Imperial College London
Department of Mathematics

Computation of Hilbert Bases of cones with Python

Final year project

Tomasz Nguyen
CID: 00644875

Supervisor: Dr Alexander Kasprzyk

This is my own work except where otherwise stated

June 11, 2015

Contents

Abstract

Finite generating sets appear naturally in many branches of mathematical pursuits. They are very useful tools for classifying objects, the most natural example being bases of vector spaces.

Rational polyhedral cones most prominently appear in linear optimisation and linear programming, but their usefulness extends to simplicial algebra and algebraic geometry and beyond. They allow to classify the feasibility regions for linear systems of inequalities and endow them with an algebraic structure. The goal of this project is to develop the necessary theory of cones to explain and build a Python implementation of Raymond Hemmecke's algorithm for finding the Hilbert bases and extremal rays of polyhedral cones [Hem02].

1 Background

Hemmecke's algorithm is the state of the art procedure for finding integral generating sets of polyhedral cones. Before diving into the details, however, we shall cover some theory of cones and set the stage for later discussion.

1.1 Lattices

Before we introduce lattices, we state a few facts about rational matrices:

Definition 1.1 (Hermite Normal Form)**.** A rational full rank matrix A is said to be in Hermite normal form if:

1. A is a non-negative non-singular lower triangular matrix
2. all entries on the main diagonal are positive
3. all maximal elements in each row are on the main diagonal

Theorem 1.2 (Hermite Normal Form Theorem)**.** *Any rational matrix can be brought to Hermite normal form by a series of* unimodular *column operations:*

- *exchanging two columns*
- *adding or subtracting a column*

Proof. See Schrijver [Sch86] □

Corollary 1.3. *A unimodular matrix has a unimodular inverse.*

Corollary 1.4. *Unimodular matrices form a subgroup of* $GL_n(\mathbb{Z})$

Definition 1.5. A lattice is a subgroup of $\mathbb{Q}^n$ that is finitely generated by $g_1, \ldots, g_m \in \mathbb{Q}^n$ as a $\mathbb{Z}$-module.

The simplest natural examples are lattices in $\mathbb{Q}^2$ spanned by integer combination of arbitrary linearly independent basis $\{b_1, b_2\}$.

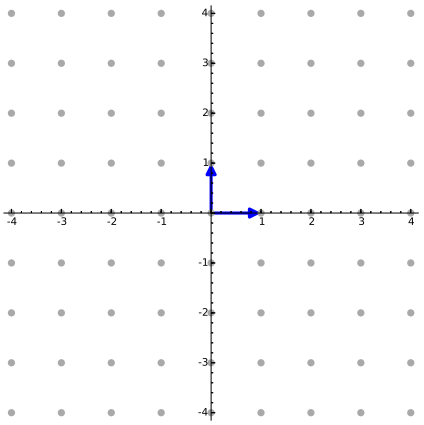

Figure 1: $\mathbb{Z}^2$ with generators $(1,0),(0,1)$

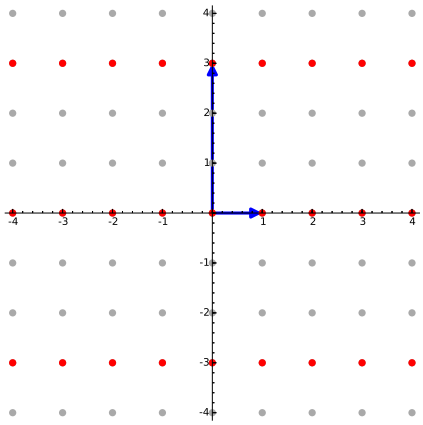

Figure 2: Sublattice of $\mathbb{Z}^2$ with generators $(1,0),(0,3)$

For our purposes we will concern ourselves only with sublattices (finitely generated $\mathbb{Z}$-submodules) of $\mathbb{Z}^n$.

It is easy to determine whether two integral bases span the same lattice over $\mathbb{Z}$:

Theorem 1.6. *Let B and C be two different bases of $\mathbb{Q}^n$. Let $\mathcal{L}(B), \mathcal{L}(C)$ be the lattices spanned by integer linear combinations of those bases. $\mathcal{L}(B) = \mathcal{L}(C)$ if and only if there exists a unimodular matrix U s.t. $B = CU$*

Proof.

$\Leftarrow$ Assume $B = CU$. Then if U is unimodular, so is its inverse U^{-1} and $B = CU, \quad C = BU^{-1}$ Since both U and U^{-1} are integer matrices, they produce integer combinations of the basis points, so: $\mathcal{L}(B) \subseteq \mathcal{L}(C)$ and $\mathcal{L}(C) \subseteq \mathcal{L}(B)$. Hence $\mathcal{L}(B) = \mathcal{L}(C)$

$\Rightarrow$ Assume that $\mathcal{L}(B) = \mathcal{L}(C)$. Then the lattice $\mathcal{L}(B)$ is a integer combination of points spanned by the basis C, so there exists a integer square

matrix V s.t. $B = VC$. Similarly there exists an integer square matrix W s.t. $C = WB$. Combining these two we get $B = VWB$. Since vectors in B are linearly independent, we must have $VW = I_n$. In particular $det(V)det(W) = det(I_n) = 1$ so $det(V) = \pm 1$ and $det(W) = \pm 1$. Since V, W have integer entries and determinants ± 1, they are unimodular.

□

Corollary 1.7. *Two integer matrices span the same lattice if and only if their Hermite normal forms are identical.*

Example 1. Looking at Figure 3 and Figure 4 we see that two different bases in $\mathbb{Q}^2$ span the same subgroup of $\mathbb{Z}^2$

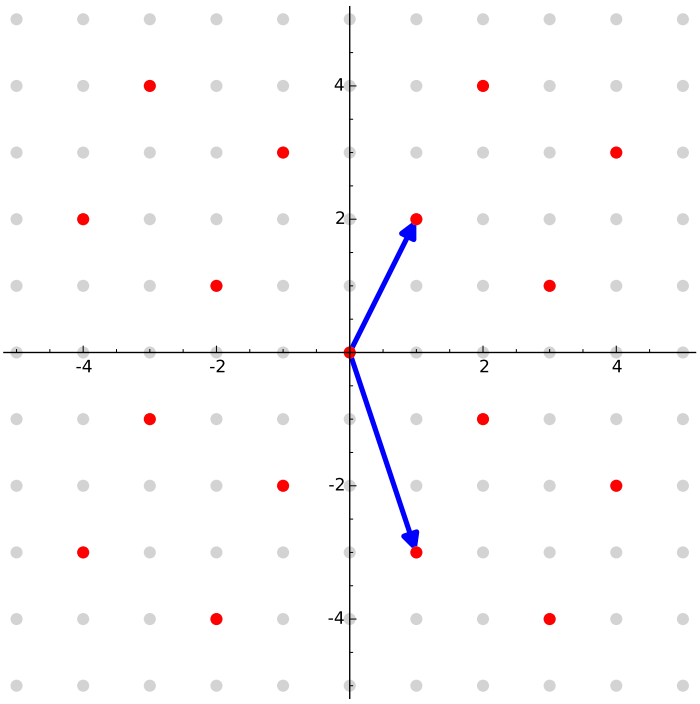

Figure 3: Integer lattice spanned by vectors $(1, 2), (1, -3)$ over $\mathbb{Z}$

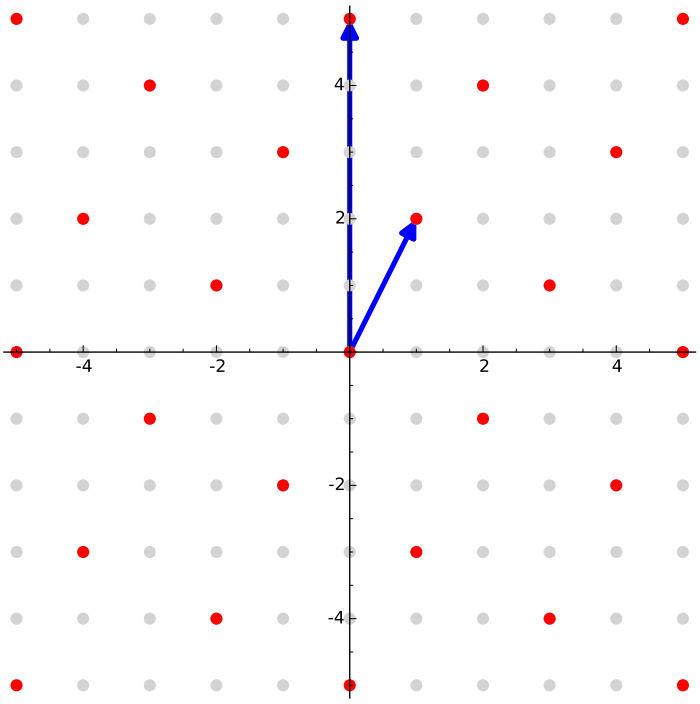

Figure 4: Integer lattice spanned by vectors $(1, 2), (0, 5)$ over $\mathbb{Z}$

Example 2. If the two matrices have different Hermite normal forms they span different lattices.

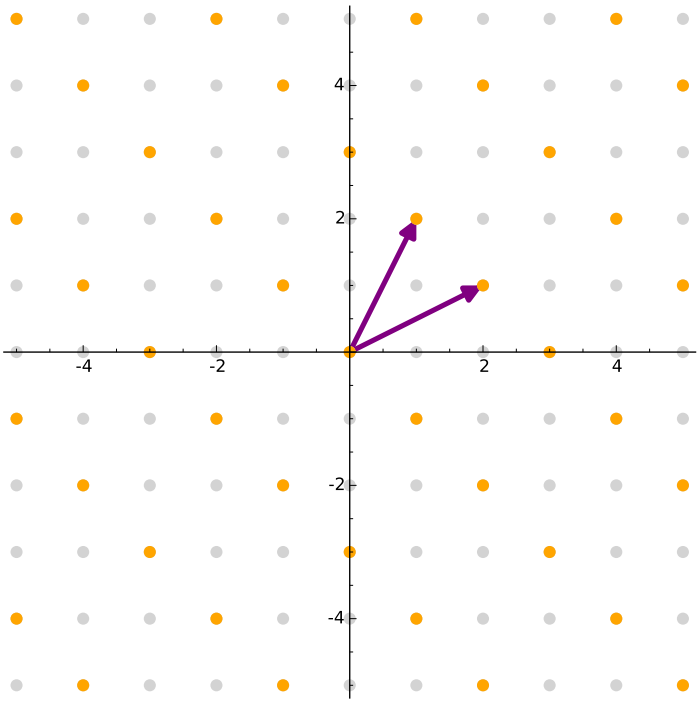

Figure 5: Integer lattice spanned by vectors $(1,2),(2,1)$ over $\mathbb{Z}$ is not contained in the lattice in Figure 3

Proposition 1.8. *The automorphism group of a lattice $\mathcal{L}$ is a subgroup of $GL_n(\mathbb{Z})$*

Proof. Automorphisms of $\mathcal{L}$ are isomorphisms of bases so they are closed under multiplication, have inverses and identity element $I_n \in GL_n(\mathbb{Z})$. Clearly these are a subgroup of $GL_n(\mathbb{Z})$ □

1.2 Polyhedral Cones

Definition 1.9. A convex cone is a set $\mathcal{C}$ such that $\forall a, b \geq 0 \quad a\mathcal{C} + b\mathcal{C} = \mathcal{C}$

Definition 1.10. A cone $\mathcal{C}$ is said to be *finitely generated* by the vectors $g_1, \ldots, g_m$ if:

$$\mathcal{C} = \{a_1 g_1 + \ldots + a_m g_m | a_1, \ldots, a_m \geq 0\}$$

Equivalently, a cone generated by $g_1, \ldots, g_m$ is the inclusion minimal cone containing $g_1, \ldots, g_m$

We will refer to convex cones as *cones*

Definition 1.11. A cone $\mathcal{C}$ is *pointed* if $\mathcal{C} \cap -\mathcal{C} = \{0\}$

Polyhedral cones are particularly interesting because they correspond to matrix inequalities:

Definition 1.12. A cone $\mathcal{C}$ is *polyhedral* if $\mathcal{C} = \{x : Ax \leq 0\}$ for some $A \in GL_n(\mathbb{R})$

Definition 1.13. A cone is $\mathcal{C}$ is rational polyhedral if it is polyhedral and A has rational coefficients.

Definition 1.14. A closed *halfspace* in $\mathbb{R}^n$ is a subset of a vector space defined by its inward pointing normal vector n:

$$half(\mathbf{n}) = \{x \in \mathbb{R}^n : \langle n, x \rangle \geq 0\}$$

This gives an equivalent definition of a polyhedral cone:

Definition 1.15. A cone is said to be *polyhedral* if it is an intersection of finitely many halfspaces

Theorem 1.16 (Farkas-Minkowski-Weyl Theorem)**.** *A cone is polyhedral if and only if it is finitely generated. [Sch86][pp. 87]*

Example 3. Consider the following set of curves in $\mathbb{R}^2$ and the area and integral points between them.

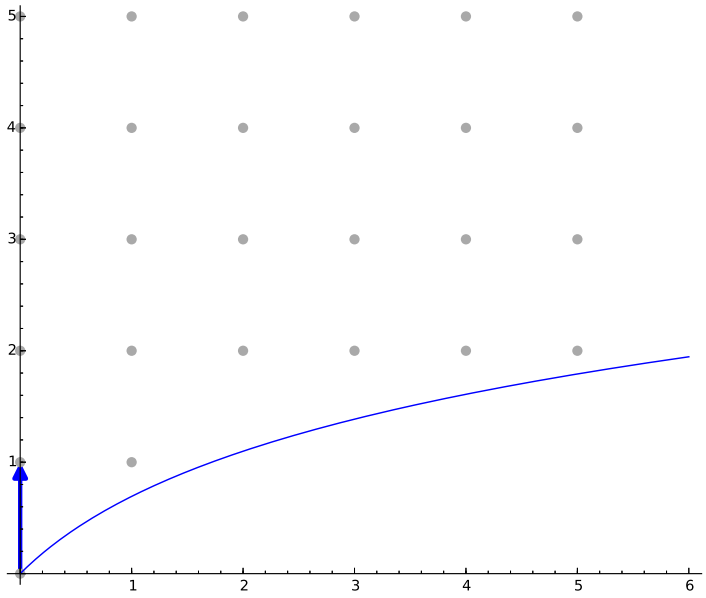

Figure 6: Lattice point set in $\mathbb{Z}^2$ contained by the cone $y \geq log(x+1)$

Clearly this set is closed under addition and contains $\{0\}$ as its only subspace but it is not finitely generated by vectors in $\mathbb{R}^2$

We can also see cones as kernels of linear transformations:

Lemma 1.17. *The solutions to* $A\mathbf{x} = 0$, $A \in \mathbb{R}^{m\times n}$ *are polyhedral cones. Moreover if* A *has rational entries, they are rational cones.*

Proof. Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n_{\geq 0}$ be solutions of $A\mathbf{x} = 0$. Then $A(\mathbf{x}_1 + \mathbf{x}_2) = A(\mathbf{x}_1) + A(\mathbf{x}_2) = 0$. Therefore $\mathbf{x}_1 + \mathbf{x}_2 \in \ker(A)$ Also trivially $0 \in \ker(A)$ □

The definitions above give rise to two ways of describing the same finitely generated cone:

1.2.1 Facet representation of a cone

We know that each halfspace is uniquely determined by its normal vectors. We also know that each polyhedral, or equivalently, finitely generated cone can be described as an intersection of finitely many halfspaces.

Definition 1.18. *Facets* of a cone $\mathcal{C}$ are given by intersections of $\mathcal{C}$ with the boundaries of closed halfspaces generating $\mathcal{C}$.

Remark. It follows immediately that halfspace normal vectors are normal to the facets of $\mathcal{C}$

What follows is that a cone can described in terms of its facet normals:

Definition 1.19. A cone $\mathcal{C}$ is given by:

$$\mathcal{C} := \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{n}_i \rangle \geq 0 \quad \forall \mathbf{n}_i\}$$

Where $\mathbf{n}_i$ are the facet normal vectors of $\mathcal{C}$

Example 4. The facets of the cone below are given by the two halflines from the origin. The normal vectors generate the cone as intersection of the halfspaces containing them.

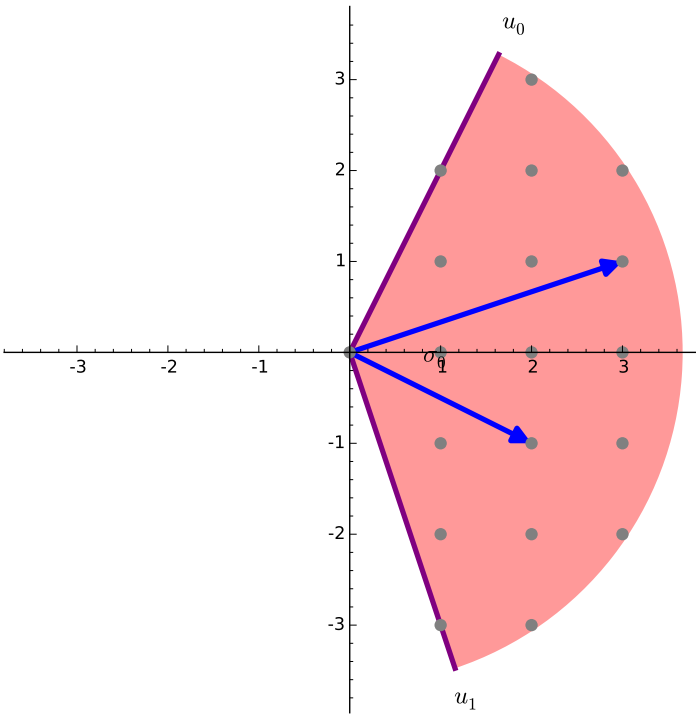

Figure 7: Facet normals of the cone generated by rational vectors $(1, 2), (1, -3)$

1.2.2 Ray representation of a cone

Definition 1.20. An *edge* (or *extremal ray*) of a rational cone $\mathcal{C}$ is a vector $r \in \mathcal{C}$ that cannot be represented as a positive rational linear combination of other vectors in $\mathcal{C}$.

Remark. Finding extremal rays of a cone is easy if we know its barycentric subdivision - we just need to find the intersections of facets with the hyperplane described by the normal vector along the barycentre.

What follows is a definition of a cone in terms of its extremal rays:

Definition 1.21. Given a set of extremal rays of a rational cone $\mathcal{C} := \{r_1, \ldots, r_s\} \subset \mathbb{Q}^n$, the cone is defined as the set of positive linear com-

binations of its edges:

$$\mathcal{C} = \left\{ x : x = \sum_{i=1}^{s} a_i r_i,\ a_i \geq 0 \right\}$$

1.2.3 Dual cone

Looking at the above we can intuit that the cone generated by the face normal vectors of $\mathcal{C}$ is also a cone:

Definition 1.22. Given a cone $\mathcal{C}$, the *dual cone* $\mathcal{C}^D$ of $\mathcal{C}$ is given by:

$$\mathcal{C}^D = \{y \in \mathbb{R}^n : \langle y, x \rangle \geq 0 \quad \forall x \in \mathcal{C}\}$$

Proposition 1.23. *The extremal rays of a dual cone $\mathcal{C}^D$ are exactly the facet normals of $\mathcal{C}$*

Proof. Observe that for facet normal vectors n_i of $\mathcal{C}$:

$$\mathcal{C} = \{x \in \mathbb{R}^n : \langle x, n_i \rangle \geq 0 \quad \forall n_i\}$$

Since dot product is bilinear, any positive linear combination of the face normals $y = a_1 n_1 + \ldots + a_m n_m, \quad a_i \geq 0$ will satisfy:

$$\langle y, x \rangle \geq 0 \qquad \forall x \in \mathcal{C}$$

So any vector in $\mathcal{C}^D$ can be written as a positive linear combination of n_i and the set $R = \{n_1, \ldots, n_k\}$ consisting of all of the distinct face normals of $\mathcal{C}$ contains the rays of $\mathcal{C}^D$

Suppose there is a n_i that is not a ray of $\mathcal{C}^D$. Then it can be written as a positive linear combination of rays of $\mathcal{C}^D$ and therefore is coplanar with some vectors $n_j, n_k,\ j, k \neq i$. So the halfspace described by n_i is wholly contained in the intersection of halfspaces described by n_i and n_k and therefore it cannot describe a facet of $\mathcal{C}$. So the set R is contained in the set of rays of $\mathcal{C}^D$ □

Corollary 1.24. *Each cone is uniquely determined by its dual*

Proof. This follows from the injectivity and idempotence of the dual operator. □

2 Some applications

Finding the generating sets of cones can help us understand the algebraic structure of classes of infinite sets. A good example comes from commutative algebra:

2.1 Cones of exponents

There is a natural similarity between pointed rational cones cones and polynomial rings. In fact, one could set up a bijective correspondence between cones in integer points in rational cones and ideals in polynomial rings.

Definition 2.1. A monomial is a polynomial of the form:

$$x_1^{\alpha_1} \dots x_n^{\alpha_n} \in k[x_1, \dots, x_n]$$

We will write $x^\alpha = x_1^{\alpha_1} \dots x_n^{\alpha_n}$ and $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{Z}^n_{\geq 0}$. The *total degree* of x^α is $|\alpha|_1 = \sum_{i=1}^n \alpha_i$ (the l_1 norm of α)

Definition 2.2. Let $I \lhd k[x_1, \dots, x_n]$. Let $A \subset \mathbb{Z}^n_{\geq 0}$ be such that:

$$I := \left\{ \sum_{\alpha \in A} h_\alpha x^\alpha \right\}$$

with finitely many $h_\alpha \in k[x_1, \dots, x_n]$ non zero. Then I is called a *monomial ideal* and we write $I = (x^\alpha | \alpha \in A)$

Remark. Note that an ideal is a monomial ideal iff each of its elements is generated by finitely many monomials with exponents in A

Lemma 2.3. *Let* $x^\alpha, x^\beta \in I \lhd k[x_1, \dots, x_n]$ *Then:* $x^\alpha = x^\beta \iff \alpha = \beta$

Proof. Since x_i are different indeterminates they need to agree in powers ($k[x_i]$ are simple submodules of $k[x_1, \ldots, x_n]$) we have:

$$\begin{aligned} x^\alpha = x^\beta &\iff x_1^{\alpha_1} \ldots x_n^{\alpha_n} = x_1^{\beta_1} \ldots x_n^{\beta_n} \\ &\iff \alpha_i = \beta_i \quad \forall i \in \{1, \ldots, n\} \\ &\iff \alpha = \beta \end{aligned}$$

□

Example 5. What do polynomials look like inside of cones? Each monomial is represented by a point and since addition of polynomials over a commutative ring is commutative, we can see polynomials as collections of points:

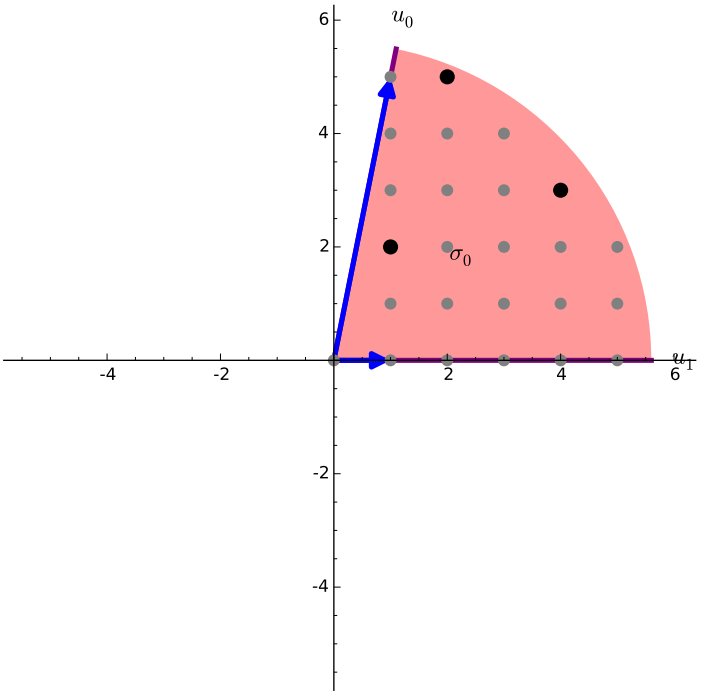

Figure 8: Polynomial $f = x^4y^2 + x^2y^6 + xy^2$ as lattice point set inside $\mathbb{Z}^2$

We see that if we consider the multiples of those monomials we see that the ideal generated by f in $k[x, y]$ is a rational, non pointed cone.

Moreover, if we notice that the minimal polynomial in our monomial set is xy^2, we can move it to the origin and see that the monomial ideal generated by those is the integral points inside the pointed rational cone spanned by the vectors $(4,1),(1,4)$

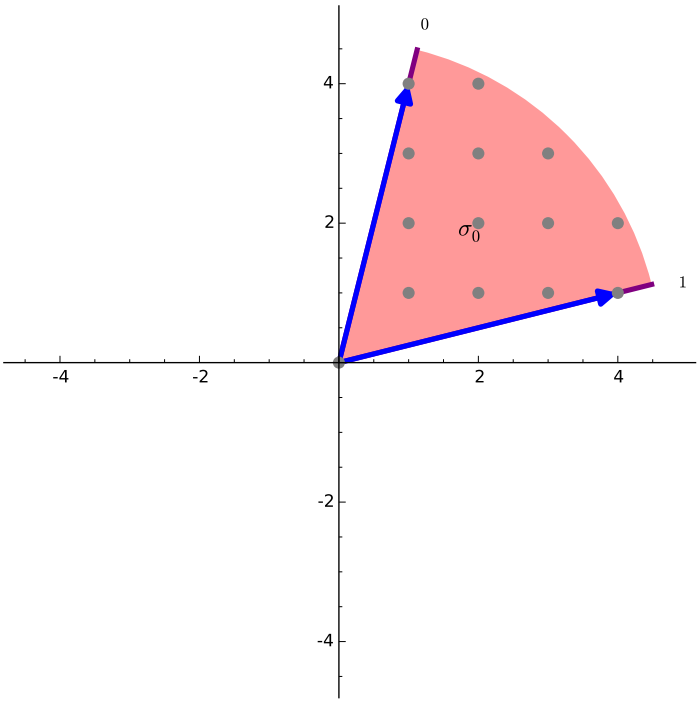

Figure 9: Monomial ideal corresponding to the monomial set $\{x^4y^2,\ x^2y^6,\ xy^2\}$

2.2 Hilbert basis of a cone

So far we have seen a way to generate a cone $\mathcal{C}$ as non-negative linear combinations of integer vectors with scalar multiplication over a field. Suppose we are interested only in the integral interior points - $\mathcal{C} \cap \mathbb{Z}^n$ generated as positive linear combinations of some integral generating set $G \subset \mathcal{C}$. Similarly to polynomial ring theory, there multiple possible choices for G, but there is a unique inclusion minimal one.

Definition 2.4. Let $\mathcal{C}$ be a cone. An *integral generating set* of $\mathcal{C}$ is a set $G \subset \mathcal{C}$ of vectors such that every integral vector $x \in \mathcal{C} \cap \mathbb{Z}^n$ can be written as a non-negative, integral linear combination of vectors in H. In other words it satisfies:

$$x \in \mathcal{C} \iff (\forall g \in G \; \exists a_g \in \mathbb{Z}_{\geq 0} \; : \; x = \sum_{g \in G} a_g g)$$

Definition 2.5. An inclusion minimal *integral generating set* $H \subset \mathcal{C}$ satisfies:

$$H = \{h \in \mathcal{C} \; : \; h \text{ is not a linear combination of vectors in } \mathcal{C} \text{ over } \mathbb{Z}_{\geq 0}\}$$

Such an integral generating set is called the *Hilbert basis* of $\mathcal{C}$

Example 6. If we remove any of blue vectors in Figure 10 there no longer is an admissible path (generated over $\mathbb{Z}_{\geq 0}$) from the origin to the point at the head of the vector. E.g. if we remove the generator $(1, 1)$ there no longer is a path from the origin to $(1, 1)$ spanned by other basis vectors, hence the basis is inclusion minimal - we cannot remove generators any further.

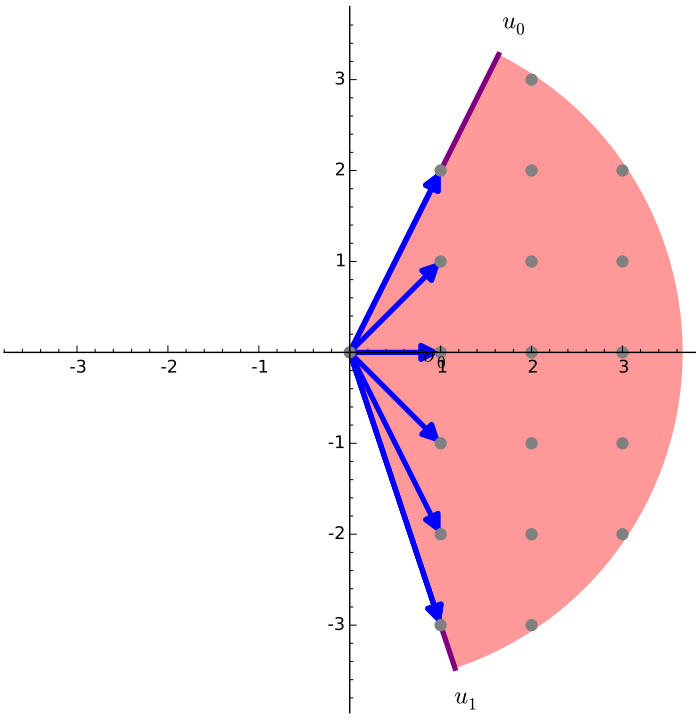

Figure 10: Hilbert Basis of the cone generated by rational vectors $(1,2),(1,-3)$

$(2,1)$ is not in H since it can be written as $(1,1)+(1,0)=(2,1)$ and $(1,2),(1,0)$ are members of the *Hilbert basis*

Every finitely generated cone admits a unique *Hilbert basis*:

Theorem 2.6. *[Sch86] Each rational polyhedral cone $\mathcal{C}$ is generated by a finite integral generating set. If $\mathcal{C}$ is pointed there is a unique Hilbert basis.*

Proof. By definition, a rational polyhedral cone can be expressed as linear combinations of finite number of rational vectors $b_1,\ldots,b_n$. Without loss of generality assume that these vectors are integral. Consider the *paralellotope* consisting of vectors:

$$\mathcal{P} := \{\lambda_1 b_1 + \ldots + \lambda_n b_n \ | 0 \leq \lambda_i \leq 1 \quad (i \ = \ 1 \ldots n)\}$$

Then all integral vectors $a_i, \ldots, a_m \in \mathcal{P}$ form an integral generating set for $\mathcal{C}$. To see that pick an arbitrary integral vector $b \in \mathcal{C}$:

$$b = \mu_1 b_1 + \ldots + \mu_n b_n$$

Then we can express b as:

$$b = \lfloor \mu_1 \rfloor b_1 + \ldots + \lfloor \mu_n \rfloor b_n + ((\mu_1 - \lfloor \mu_1 \rfloor)b_1 + \ldots + (\mu_n - \lfloor \mu_n \rfloor)b_n)$$
$$c_a = b - \lfloor \mu_1 \rfloor b_1 + \ldots + \lfloor \mu_n \rfloor b_n = ((\mu_1 - \lfloor \mu_1 \rfloor)b_1 + \ldots + (\mu_n - \lfloor \mu_n \rfloor)b_n)$$

Since $0 \leq (\mu_i - \lfloor \mu_i \rfloor) \leq 1$ for all i we have that $c_a \in \mathcal{P}$. Moreover, since $b - \lfloor \mu_1 \rfloor b_1 + \ldots + \lfloor \mu_n \rfloor b_n$ is integral, $c_a = a_i$ for some i. Also, each $b_k \in \mathcal{P}$ we can write any b as a non-negative integral combination of $a_1, \ldots, a_m$, therefore $a_1, \ldots, a_m$ form an *integral generating set*.

Now suppose C is pointed (it contains the zero vector). Write k for the dimension of the ambient space and define:

$$H := \{a \in \mathcal{C} \cap \mathbb{Z}^k | a \neq 0, \forall x, y \in \mathcal{C} \cap \mathbb{Z}^k \ a \neq x + y\}$$

Clearly any *integral generating set* must contain H as a subset. To see that H is itself a *integral generating set*, let b be a vector such that $bx > 0 \ \forall x \in \mathcal{C} \setminus \{0\}$. This induces a partial ordering on $\mathcal{C}$. Suppose there exists $c \in \mathcal{C}$ such that c cannot be written as a non-negative integral combination of vectors in H. By Zorn's lemma we can choose such a c such that bc is minimal. By definition of a pointed rational cone $c = c_1 + c_2$ for some $c_1, c_2 \in \mathcal{C} \setminus \{0\}$. Since $bc_1 < bc$ and $bc_2 < bc$ we see that c_1, c_2 are non-negative integral combinations of vectors in H and by definition of dot product, so is c. Hence H is a *Hilbert basis* that is contained in every other *integral generating set* and therefore it is inclusion minimal. □

2.3 Gordan-Dickson Lemma

We define a partial order on $\mathbb{Z}^n$. For $x, y \in \mathbb{Z}^n_{\geq 0}$ we have:

$$x \leq y \iff \forall i = 1, \ldots, n \quad x^{(i)} \leq y^{(i)}$$

Lemma 2.7 (Gordan-Dickson Lemma, sequence version)**.** *[Hem06] Let* $S = \{p_1, p_2, \ldots\}$ *be a sequence of points in* $\mathbb{Z}^n_{\geq 0}$ *such that*

$$p_i \not\leq p_j \iff i < j$$

. Then the sequence $\{p_1, p_2, \ldots\}$ *is finite.*

Remark. What this theorem is saying is that every *decreasing* sequence of vectors in $\mathbb{Z}^n_{\geq 0}$ has to terminate. This gives a halting condition for algorithms generating decreasing sequences with respect to a partial order compatible with the Gordan-Dickson Lemma.

Proof. We will proceed by induction on n the number of variables. Clearly if $n < 1$ this is equivalent to the existence of total well orders in $\mathbb{Z}$. Assume that the lemma is true for $n \leq k$. Suppose there is an infinite decreasing sequence in $\mathbb{Z}^{k+1}_{\geq 0}$ such that $p_i \not\leq p_j$ whenever $i < j$.

Consider the last elements in vectors S. Construct a new sequence sequence C according to the rule:

1. Find the first index j such that the last element of p_j is minimal among all p_i
2. Add p_j to C
3. Repeat the procedure on the remaining elements of the sequence $\{p_1, p_2, \ldots\}$ with p_j omitted.

The result is a subsequence of S with the last components in increasing order. Clearly $p_{j_i} \not\leq p_{j_2}$ whenever $i < j$. Since the sequence S is not finite,

we can repeat the procedure infinitely many times and C is also an infinite subsequence of S.

By projecting the elements of C onto first k coordinates we construct an infinite decreasing sequence $\{r_1, r_2, \ldots\}$s.t. $r_i \not\leq r_j$ whenever $i < j$, a contradiction. □

Lemma 2.8 (Gordan-Dickson Lemma, set version)**.** *[Hem06] Every infinite set $S \subseteq \mathbb{Z}^n_{\geq 0}$ contains only finitely many $\leq$-minimal points.*

Proof. Assume there are infinitely many $\leq$-minimal points in S. By ordering them according to the rule as in Lemma 2.7, we obtain a sequence contradicting Lemma 2.7. □

Lemma 2.7 implies Lemma 2.8. The converse is also true:

Lemma 2.9. *[Hem06] Every $\leq$ decreasing sequence of points in $\mathbb{Z}^n_{\geq 0}$ is finite iff every infinite set $S \subseteq \mathbb{Z}^n_{\geq 0}$ has finitely many $\leq$-minimal points*

Proof. We saw $\implies$ in the proof of Lemma 2.8. To see $\impliedby$ let $C = \{p_1, p_2, \ldots\}$ be an infinite sequence of points in $\mathbb{Z}^n_{\geq 0}$ with $p_i \not\leq p_j$ whenever $i < j$. Then as a set C contains finitely many $\leq$-minimal points. Let k denote the maximal index among those, then for $j > k$ there exists at least one $\leq$-minimal point in C with index $i \leq k < j$ with $p_i \leq j$. But in C $p_i \not\leq p_j$ whenever $i < j$ by construction, a contradiction. Hence C must be finite. □

2.4 The final piece

The last element we need to construct algorithms for finding bases is a structure theorem for rational polyhedral cones. We need to recognise some sort of group-like structure to be able to change the generators of a cone (analogously to change of bases of vector spaces).

Lemma 2.10 (Gordan's Lemma)**.** *Let $\mathcal{C} \subset \mathbb{Q}^n$ be the cone generated by $\{g_1, \ldots, g_n\} \subset \mathbb{Z}^n$. Then $\mathcal{C} \cap \mathbb{Z}^n$ is an affine monoid*

Proof. Let $x, y \in \mathcal{C} \cap \mathbb{Z}^n$. Since:

$$x = \sum_{i=1}^{n} \alpha_i g_i, \quad y = \sum_{i=1}^{n} \gamma_i g_i$$

Let $g_i^{(j)}$ be the j'th component of the vector g_i. Since we can divide each vector g_i by the gcd of all of its components and still get a generator of a cone $\mathcal{C}$ in $\mathbb{Q}^n$, WLOG we can assume that $gcd(g_i^{(1)}, \ldots, g_i^{(n)}) = 1$. Now we see that $x, y \in \mathcal{C} \cap \mathbb{Z}^n$ if and only if $\alpha_i, \gamma_i \in \mathbb{Z}$. So the addition of vectors inside $\mathcal{C} \cap \mathbb{Z}^n$ is well defined, as well as multiplication by the integer scalars.

The additive identity 0 corresponds to all $\alpha_i, \gamma_i = 0$. From the associativity, distributivity and commutativity of multiplication and addition in $\mathbb{Q}^n$ we get the commutative monoid structure. □

Corollary 2.11. *A pointed rational polyhedral cone $\mathcal{C}$ with a Hilbert Basis $H \subset \mathcal{C} \cap \mathbb{Z}^n$ is a free monoid with the alphabet consisting of the generators in H.*

3 Why calculating integral generating sets is hard

Theorem 2.6 shows the existence of generating sets, but it is not constructive.

A *parallelotope* generated by integral vectors $B = b_1, \ldots, b_n$ is given by:

$$\mathcal{P} := \{\lambda_1 b_1 + \ldots + \lambda_n b_n \mid 0 \leq \lambda_i \leq 1\}$$

In the proof of the Theorem 2.6 we have shown that the integral vectors $a_1, \ldots, a_m \in \mathcal{P}$ form the integral generating set for cone(B).

Example 7. Looking at Figure 11 we can easily see that the interior of the cone in question can be tiled into repeating parallelotopes as in Figure 12

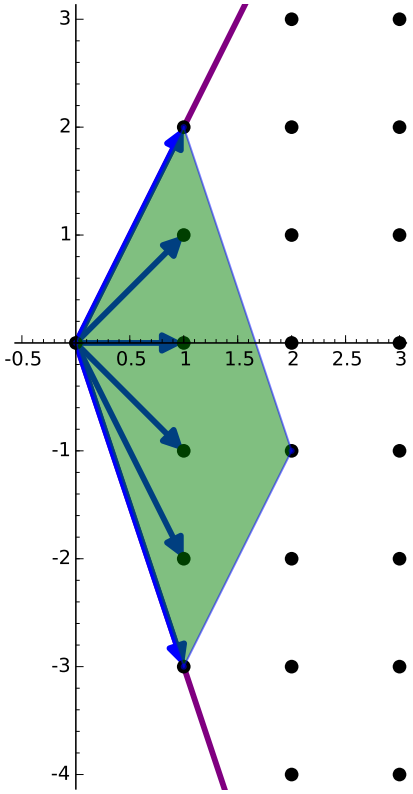

Figure 11: A parallelotope spanned by the generators of a cone $\text{cone}(\{(1,2),(1,-3)\})$ with its Hilbert basis

Note that the inclusion minimal Hilbert basis is contained entirely inside of the parallelotope as shown in the proof of Theorem 2.6

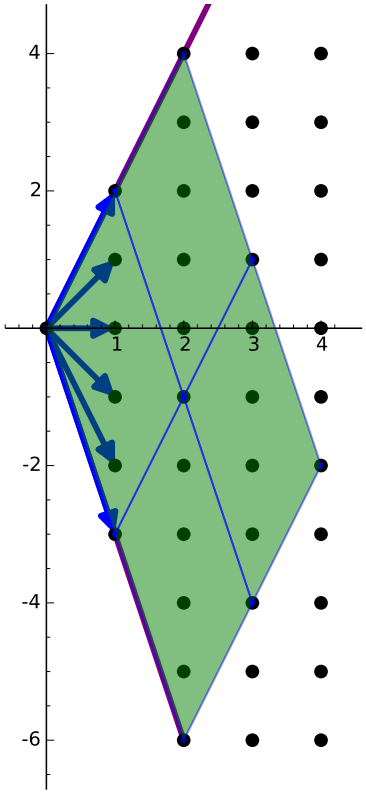

Figure 12: Tesselation of the cone into similar parallelotopes

The proof of Theorem 2.6 tells us a very important fact about Hilbert bases of cones - we only need to consider the parallelotope spanned by the generating set we wish to transform into a Hilbert basis - the fundamental domain of a problem. This however, does not make the computation easy. For Hilbert bases we get the following result:

Theorem 3.1 (Sebö [Seb90]). *Let $\mathcal{C}$ be a pointed cone and $H \subseteq \mathbb{Z}^n$ be its Hilbert basis. If $w \in cone(H) \cap \mathbb{Z}^n$ (i.e. w is an integral point in the cone generated by H), then w is the positive integer linear combination of at most $2n-1$ elements onf H.*

Remark. This theorem gives us a complexity bound on verifying whether a generating set is a Hilbert basis. Given a generating set H, we need to check

if an arbitrary vector in cone(H) is an admissible positive linear combination of members of H

Proof. Let $H = \{a_1, \ldots, a_k\}$ and express $w = \sum_{i=1}^{k} \lambda_i a_i$ in a way such that $\sum_{i=1}^{k} \lambda_i$ is maximal (choose the longest path from the origin to w in terms of the members of H). Since the cones are commutative monoids, we can minimise the number of the generators in the expression for each path, by reordering terms and expressing them in terms of one another, giving us the representation of w in terms of linearly independent vectors. Since the cone is contained in $\mathbb{Z}^n$ by rank-nullity, there are at most n of those, so WLOG we can assume that $\lambda_i = 0$ if $i > n$ and write $w = \sum_{i=1}^{n} \lambda_i a_i$ Let

$$w_0 = w - \sum_{i=1}^{n} \lfloor \lambda_i \rfloor a_i = \sum_{i=1}^{n} (\lambda_i - \lfloor \lambda_i \rfloor) a_i \in \mathcal{P}$$

Where $\mathcal{P}$ is the parallelotope generated by H as in the proof of Theorem 2.6.

$w_0 \in \mathcal{C} \cap \mathbb{Z}^n$ only if there exist $\alpha_i \in \mathbb{Z}_{\geq 0}$ such that $w_0 = \sum_{i=1}^{k} \alpha_i a_i$. Now

$$w = \sum_{i=1}^{k} \alpha_i a_i + \sum_{i=1}^{n} \lfloor \lambda_i \rfloor a_i$$

$\sum_{i=1}^{k} \alpha_i \leq n - 1$, otherwise we have:

$$\sum_{i=1}^{k} \alpha_i \geq n > \sum_{i=1}^{k} (\lambda_i - \lfloor \lambda_i \rfloor)$$

Which in turn implies:

$$\sum_{i=1}^{k} \alpha_i + \sum_{i=1}^{k} (\lambda_i - \lfloor \lambda_i \rfloor) > \sum_{i=1}^{k} \lambda_i$$

Since $\sum_{i=1}^{k} \lambda_i$ was chosen to be maximal, we know that

$$\sum_{i=1}^{k} \alpha_i + \sum_{i=1}^{k} (\lambda_i - \lfloor \lambda_i \rfloor) \leq 2n - 1$$

□

This shows that a "naive" algorithm, taking a generating set H and trying to rewrite it in terms of the vectors contained in the paralellotope spanned by H would have to perform $\binom{|H|}{2n-1}$ checks for each vector in H, giving a runtime increasing exponentially in the dimensionality of the ambient space.

Example 8. Going back to Example 7. We see that the number of points in the parallelotope is determined by the spread between the generators. Consider the parallelotope spanned by $(1, 2)$ and $(1, -3)$:

$$\mathcal{P} := \{\lambda_1 \times (1, 2) + \lambda_2 \times (1, -3) | 0 \leq \lambda_i \leq 1\}$$

We see that the Hilbert basis of the cone spanned by $(1, 2), (1, -3)$ are all of the points in $\mathcal{P} \cap \mathbb{Z}^2$ apart from the origin and $(1, 2) + (1, -3) = (2, -1)$. It is easy to see that any cone with rays $(1, y_1), (1, y_2),\ y_1 > y_2,\ y_1, y_2 \in \mathbb{Z}$ will have a Hilbert basis consisting of vectors:

$$H := \{(1, y) \mid y_2 \leq y \leq y_1,\ y \in \mathbb{Z}\}$$

We can construct a cone with an arbitrarily large Hilbert basis even in low-dimensional spaces. While enumerating the integral points in the parallelotope $\mathcal{P}$ is guaranteed to give us the superset of the Hilbert basis, extracting the minimal alphabet requires is a problem that not only grows exponentially in the dimensionality of the cone, we are able to construct an arbitrarily difficult problem in any dimension - Hilbert bases are in no way bound by the properties of the ambient space.

Luckily, we have not yet used all that we know about lattice point sets. Thanks to Gordan-Dickson Lemma (Lemma 2.9), given a partial ordering of lattice points we can produce a finitely terminating algorithm that will reduce any generating set to an inclusion minimal one. An example of such an algorithm is Buchberger's Gröbner basis algorithm Let k be a field and let $I \triangleleft k[x_1, \ldots, x_n]$ be an ideal. A finite subset $\{g_1, \ldots, g_m\} \subset I$ is a *Gröbner*

basis for I if it satisfies:

$$(LT(g_1), \ldots, LT(g_m)) = (LT(I))$$

where $(LT(I))$ is the monomial ideal generated by the leading terms of polynomials in I

Algorithm 3.2. Let $G = \{g_1, \ldots, g_m\}$ be a generating set for an ideal $I \triangleleft k[x_1, \ldots, x_n]$ where k is a field.

Fix a term ordering $\leq$. We define the *S-polynomial* as:

$$S(f,g) = \frac{LCM(LM(f), LM(g))}{LT(f)} f - \frac{LCM(LM(f), LM(g))}{LT(g)} g$$

Where $LM(f)$ is the *leading monomial* (the largest monomial with respect to $\leq$) and LT is the leading term - $\alpha * LM(f)$ for some $\alpha \in k$.

We define $\overline{f}^G$ to be the remainder from performing the division algorithm on f by the members of G.

Data: G generating set for I

Result: B a basis for I

while $\exists f, g \in G \ : \ \overline{S(f,g)}^G \neq 0$ **do**

  $G \leftarrow G \cup \{\overline{S(f,g)}^G\}$ `// (completion step)`

end

$B \leftarrow G$

return B

Lemma 3.3. *Algorithm 3.2 terminates and produces a Gröbner basis for* I

Proof. Clearly at each stage of the algorithm, the set G is a generating set for I. If the algorithm terminates, the returned set B satisfies:

$$\forall f, g \in B \quad (\overline{S(f,g)}^B = 0)$$

Hence by Buchberger's Criterion [CLO07 p: pp. 85-87] it is a Gröbner basis.

To see that the algorithm terminates assume for contradiction that the algorithm does not terminate. Let $G_i = \{g | g \in G \text{at } i\text{'th iteration}\}$. Let $f, g \in G_i$. Then $\overline{S(f,g)}^{G}_{i+1} = 0$ by construction. Let r_i be the remainder added to G_i. By the division algorithm we have $LT(r_i) \nleq LT(r_j) \iff i < j$ Since the algorithm does not terminate we can create an infinite descending sequence of $\{LT(r_1), LT(r_2), \ldots\}$ by choosing just one remainder from the residues added at each iteration. So if the algorithm does not terminate we can construct a descending sequence contradicting Gordan-Dickson Lemma. Thus the algorithm must terminate. □

Proof of termination of the Algorithm 3.2 shows the usefulness of the Gordan-Dickson Lemma. It is also important to note the key features of the Buchberger's algorithm. At each stage we produce a *S-polynomial* f that lies in I (by definition of the *S-polynomial*) but cannot be generated by G. Then we reduce f with respect to G and add the residue so that the new generating set generates f. This pattern is known as *Critical Pair/Completion*[Buc85] and is the foundation of Hemmecke's algorithm.

4 Disassembling Hemmecke's algorithm

Rather than dive in head-first, it is more informative to reconstruct the algorithm step-by-step:

4.1 Positive sum property

We begin by introducing a partial order $\sqsubseteq$ on $\mathbb{R}^n$:

Definition 4.1. Let $u, v \in \mathbb{Z}^n_{\geq 0}$. $u \sqsubseteq_{j+1} v$ iff:

1. $u^{(i)} \leq v^{(i)}, \quad i = 1, \ldots, j$
2. $|u^{(j+1)}| \leq |v^{(j+1)}|$
3. $u^{(j+1)} v^{(j+1)} \geq 0$

Lemma 4.2. *For $v \in \mathbb{Z}^n$ define $v^+ := max(0, v)$ and $v^- := min(0, -v)$. Then*

$$u \sqsubseteq_n v \iff (u^+, u^-) \leq (v^+, v^-)$$

Proof. This follows immediately from the definitions of $\leq$ and $\sqsubseteq$ □

With this in hand, we see that we can generalise the Gordan-Dickson lemma to $\sqsubseteq$ on $\mathbb{Z}^n$.

Lemma 4.3 (Gordan-Dickson Lemma, $\sqsubseteq$ version)**.**

1. *If $P := \{p_1, p_2, \ldots, \} \sqsubseteq \mathbb{Z}^N$ is a sequence of points, such that $i < j \implies p_i \not\sqsubseteq p_j$ then P is finite*

2. *If $S \sqsubseteq \mathbb{Z}^n$ is an infinite set, then it has finitely many $\sqsubseteq$ minimal points*

Lemma 4.4. *[Hem06] Let, $u, v, w \in \mathbb{Z}^n$ with $u \sqsubseteq v - w$. Then for each component of $v - u$ and $w + u$ the following holds:*

If $(v - w)^{(i)} \geq 0$ *Then* $w^{(i)} \leq (v - u)^{(i)} \leq v^{(i)}$ *and* $w^{(i)} \leq (v + u)^{(i)} \leq v^{(i)}$

If $(v - w)^{(i)} \leq 0$ *Then* $w^{(i)} \geq (v - u)^{(i)} \geq v^{(i)}$ *and* $w^{(i)} \geq (v + u)^{(i)} \geq v^{(i)}$

Definition 4.5. $v \in \mathbb{Z}^n$ is $\sqsubseteq$*-representable* with respect to $G \sqsubseteq \mathbb{Z}^n$ if v can be written as a positive linear combination $v = \sum_{i \in I} \alpha_i g_i$ with $g_i \sqsubseteq v$, and $g_i \in G \; \forall i \in I$.

A set G has the *positive sum property* with respect to S if $G \subset S$ and every $v \in S$ is $\sqsubseteq$-representable with respect to G

4.2 Connecting cones and lattices

Without loss of generality we can assume that the rays of a cone are linearly independent. Otherwise we proceed with analogous discussion in lower number of dimensions.

Since each cone is uniquely determined by its generators (by definition of being finitely generated), so writing these as columns of a matrix gives us a unique lattice in $\mathbb{Z}^n$ up to change of basis. Moreover, if we bring the generators into Hermite normal form, we get a subgroup of $\mathbb{Z}^n$ that is finitely generated as a $\mathbb{Z}$-module.

The strategy is to find an injective map taking a cone and its integer interior into the intersection of some sublattice of $\mathbb{Z}^n$ with the positive orthant of $\mathbb{Q}^n$ and use the Gordan-Dickson Lemma to construct a finitely terminating algorithm producing decreasing sequences of vectors.

Theorem 4.6 (Gordan's Theorem)**.** *Let $A \in \mathbb{Q}^{m\times n}$ Exactly one of the following is true:*

1. $\exists y \in \mathbb{Q}^m \quad s.t.\ y^T A > 0$
2. $\exists x \in \mathbb{Q}^n_{>0} \quad s.t.\ Ax = 0$

Proof. Suppose both are true. Then let $z^T = y^T A > 0$ and $0 = Ax$ for some x. This gives a contradiction:

$$0 = y^T(Ax) = z^T x > 0$$

Let

$$S_1 = \{z \in \mathbb{Q}^n \ : \ z = y^T A,\ y \in \mathbb{Q}^n\} \qquad S_2 = \mathbb{Q}^n_{>0}$$

Then if (1) has no solutions these sets are disjoint. Assume that it is the case; then by Farkas' lemma they are contained in two disjoint closed halfspaces with a common boundary. Let $x \geq 0$ be the unit vector normal to the separating hyperplane. Then we get:

$$x^T z \geq 0 \quad \forall z \in S_2, \qquad (Ax)^T y \leq 0 \quad \forall y \in \mathbb{Q}^n$$

Assume that $y = Ax \neq 0$. We see that $||y|| = y^T y \leq 0$ So either $Ax = 0,\ x \neq 0$ or (1) is true. □

Observe that for cone $\mathcal{C}$ generated by the (linearly independent) columns of an integral $n \times m$ matrix A we have:

$$A^T y \geq 0 \ \forall y \in \mathcal{C}^D \qquad ker(A) \cap \mathbb{Q}^m_{\geq 0} = 0$$

By pointedness of $\mathcal{C}^D$ and Gordan's Theorem we get:

$$ker(A^T) \cap \mathcal{C}^D = 0$$

We infer two facts:

Lemma 4.7. *A^T is a $\mathbb{Z}$-module homomorphism sending $\mathbb{Z}^n$ to the sublattice of $\mathbb{Z}^m$ spanned by the columns of A^T*

Proof. By acting with A^T on the lattice basis for $\mathbb{Z}^n$ we get a sublattice basis in $\mathbb{Z}^m$ spanned by the columns of A^T. □

Lemma 4.8. *A^T is an injective monoid isomorphism taking $\mathcal{C}^D$ to $\mathbb{Q}^m_{\geq 0}$.*

Proof. The restriction of A^T to $\mathcal{C}^D$ has the morphism property, is injective and preserves the additive identity. Moreover since A^T is a module homomorphism, it preserves the lattice order so there is a surjective left inverse mapping from $\mathbb{Q}^m_{\geq 0}$ to $\mathcal{C}^D$. □

Corollary 4.9. *Let $\mathcal{L}$ be the $\mathbb{Z}$-submodule of $\mathbb{Z}^m$ spanned by the columns of A^T. Then A^T sends $\mathbb{Z}^n \cap \mathcal{C}^D$ to $\mathcal{L} \cap \mathbb{Q}^m_{>0}$.*

4.3 Project and lift algorithm

So far we have developed a way to embed $\mathcal{C}^D$ in a positive orthant of a higher dimensional space (failing that we can solve the dual problem). Moreover, we have managed to map integer interior of $\mathcal{C}^D$ to lattice points with known generators. What remains is to figure out a way to enumerate the minimal generators of the new lattice point set.

Let $G := \{p_1, \ldots, p_s\} \subset \mathbb{Z}^n$ generate a sublattice $\mathcal{L} \subseteq \mathbb{Z}^n$ over $\mathbb{Z}$. We will look at the monoids $(\mathcal{L} \cap \mathbb{R}^n_{\geq 0}, +)$

By discussion in subsection 4.2 we can assume that we have transformed the generators into Hermite normal form, i.e:

$$\begin{array}{c} \begin{matrix} p_1 & p_2 & \cdots & p_s \end{matrix} \\ \begin{bmatrix} p_{1,1} & 0 & \dots & 0 \\ p_{1,2} & p_{2,2} & & 0 \\ \vdots & & \ddots & \vdots \\ p_{1,s} & \cdots & \cdots & p_{s,s} \\ \vdots & & & \vdots \\ p_{1,n} & \cdots & \cdots & p_{s,n} \end{bmatrix} \end{array}$$

Definition 4.10. Let $G := \{p_1, \dots, p_s\}$ be vectors in $\mathbb{Z}^n$. Let $\mathcal{L}$ be a $\mathbb{Z}$-submodule of $\mathbb{Z}^n$ (an integer sublattice of $\mathbb{Z}^n$) finitely generated by G. For any $m > j$ define $\pi_j^m : \mathbb{Q}^m \to \mathbb{Q}^j$ to be the projection onto the first j coordinates. Let $K_j := \{\pi_j^n(v) \quad v \in \mathcal{L}\}$ be the *projection of the lattice* $\mathcal{L}$ *to first* j *coordinates* and let $K_j^+ := K_j \cap (\mathbb{Q}_{\geq 0}^{j-1} \times \mathbb{Q}_{\geq 0})$ and $K_j^- := K_j \cap (\mathbb{Q}_{\geq 0}^{j-1} \times \mathbb{Q}_{\leq 0})$.

Remark. Note that $H_1^+ = \{p_{1,1}\}$

Lemma 4.11. *The Hilbert basis of* K_j^+ *is the inclusion minimal set of integral vectors* $G \subset K_j^+ \setminus \{0\}$ *with the* positive sum property *with respect to* $K_j^+ \setminus \{0\}$

There are three components of the algorithm using the same structure as Buchberger's Gröbner basis algorithm.

Definition 4.12 (S-vectors)**.** [Hem02] This is the analogue of the computation of *S-polynomials* in Algorithm 3.2. The set of *S-vectors* (or a critical vector) is represented as:

$$S(f,g) := \begin{cases} f+g & \text{if the last components of } f \text{ and } g \text{ differ in sign} \\ \emptyset & \text{otherwise} \end{cases}$$

The NormalForm is a boolean function that checks if a critical vector f is $\sqsubseteq$ representable with respect to a set of $\sqsubseteq$-minimal vectors $G \subseteq K_{j+1}^{+}$. It corresponds to performing the division algorithm to get a remainder of a polynomial modulo generating set (Algorithm 3.2)

Definition 4.13 (NormalForm)**.** [Hem02]

$$NormalForm(f, G) := \begin{cases} 0 & \text{if} \quad \exists g \in G \text{ s.t. } f \sqsubseteq_{j+1} g \\ 1 & \text{otherwise} \end{cases}$$

At j'th iteration of the algorithm we consider the projection of the input generators onto K_j (4.16) and lift them up to a generating for $K_{j+1}^{+} \cup K_{j+1}^{-}$. We write (h, h') for a vector h with an integer h' added at the end:

Definition 4.14 (Lifting step)**.** [Hem02]

$$F := \begin{cases} \bigcup_{h \in H_j^{+}} \{(h, h') : (h, h') \in K_{j+1}\} & \text{if } j \geq s \\ \bigcup_{h \in H_j^{+}} \{(h, h') : (h, h') \in K_{j+1}\} \cup \{\pi_{j+1}^{n}(p_{j+1}), -\pi_{j+1}^{n}(p_{j+1})\} & \text{if } j < s \end{cases}$$

The iterative step for computing H_{j+1}^{+} from H_j^{+} is given below:

Algorithm 4.15.

With some minor modifications we can also use the above algorithm to compute the extremal rays of cones. Instead of taking integral vectors as the input generators, we take a matrix in the same shape as before, but with rational entries - this can be easily achieved by computing an upper triangular matrix via Gaussian reduction and transposing it.

Definition 4.16. [Hem02] Let $G := \{p_1, \ldots, p_s\}$ be vectors in $\mathbb{Z}^n$. Let L be a $\mathbb{Q}$-submodule of $\mathbb{Q}^n$ finitely generated by G. For any $m > j$ define $\pi_j^m : \mathbb{Q}^m \to \mathbb{Q}^j$ to be the projection onto the first j coordinates. Let $K_j := \{\pi_j^n(v) \;\; v \in L\}$ be the *projection of the lattice L to first j coordinates* and let $\bar{K}_j^{+} := \bar{K}_j \cap (\mathbb{Q}_{\geq 0}^{j-1} \times \mathbb{Q}_{\geq 0})$ and $\bar{K}_j^{-} := \bar{K}_j \cap (\mathbb{Q}_{\geq 0}^{j-1} \times \mathbb{Q}_{\leq 0})$.

thm[H] **Data**: input set F as described above

Result: A set G containing $H^+_{j+1} \cup H^-_{j+1}$

$G := F$

$C := \bigcup_{f,g \in G} S(f,g)$

while $C \neq \emptyset$ **do**

  $s :=$ an element of C with minimal norm

  $C := C \setminus \{s\}$

  if $normalForm(s, G)$ **then**

    $C := C \cup \bigcup_{g \in G} S(f,g)$

    $G := G \cup \{s\}$

  end

end

return G

Algorithm 4.1: Critical Pair/Completion algorithm for computing generating sets of cones [Hem02]

We redefine the partial order on $\mathbb{Q}^n_{\{} \geq >\}$:

$$f \leq g \iff supp(f) \subseteq supp(g)$$

This also changes the norm we are using for ordering the vectors in C - we will order them by increasing value of $|supp(g)|$ And add new definitions for S-vectors and the lifting step:

Definition 4.17 (S-vectors for extremal rays)**.** [Hem02] (v', w' denote the last components of vectors v, w)

$$S(f,g) := \begin{cases} f - (f'/g')g & \text{if } f'g' < 0 \\ \emptyset & \text{otherwise} \end{cases}$$

Definition 4.18 (Lifting step)**.** [Hem02]

$$F := \begin{cases} \bigcup_{h \in R^+_j} \{(h,h') : (h,h') \in \bar{K}_{j+1}\} & \text{if } j \geq s \\ \bigcup_{h \in R^+_j} \{(h,h') : (h,h') \in \bar{K}_{j+1}\} \cup \{\pi^n_{j+1}(p_{j+1}), -\pi^n_{j+1}(p_{j+1})\} & \text{if } j < s \end{cases}$$

Remark. Note that $R_1^+ = \{p_{1,1}\}$

By replacing the appropriate function calls in Algorithm 2 the output is the linearly independent rational vectors that cannot be written as a linear combination of any other vectors in $L \cap \mathbb{Q}^n_{\geq 0}$

We now have a way to compute the Hilbert basis and extremal rays of the dual of the cone $\mathcal{C}^D$ where $\mathcal{C}$ has extremal rays $r_1, \ldots, r_m$. If we want to compute the Hilbert basis of $\mathcal{C}$, we compute the extremal rays of $\mathcal{C}^D$ and then compute the Hilbert basis for $(\mathcal{C}^D)^D$ which gives the Hilbert basis for $\mathcal{C}$.

4.4 Example: Computing the Hilbert Basis from extremal rays

Let $\mathcal{C}$ be a cone generated over $\mathbb{Q}$ by integral rays $(1,2), (2,-3)$. We begin by computing the rays of the dual: $\mathcal{C}^D = \text{cone}((2,-1),(3,2))$. We bring the generators to the transpose Hermite normal form:

$$\begin{matrix} & r_1 & r_2 \end{matrix} \quad A = \begin{bmatrix} 2 & 3 \\ -1 & 2 \end{bmatrix} \quad \rightarrow \quad \begin{matrix} & r_1 & r_2 \end{matrix} \quad A^T = \begin{bmatrix} 2 & -1 \\ 3 & 2 \end{bmatrix} \quad \rightarrow \quad \begin{matrix} & r_1 & r_2 \end{matrix} \quad A^T_{hnf} = \begin{bmatrix} 1 & 0 \\ 3 & 7 \end{bmatrix}$$

We begin with H_2^+ and compute the set of critical vectors C:

$$\begin{matrix} & h_1 & h_2 & h_3 \end{matrix} \quad G = H_2^+ = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 7 & -7 \end{bmatrix} \quad \xrightarrow{\text{S-vector}} \quad \begin{matrix} & s_1 \end{matrix} \quad C = \begin{bmatrix} 1 \\ -4 \end{bmatrix}$$

We see that $s_1 \sqsubseteq_2 G$ so we extend the set of critical vectors and add s_1 to G:

$$\begin{matrix} & s_2 & s_3 \end{matrix} \quad C = \begin{bmatrix} 1 & 2 \\ 3 & -1 \end{bmatrix} \qquad \begin{matrix} & h_1 & h_2 & h_3 & s_1 \end{matrix} \quad G = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 3 & 7 & -7 & -4 \end{bmatrix}$$

We pick $s_3 \in C$ since it has the minimal norm and repeat the previous step:

$$C = \begin{array}{c} \begin{matrix} s_2 & s_4 & s_5 \end{matrix} \\ \begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 2 \end{bmatrix} \end{array} \qquad G = \begin{array}{c} \begin{matrix} h_1 & h_2 & h_3 & s_1 & s_2 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 2 \\ 3 & 7 & -7 & -4 & -1 \end{bmatrix} \end{array}$$

Carrying on we get the following sequence of sets C and G (note that for s_3 we don't do anything since it is already in G):

$$s_2 = (1,3)^T \quad C = \begin{array}{c} \begin{matrix} s_4 & s_5 \end{matrix} \\ \begin{bmatrix} 2 & 3 \\ 6 & 2 \end{bmatrix} \end{array}$$

$$G = \begin{array}{c} \begin{matrix} h_1 & h_2 & h_3 & s_1 & s_2 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 2 \\ 3 & 7 & -7 & -4 & -1 \end{bmatrix} \end{array}$$

$$s_5 = (3,2)^T \quad C = \begin{array}{c} \begin{matrix} s_4 & s_6 & s_7 & s_8 \end{matrix} \\ \begin{bmatrix} 2 & 3 & 4 & 5 \\ 6 & -5 & -2 & 1 \end{bmatrix} \end{array}$$

$$G = \begin{array}{c} \begin{matrix} h_1 & h_2 & h_3 & s_1 & s_2 & s_5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 2 & 3 \\ 3 & 7 & -7 & -4 & -1 & 2 \end{bmatrix} \end{array}$$

$$s_7 = (4,-2)^T \quad C = \begin{array}{c} \begin{matrix} s_4 & s_6 & s_8 \end{matrix} \\ \begin{bmatrix} 2 & 3 & 5 \\ 6 & -5 & 1 \end{bmatrix} \end{array}$$

$$G = \begin{array}{c} \begin{matrix} h_1 & h_2 & h_3 & s_1 & s_2 & s_5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 2 & 3 \\ 3 & 7 & -7 & -4 & -1 & 2 \end{bmatrix} \end{array}$$

As we add more vectors to G, the new additions do not necessarily admit a critical pair:

$$s_8 = (5,1)^T \quad C = \begin{matrix} s_4 & s_6 & s_9 & s_{10} & s_{11} \end{matrix} \begin{bmatrix} 2 & 3 & 5 & 6 & 7 \\ 6 & -5 & -6 & -3 & 0 \end{bmatrix}$$

$$G = \begin{matrix} h_1 & h_2 & h_3 & s_1 & s_2 & s_5 & s_8 \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 2 & 3 & 5 \\ 3 & 7 & -7 & -4 & -1 & 2 & 1 \end{bmatrix}$$

$$s_{11} = (7,0)^T \quad C = \begin{matrix} s_4 & s_6 & s_9 & s_{10} \end{matrix} \begin{bmatrix} 2 & 3 & 5 & 6 \\ 6 & -5 & -6 & -3 \end{bmatrix}$$

$$G = \begin{matrix} h_1 & h_2 & h_3 & s_1 & s_2 & s_5 & s_8 & s_{11} \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 2 & 3 & 5 & 7 \\ 3 & 7 & -7 & -4 & -1 & 2 & 1 & 0 \end{bmatrix}$$

Finally the remaining vectors in C are $\sqsubseteq$-representable with respect to G so we can discard them:

$$s_{10} = (6, -3)^T \quad C = \begin{matrix} \\ \\ \end{matrix} \begin{matrix} s_4 & s_6 & s_9 \\ \end{matrix} \begin{bmatrix} 2 & 3 & 5 \\ 6 & -5 & -6 \end{bmatrix}$$

$$G = \begin{matrix} \\ \\ \end{matrix} \begin{matrix} h_1 & h_2 & h_3 & s_1 & s_2 & s_5 & s_8 & s_{11} \\ \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 2 & 3 & 5 & 7 \\ 3 & 7 & -7 & -4 & -1 & 2 & 1 & 0 \end{bmatrix}$$

$$\vdots$$

$$C = \emptyset$$

$$G = \begin{matrix} \\ \\ \end{matrix} \begin{matrix} h_1 & h_2 & h_3 & s_1 & s_2 & s_5 & s_8 & s_{11} \\ \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 2 & 3 & 5 & 7 \\ 3 & 7 & -7 & -4 & -1 & 2 & 1 & 0 \end{bmatrix}$$

By computing the preimage of the set $G \cap \mathbb{Q}^2_{\geq 0}$ under A^T we obtain the Hilbert basis for $\mathcal{C}$:

$$H = \begin{matrix} \\ \\ \end{matrix} \begin{matrix} h_1 & h_2 & h_3 & h_4 & h_5 \\ \end{matrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 2 \\ -1 & 0 & 1 & 2 & -3 \end{bmatrix}$$

4.5 Example: Computing the rays of a dual cone

In $\mathbb{Q}^2$ using Hemmecke's algorithm makes little sense, so we will use a 3-dimensional cone with 4 rays as an example. Let $\mathcal{C}$ with integral rays given

by the columns of the matrix A:

$$A = \begin{matrix} r_1 & r_2 & r_3 & r_4 \end{matrix} \begin{bmatrix} 2 & 0 & 1 & 0 \\ -5 & 1 & 1 & 0 \\ 11 & 0 & 0 & 1 \end{bmatrix}$$

The corresponding transposed matrix in upper triangular reduced row echelon form is given by:

$$A^T_{rref} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{1}{11} & \frac{7}{11} & \frac{-2}{11} \end{bmatrix}$$

Since the first 3 rows of A^T_{rref} are a diagonal matrix, we can begin with the input set R_3^+

$$G = R_3^+ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{1}{11} & \frac{7}{11} & \frac{-2}{11} \end{bmatrix}$$

Computing the critical pairs we get:

$$C = \begin{matrix} c_1 & c_2 & c_3 & c_4 \end{matrix} \begin{bmatrix} 0 & 0 & 1 & 2 \\ \frac{2}{7} & 1 & 0 & 0 \\ 1 & \frac{7}{2} & \frac{1}{2} & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

We pick a vector with the minimal value of $|supp(s)|$:

$$s_1 = (2, 0, 1, 0)^T$$

Since $supp(s_1) \nsubseteq supp(g) \forall g \in G$ we add it to G. No $S - vector$ computation is required since all of the vectors in C have 0 in the last coordinate. Afterwards we proceed with new s_2:

$$s_2 = (1, 0, \frac{1}{2}, 0)$$

Since $supp(s_2) \subseteq supp(s_1)$, we discard it. Proceeding in this fashion with the remaining vectors in C, we obtain the final contents of set G:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & \frac{7}{2} & 1 \\ \frac{1}{11} & \frac{7}{11} & \frac{-2}{11} & 0 & 0 \end{bmatrix}$$

By picking the vectors with non-negative last component and computing their preimage under A^T, we get a set of integral extremal rays of $\mathcal{C}^D$:

$$B = \begin{array}{c} \begin{matrix} r_1 & r_2 & r_3 & r_4 \end{matrix} \\ \begin{bmatrix} 7 & 0 & 0 & 1 \\ -11 & 1 & 5 & 0 \\ 11 & 0 & 2 & 0 \end{bmatrix} \end{array}$$

5 A Python implementation

Most of the matrix algebra and number theoretic tools are inherited from SymPy [Sym14] since it provides tools for exact representation of rationals. The code was written using OOP pattern and the test suite and the source are available online (Appendix A)

5.1 Project and lift algorithm

Most of the functionality is contained in a separate module in `bases_computation.py`.

The `preimage` function computes the preimage of a vector under the transpose operation.

```
def preimage(matrix, vectors, VectorClass):
    nullspaces = [(-1 * v).row_join(matrix).nullspace() for v in vectors]

    nullspaces = [to_postive_halfspace(ns[0]) for ns in nullspaces]
    vectors = [VectorClass(ns[1:matrix.cols+1]) for ns in nullspaces]

    return set(map(convert_to_ZZ, vectors))
```

The `to_positive_halfspace` and `convert_to_ZZ` functions help compute a representation of a vector in the positive halfspace of $\mathbb{Z}^n$

```

def to_postive_halfspace(vector):
    # make sure 1 in the first position
    if vector[0] < 0:
        vector = vector * abs(vector[0])/vector[0]
    return vector

def convert_to_ZZ(vector):

    for i in range(len(vector)):
        if(vector[i].is_Rational):
            vector = vector[i].q * vector

    return vector

```

The `critical_pairs(f, G)` function computes the critical pairs (S-vectors) of a vector f with respect to a generating set G. It returns a set of values for easy membership testing. `is_irreducible` checks if a vector is not $\sqsubseteq$-representable for a choice of partial ordering $\sqsubseteq$ appropriate for either an

extremal ray or a Hilbert basis vector. The `not any()` construction allows the verification process to halt and return when the first vector in G failing the test is found.

```
30 def critical_pairs(s, G):
31     return {v for v in map(s.s_vector, G) if not v.origin }
32
33 def is_irreducible(s, G):
34     return not any(g <= s for g in G)
35
```

Using generator compatible function calls has the benefit of being easily portable to a distributed system.

The main loop of the program is realised in the `construct_generating_set` call. It takes in a matrix A of generators of the dual cone $\mathcal{C}$ as columns in column Hermite normal form and the output type of vector. This allows to reuse almost all of the algorithm, with changes only made to specifications of the vectors in the type of generating set we wish to compute.

```
36 def construct_generating_set(A, VectorClass=BasisElement):
37     H = {VectorClass([A[0, 0]])}
38     s, n = A.shape
39 
40     for j in range(1, n):
41         K = A.T[:j+1, :j+1]
42         F = set()
43         if j < s:
44             F.add(VectorClass(K[:,-1]))
45             F.add(VectorClass(-1*K[:,-1]))
46         for h in H:
47             F.add(h.lift(K))
48 
49         H = cpc(F)
50         # pick H^+
51         H -= set(filter(lambda x: x[-1] < 0, H))
52 
53     return H
```

It returns the generating set H^+ for $colspan(A) \cap \mathbb{R}^n_{\geq 0}$ (the Hilbert basis) or the set of rays of $colspan(A) \cap \mathbb{R}^n_{\geq 0}$. It creates the appropriate set F at each iteration by calling a generator-type specific lift method.

The bulk of each iteration $j \to j+1$ happens inside of a `cpc` call. It takes in the input set F as described in Definitions 4.14 and 4.18 and returns a completed set $H^+_{j+1} \cup H^-_{j+1}$.

```
55  def cpc(G):
56      C = reduce(set.union, (critical_pairs(f, G) for f in G))
57
58      while len(C):
59
60          s = min(C,key=methodcaller('norm'))
61          C.remove(s)
62
63          if is_irreducible(s, G):
64              G.add(s)
65              C |= critical_pairs(s, G)
66
67      return G
```

5.2 Vectors

Vectors are simply matrices with 1 column. They come with instance methods enabling us to overload the operators and perform the basic evaluations using natural syntax. Depending on the type of the generating set required the `__leq__` method will perform a different calculation depending whether we are looking for extremal rays of a cone or its Hilbert basis.

The base class for the generating set vectors is `BaseVector`, a subclass of a sympy class `ImmutableMatrix`. It comes with just one property - `origin` which returns `True` if `self` is a zero vector. This is due to a bug in sympy causing each instance of `ImmutableMatrix` to return `True` as the value of `is_zero` property.

```
5  class BaseVector(ImmutableMatrix):
6      """docstring for BaseVector"""
7
8      def __init__(self, *args, **kwargs):
9          super(BaseVector, self).__init__(*args, **kwargs)
10
11     @property
12     def origin(self):
13         return self.as_mutable().is_zero
14
```

Hilbert basis vectors Hilbert basis vector specific functionality is encapsulated in `BasisVector` class. It uses `__geq__` to encapsulate the $\sqsubseteq$ relation (Definition 4.1). It returns the l_1 norm for the `norm` call.

```
16 class BasisElement(BaseVector):
17     """docstring for BasisElement"""
18     def __init__(self, *args, **kwargs):
19         super(BasisElement, self).__init__(*args, **kwargs)
20
21     def __le__(self, other):
22         return all(
23             [u_i <= v_i for u_i, v_i in itertools.izip(self[:-1], other[:-1])]
24             ) and abs(self[-1]) <= abs(other[-1]) and self[-1]*other[-1] >= 0
25
26     def norm(self):
27         return super(BasisElement, self).norm(1)
```

It also returns the *s-vector* if the other vector has the last coefficient in opposite sign than `self` and zero vector otherwise. It also handles lifting

the vectors from H_j^+ to K_{j+1}:

```
29      def s_vector(self, other):
30          if self[-1]*other[-1] < 0:
31              return self + other
32          else:
33              return BasisElement([0])
34
35      def lift(self, M):
36          if M.rows == M.cols:
37              i = 0
38              h = self.as_mutable().col_join(ImmutableMatrix([0]))
39              while True:
40                  h[-1] = i
41                  echelon_form = M.row_join(h).rref()[0]
42                  coeffs = echelon_form[:,-1].values()
43                  if all(val.is_integer for val in coeffs):
44                      return BasisElement(h)
45                  i = i + 1
46          else:
47              return M * M[:M.cols,:M.cols].solve(self[:M.cols,:M.cols])
```

Extremal rays `ExtremalRay` is the base class for extremal rays of cones.

There are several differences with `BasisElement` class.

The partial order is defined as inclusion of supports:

```
50 class ExtremeRay(BaseVector):
51     """docstring for ExtremeRay"""
52     def __init__(self, *args, **kwargs):
53         super(ExtremeRay, self).__init__(*args, **kwargs)
54 
55     def __le__(self, other):
56         return self.support <= other.support
57 
58     @property
59     def support(self):
60         support = set()
61         for index, el in enumerate(self):
62             if el != 0:
63                 support.add(index)
64         return support
```

The lifting procedure finds the first appropriate ray of K_{j+1} instead of looking for solutions in the positive orthant:

```
66     def lift(self, M):
67         if M.rows == M.cols:
68             comb = M[:self.rows, :self.rows].solve(self)
69             return M[:, :self.rows] * comb
70 
71         else:
72             return M * M[:M.cols,:M.cols].solve(self[:M.cols,:M.cols])
```

The norm is the length of the support of `self` and the `s_vector` computation is adjusted to ensure we produce decreasing sequences of vectors ordered by inclusion of supports:

```
74      def norm(self):
75          return len(self.support)
76
77      def s_vector(self, other):
78          if self[-1]*other[-1] < 0:
79              return self - (self[-1]/other[-1])*other
80          else:
81              return ExtremeRay([0])
```

5.3 Cone

The base entity in the algorithm is a cone. Since we are only interested in computing the minimal generating sets of cones, we can get away by describing the cone as a matrix where the column vectors represent the rays of the cone

```
7   class Cone(ImmutableMatrix):
8       """docstring for Cone"""
9
10      _dual = None
11      _rays = None
12
13      def __init__(self, *args, **kwargs):
14          super(Cone, self).__init__()
```

The constructor assumes that the rays are in correct form and does not verify if any of the rays is an integral multiple of some vector also contained in $\mathcal{C}$.

Given the rays of the cone as columns of the input matrix, an instance of the `Cone` class will construct a dual cone when the `dual` property is first

accessed:

```
16      @property
17      def rays(self):
18          if self._rays is None:
19              self._rays = {ExtremeRay([self.col(i)]) for i in range(self.cols)}
20          return self._rays
21
22      @property
23      def dual(self):
24          if self._dual is None:
25              rays = self._compute_dual()
26              col_matrix = Matrix.hstack(*[ray.as_mutable() for ray in rays])
27              self._dual = Cone(col_matrix)
28          return self._dual
29
30
31      def _compute_dual(self):
32          A, BC = self.rref()
33
34          dual_rays = construct_generating_set(Matrix(A), ExtremeRay)
35          dual_rays = preimage(self.T, dual_rays, ExtremeRay)
36
37          return dual_rays
```

`Cone` will use a similar construction using the facet normals (rays of the dual cone) to obtain the Hilbert basis of a cone:

```
39      def hilbert_basis(self):
40          dual = self.dual
41          return self._hb_from_dual(dual.T)
42
43
44      def _hb_from_dual(self, dual):
45          A, BC = hnf_row(dual.T)
46          basis = construct_generating_set(A)
47
48          return preimage(dual, basis, BasisElement)
```

5.4 Hermite Normal Form

Hermite normal form is computed using a parallelisable algorithm by Jäger and Wagner [JW09]. The main logic can be made run on a parallel system by performing appropriate changes to the main loop.

The reduction step is obtained by repeated computation of the *gcd* of entries via an extended Euclidean algorithm. The row version of the HNF is obtained by transposing the result of the column algorithm ran on the transpose of input (see Appendix A)

6 Conclusions

Hemmecke's algorithm exploits the dual nature of $\mathbb{Z}$ and $\mathbb{Q}$ - the $\mathbb{Z}$ generators of cones exhibit asymptotic behaviour during completion step, while the $\mathbb{Q}$-generators are reduced and ordered based on an argument similar to the construction of ordinals. Unfortunately, the ordering does not allow us to systematize the completion procedure in a way that is easily implemented in parallel. Since we need to consider the critical vectors by increasing norm, each vector needs to 'wait' for its predecessor to finish computing

critical vectors - it is possible that we might compute a new 'smallest' one. Parallelising multiple reducibility checks should yield theoretical speed increases, but this is a computationally efficient procedure and the overhead from managing multiple threads or processes might negate any benefits for smaller computations. What is interesting is that the completion step is not associative, so perhaps there is a quasigroup structure that could extract additional information from the lattice structure and help find another breakable symmetry in the completion process.

The run times for some test cases varied almost thousand-fold despite the bases and generators differing in only one element in $\mathbb{Z}^3$ (the corresponding test case has been marked with 'slow', see Appendix Appendix A) This is due to the selection strategy for the elements of C. Since there is no obvious order in which the critical vectors are generated, the choice of the selection strategy when $j > s$ is non trivial and depends on the structure of the problem.

The theoretical benefits Hemmecke's algorithm are clear. Since it works by extending smaller sets and performs computations only when strictly necessary, its applicability comes down to the problem on hand. As mentioned above, some matrices are more 'difficult' than others and lend themselves better to other solutions - as we have found out with Dr Kasprzyk, the problematic test case is solved much faster using barycentric subdivision.

References

[BG05] Winfried Bruns and Joseph Gubeladze. *Polytopes, rings and K-theory.* Tech. rep. 2005.

[Buc85] Bruno Buchberger. "Basic features and development of the critical-pair/completion procedure". In: *Springer-Verlag New York, Inc.* (Dec. 1985), pp. 1–45. URL: http://dl.acm.org/citation.cfm?id=4933.4934.

[CLO07] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms.* Undergraduate Texts in Mathematics. New York, NY: Springer New York, 2007. ISBN: 978-0-387-35650-1. URL: http://link.springer.com/10.1007/978-0-387-35651-8.

[Hem02] Raymond Hemmecke. "On the computation of Hilbert bases and extreme rays of conesls". In: (2002), pp. 1–10. URL: http://arxiv.org/abs/math/0203105.

[Hem06] Raymond Hemmecke. "Representations of lattice point sets". PhD thesis. University Magdeburg, 2006.

[JW09] Gerold Jäger and Clemens Wagner. "Efficient parallelizations of Hermite and Smith normal form algorithms". In: 35.6 (2009), pp. 345–357.

[Mos] Maria Moszy. *Selected Topics in Convex Geometry.* ISBN: 9780817643966.

[Sch86] Alexander Schrijver. "Theory of linear and integer programming". In: *John Wiley & Sons, Inc.* (June 1986). URL: http://dl.acm.org/citation.cfm?id=17634.

[Seb90] András Sebö. "Hilbert Bases, Caratheodory's Theorem and Combinatorial Optimization". In: *University of Waterloo Press* (May 1990), pp. 431–455. URL: http://dl.acm.org/citation.cfm?id=645585.757886.

[Sym14] SymPy Development Team. *SymPy: Python library for symbolic mathematics*. 2014. URL: `http://www.sympy.org`.

A Additional code listings

The full source code and test cases are available at `https://github.com/swist/M4R`

```
from sympy import pprint, Matrix, floor
from sympy.core.numbers import igcdex


def phi(a, b):
    if b != 0:
        return a - (floor(a/b) * b)
    else:
        return 0


def row_one_gcd(A, i, j, l):
    assert i < A.rows
    assert j < l < A.cols
    if A[i, j] != 0 or A[i, l] != 0:
        u, v, d = igcdex(A[i, j], A[i, l])
        reduced = Matrix([
            [u, -A[i, l]/d],
            [v, A[i, j]/d]
        ])
        s = A.extract(range(A.rows), [j, l]) * (reduced)
        A[:, j] = s[:, 0]
        A[:, l] = s[:, 1]
    return A


```

Listing 1: Code listing for the Hermite normal form computation