is equal to the number of facets of P. The w_i are the inward-pointing primitive facet normals, and the c_i are necessarily integers. The pairing matrix PM is the $n_f \times n_v$ matrix

$$PM_{ij} = \sum_{k=1}^{n} w_{ik}v_{jk} + c_i = \langle w_i, v_j \rangle + c_i$$

with integral coefficients.

The order of the columns of PM corresponds to an order of the vertices of P, and the order of the rows of PM corresponds to an order of the facets of P. Let $\rho = (r, c) \in S_{n_f} \times S_{n_v}$ act on PM via

$$(\rho PM)_{ij} = PM_{r(i)c(j)}.$$

---

[6]Palp 1.1, updated November 2, 2006. http://hep.itp.tuwien.ac.at/~kreuzer/CY/palp/palp-1.1.tar.gz

A.2. The maximal pairing matrix.

Let PM^{max} denote the maximal lexicographic matrix (when reading row by row) obtained from PM by reordering rows and columns, so that

$$PM^{\mathrm{max}} := \max \left\{ \rho PM \mid \rho \in S_{n_f} \times S_{n_v} \right\}.$$

It can happen that $\mathrm{Aut}(PM) \leq S_{n_f} \times S_{n_v}$ is non-trivial, say $|\mathrm{Aut}(PM)| = n_s$. Then we have n_s permutations $\{\rho_i\}_{i=1}^{n_s}$ such that $\rho_i PM = PM^{\mathrm{max}}$, and n_s corresponding orders for the vertices of the polytope. Our main task is to compute PM^{max} and $\{\rho_i\}_{i=1}^{n_s}$ from PM. This will be done by induction on the rows of PM^{max}.

A.3. Constructing the first row.

We begin by constructing the first row of PM^{max}.

Line:	348 (`Aux_vNF_Line`)
Input:	The paring matrix PM.
Output:	An array of permutations giving the first row of PM^{max}.

Set $n_s = 1$ and maximise the first row of PM, i.e. find a permutation $c_1 \in S_{n_v}$ such that $PM_{1c_1(i)} \leq PM_{1c_1(j)}$, $j \leq i$:

    $n_s \leftarrow 1$
    $(r_1, c_1) \leftarrow (1_{S_{n_f}}, 1_{S_{n_v}})$
    for $j = 1$ **to** n_v **do**
        $m \leftarrow \mathrm{IndexOfMax}\{PM_{1i} \mid i \geq j\}$
        if $m > 1$ **then**
            $c_1 \leftarrow c_1(j\,m + j - 1)$
        end if
    end for
    $b \leftarrow PM_1$

Suppose we have computed the first $k - 1$ lines, n_s of which could be chosen to be the first row of PM^{max} (i.e. up to reordering of the facets they are maximal among other lines and equal to the reference line, denoted b). Then we have integers $1 \leq k_i \leq k - 1 < n_f$ with corresponding permutations $\rho_i = (r_i, c_i) \in S_{n_f} \times S_{n_v}$, $i = 1, \dots, n_s$, and a reference line defined by $b := PM_{k_1}$ such that:

$$PM_{k_i c_i(j)} = b_{c_1(j)}, \qquad i = 1, \dots, n_s, j = 1, \dots, n_v.$$

Set $r_i = (1\,k_i)$ to be the permutation which moves the line in question to the first row of PM. Now we consider the k-th row of PM. Find the maximal element $\max_j\{PM_{kj}\}$, say PM_{km}, and let $c_{n_s+1} = (1\,m)$. We compare this against the reference line. If $PM_{k c_{n_s+1}(1)} < b_{c_1(1)}$ then continue with the next line (or stop if we are at the last line), otherwise continue constructing c_{n_s+1}. If $\max_{j>1}\{PM_{k c_{n_s+1}(j)}\} = PM_{k c_{n_s+1}(m)}$ then let $c_{n_s+1} \mapsto c_{n_s+1}(2\,m)$ and verify that $PM_{k c_{n_s+1}(2)} < b_{c_1(2)}$; if this inequality fails to hold then continue with the next element.

If the line k is not less than the reference line b then we set $r_{n_s+1} = (1\,k)$ and have two cases to consider:

  (1) If $PM_{k c_{n_s+1}(j)} = b_{c_1(j)}$, $j = 1, \dots, n_v$, then we have a new case of symmetry. We set $k_{n_s+1} := k$ and increment the number of symmetries n_s.

  (2) Otherwise we have found a (lexicographically) bigger row and so obtain a new reference line. We set $b := PM_k$, $k_1 := k$, and $\rho_1 := (r_{n_s+1}, c_{n_s+1})$, and reset the number of symmetries n_s.

for $k = 2$ **to** n_f **do**

$\quad (r_{n_s+1}, c_{n_s+1}) \leftarrow (1_{S_{n_f}}, 1_{S_{n_v}})$

$\quad m \leftarrow \text{IndexOfMax}\{PM_{kc_{n_s+1}(j)} \mid j \geq 1\}$

$\quad$ **if** $m > 1$ **then**

$\quad\quad c_{n_s+1} \leftarrow c_{n_s+1}(1\,m)$

$\quad$ **end if**

$\quad d \leftarrow PM_{kc_{n_s+1}(1)} - b_{c_1(1)}$

$\quad$ **if** $d < 0$ **then**

$\quad\quad$ **continue**

$\quad$ **end if**

$\quad$ **for** $i = 2$ **to** n_v **do**

$\quad\quad m \leftarrow \text{IndexOfMax}\{PM_{kc_{n_s+1}(j)} \mid j \geq i\}$

$\quad\quad$ **if** $m > 1$ **then**

$\quad\quad\quad c_{n_s+1} \leftarrow c_{n_s+1}(i\,m + i - 1)$

$\quad\quad$ **end if**

$\quad\quad$ **if** d=0 **then**

$\quad\quad\quad d \leftarrow PM_{kc_{n_s+1}(i)} - b_{c_1(i)}$

$\quad\quad\quad$ **if** $d < 0$ **then**

$\quad\quad\quad\quad$ **break**

$\quad\quad\quad$ **end if**

$\quad\quad$ **end if**

$\quad$ **end for**

$\quad$ **if** $d < 0$ **then**

$\quad\quad$ **continue**

$\quad$ **end if**

$\quad r_{n_s+1} \leftarrow r_{n_s+1}(1\,k)$

$\quad$ **if** $d = 0$ **then**

$\quad\quad n_s \leftarrow n_s + 1$

$\quad$ **else**

$\quad\quad (r_1, c_1) \leftarrow (r_{n_s+1}, c_{n_s+1})$

$\quad\quad n_s \leftarrow 1$

$\quad\quad b \leftarrow PM_k$

$\quad$ **end if**

end for

A.4. **Computing the restricted automorphism group, step I.** Once the first row of $PM^{\max}$ has been constructed, it imposes restrictions on any future column permutations: they must fix the first row.

Line:	376 (`Aux_vNF_Line`)
Input:	The first line of the maximal pairing matrix.
Output:	The array S capturing its automorphism group.

Suppose that the row is equal to blocks of a_i's, each of size n_i, $i = 1, \ldots, k$, where $\sum_{i=1}^{k} n_i = n_v$:

$$\left(\begin{array}{ccc|ccc|c|ccc} a_1 & \ldots & a_1 & a_2 & \ldots & a_2 & \ldots & a_k & \ldots & a_k \end{array} \right).$$

It is clear that if we had such a row, the only permutations of columns allowed in the construction of later rows will be those factoring through $S_{n_1} \times S_{n_2} \times \ldots \times S_{n_k}$. The symmetry of this row is encoded in an array S such that if $S(i) = j$ and $S(S(i)) = S(j) = h$ then the index i is in the block delimited by the indices j and h (depending on whichever is greater). We represent S as an array

$$\left(\begin{array}{cccc|c} n_1 & 1 & \ldots & 1 & n_1 + n_2 \quad n_1 + 1 \quad \ldots \quad n_1 + 1 \; \Big| \; \ldots \end{array} \right.$$
$$\left. \Big| \; n_v \quad 1 + \sum_{i=1}^{k-1} n_i \quad \ldots \quad 1 + \sum_{i=1}^{k-1} n_i \; \right)$$

Example A.1. The symmetries of the row $(5\ 5\ 5\ 5\ 4\ 3\ 3\ 2\ 2\ 2\ 1\ 0\ 0)$ are encoded by the array

$$S = \left(\begin{array}{cccc|c|cc|ccc|c|cc} 4 & 1 & 1 & 1 & 5 & 7 & 6 & 10 & 8 & 8 & 11 & 13 & 12 \end{array} \right).$$

When $S = (1\ 2\ \ldots\ n_v)$ the columns are fixed and we may only permute the rows. The computation of S is summarised in the following pseudo-code:

$S \leftarrow (1\ 2\ \ldots\ n_v)$
for $i = 2$ **to** n_v **do**
    if $PM_{r_1(1)c_1(i-1)} = PM_{r_1(1)c_1(i)}$ **then**
        $S(i) \leftarrow S(i-1)$
        $S^2(i) \leftarrow S(S(i)) + 1$
    else
        $S(i) \leftarrow i$
    end if
end for

A.5. **Constructing the k-th row.** Proceeding by induction on the rows, we construct the remaining rows of $PM^{\max}$.

Line:	289 (`Aux_vNF_Line`)
Input:	PM, the permutations $\{p_i\}_{i=1}^{n_s}$, and the array S.
Output:	The k-th line of the maximal pairing matrix.

Assume we have computed the first $l-1 < n_f - 1$ rows of $PM^{\max}$ and the associated symmetry array S (notice that the last row of $PM^{\max}$ need not be computed as it is completely determined), together with n_s distinct permutations $\rho_i = (r_i, c_i) \in S_{n_f} \times S_{n_v}$ such that

$$PM_{kj}^{\max} = PM_{r_i(k)c_i(j)} \qquad \text{for all } 1 \le j \le n_v, 1 \le k < l, 1 \le i \le n_s.$$

We have to consider each configuration given by the permutations $\{\rho_i\}_{i=1}^{n_s}$. For each configuration we generally obtain n_ρ ways to construct the line l, moreover some constructions might give a smaller line, hence n_s will have to be updated as we proceed. Let $\tilde{n}_s$ record the initial value of n_s.

First consider the case $k = \tilde{n}_s$. We will construct a candidate line for the l-th row of $PM^{\max}$; this will be our reference line against which the other cases will be compared. If a greater candidate is found, all the preceding computations will have to be deleted and redone with the new candidate. If a given case lead to a smaller line than the reference, it will have to be deleted.

Initially set the local number of symmetries, n_ρ, to zero and initialise the permutation $\tilde{\rho}_{n_\rho} = \rho_k$. We start with the line $\tilde{r}_{n_\rho}(l)$ by finding the maximal element of the first symmetry block. Suppose that

$$\max\left\{PM_{\tilde{r}_{n_\rho}(l)\tilde{c}_{n_\rho}(i)} \mid 1 \leq i \leq S(1)\right\} = PM_{\tilde{r}_{n_\rho}(l)\tilde{c}_{n_\rho}(m)}.$$

Then we update $\tilde{c}_{n_\rho}$ to $\tilde{c}_{n_\rho}(1\,m)$. This maximal value is saved in the reference line which we denote l_r (if it were already constructed, $k < \tilde{n}_s$, we move straight to the tests below). We increment n_ρ by one to reflect this new candidate, initialise $\tilde{\rho}_{n_\rho} = p_k$, and proceed to consider the maximal entries in the first symmetry block for other lines $r_k(s)$, $s = l+1, \ldots, n_f$.

Inductively, suppose we have considered $s-1$ lines where n_ρ of them have a maximal element in the first symmetry block equal to the one of the reference line $l_r(1)$, and the others have smaller values. We also have $\tilde{r}_{n_\rho} = r_k$ from the initialisation. Consider the line $\tilde{r}_{n_\rho}(s)$ and find its maximal element in $1, \ldots, S(1)$ as above, updating $\tilde{c}_{n_\rho}$. Now if $PM_{\tilde{r}_{n_\rho}(s)\tilde{c}_{n_\rho}(1)} < l_r(1)$ then proceed to the case $s+1$, if possible. Otherwise $\tilde{r}_{n_\rho} \mapsto \tilde{r}_{n_\rho}(l\,s)$ and there are two possibilities: if $PM_{\tilde{r}_{n_\rho}(s)\tilde{c}_{n_\rho}(1)} = l_r(1)$ then increase the number of symmetries $n_\rho \mapsto n_\rho + 1$ and move to $s+1$, after initialising the new permutation $\tilde{\rho}_{n_\rho} = \rho_k$; if $PM_{\tilde{r}_{n_\rho+1}(s)\tilde{c}_{n_\rho+1}(1)} > l_r(1)$ then redefine the first element of the reference line $l_r(1) := PM_{\tilde{r}_{n_\rho}(s)\tilde{c}_{n_\rho}(1)}$, update the first permutation $\tilde{\rho}_0 = \tilde{\rho}_{n_\rho}$, and reset $n_\rho = 1$ ready for the next permutation $\tilde{\rho}_{n_\rho} = \rho_k$.

$c \leftarrow 1$
$n_\rho \leftarrow 0$
$ccf \leftarrow cf$
$(\tilde{r}_{n_\rho}, \tilde{c}_{n_\rho}) \leftarrow (r_k, c_k)$
for $s = l$ **to** n_f **do**
    for $j = 2$ **to** $S(1)$ **do**
        if $PM_{\tilde{r}_{n_\rho}(s)\tilde{c}_{n_\rho}(c)} < PM_{\tilde{r}_{n_\rho}(s)\tilde{c}_{n_\rho}(j)}$ **then**
            $\tilde{c}_{n_\rho} \leftarrow \tilde{c}_{n_\rho}(c\,j)$
        end if
    end for
    if $ccf = 0$ **then**
        $l_r(1) \leftarrow PM_{\tilde{r}_{n_\rho}(s)\tilde{c}_{n_\rho}(1)}$
        $\tilde{r}_{n_\rho} \leftarrow \tilde{r}_{n_\rho}(l\,s)$
        $n_\rho \leftarrow n_\rho + 1$
        $ccf \leftarrow 1$
        $(\tilde{r}_{n_\rho}, \tilde{c}_{n_\rho}) \leftarrow (r_k, c_k)$
    else
        $d \leftarrow PM_{\tilde{r}_{n_\rho}(s)\tilde{c}_{n_\rho}(1)} - l_r(1)$
        if $d < 0$ **then**
            continue
        else if $d = 0$ **then**
            $\tilde{r}_{n_\rho} \leftarrow \tilde{r}_{n_\rho}(l\,s)$
            $n_\rho \leftarrow n_\rho + 1$
            $(\tilde{r}_{n_\rho}, \tilde{c}_{n_\rho}) \leftarrow (r_k, c_k)$
        else
            $l_r(1) \leftarrow PM_{\tilde{r}_{n_\rho}(s)\tilde{c}_{n_\rho}(1)}$
            $cf \leftarrow 0$
            $(\tilde{r}_1, \tilde{c}_1) \leftarrow (\tilde{r}_{n_\rho}, \tilde{c}_{n_\rho})$

$$n_\rho \leftarrow 1$$
$$(\tilde{r}_{n_\rho}, \tilde{c}_{n_\rho}) \leftarrow (r_k, c_k)$$
$$n_s \leftarrow k$$
$$\tilde{r}_{n_\rho} \leftarrow \tilde{r}_{n_\rho}(l\,s)$$
        end if
      end if
    end for

Note that the initial value of the *comparison flag cf* is 0. This indicates that the reference line has not been initialised; it is also reset to zero when a greater candidate is found. We will see later how cf is updated.

We need to construct other elements of l_r. Inductively, suppose we are constructing the entry i of l_r and we have n_ρ symmetries with corresponding permutations $\tilde{\rho}_j$, $j = 0, \ldots, n_\rho - 1$. If $n_\rho = 0$ we move to the next configuration $k - 1$ after having updated the symmetries accordingly, i.e. we do not save the current configuration. Otherwise, start with the last $j = n_\rho - 1$. Determine where the corresponding block of symmetry ends for i by looking at the maximum of $S(i)$ and $S^2(i)$, which we will call h. Then compute

$$\max\left\{ PM_{\tilde{r}_j(l)\tilde{c}_j(\lambda)} \mid i \leq \lambda \leq h \right\} = PM_{\tilde{r}_j(l)\tilde{c}_j(m)}$$

and update $\tilde{c}_j \mapsto \tilde{c}_j(i\,m)$. This value is saved in the reference line $l_r(i)$. Then we consider (inductively) any cases of symmetry with $j < n_\rho - 1$ and compute the i-th entry in the same manner as above: if $PM_{\tilde{r}_j(l)\tilde{c}_j(i)} = l_r(i)$ then continue with the next j; if $PM_{\tilde{r}_j(l)\tilde{c}_j(i)} < l_r(i)$ then the current case is removed and we update $n_\rho \mapsto n_\rho - 1$; finally if $PM_{\tilde{r}_j(l)\tilde{c}_j(i)} > l_r(i)$ then all cases previously considered are irrelevant, so we let $n_\rho = j + 1$ and the reference line is updated $l_r(i) = PM_{\tilde{r}_j(l)\tilde{c}_j(i)}$.

  for $c = 2$ **to** n_v **do**
    $h \leftarrow S(c)$
    $ccf \leftarrow cf$
    if $h < c$ **then**
      $h \leftarrow S(h)$
    end if
    $s \leftarrow n_\rho$
    while $s > 0$ **do**
      $s \leftarrow s - 1$
      for $j = c + 1$ **to** h **do**
        if $PM_{\tilde{r}_s(l)\tilde{c}_s(c)} < PM_{\tilde{r}_s(l)\tilde{c}_s(j)}$ **then**
          $\tilde{c}_s \leftarrow \tilde{c}_s(cj)$
        end if
      end for
      if $ccf = 0$ **then**
        $l_r(c) \leftarrow PM_{\tilde{r}_s(l)\tilde{c}_s(c)}$
        $ccf \leftarrow 1$
      else
        $d \leftarrow PM_{\tilde{r}_s(l)\tilde{c}_s(c)} - l_r(c)$
        if $d < 0$ **then**
          $n_\rho \leftarrow n_\rho - 1$

> **if** $n_\rho > s$ **then**
>> $(\tilde{r}_s, \tilde{c}_s) \leftarrow (\tilde{r}_{n_\rho}, \tilde{c}_{n_\rho})$
>
> **end if**
> **else if** $d > 0$ **then**
>> $l_r(c) \leftarrow PM_{\tilde{r}_s(l)\tilde{c}_s(c)}$
>> $cf \leftarrow 0$
>> $n_\rho \leftarrow s + 1$
>> $n_s \leftarrow k$
>
> **end if**
> **end if**
> **end while**
end for

A.6. Updating the set of permutations.

The last step in the construction of the line l is to organise the new symmetries for a given case k.

Line:	333 (`Aux_vNF_Line`)
Input:	The permutations $\{p_i\}_{i=1}^{n_s}$ and the newly computed $\{\tilde{\rho}_i\}_{i=0}^{n_\rho - 1}$.
Output:	The updated set $\{p_i\}_{i=1}^{n_s}$.

Recall that $\tilde{n}_s$ denotes the number of symmetries we had before performing the computations for the line l of $PM^{\max}$, and $n_s \leq \tilde{n}_s$ represents the updated number symmetries. Our current construction of the line l may well introduce new symmetries, so-called *local symmetries*, of which there are n_ρ. We can have $n_\rho = 0$, in which case all the configurations in the case k lead to a smaller candidate for l. When $n_\rho > 0$ the local symmetries are represented by the set $\{\tilde{\rho}_i\}_{i=0}^{n_\rho - 1}$ of new permutations.

We now update the array of all permutations. If $n_s > k$ we set $\rho_k = \rho_{n_s}$; we want the set of permutations $\{\rho_i\}_{i=1}^{n_s}$ to be updated so that the only cases which need to be considered are those with index $i < k$. Since we are appending n_ρ new permutations at end for the indices $i \geq n_s$, so $n_s \to n_s + n_\rho - 1$. If $n_\rho = 0$ then nothing is appended and n_s decreases by one as required. Finally, we update the comparison flag cf to reflect the current number of symmetries.

> $n_s \leftarrow n_s - 1$
> **if** $n_s > k - 1$ **then**
>> $(r_k, c_k) \leftarrow (r_{n_s+1}, c_{n_s+1})$
>
> **end if**
> $cf \leftarrow n_s + n_\rho$
> **for** $s = 0$ **to** $n_\rho - 1$ **do**
>> $(r_{n_s+1}, c_{n_s+1}) \leftarrow (\tilde{r}_s, \tilde{c}_s)$
>> $n_s \leftarrow n_s + 1$
>
> **end for**

A.7. Computing the restricted automorphism group, step II.

Once a new row of $PM^{\max}$ has been compute we need to update S to reflect the symmetries of this row. This is done by restricting the blocks previously delimited by S to reflect any additional constraints imposed by the row.

Example A.2. Continuing Example A.1, suppose that the second row of the candidate $PM^{\max}$ has been computed, and that the two rows are given by

$$\begin{pmatrix} 5 & 5 & 5 & 5 & 4 & 3 & 3 & 2 & 2 & 2 & 1 & 0 & 0 \\ 4 & 3 & 3 & 3 & 3 & 2 & 2 & 2 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The corresponding array S is

$$\begin{pmatrix} 1 & \big| & 4 & 2 & 2 & \big| & 5 & \big| & 7 & 6 & \big| & 8 & \big| & 9 & \big| & 10 & \big| & 11 & \big| & 13 & 12 \end{pmatrix}.$$

Line:	376 (`Aux_vNF_Line`)
Input:	The newly computed upper block of the maximal pairing matrix.
Output:	The updated array S capturing the automorphism group of the matrix.

$c \leftarrow 1$
while $c < n_v + 1$ **do**
    $s \leftarrow S(c) + 1$
    $S(c) \leftarrow c$
    $c \leftarrow c + 1$
    while $c < s$ **do**
        if $PM_{r_1(l)c_1(c)} = PM_{r_1(l)c_1(c-1)}$ **then**
            $S(c) \leftarrow S(c-1)$
            $S^2(c) \leftarrow S^2(c) + 1$
        else
            $S(c) \leftarrow c$
        end if
        $c \leftarrow c + 1$
    end while
end while

A.8. **Computing the normal form of the polytope.** Inductively, we have obtained n_s permutations $\{\rho_i = (r_i, c_i)\}_{i=1}^{n_s}$ such that $\rho_i PM = PM^{\max}$. We are really only interested in the permutations of the columns, since they correspond to permutations of the vertices of P. The PALP algorithm computes a new order for the columns of $PM^{\max}$ based on the following: the maximum coefficient in the column; the sum of the coefficients in the column; and the relative position of the column in $PM^{\max}$. Let $\rho_c \in S_{n_v}$ denote this column permutation.

Line:	216 (`New_pNF_Order`)
Input:	The maximal pairing matrix $PM^{\max}$.
Output:	The column permutation $\rho_c \in S_{n_v}$.

$PM^{\max} \leftarrow p_1 PM$
$p_c \leftarrow 1_{S_{n_v}}$
$M^{\max} \leftarrow \left\{ \max_{1 \le i \le n_f} \left\{ PM_{ij}^{\max} \right\} \mid 1 \le j \le n_v \right\}$
$S^{\max} \leftarrow \left\{ \sum_{1 \le i \le n_f} PM_{ij}^{\max} \mid 1 \le j \le n_v \right\}$
for $i = 1$ **to** n_v **do**

$k \leftarrow i$
for $j = i + 1$ **to** n_v **do**
    if $(M_j^{\max} < M_k^{\max}) \vee ((M_j^{\max} = M_k^{\max}) \wedge (S_j^{\max} < S_k^{\max}))$ **then**
        $k \leftarrow j$
    end if
end for
if $k \neq i$ **then**
    $M^{\max} \leftarrow \mathrm{SwapRow}(M^{\max}, i, k)$
    $S^{\max} \leftarrow \mathrm{SwapRow}(S^{\max}, i, k)$
    $p_c \leftarrow p_c(i\,k)$
end if
end for

Given the column permutations ρ_c and c_i, $i = 1 \ldots, n_s$, we obtain a permutation of the vertices of P, and hence of the columns of the vertex matrix V. We let V_i denote this reordered vertex matrix. The remaining freedom – the action of $\mathrm{GL}_n(\mathbb{Z})$ corresponding to the choice of lattice basis – is removed by computing the Hermite normal form $H(V_i)$.

Line:	134 (`GLZ_Make_Trian_NF`)
Input:	A matrix with integer coefficents.
Output:	The Hermite normal form of the matrix.

The PALP normal form is simply the minimum amongst the $H(V_i)$.

Line:	399 (`Aux_Make_Triang`)
Input:	The column permutations ρ_c and $\{c_i\}_{i=1}^{n_s}$, and the vertex matrix V.
Output:	The normal form.

Appendix B. Calculating the maximum pairing matrix

Let M be an $n_r \times n_c$ matrix. Recall that we define an action of $\sigma = (\sigma_r, \sigma_c) \in S_{n_r} \times S_{n_c}$ on the rows and columns of M via $(\sigma M)_{ij} := M_{\sigma_r(i), \sigma_c(j)}$, and that we call two matrices M and M' isomorphic if there exists some permutation $\sigma \in S_{n_r} \times S_{n_c}$ such that $\sigma(M) = M'$. We begin by briefly describing one approach to determining when two matrices are isomorphic.

Given a matrix M, we associate a bipartite graph $G(M)$ with $n_r + n_c$ vertices, where the vertices v_i, v_{n_r+j} are connected by an edge E_{ij} for all $1 \leq i \leq n_r$, $1 \leq j \leq n_c$. Each edge E_{ij} is labelled with the corresponding value M_{ij}. The vertices v_i, $1 \leq i \leq n_r$, are labelled with one colour, whilst the vertices v_{n_r+j}, $1 \leq j \leq n_c$, are labelled with a second colour. This distinguishes between vertices representing rows of M and vertices representing columns of M. Clearly two matrices M and M' are isomorphic if and only if the graphs $G(M)$ and $G(M')$ are isomorphic. We note also that the automorphism group $\mathrm{Aut}(M) \leq S_{n_r} \times S_{n_c}$ is given by the automorphism group of $G(M)$.

We now describe a recursive algorithm to compute $PM^{\max}$ from PM. For readability, we shall split this algorithm into three parts, with a brief discussion preceding each part.

Input:	A matrix PM.
Output:	The maximal matrix $PM^{\max}$.

Throughout we set n_r and n_c equal to, respectively, the number of rows and the number of columns of the input matrix PM. A vector s of length n_c is used to represent the permitted permutations of the columns of PM. Initially s is defined as

$$s = (a, \ldots, a) \in \mathbb{Z}^{n_c},$$

where $a := 1 + \max PM_{ij}$ is larger than any entry of the matrix PM. At each step of the recursion, the value of n_c remains unchanged, but the value of n_r will decrease by one as a row of PM is removed from consideration. The vector s will be modified to reflect the symmetries of the previously steps; two coefficients s_j and s_k are equal if and only if the columns j and k can be exchanged without affecting the computations so far. By construction s will always satisfy:

(1) either $s_j = s_{j+1}$ or $s_j = s_{j+1} + 1$, for each $1 \le j < n_c$;
(2) $s_{n_c} = a$.

The first stage is to calculate the maximum possible row $R^{\max}$ of PM, where each row is sorted in decreasing order. Once done, we update the vector s to reflect the possible column permutations that will leave $R^{\max}$ unchanged.

$\tilde{R}_i \leftarrow \text{Sort}_{\ge}\{(s_j, PM_{ij}) \mid 1 \le j \le n_c\}$
$R_i \leftarrow (\tilde{R}_{ij2} \mid 1 \le j \le n_c)$
$R^{\max} \leftarrow \max\{R_i \mid 1 \le i \le n_r\}$
$s' \leftarrow s$
for $j = n_c - 1$ **to** 1 **by** -1 **do**
    if $s_j = s_{j+1} \wedge R_j^{\max} \ne R_{j+1}^{\max}$ **then**
        for $k = 1$ **to** j **do**
            $s'_k \leftarrow s'_k + 1$
        end for
    end if
end for

Next we collect together all non-isomorphic ways of writing PM with $R^{\max}$ as the first row. These possibilities are recorded in the set $\mathcal{M}$.

$\mathcal{M} \leftarrow \{\}$
for $i = 1$ **to** n_r such that $R_i = R^{\max}$ **do**
    $M \leftarrow \text{SwapRow}(PM, 1, i)$
    $T \leftarrow \text{Sort}_{\ge}\{(s_j, M_{1j}, j) \mid 1 \le j \le n_c\}$
    $\tau \leftarrow$ permutation in S_{n_c} sending j to T_{j3}
    $M \leftarrow \tau(M)$
    $\tilde{M} \leftarrow M$
    $\tilde{M}_1 \leftarrow s'$
    if $\bigwedge_{M' \in \mathcal{M}} \left(\tilde{M}' \not\cong \tilde{M}\right)$ **then**
        $\mathcal{M} \leftarrow \mathcal{M} \cup \{M\}$
    end if
end for

When all possible symmetries of the columns have been exhausted, the vector s' will be equal to the sequence

$$(a + n_c - 1, a + n_c - 2, \ldots, a).$$

If this is the case, then $PM^{\max}$ is the maximum matrix in $\mathcal{M}$, once the rows have been placed in decreasing order. If there remain symmetries to explore, then we recurse on each of the matrices in $\mathcal{M}$ using the new permutation vector s'; $PM^{\max}$ is given by the largest resulting matrix.

> **if** $n_r = 1$ **then**
> $PM^{\max} \leftarrow R^{\max}$
> **else if** $s'_1 = s'_{n_c} + n_c - 1$ **then**
> $PM^{\max} \leftarrow \max \{ \text{SortRows}_{\geq}(M) \mid M \in \mathcal{M} \}$
> **else**
> $\mathcal{M}' \leftarrow \{\}$
>   **for** $M \in \mathcal{M}$ **do**
> $M' \leftarrow \text{RemoveRow}(M, 1)$
> $M' \leftarrow (\text{recurse with } PM \leftarrow M' \text{ and } s \leftarrow s')$
> $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{M'\}$
>   **end for**
> $PM^{\max} \leftarrow \text{VerticalJoin}(R^{\max}, \max \mathcal{M}')$
> **end if**

References

[BBK09]   Gavin Brown, Jarosław Buczyński, and Alexander M. Kasprzyk, *Convex polytopes and polyhedra*, Handbook of Magma Functions, Edition 2.16, November 2009, available online at http://magma.maths.usyd.edu.au/.

[BCP97]   Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265, Computational algebra and number theory (London, 1993).

[BDSS09]   David Bremner, Mathieu Dutour Sikirić, and Achill Schürmann, *Polyhedral representation conversion up to symmetries*, Polyhedral computation, CRM Proc. Lecture Notes, vol. 48, Amer. Math. Soc., Providence, RI, 2009, pp. 45–71.

[BEK84]   Jürgen Bokowski, Günter Ewald, and Peter Kleinschmidt, *On combinatorial and affine automorphisms of polytopes*, Israel J. Math. **47** (1984), no. 2-3, 123–130.

[BK]   Gavin Brown and Alexander M Kasprzyk, *The Graded Ring Database*, online, access via http://www.grdb.co.uk/.

[BSP+12]   David Bremner, Mathieu Dutour Sikirić, Dmitrii V. Pasechnik, Thomas Rehn, and Achill Schürmann, *Computing symmetry groups of polyhedra*, arXiv:1210.0206 [math.CO].

[Kre10]   Maximilian Kreuzer, *PALP++ project proposal*, unfinished draft, available online at http://hep.itp.tuwien.ac.at/∼www/palp++.pdf, September 2010.

[KS98]   Maximilian Kreuzer and Harald Skarke, *Classification of reflexive polyhedra in three dimensions*, Adv. Theor. Math. Phys. **2** (1998), no. 4, 853–871.

[KS00]   _____, *Complete classification of reflexive polyhedra in four dimensions*, Adv. Theor. Math. Phys. **4** (2000), no. 6, 1209–1230.

[KS03]   Volker Kaibel and Alexander Schwartz, *On the complexity of polytope isomorphism problems*, Graphs Combin. **19** (2003), no. 2, 215–230.

[KS04]   Maximilian Kreuzer and Harald Skarke, *PALP, a package for analyzing lattice polytopes with applications to toric geometry*, Computer Phys. Comm. **157** (2004), 87–106.

[McK]   Brendan D. McKay, *The Nauty graph automorphism software*, online, access via http://cs.anu.edu.au/∼bdm/nauty/.

[McK81]   _____ , *Practical graph isomorphism*, Proceedings of the Tenth Manitoba Conference on Numerical Mathematics and Computing, Vol. I (Winnipeg, Man., 1980), vol. 30, 1981, pp. 45–87.

[MdlBW09] C. Mears, M. Garcia de la Banda, and M. Wallace, *On implementing symmetry detection*, Constraints **14** (2009), no. 4, 443–477.

[Øbr07]   Mikkel Øbro, *An algorithm for the classification of smooth Fano polytopes*, `arXiv:0704.0049v1 [math.CO]`, classifications available from `http://www.grdb.co.uk/`.

[Pug05]   Jean-Francois Puget, *Automatic detection of variable and value symmetries*, Principles and Practice of Constraint Programming (Peter van Beek, ed.), vol. 3709, Springer, 2005, pp. 475–489.

[S⁺]   W. A. Stein et al., *Sage Mathematics Software*, available online at `http://www.sagemath.org/`.

Trinity College, University of Cambridge, Cambridge, CB2 1TQ, UK
E-mail address: `roland.grinis09@imperial.ac.uk`

Department of Mathematics, Imperial College London, London, SW7 2AZ, UK
E-mail address: `a.m.kasprzyk@imperial.ac.uk`