A New View on HJLS and PSLQ: Sums and Projections of Lattices

Jingwei Chen[†,‡]
[†]Chengdu Institute of Computer Application, CAS
[‡]CNRS, ENS de Lyon, UCBL, Université de Lyon Laboratoire LIP
velen.chan@163.com

Damien Stehlé
CNRS, ENS de Lyon, Inria UCBL, Université de Lyon Laboratoire LIP
damien.stehle@gmail.com

Gilles Villard
CNRS, ENS de Lyon, Inria, UCBL, Université de Lyon Laboratoire LIP
gilles.villard@ens-lyon.fr

ABSTRACT

The HJLS and PSLQ algorithms are the de facto standards for discovering non-trivial integer relations between a given tuple of real numbers. In this work, we provide a new interpretation of these algorithms, in a more general and powerful algebraic setup: we view them as special cases of algorithms that compute the intersection between a lattice and a vector subspace. Further, we extract from them the first algorithm for manipulating finitely generated additive subgroups of a euclidean space, including projections of lattices and finite sums of lattices. We adapt the analyses of HJLS and PSLQ to derive correctness and convergence guarantees.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms—*Algebraic algorithms*

Keywords

integer relation, lattice, HJLS, PSLQ, LLL

1. INTRODUCTION

A vector $\boldsymbol{m} \in \mathbb{Z}^n \setminus \{\boldsymbol{0}\}$ is called an *integer relation* for $\boldsymbol{x} \in \mathbb{R}^n$ if $\boldsymbol{x} \cdot \boldsymbol{m}^T = 0$. The HJLS algorithm [7, Sec. 3], proposed by Håstad, Just, Lagarias and Schnorr in 1986, was the first algorithm for discovering such a relation (or proving that no small relation exists) that consumed a number of real arithmetic operations polynomial in n and the bit-size of the relation bound. In 1992, Ferguson and Bailey published the other de facto standard algorithm for this task, the PSLQ algorithm [5] (see also [6] for a simplified analysis). We refer to the introduction of [7], and to [6, Sec. 9] for a historical perspective on integer relation finding. Our computational model will assume exact operations on real numbers. In this model, Meichsner has shown in [10, Sec. 2.3.1] that PSLQ is essentially equivalent to HJLS (see also [2, App. B, Th. 7] and the comments in Section 2).

Given as input $\boldsymbol{x} \in \mathbb{R}^n$, HJLS aims at finding a nonzero element in the intersection between the integer lattice $\Lambda = \mathbb{Z}^n$ and the $(n-1)$-dimensional vector subspace $E = \mathrm{Span}(\boldsymbol{x})^\perp \subseteq \mathbb{R}^n$. It proceeds as follows. (1) It first projects the rows of the identity matrix (which forms a basis of Λ) onto E. This leads to n vectors belonging to a vector space of dimension $n-1$. The set of all integer linear combinations of these n vectors may not be a lattice: in full generality, it is only guaranteed to be a *finitely generated additive subgroup*, or *fgas* for short, of $\mathbb{R}^n$ (fgas's are studied in detail in Section 3). (2) It performs unimodular operations (swaps and integral translations) on these n vectors, in a fashion akin to (though different from) the LLL algorithm [8]. This aims at removing the linear dependencies between the fgas generators. (3) It stops computing with the fgas if it finds $n-1$ vectors belonging to the same $(n-2)$-dimensional vector subspace and an n-th vector that is linearly independent with those first $n-1$ vectors. This n-th vector contains a component that cannot be shortened any further using any linear combination of the previous vectors. At this stage, the inverse of the unimodular transformation matrix contains a non-trivial integer relation for $\boldsymbol{x}$. The computationally expensive step of HJLS is the second one, i.e., the manipulation of the fgas representation.

Our results. Our first contribution is to propose a new view on HJLS, and hence PSLQ, in a more general algebraic setup. It (partially) solves a special case of the following lattice and vector space intersection problem `Intersect`: given as inputs a basis of a lattice $\Lambda \subseteq \mathbb{R}^m$ and a basis of the vector subspace $E \subseteq \mathbb{R}^m$, the goal is to find a basis of the lattice $\Lambda \cap E$ (i.e., in the case of HJLS, the lattice of all integer relations). The main step of HJLS for (partially) solving (a particular case of) this problem, i.e., Step (2), is itself closely related to the following structural problem on fgas's. The topological closure $\overline{\mathcal{S}}$ of any fgas $\mathcal{S} \subseteq \mathbb{R}^m$ is the orthogonal sum of a unique lattice component Λ and a unique vector subspace component E, i.e., $\overline{\mathcal{S}} = \Lambda \bigoplus E$. The `Decomp` problem takes as input an fgas $\mathcal{S}$ described by a generating set and returns bases of Λ and E. We exhibit a duality relationship between the `Intersect` and `Decomp` problems that was somewhat implicit in HJLS.

Apart from putting HJLS in a broader context, this new view leads to the first algorithm, which we call `Decomp_HJLS`, for decomposing fgas's. Prior to this work, only special cases

ISSAC'13, June 26–29, 2013, Boston, Massachusetts, USA.

were handled: Pohst's MLLL algorithm [12] (see also [7, Sec. 2]) enables the computation of a basis of a lattice given by linearly dependent lattice vectors; and special cases of fgas's, corresponding to integer relations detection instances, were handled by HJLS and PSLQ. We describe the `Decomp_HJLS` algorithm in details, provide a correctness proof and analyze its convergence by adapting similar analyzes from [6] (which are essentially the same as in [7]). We show that it consumes a number of iterations (akin to LLL swaps) that is $\mathcal{O}(r^3 + r^2 \log \frac{X}{\lambda_1(\Lambda)})$, where r is the rank of the input fgas, X is an upper bound on the euclidean norms of the input generators and $\lambda_1(\Lambda)$ is the minimum of the lattice component Λ. For an fgas $\mathcal{S} \subseteq \mathbb{R}^m$ with n generators, an iteration consumes $\mathcal{O}(nm^2)$ arithmetic operations. Additionally, we prove that the returned lattice basis is reduced, for a notion of reduction that is similar to the LLL reduction.

Finally, we investigate a folklore strategy for solving problems similar to `Decomp`. This approach can be traced back to the original LLL article [8, p. 525]. It consists in embedding the input fgas into a higher-dimensional lattice, and calls the LLL algorithm. In order to ensure that the lattice component of the fgas can be read from the LLL output, we modify the underlying inner product by multiplying a subpart of the LLL input basis by a very small weight. More specifically, if we aim at decomposing the fgas spanned by the rows of a matrix $A \in \mathbb{R}^{n\times m}$, the `Decomp_LLL` algorithm will call LLL on the lattice basis $\left(c^{-1}\cdot I_n | A\right)$, where I_n denotes the n-dimensional identity matrix and $c > 0$. For a sufficiently large c, it is (heuristically) expected the lattice component of the fgas will appear in the bottom right corner of the LLL output.

Notation. All our vectors are row vectors and are denoted in bold. If $\boldsymbol{b}$ is a vector, then $\|\boldsymbol{b}\|$ denotes its euclidean norm. We let $\langle \boldsymbol{b}, \boldsymbol{c}\rangle$ denote the usual inner product between two real vectors $\boldsymbol{b}$ and $\boldsymbol{c}$ sharing the same dimension. If $\boldsymbol{b} \in \mathbb{R}^n$ is a vector and $E \subseteq \mathbb{R}^n$ is a vector space, we let $\pi(\boldsymbol{b}, E)$ denote the orthogonal projection of $\boldsymbol{b}$ onto E. Throughout this paper, we assume exact computations on real numbers. The unit operations are addition, substraction, multiplication, division, comparison of two real numbers, and the floor and square root functions.

2. REMINDERS

We give some brief reminders on lattices, and on the HJLS and PSLQ algorithms. For a comprehensive introduction to lattices, we refer the reader to [13].

LQ decomposition. Let $A \in \mathbb{R}^{n\times m}$ be a matrix of rank r. It has a unique LQ decomposition $A = L \cdot Q$, where the Q-factor $Q \in \mathbb{R}^{r\times m}$ has orthonormal rows (i.e., $QQ^T = I_r$), and the L-factor $L \in \mathbb{R}^{n\times r}$ satisfies the following property: there exist *diagonal indices* $1 \le k_1 < \ldots < k_r \le n$, such that $l_{i,j} = 0$ for all $i < k_j$, and $l_{k_j,j} > 0$ for all $j \le r$ (when $n = r$, the L-factor is lower-triangular with positive diagonal coefficients). The LQ decomposition of A is equivalent to the more classical QR decomposition of A^T.

Definition 2.1. *Let $L = (l_{i,j}) \in \mathbb{R}^{n\times r}$ be a lower trapezoidal matrix with rank r and diagonal indices $k_1 < \ldots < k_r$. We say L is* size-reduced *if $|l_{i,j}| \le \frac{1}{2}\left|l_{k_j,j}\right|$ holds for $i > k_j$.*

Given L, it is possible to find a unimodular matrix $U \in \mathrm{GL}_n(\mathbb{Z})$ such that $U \cdot L$ is size-reduced. Computing U and updating $U \cdot L$ can be achieved within $\mathcal{O}(n^3)$ real arithmetic operations.

Lattices. A euclidean *lattice* $\Lambda \subseteq \mathbb{R}^m$ is a discrete (additive) subgroup of $\mathbb{R}^m$. A *basis* of Λ consists of n linearly independent vectors $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n \in \mathbb{R}^m$ such that $\Lambda = \sum_i \mathbb{Z}\boldsymbol{b}_i$. We say that $B = (\boldsymbol{b}_1^T, \ldots, \boldsymbol{b}_n^T)^T \in \mathbb{R}^{n\times m}$ is a basis matrix of Λ. The integer n is called the *dimension* of Λ. If $n \ge 2$, then Λ has infinitely many bases, that exactly consist in the rows of $U\cdot B$ where B is an arbitrary basis matrix of Λ and U ranges over $\mathrm{GL}_n(\mathbb{Z})$. The *i-th successive minimum* $\lambda_i(\Lambda)$ (for $i \le n$) is defined as the radius of the smallest ball that contains i linearly independent vectors of Λ. The *dual lattice* $\widehat{\Lambda}$ of Λ is defined as $\widehat{\Lambda} = \{\boldsymbol{x} \in \mathrm{Span}(\Lambda) : \forall \boldsymbol{b} \in \Lambda, \langle \boldsymbol{b}, \boldsymbol{x}\rangle \in \mathbb{Z}\}$. If B is a basis matrix of Λ, then $(BB^T)^{-1}B$ is a basis of $\widehat{\Lambda}$, called the *dual basis* of B.

Weakly-reduced bases. Weak reduction is a weakening of the classical notion of LLL reduction. It is very similar to the semi-reduction of [14]. Let $B = (\boldsymbol{b}_1^T, \cdots, \boldsymbol{b}_n^T)^T \in \mathbb{R}^{n\times m}$ be the basis matrix of a lattice Λ, and $L = (l_{i,j})$ be its L-factor. We say the basis $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n$ is *weakly-reduced* with parameters $\gamma > 2/\sqrt{3}$ and $C \ge 1$ if L is size-reduced and satisfies the (generalized) Schönhage condition $l_{j,j} \le C \cdot \gamma^i \cdot l_{i,i}$ for $1 \le j \le i \le n$. Note that a LLL-reduced basis is always weakly-reduced, with $C = 1$. If a lattice basis is weakly-reduced, then

$$\begin{gathered}\|\boldsymbol{b}_i\| \le \sqrt{n} C\gamma^i \cdot l_{i,i}, \\ (\sqrt{n}C^2\gamma^{2i})^{-1} \cdot \lambda_i(\Lambda) \le \|\boldsymbol{b}_i\| \le \sqrt{n}C^2\gamma^{2n} \cdot \lambda_i(\Lambda).\end{gathered} \tag{2.1}$$

HJLS-PSLQ. We recall HJLS [7, Sec. 3] using the PSLQ setting [6]. We call the resulting algorithm HJLS-PSLQ (Algorithm 1). Given $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$, HJLS-PSLQ either returns an integer relation for $\boldsymbol{x}$, or gives a lower bound on $\lambda_1(\Lambda_x)$, where Λ_x is the lattice of all integer relations for $\boldsymbol{x}$, and $\lambda_1(\Lambda_x)$ is the norm of any shortest nonzero vector in Λ_x. The updates of U and Q at Steps 1b, 2a, 2b and 2c are implemented so as to maintain the relationship $UL_x = LQ$ at any stage of the execution. Note that storing and updating Q is not necessary for the execution of the algorithm (and does not appear in [6]). It has been added for easing explanations in Section 4.

Algorithm 1 (HJLS-PSLQ).

Input: $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$ with $x_i \neq 0$ for $i \le n$, $M > 0$ and $\gamma > 2/\sqrt{3}$.

Output: Either return an integer relation for $\boldsymbol{x}$, or claim that $\lambda_1(\Lambda_x) > M$.

1. (a) Normalize $\boldsymbol{x}$, i.e., set $\boldsymbol{x} := \boldsymbol{x}/\|\boldsymbol{x}\|$; set $U := I_n$ and $Q = I_{n-1}$.
   (b) Compute the Q-factor $(\boldsymbol{x}^T|L_x)^T$ of $(\boldsymbol{x}|I_n)^T$; set $L := L_x$; size-reduce L and update U.
2. While $l_{n-1,n-1} \neq 0$ and $\max_i l_{i,i} \ge 1/M$ do
   (a) Choose k such that $\gamma^k \cdot l_{k,k} = \max_{j<n} \gamma^j \cdot l_{j,j}$; swap the k-th and $(k+1)$-th rows of L and update U;
   (b) Compute the LQ decomposition of L; replace L by its L-factor and update Q.
   (c) Size-reduce L and update U.
3. If $l_{n-1,n-1} \neq 0$, return "$\lambda_1(\Lambda_x) > M$". Else return the last column of U^{-1}.

For the proof of termination, it suffices to enforce the partial size-reduction condition $|l_{k+1,k}| \le \frac{1}{2} l_{k,k}$ before swapping

(HJLS), instead of full size-reduction (PSLQ). Along with a stronger size-reduction, PSLQ may have a slightly faster termination for specific cases, due to a refined while loop test. PSLQ has been proposed with the additional nullity test of $(\boldsymbol{x}\cdot U^{-1})_j$ for some $j \leq n$, possibly leading to the early output of the j-th column of U^{-1}. Apart from the latter test, if HJLS is implemented with full size-reduction then PSLQ is equivalent to HJLS [10, Sec. 2.3.1] (see also [2, App. B, Th. 7]). Full size-reduction is essentially irrelevant in the exact real number model. Indeed, HJLS works correctly and consumes $\mathcal{O}(n^3 + n^2\lambda_1(\Lambda_x))$ iterations. The same bound has been established for PSLQ. We note that without loss of generality HJLS has been initially stated with $\gamma = \sqrt{2}$. Both the roles of the size-reduction and the parameter γ may be important in a bit complexity model for keeping integer bit sizes small [8, 7], or in a model based on approximate real number operations for mastering the required number precision [6]. This is outside the scope of the present paper.

3. DECOMP AND INTERSECT

In this section, we provide efficient reductions in both directions between the problem of computing the decomposition of an fgas (`Decomp`) and the problem of computing the intersection of a lattice and a vector subspace (`Intersect`), assuming exact computations over the reals.

3.1 FGAS of a euclidean space

Given $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n \in \mathbb{R}^m$, the *finitely generated additive subgroup* (*fgas* for short) spanned by the $\boldsymbol{a}_i$'s is the set of all integral linear combinations of the $\boldsymbol{a}_i$'s:

$$\mathcal{S} = \sum_{i=1}^{n} \mathbb{Z}\boldsymbol{a}_i = \left\{ \sum_{i=1}^{n} z_i \boldsymbol{a}_i : z_i \in \mathbb{Z} \right\} \subseteq \mathbb{R}^m. \tag{3.1}$$

Given an fgas $\mathcal{S}$ as in (3.1), the matrix $A \in \mathbb{R}^{n\times m}$ whose i-th row is $\boldsymbol{a}_i$ is called a *generating matrix* of $\mathcal{S}$. The rank of A is called the *rank* of the fgas. If a matrix $U \in \mathrm{GL}_n(\mathbb{Z})$, then $U \cdot A$ is also a generating matrix of $\mathcal{S}$.

When the vectors $\boldsymbol{a}_i$ are linearly independent, then the set $\mathcal{S}$ is a lattice and the $\boldsymbol{a}_i$'s form a basis of the lattice. If the $\boldsymbol{a}_i$'s are linearly dependent, but $\mathcal{S}$ can be written as $\mathcal{S} = \sum_{i=1}^{d} \mathbb{Z}\boldsymbol{b}_i$ for some linearly independent $\boldsymbol{b}_i$'s, then $\mathcal{S}$ is also a lattice. In this case, the $\boldsymbol{a}_i$'s are not a basis of $\mathcal{S}$ and $\dim(\mathcal{S}) < n$.

The situation that we are mostly interested in the present work is when $\mathcal{S}$ is not a lattice. The simplest example may be the fgas $\mathbb{Z}+\alpha\mathbb{Z}$ with $\alpha \notin \mathbb{Q}$: it contains non-zero elements that are arbitrarily close to 0, and thus cannot be a lattice. More generally, an fgas can always be viewed as a *finite sum of lattices.*

Fgas's can also be viewed as *orthogonal projections of lattices onto vector subspaces.* Let $\Lambda = \sum_i \mathbb{Z}\boldsymbol{b}_i \subseteq \mathbb{R}^m$ be a lattice and $E \subseteq \mathbb{R}^m$ be a vector subspace. The *orthogonal projection* of Λ onto E, i.e., the set $\pi(\Lambda, E) = \{\boldsymbol{v}_1 \in E : \exists \boldsymbol{v}_2 \in E^\perp, \boldsymbol{v}_1+\boldsymbol{v}_2 \in \Lambda\}$, is an fgas of $\mathbb{R}^m$: it is spanned by the projections of the $\boldsymbol{b}_i$'s. Conversely, given an fgas $\mathcal{S}$ with generating matrix $A \in \mathbb{R}^{n\times m}$, let $\Lambda \subseteq \mathbb{R}^{n+m}$ be the lattice generated by the rows of $(I_n|A)$ and $E = \mathrm{Span}(I_n|0)^\perp \subseteq \mathbb{R}^{n+m}$. Then $\mathcal{S} = \pi(\Lambda, E)$.

3.2 The Decomp and Intersect problems

Consider the topological closure $\overline{\mathcal{S}}$ of an fgas $\mathcal{S} \subseteq \mathbb{R}^m$, i.e., the set of all limits of converging sequences of $\mathcal{S}$ (which is hence a closed additive subgroup in $\mathbb{R}^m$). By [9, Th. 1.1.2] (see also [3, Chap. VII, Th. 2]), there exists a unique lattice $\Lambda \subseteq \mathbb{R}^m$ and a unique vector subspace $E \subseteq \mathbb{R}^m$ such that their direct sum is $\overline{\mathcal{S}}$, and the vector space $\mathrm{Span}(\Lambda)$ spanned by Λ is orthogonal to E. We denote the latter decomposition by $\overline{\mathcal{S}} = \Lambda \bigoplus E$. More explicitly, if $\operatorname{rank}\mathcal{S} = \dim(\mathrm{Span}(\mathcal{S})) = r \leq m$, then there exist $0 \leq d \leq r$, $(\boldsymbol{b}_i)_{i\leq d}$ and $(\boldsymbol{e}_i)_{i\leq r-d}$ in $\mathbb{R}^m$ such that:

- $\overline{\mathcal{S}} = \sum_{i\leq d} \mathbb{Z}\boldsymbol{b}_i + \sum_{i\leq r-d} \mathbb{R}\boldsymbol{e}_i$;
- the r vectors $\boldsymbol{b}_i$ $(i \leq d)$ and $\boldsymbol{e}_i$ $(i \leq r-d)$ are linearly independent;
- for any $i \leq d$ and $j \leq r-d$, we have $\langle \boldsymbol{b}_i, \boldsymbol{e}_j \rangle = 0$.

Then $(\boldsymbol{b}_i)_{i\leq d}$ and $(\boldsymbol{e}_i)_{i\leq r-d}$ are bases of Λ and E, respectively. We call Λ and E the *lattice and vector space components* of $\mathcal{S}$, respectively, and define the *ΛE decomposition* of $\mathcal{S}$ as (Λ, E). The `Decomp` problem is the associated computational task.

Definition 3.1. *The* `Decomp` *problem is as follows: Given as input a finite generating set of an fgas $\mathcal{S}$, the goal is to compute its ΛE decomposition, i.e., find bases for the lattice and vector space components Λ and E.*

The following result, at the core of the correctness analysis of our decomposition algorithm of Section 5, reduces `Decomp` to the task of obtaining an fgas generating set that contains sufficiently many linear independencies.

Lemma 3.2. *Let $(\boldsymbol{a}_i)_{i\leq n}$ be a generating set of an fgas $\mathcal{S}$ with ΛE decomposition $\overline{\mathcal{S}} = \Lambda \bigoplus E$. Define $\boldsymbol{a}'_j$ as the projection of $\boldsymbol{a}_j$ orthogonally to $\mathrm{Span}(\boldsymbol{a}_i)_{i\leq n-k}$, for $n-k+1 \leq j \leq n$ and some $k < n$, and assume the $\boldsymbol{a}'_j$'s are linearly independent. Then $\boldsymbol{a}'_{n-k+1}, \ldots, \boldsymbol{a}'_n$ form a basis of a projection of Λ and $E \subseteq \mathrm{Span}(\boldsymbol{a}_i)_{i\leq n-k}$. Further, if $k = \dim\Lambda$, then $\Lambda = \sum_{n-k+1\leq i\leq n} \mathbb{Z}\cdot\boldsymbol{a}'_i$ and $E = \mathrm{Span}(\boldsymbol{a}_i)_{i\leq n-k}$.*

The proof derives from the definition of the ΛE decomposition. The vector space component E is the largest vector subspace of $\mathrm{Span}(\mathcal{S})$ that is contained in $\overline{\mathcal{S}}$. This characterisation of E implies that it is contained in $\mathrm{Span}(\boldsymbol{a}_i)_{i\leq n-k}$. Indeed, the projections $\boldsymbol{a}'_j$ are linearly independent and lead to a discrete subgroup that must be orthogonal to E. By unicity of the ΛE decomposition, the vectors $\boldsymbol{a}'_{n-k+1}, \ldots, \boldsymbol{a}'_n$ form a basis of a projection of the lattice component Λ.

We now introduce another problem, `Intersect`, which generalizes the integer relation finding problem.

Definition 3.3. *The* `Intersect` *problem is as follows: Given as inputs a basis of a lattice Λ and a basis of a vector subspace E, the goal is to find a basis of the lattice $\Lambda \cap E$.*

Finding a non-zero integer relation corresponds to taking $\Lambda = \mathbb{Z}^n$ and $E = \mathrm{Span}(\boldsymbol{x})^\perp$, and asking for one vector in $\Lambda \cap E$. In that case, `Intersect` aims at finding a description of all integer relations for $\boldsymbol{x}$. When E is arbitrary but Λ remains $\mathbb{Z}^n$, `Intersect` corresponds to the task of finding all *simultaneous integer relations.* These special cases are considered in [7].

3.3 Relationship between the problems

The `Decomp` and `Intersect` problems turn out to be closely related. To explain this relationship, we need the concept of dual lattice of an fgas. The facts of this subsection are adapted from basic techniques on lattices (see, e.g.,[4]).

DEFINITION 3.4. *The* dual lattice $\widehat{\mathcal{S}}$ *of an fgas* $\mathcal{S}$ *is defined as* $\widehat{\mathcal{S}} = \{\boldsymbol{x} \in \mathrm{Span}(\mathcal{S}) : \forall \boldsymbol{b} \in \mathcal{S}, \langle \boldsymbol{x}, \boldsymbol{b} \rangle \in \mathbb{Z}\}$.

We could equivalently define $\widehat{\mathcal{S}}$ as $\{\boldsymbol{x} \in \mathrm{Span}(\mathcal{S}) : \forall \boldsymbol{b} \in \overline{\mathcal{S}}, \langle \boldsymbol{x}, \boldsymbol{b} \rangle \in \mathbb{Z}\}$. Indeed, for all $\boldsymbol{b} \in \overline{\mathcal{S}}$, there exists a converging sequence $(\boldsymbol{b}_i)_i$ in $\mathcal{S}$ such that $\boldsymbol{b}_i \to \boldsymbol{b}$ as $i \to \infty$. Thus, for all $\boldsymbol{x} \in \widehat{\mathcal{S}}$, we have $\langle \boldsymbol{x}, \boldsymbol{b} \rangle = \langle \boldsymbol{x}, \lim_i \boldsymbol{b}_i \rangle = \lim_i \langle \boldsymbol{x}, \boldsymbol{b}_i \rangle \in \mathbb{Z}$. We will freely use both definitions.

Note further that if $\mathcal{S}$ is a lattice, then $\widehat{\mathcal{S}}$ is exactly the dual lattice of $\mathcal{S}$. Interestingly, $\widehat{\mathcal{S}}$ is always a lattice, even if $\mathcal{S}$ is not a lattice.

LEMMA 3.5. *Let* $\mathcal{S}$ *be an fgas and* Λ *its lattice component. Then* $\widehat{\Lambda} = \widehat{\mathcal{S}}$.

PROOF. Let $\overline{\mathcal{S}} = \Lambda \bigoplus E$ be the ΛE decomposition of $\mathcal{S}$. Recall that for any $\boldsymbol{x} \in \overline{\mathcal{S}}$, there exist unique $\boldsymbol{x}_\Lambda \in \Lambda$ and $\boldsymbol{x}_E \in E$ such that $\boldsymbol{x} = \boldsymbol{x}_\Lambda + \boldsymbol{x}_E$ and $\langle \boldsymbol{x}_\Lambda, \boldsymbol{x}_E \rangle = 0$.

We first prove that $\widehat{\Lambda} \subseteq \widehat{\mathcal{S}}$. For all $\widehat{\boldsymbol{x}} \in \widehat{\Lambda}$ and all $\boldsymbol{x} \in \overline{\mathcal{S}}$, we have $\langle \widehat{\boldsymbol{x}}, \boldsymbol{x} \rangle = \langle \widehat{\boldsymbol{x}}, \boldsymbol{x}_\Lambda \rangle + \langle \widehat{\boldsymbol{x}}, \boldsymbol{x}_E \rangle = \langle \widehat{\boldsymbol{x}}, \boldsymbol{x}_\Lambda \rangle \in \mathbb{Z}$, where the second equality follows from the orthogonality between the vector subspaces E and $\mathrm{Span}(\Lambda)$, and $\langle \widehat{\boldsymbol{x}}, \boldsymbol{x}_\Lambda \rangle \in \mathbb{Z}$ derives from the definition of $\widehat{\Lambda}$.

Further, for all $\widehat{\boldsymbol{x}} \in \widehat{\mathcal{S}}$ and all $\boldsymbol{x} \in \Lambda \subseteq \overline{\mathcal{S}}$, it follows from the second definition of $\widehat{\mathcal{S}}$ that $\langle \widehat{\boldsymbol{x}}, \boldsymbol{x} \rangle \in \mathbb{Z}$, i.e., we have $\widehat{\boldsymbol{x}} \in \widehat{\Lambda}$. This completes the proof. □

From Lemma 3.5, we derive the following alternative definition of the lattice component of an fgas.

LEMMA 3.6. *Let* $\mathcal{S}$ *be an fgas and* Λ *its lattice component. Then* $\Lambda = \widehat{\widehat{\mathcal{S}}}$.

Let $\Lambda \subseteq \mathbb{R}^m$ be a lattice and $E \subseteq \mathbb{R}^m$ a vector subspace. If $\pi(\widehat{\Lambda}, E)$ happens to be a lattice, then it is exactly $\widehat{\Lambda \cap E}$ (see, e.g., [9, Prop. 1.3.4]). However, in general, the fgas $\pi(\widehat{\Lambda}, E)$ may not be a lattice. Using Definition 3.4, we can prove the following result, which plays a key role in the relationship between `Intersect` and `Decomp`.

LEMMA 3.7. *For any lattice* $\Lambda \subseteq \mathbb{R}^m$ *and a vector subspace* $E \subseteq \mathbb{R}^m$*, we have* $\Lambda \cap E = \widehat{\pi(\widehat{\Lambda}, E)}$.

PROOF. Let $\boldsymbol{b} \in \Lambda \cap E$ and $\boldsymbol{y} \in \pi(\widehat{\Lambda}, E)$. There exist $\widehat{\boldsymbol{b}} \in \widehat{\Lambda}$ and $\boldsymbol{y}' \in E^\perp$ such that $\widehat{\boldsymbol{b}} = \boldsymbol{y} + \boldsymbol{y}'$. Then

$$\langle \boldsymbol{b}, \boldsymbol{y} \rangle = \langle \boldsymbol{b}, \widehat{\boldsymbol{b}} \rangle - \langle \boldsymbol{b}, \boldsymbol{y}' \rangle = \langle \boldsymbol{b}, \widehat{\boldsymbol{b}} \rangle \in \mathbb{Z}.$$

Hence $\Lambda \cap E \subseteq \widehat{\pi(\widehat{\Lambda}, E)}$.

Now, let $\boldsymbol{b} \in \widehat{\pi(\widehat{\Lambda}, E)}$. By definition, we have

$$\boldsymbol{b} \in \mathrm{Span}(\pi(\widehat{\Lambda}, E)) \subseteq E.$$

Moreover, for all $\widehat{\boldsymbol{b}} \in \widehat{\Lambda}$, using $\widehat{\boldsymbol{b}} = \pi(\widehat{\boldsymbol{b}}, E) + \pi(\widehat{\boldsymbol{b}}, E^\perp)$:

$$\langle \boldsymbol{b}, \widehat{\boldsymbol{b}} \rangle = \langle \boldsymbol{b}, \pi(\widehat{\boldsymbol{b}}, E) \rangle + \langle \boldsymbol{b}, \pi(\widehat{\boldsymbol{b}}, E^\perp) \rangle = \langle \boldsymbol{b}, \pi(\widehat{\boldsymbol{b}}, E) \rangle \in \mathbb{Z}.$$

Hence $\boldsymbol{b} \in \widehat{\widehat{\Lambda}} = \Lambda$. We obtain that $\boldsymbol{b} \in \Lambda \cap E$, which completes the proof. □

Reducing `Decomp` to `Intersect`. Suppose we are given a generating set $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n \in \mathbb{R}^m$ of an fgas $\mathcal{S}$. Our goal is to find the ΛE decomposition of $\mathcal{S}$, using an oracle that solves `Intersect`. It suffices to find a basis of the lattice component Λ, which, by Lemma 3.6, satisfies $\Lambda = \widehat{\widehat{\mathcal{S}}}$.

Recall that we can construct a lattice Λ' and a vector space E such that (see the end of Section 3.1) $\mathcal{S} = \pi(\Lambda', E)$. From Lemma 3.7, the lattice component Λ of $\mathcal{S}$ is the dual lattice of $\widehat{\Lambda'} \cap E$. This means we can get a basis of $\widehat{\Lambda}$ by calling the `Intersect` oracle on $\widehat{\Lambda'}$ and E, and then computing the dual basis of the returned basis.

Reducing `Intersect` to `Decomp`. Assume we are given a basis $(\boldsymbol{b}_i)_i$ of a lattice $\Lambda \subseteq \mathbb{R}^m$ and a basis $(\boldsymbol{e}_i)_i$ of a vector subspace $E \subseteq \mathbb{R}^m$. We aim at computing a basis of the lattice $\Lambda \cap E$, using an oracle that solves `Decomp`.

We first compute the dual basis $(\widehat{\boldsymbol{b}}_i)_i$ of $(\boldsymbol{b}_i)_i$. Then we compute the projections $\widehat{\boldsymbol{b}}'_i = \pi(\widehat{\boldsymbol{b}}_i, E)$, for all i. Let $\mathcal{S}$ denote the fgas spanned by $(\widehat{\boldsymbol{b}}'_i)_i$. We now use the `Decomp` oracle on $\mathcal{S}$ to obtain a basis of the lattice component Λ' of $\mathcal{S}$. Then, by Lemma 3.7, the dual basis of the oracle output is a basis of $\Lambda \cap E$.

4. A NEW VIEW ON HJLS-PSLQ

We explain the principle of HJLS-PSLQ described in Section 2, by using the results of Section 3. At a high level, HJLS-PSLQ proceeds as in the `Intersect` to `Decomp` reduction from Section 3. The algorithm in Section 2 halts as soon as a relation is found. Hence it only partially solves `Decomp`, on the specific input under scope, and, as a result, only partially solves `Intersect`. The full decomposition will be studied in Section 5.

Step 1 revisited: Projection of $\mathbb{Z}^n$ on $\mathrm{Span}(\boldsymbol{x})^\perp$. The reduction from `Intersect` to `Decomp` starts by projecting $\widehat{\Lambda}$ onto E. In our case, we have $\Lambda = \widehat{\Lambda} = \mathbb{Z}^n$ (the lattice under scope is self-dual) and $E = \mathrm{Span}(\boldsymbol{x})^\perp$. The start of the reduction matches with the main component of Step 1, which is the computation of the Q-factor $Q_x := (\boldsymbol{x}^T | L_x)^T$ of $(\boldsymbol{x}^T | I_n)^T$, considering $\boldsymbol{x}$ after normalization. Using that $Q_x^T \cdot Q_x = I_n$ we now observe that L_x satisfies the following equation

$$\begin{pmatrix} \boldsymbol{x} \\ I_n \end{pmatrix} = \begin{pmatrix} 1 & \\ \boldsymbol{x}^T & L_x \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x} \\ L_x^T \end{pmatrix}.$$

By construction, the matrix L_x is lower trapezoidal. Indeed, since the i-th row of Q_x is orthogonal to the linear span of the first $i-1$ rows, and as this linear span contains the first $i-2$ unit vectors, the first $i-2$ coordinates of this i-th row of Q_x are zero. Hence the equation above provides the LQ decomposition of $\left(\boldsymbol{x}^T | I_n\right)^T$. It is worth noticing the unusual fact that L_x is involved in both the L-factor and the Q-factor. Also, as a consequence of the equation above, we have that the matrix $\pi_x = I_n - \boldsymbol{x}^T \boldsymbol{x} = L_x L_x^T$ corresponds to the orthogonal projection that maps $\mathbb{R}^n$ to $\mathrm{Span}(\boldsymbol{x})^\perp$. Therefore the rows of L_x are the coordinate vectors of the rows of π_x with respect to the normalized orthogonal basis of $\mathrm{Span}(\boldsymbol{x})^\perp$ given by the $n-1$ rows of L_x^T. Overall, we obtain that $(\boldsymbol{0}|L_x) \cdot Q_x$ is a generating matrix of the fgas $\mathcal{S}_x = \pi(\mathbb{Z}^n, \mathrm{Span}(\boldsymbol{x})^\perp)$.

Step 2 revisited: A partial solution to `Decomp`. Since $(\boldsymbol{0}|L_x) \cdot Q_x$ is a generating matrix of the fgas $\mathcal{S}_x$, the while loop of HJLS-PSLQ only considers this fgas (in fact, HJLS-PSLQ only works on L_x since its only requires U). In Section 5, we will show that a generalization of the while loop may be used to solve the `Decomp` problem. By Lemma 3.7, finding a basis of the lattice component of $\mathcal{S}_x$ suffices to find all integer relations of $\boldsymbol{x}$: indeed, the dual basis is a basis of the integer relation lattice. However, when HJLS-PSLQ terminates, we may not have the full lattice component Λ' of $\mathcal{S}_x$. If the loop stops because $l_{n-1,n-1} = 0$, then we have found a projection to a 1-dimensional subspace of a vector belonging to the lattice component. In this sense, Step 2 of HJLS-PSLQ partially solves `Decomp` on input $\mathcal{S}_x$. It gets the full solution only when $\dim(\mathbb{Z}^n \cap \mathrm{Span}(\boldsymbol{x})^\perp) = \dim(\Lambda') = 1$.

Step 3 revisited: Getting back to `Intersect`. Suppose HJLS-PSLQ exits the while loop because $l_{n-1,n-1} = 0$. Because of the shape of L (see Lemma 3.2), it has found a 1-dimensional projection of a non-zero basis vector of Λ', orthogonally to the first vectors of that basis of Λ'. This vector is:

$$\boldsymbol{b} := (\mathbf{0}|l_{n-1,n-1}) \cdot \mathrm{diag}(1, Q) \cdot Q_x.$$

Its dual, when considered as a basis, is

$$\widehat{\boldsymbol{b}} = \boldsymbol{b}/\|\boldsymbol{b}\|^2 = (\mathbf{0}|l_{n-1,n-1}^{-1}) \cdot \mathrm{diag}(1, Q) \cdot Q_x.$$

As $\widehat{\boldsymbol{b}}$ is a projection of a non-zero basis vector of Λ', orthogonally to the first vectors of that basis, we have that $\widehat{\boldsymbol{b}}$ belongs to $\widehat{\Lambda'} = \mathbb{Z}^n \cap \mathrm{Span}(\boldsymbol{x})^\perp$. Because of the specific shape of Q_x, we obtain

$$\begin{aligned}\widehat{\boldsymbol{b}} &= (\mathbf{0}|l_{n-1,n-1}^{-1}) \cdot \left(\begin{array}{c|c} 1 & \\ \hline & Q \end{array}\right) \cdot \left(\begin{array}{c} \boldsymbol{x} \\ \hline L_x^T \end{array}\right) \\ &= (\mathbf{0}|l_{n-1,n-1}^{-1}) \cdot \left(\begin{array}{c} \boldsymbol{x} \\ \hline QL_x^T \end{array}\right).\end{aligned}$$

Now, as $UL_x = LQ$, we obtain that $\widehat{\boldsymbol{b}} = (\mathbf{0}|l_{n-1,n-1}^{-1}) \cdot (\boldsymbol{x}^T|U^{-1}L)^T = (\mathbf{0}|1)U^{-T}$. This explains why the relation is embedded in the inverse of the transformation matrix. Note that this is somewhat unexpected, and derives from the uncommon similarity between L_x and Q_x.

A numerical example. Consider the input $(1, \sqrt{2}, 2)$. After normalization, it becomes $\boldsymbol{x} = (\frac{1}{\sqrt{7}}, \frac{\sqrt{2}}{\sqrt{7}}, \frac{2}{\sqrt{7}})$. At the beginning, we have

$$L_x = \begin{pmatrix} \frac{6}{\sqrt{42}} & 0 \\ -\frac{\sqrt{2}}{\sqrt{42}} & \frac{\sqrt{2}}{\sqrt{3}} \\ -\frac{2}{\sqrt{42}} & -\frac{1}{\sqrt{3}} \end{pmatrix} \text{ and } Q_x = \begin{pmatrix} \frac{1}{\sqrt{7}} & \frac{\sqrt{2}}{\sqrt{7}} & \frac{2}{\sqrt{7}} \\ \frac{\sqrt{6}}{\sqrt{42}} & -\frac{\sqrt{2}}{\sqrt{42}} & -\frac{2}{\sqrt{42}} \\ 0 & \frac{\sqrt{2}}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{pmatrix}.$$

The matrix $(\mathbf{0}|L_x) \cdot Q_x$ is a generating matrix of the fgas $\mathcal{S}_x = \pi(\mathbb{Z}^n, \mathrm{Span}(\boldsymbol{x})^\perp)$. After 5 loop iterations, HJLS-PSLQ terminates. At that stage, we obtain

$$L = \begin{pmatrix} \frac{15-10\sqrt{2}}{\sqrt{35}} & 0 \\ -\frac{5(-41+29\sqrt{2})}{\sqrt{35}(-3+2\sqrt{2})} & 0 \\ \frac{41\sqrt{2}-58}{\sqrt{35}(-3+2\sqrt{2})} & \frac{1}{\sqrt{5}} \end{pmatrix},$$

$$U = \begin{pmatrix} -2 & -3 & -4 \\ 5 & 7 & 10 \\ -1 & -2 & -3 \end{pmatrix},\ Q = \begin{pmatrix} \frac{-4+3\sqrt{2}}{\sqrt{30}(3-2\sqrt{2})} & -\frac{\sqrt{14}}{\sqrt{15}} \\ \frac{\sqrt{14}}{\sqrt{15}} & \frac{-24+17\sqrt{2}}{\sqrt{30}(17-12\sqrt{2})} \end{pmatrix}.$$

Thanks to the shape of L, the ΛE decomposition $\overline{\mathcal{S}}_x = \Lambda \bigoplus E$ can be derived from $(\mathbf{0}|L)$. In this precise case, HJLS-PSLQ discloses the full lattice component. Thanks to Lemma 3.7, we have $\Lambda = \widehat{\Lambda}_x$, and hence $\dim(\Lambda) = \dim(\widehat{\Lambda}_x) = \dim(\Lambda_x) = 1$ (as $\boldsymbol{x}$ contains two rational entries and one irrational entry). Using the matrix factorisation above, we obtain

$$\Lambda = \mathbb{Z} \cdot (0, 0, 1/\sqrt{5}) \cdot \mathrm{diag}\,(1, Q) \cdot Q_x = \mathbb{Z} \cdot (2/5, 0, -1/5)$$

and $E = (0, 1, 0) \cdot \mathrm{diag}(1, Q) \cdot Q_x$. By Lemma 3.7, we obtain $\mathbb{Z}^3 \cap \mathrm{Span}(\boldsymbol{x})^\perp = \widehat{\Lambda} = \mathbb{Z} \cdot (2, 0, -1)$. Note that we recovered the last column vector of U^{-1}.

5. SOLVING `DECOMP` *À LA* HJLS

Let $A \in \mathbb{R}^{n\times m}$ be a generating matrix of an fgas $\mathcal{S}$ and $\overline{\mathcal{S}} = \Lambda \bigoplus E$ be the ΛE decomposition of $\mathcal{S}$ with $\dim(\Lambda) = d$. In this section, we present and analyze an algorithm, named `Decomp_HJLS`, for solving the `Decomp` problem.

Note that `Decomp_HJLS` requires as input the dimension d of the lattice component. One might ask whether there exists an algorithm, based on the unit cost model over the reals, solving the problem without knowing d before. This is actually not the case: In [1], Babai, Just and Meyer auf der Heide showed that, in this model, it is not possible to decide whether there exists a relation for given input $\boldsymbol{x} \in \mathbb{R}^n$. Computing the dimension of the lattice component of an fgas would allow us to solve that decision problem.

5.1 The `Decomp_HJLS` algorithm

`Decomp_HJLS`, given as Algorithm 2, is a full fgas decomposition. It is derived, thanks to the new algebraic view, from the Simultaneous Relations Algorithm in [7, Sec. 5]. The latter is a generalization of the Small Integer Relation Algorithm of Section 2 which contains, as we have seen, a partial decomposition algorithm. We keep using the PSLQ setting and follow the lines of [11, Sec. 2.5]. In particular we adopt a slight change, with respect to [7, Sec. 5], in the swapping strategy. (The index κ' we select, hereafter at Step 2c of Algorithm 2, may differ from $\kappa + 1$.) However, as for differences between HJLS and PSLQ we have seen in Section 2, there is no impact on the asymptotic number of iterations.

We introduce the next definition to describe different stages in the execution of the algorithm, using the shape of the current L-factor L.

Definition 5.1. *Let $0 \le \ell \le r$. If a lower trapezoidal matrix $L \in \mathbb{R}^{n\times r}$ can be written as*

$$L = \begin{pmatrix} M & \\ F & \\ G & N \end{pmatrix},$$

with $F \in \mathbb{R}^{(n-r)\times(r-\ell)}$, $G \in \mathbb{R}^{\ell\times(r-\ell)}$, and both $N \in \mathbb{R}^{\ell\times\ell}$ and $M \in \mathbb{R}^{(r-\ell)\times(r-\ell)}$ are lower triangular with positive diagonal coefficients, then we say that L has shape Trap(ℓ).

`Decomp_HJLS` takes as input an fgas generating matrix. It also requires the dimension of the lattice component (see the end of Section 3.2). Without loss of generality, we may assume that the initial L-factor $L^{(0)}$ has shape Trap(0) (this is provided by Step 1a). The objective of `Decomp_HJLS` is to apply unimodular transformations (namely, size-reductions and swaps) to a current generating matrix $L \cdot Q$ of the input fgas, in order to eventually obtain an L-factor that has shape Trap(d), where d is the dimension of the lattice component. These unimodular transformations are applied through successive loop iterations (Step 2), that progressively modify the shape of the current L-factor from Trap(0) to Trap(1), ..., and eventually to Trap(d). When the latter event occurs, the algorithm exits the while loop and moves on to Step 3: the lattice component can now be extracted by taking the last d rows of L and cancelling their first $r-d$ columns, where r is the rank of L.

Algorithm 2 (Decomp_HJLS).

Input: A generating matrix $A = (\boldsymbol{a}_1^T, \cdots, \boldsymbol{a}_n^T)^T \in \mathbb{R}^{n\times m}$ of an fgas $\mathcal{S}$ with $\max_{i\le m} \|\boldsymbol{a}_i\|_2 \le X$; a positive integer d as the dimension of the lattice component Λ of $\mathcal{S}$; a parameter $\gamma > 2/\sqrt{3}$.

Output: A basis matrix of Λ.

1. (a) Compute $r = \operatorname{rank}(A)$. If $d = r$, then return $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_r$. Else, using row pivoting, ensure that the first r rows of A are linearly independent.
   (b) Compute the LQ decomposition $A = L_0 \cdot Q_0$.
   (c) Set $L := L_0$ and size-reduce it; set $Q := Q_0$ and $\ell := 0$.
2. While $l_{r-d+1,r-d+1} \neq 0$ do
   (a) Choose κ such that $\gamma^\kappa \cdot l_{\kappa,\kappa} = \max_{k\le r-\ell} \gamma^k \cdot l_{k,k}$.
   (b) If $\kappa < r-\ell$, then swap the κ-th and the $(\kappa+1)$-th rows of L; compute the LQ decomposition of L; replace L by its L-factor and update Q.
   (c) Else swap the κ-th and κ'-th rows of L, where $\kappa' \ge \kappa+1$ is the largest index such that $|l_{\kappa',\kappa}| = \max_{\kappa+1\le k\le n-\ell} |l_{k,\kappa}|$. If $l_{\kappa,\kappa} = 0$, set $\ell := \ell + 1$.
   (d) Size-reduce L.
3. Return $\left(\mathbf{0}_{d\times(r-d)} | (l_{i,j})_{i\in[n-d+1,n], j\in[r-d+1,r]}\right) \cdot Q$.

In the remainder of this section, we let $L^{(t)} = (l_{i,j}^{(t)})$ denote the matrix L at the beginning of the t-th loop iteration of `Decomp_HJLS`. We also let $\ell(t)$ and $\kappa(t)$ respectively denote the values of ℓ and κ at the end of Step 2a of the t-th loop iteration. We let τ denote the total number of loop iterations and $L^{(\tau+1)}$ and $Q^{(\tau+1)}$ respectively denote the values of L and Q at Step 3.

5.2 The correctness of `Decomp_HJLS`

Note that if $\kappa(t) = r - \ell(t)$ and $l^{(t)}_{\kappa'(t),r-\ell(t)} = 0$, then $l^{(t+1)}_{r-\ell,r-\ell} = 0$. This is the only situation that transforms L from shapes Trap(ℓ) to Trap($\ell+1$), i.e., that decrements (resp. increments) the dimension of the triangular matrix M (resp. N) from Definition 5.1. Indeed, LQ decompositions, size-reductions and swaps of consecutive vectors of indices $\kappa < r-\ell$ preserve the trapezoidal shape of L.

The two lemmas below give insight on the execution of the algorithm. They will be useful especially for proving that `Decomp_HJLS` terminates, and bounding the number of iterations. On the one hand, the maximum of the diagonal coefficients of the M-part of the current L-factor does not increase during the successive loop iterations (Lemma 5.2). On the other hand, because of the existence of the lattice component, which is linearly independent from the vector space component, these diagonal coefficients cannot decrease arbitrarily while maintaining the dimension of M. As long as the lattice component has not been fully discovered, this maximum must remain larger than the first minimum of that lattice (Lemma 5.3).

LEMMA 5.2. *For any $t \in [1,\tau]$, we have $\max_i l_{i,i}^{(t+1)} \le \max_i l_{i,i}^{(t)}$, where i ranges over $[1, r-\ell(t+1)]$ and $[1, r-\ell(t)]$ respectively.*

The proof is standard. The only $l_{i,i}$'s that may change are those that correspond to the swapped vectors, and the non-increase of the maximum of this or these $l_{i,i}$'s originates from the choice of the swapping index.

LEMMA 5.3. *Let Λ be the lattice component of the input fgas, and $d = \dim(\Lambda) \ge 1$. Then, for any $t \in [1,\tau]$, we have*

$$\lambda_1(\Lambda) \le \max_{i\le r-\ell(t)} l_{i,i}^{(t)}.$$

PROOF. The matrix $L^{(\tau+1)}$ has shape Trap(d), and

$$\left(\mathbf{0}^{r-d}, l^{(\tau+1)}_{n-d+1,r-d+1}, \mathbf{0}^{d-1}\right) \cdot Q^{(\tau+1)}$$

belongs to Λ (by Lemma 3.2). As the matrix $Q^{(\tau+1)}$ is orthogonal, it has norm $l^{(\tau+1)}_{n-d+1,r-d+1}$. We thus have $\lambda_1(\Lambda) \le l^{(\tau+1)}_{n-d+1,r-d+1}$. Now, as τ is the last loop iteration, Step 2c must have been considered at that loop iteration, with a swap between rows $\kappa(\tau) = r-d+1$ and $n-d+1$ of $L^{(\tau)}$. We thus obtain:

$$\begin{aligned}\lambda_1(\Lambda) &\le l^{(\tau+1)}_{n-d+1,r-d+1} = l^{(\tau)}_{r-d+1,r-d+1} \\ &\le \max_{i\le r-d+1} l^{(\tau)}_{i,i} \quad \le \max_{i\le r-\ell(t)} l^{(t)}_{i,i}\ .\end{aligned}$$

The last inequality follows from Lemma 5.2. □

We now prove the correctness of the `Decomp_HJLS` algorithm, i.e., that it returns a basis of the lattice component of the input fgas. We also prove that the returned lattice basis is weakly-reduced (see Section 2), and hence that the successive basis vectors are relatively short compared to the successive lattice minima (by Equation (2.1)).

THEOREM 5.4. *If the `Decomp_HJLS` algorithm terminates (which will follow from Theorem 5.6), then it is correct: given a generating matrix of a rank r fgas $\mathcal{S}$ as input and the dimension d of its lattice component, it returns a weakly-reduced basis, with parameters γ and $C = \gamma^{r-d}$, of the lattice component of $\mathcal{S}$.*

PROOF. At the end of the while loop in `Decomp_HJLS`, the L-factor $L^{(\tau+1)}$ has shape Trap(d), where $d = \dim(\Lambda)$. As we only apply unimodular operations to the row vectors, the fgas $\mathbb{Z}^n \cdot L^{(\tau+1)} \cdot Q^{(\tau+1)}$ matches the input fgas $\mathbb{Z}^n \cdot A$. Let $\Lambda' = \mathbb{Z}^d \cdot \left(\mathbf{0}_{d\times(r-d)} | (l^{(\tau+1)}_{i,j})_{i\in[n-d+1,n], j\in[r-d+1,r]}\right) \cdot Q^{(\tau+1)}$ denote the output of `Decomp_HJLS`. By Lemma 3.2, the lattice Λ' is exactly the lattice component Λ.

Let $L' \in \mathbb{R}^{d\times d}$ be the matrix corresponding to the bottom right d rows and d columns of L. We now check that L' is size-reduced and satisfies the Schönhage conditions. Thanks to the size-reductions of Steps 1c and 2d, the whole matrix $L^{(\tau+1)}$ is size-reduced. It remains to show that $l'_{j,j} \le \gamma^{r-d+i} \cdot l'_{i,i}$ for all $1 \le j < i \le d$. For this purpose, we consider two moments $t_i < t_j$ during the execution of the algorithm: the t_i-th (resp. t_j-th) loop iteration is the first one such that $L^{(t)}$ has shape Trap($d-i+1$) (resp. Trap($d-j+1$)). By construction of t_i and t_j, we have:

$$\begin{aligned} l'_{i,i} &= l^{(\tau+1)}_{n-d+i,r-d+i} = l^{(t_i)}_{n-d+i,r-d+i}, \\ l'_{j,j} &= l^{(\tau+1)}_{n-d+j,r-d+j} = l^{(t_j)}_{n-d+j,r-d+j}. \end{aligned}$$

As t_i and t_j are chosen minimal, Step 2c was considered at iterations $t_i - 1$ and $t_j - 1$. We thus have $\kappa(t_i - 1) = r-d+i$

and $\kappa(t_j-1) = r-d+j$. Thanks to the choice of κ at Step 2a, we have (recall that $\ell(t_i-1) = d-i$ and $\ell(t_j-1) = d-j$):

$$l'_{i,i} = l^{(t_i)}_{n-d+i,r-d+i} = \gamma^{-(r-d+i)} \cdot \max_{k \le r-d+i} \gamma^k \cdot l^{(t_i-1)}_{k,k},$$

$$l'_{j,j} = l^{(t_j)}_{n-d+i,r-d+i} = \gamma^{-(r-d+j)} \cdot \max_{k \le r-d+j} \gamma^k \cdot l^{(t_j-1)}_{k,k}.$$

Using Lemma 5.2 and the fact that $t_i < t_j$, we conclude that:

$$\begin{aligned} l'_{j,j} &\le \max_{k \le r-d+j} l^{(t_j-1)}_{k,k} \le \max_{k \le r-d+i} l^{(t_i-1)}_{k,k} \\ &\le \max_{k \le r-d+i} \gamma^k \cdot l^{(t_i-1)}_{k,k} = \gamma^{r-d+i} \cdot l'_{i,i}, \end{aligned}$$

which completes the proof. □

Integer relation algorithms may not have any a priori information on the set of solutions. As mentioned previously, under the exact real arithmetic model, it is impossible to decide whether there exists an integer relation for a given $\boldsymbol{x} \in \mathbb{R}^n$. Hence, as we have seen with HJLS-PSLQ, they have been designed for only ruling out the existence of small relations. Similarly, in our more general context, we can rule out the existence of some large invariants in the lattice component. If the target dimension is not known in advance, then `Decomp_HJLS` may not return a basis of the lattice component. However, if the input integer d is smaller than the dimension d' of the lattice component Λ, then `Decomp_HJLS` returns a d-dimensional lattice that is a projection of Λ orthogonally to a sublattice of Λ and one can prove that:

$$\lambda_{d'-d}(\Lambda) \le \sqrt{2r}\gamma^{2r} \cdot \max_{k \le r-d} l^{(\tau+1)}_{k,k}, \qquad (5.1)$$

where τ is the total number of iterations of `Decomp_HJLS` with integer input d.

5.3 Speed of convergence of `Decomp_HJLS`

We adapt the convergence analyses from [7] to `Decomp_HJLS`. We will use the following notations. For each iteration $t \ge 1$, we define

$$\pi^{(t)}_j := \begin{cases} l^{(t)}_{j,j} & \text{if } l^{(t)}_{j,j} \ne 0, \\ l^{(t)}_{n-r+j,j} & \text{if } l^{(t)}_{j,j} = 0 \end{cases}$$

and

$$\Pi(t) := \prod_{i=1}^{r-1} \prod_{j=1}^{i} \max\left(\pi^{(t)}_j,\, \gamma^{-r-1} \cdot \lambda_1(\Lambda)\right).$$

The following result allows us to quantify progress during the execution of the algorithm: at every loop iteration, the potential function $\Pi(t)$ decreases significantly.

Lemma 5.5. *Let $\beta = 1/\sqrt{1/\gamma^2 + 1/4} > 1$. Then for any loop iteration $t \in [1,\tau]$, we have $\Pi(t) \ge \beta \cdot \Pi(t+1)$. Further, we also have $\Pi(1) \le X^{\frac{r(r-1)}{2}}$ with $X = \max_{i \le n} \|\boldsymbol{a}_i\|$ and $\Pi(\tau+1) \ge \left(\gamma^{r+1}\right)^{-\frac{r(r-1)}{2}} \cdot \lambda_1(\Lambda)^{\frac{r(r-1)}{2}}$.*

Proof. The proof of the first claim is similar to the proofs of [7, Th. 3.2] and [6, Lem. 9]. We omit it here. The upper bound on $\Pi(1)$ follows from

$$\gamma^{-r-1} \cdot \lambda_1(\Lambda) \le \lambda_1(\Lambda) \le \max_{i \le r} l^{(1)}_{i,i} \le \max_i \|\boldsymbol{a}_i\| = X,$$

where the second inequality follows from Lemma 5.3 with $t = 1$. The last item follows from $\max(\pi^{(\tau+1)}_j,\, \gamma^{-r-1} \cdot \lambda_1(\Lambda)) \ge \gamma^{-r-1} \cdot \lambda_1(\Lambda)$. □

The following result directly follows from Lemma 5.5.

Theorem 5.6. *The number of loop iterations consumed by `Decomp_HJLS` is $\mathcal{O}(r^3 + r^2 \log \frac{X}{\lambda_1(\Lambda)})$. The number of arithmetic operations consumed at each loop iteration is $\mathcal{O}(nm^2)$.*

6. USING LLL TO SOLVE `DECOMP`

In this section, we provide elements of analysis for a folklore method to solve problems akin to `Decomp` using lattice reduction, such as LLL.

Directly calling LLL on the input fgas generating matrix does not work: it may launch an infinite loop and fail to disclose the lattice component. For instance, consider

$$A = \begin{pmatrix} 1 & 0 \\ \sqrt{2} & 0 \\ x & 2 \end{pmatrix},$$

where $x \in \mathbb{R}$ is arbitrary. LLL keeps swapping (and size-reducing) the first two rows forever, and fails to disclose the lattice component $\mathbb{Z} \cdot (0, 2)$. A crucial point here (see the discussion in [7, Sec. 1-2]) is the fact that the swap strategy is not global enough.

6.1 The `Decomp_LLL` algorithm

In `Decomp_LLL`, we lift the input fgas generating matrix $A \in \mathbb{R}^{n \times m}$ to a lattice basis $A_c := (c^{-1} I_n | A) \in \mathbb{R}^{n \times (m+n)}$, where $c > 0$ is a parameter. LLL will be called on A_c. This creates a unimodular matrix U such that $U \cdot A_c$ is LLL-reduced. The output matrix $U \cdot A_c$ is of the shape $(c^{-1}U | U \cdot A)$, and hence the right hand side $U \cdot A$ of the LLL output is a generating matrix for the input fgas. The goal is to set c sufficiently large so that in A_c there exists a gap between those vectors corresponding the lattice component and those vectors corresponding the vector space component. Ideally, the first vectors of $U \cdot A$ should be very small, because of the LLL-reduction of $U \cdot A_c$: for a large c, the matrix U can get quite large to decrease the right hand side of A_c. Oppositely, by linear independence, the vectors belonging to the lattice component of the input fgas cannot be shortened arbitrarily. They will always lead to large vectors in the lattice spanned by A_c, even for very large values of c. Overall, the key point in `Decomp_LLL` is the choice of the parameter c.

Algorithm 3 (Decomp_LLL).

Input: A generating matrix $A = (\boldsymbol{a}_1^T, \cdots, \boldsymbol{a}_n^T)^T \in \mathbb{R}^{n \times m}$ of an fgas $\mathcal{S}$; the dimension d of the lattice component Λ of $\mathcal{S}$; a parameter $c > 0$.

Output: Hopefully, a basis of Λ.

1. Define $A_c := (c^{-1} \cdot I_n | A)$.
2. Call LLL on input A_c; let A'_c denote the output basis.
3. Let $\pi_m(A'_c)$ denote the submatrix of A'_c consisting in the last m columns. Compute the LQ decomposition of $\pi_m(A'_c) = L \cdot Q$; define

$$L' := \left(\mathbf{0}_{d \times (r-d)} | (l_{i,j})_{i \in [n-d+1,n], j \in [r-d+1,r]}\right)$$

with $r = \operatorname{rank}(A)$; return $L' \cdot Q$.

Theorem 6.1. *Let $c > 0$ and Λ_c denote the lattice spanned by the A_c matrix of Step 1. If $2^{\frac{n-1}{2}} \cdot \lambda_{n-d}(\Lambda_c) < \lambda_1(\Lambda)$, then Algorithm 3 works correctly: it outputs a basis of the lattice component of the input fgas. Further, for any input A,* *there exists a threshold $c_0 > 0$ such that $2^{\frac{n-1}{2}} \cdot \lambda_{n-d}(\Lambda_c) < \lambda_1(\Lambda)$ holds for any $c > c_0$. Finally, Algorithm 3 consumes $\mathcal{O}(n^2 \log(cX))$ LLL swaps, where $X = \max_i \|\boldsymbol{a}_i\|$.*

PROOF. Since $\dim(\Lambda) = d$, there is a unimodular matrix U such that the L-factor of UA has shape $\mathrm{Trap}(d)$ (see Definition 5.1) and the first $n-d$ vectors of UA have norms $\leq 2^{-n}\lambda_1(\Lambda)$. For example, we can use the while loop in `Decomp_HJLS` to generate such a U that makes the M-part small enough (using the notation from Definition 5.1). Then, choosing $c > \max_{i \leq n-d}(2^n \|\boldsymbol{u}_i\|/\lambda_1(\Lambda))$ implies that $\lambda_{n-d}(\Lambda_c) < 2^{\frac{1-n}{2}} \cdot \lambda_1(\Lambda)$, where $\boldsymbol{u}_i$ is the i-th row of U.

Write $A'_c = (\boldsymbol{a}_1'^T, \cdots, \boldsymbol{a}_n'^T)^T$. Since the basis $\boldsymbol{a}'_1, \cdots, \boldsymbol{a}'_n$ is LLL-reduced, it follows that

$$\forall i \leq n-d: \quad \|\boldsymbol{a}'_i\| \leq 2^{(n-1)/2} \cdot \lambda_i(\Lambda_c) \leq 2^{(n-1)/2} \cdot \lambda_{n-d}(\Lambda_c).$$

Hence the condition on c implies that $\|\boldsymbol{a}'_i\| < \lambda_1(\Lambda)$ for $1 \leq i \leq n-d$. Let $\pi_m(\boldsymbol{a}'_i)$ denote the vector in $\mathbb{R}^m$ consisting in keeping only the last m components of $\boldsymbol{a}'_i$. Then for $i \leq n-d$, it follows that $\pi_m(\boldsymbol{a}'_i) \in \mathcal{S}$ and $\|\pi_m(\boldsymbol{a}'_i)\| < \lambda_1(\Lambda)$. Thus $\pi_m(\boldsymbol{a}'_i) \in E$, where E is the vector space component of $\overline{\mathcal{S}} = \Lambda \bigoplus E$. Since $\dim(\Lambda) = d$, it follows from Lemma 3.2 that $E = \mathrm{Span}_{i \leq n-d}(\pi_m(\boldsymbol{a}'_i))$, and that the output is exactly a basis of the lattice component Λ. Recall that in the classical LLL analysis for integral inputs, the number of iterations is at most $\mathcal{O}(n^2 \log K)$, where K is the maximum of the norms of the input vectors. For Algorithm 3, we can map the matrix A_c to $c \cdot A_c$, and then the new vectors have norms less than cX. $\square$

In practice, the parameter c may need to be arbitrary large. Consider the fgas generated by the rows of

$$A = \begin{pmatrix} 0 & 1 \\ 1/c_0 & 1 \\ 3 & 0 \end{pmatrix}$$

with c_0 a large irrational number. Its lattice component is $\mathbb{Z} \cdot (0, 1)$. If we choose $2 \leq c \leq c_0$ in Algorithm 3, then after LLL reduction, the first two rows of the submatrix UA of $(c^{-1}U|UA)$ will be $(1/c_0, 0)$ and $(0, 1)$. In this case, `Decomp_LLL` fails to disclose the lattice component, which means that we should choose $c > c_0$. Thus, when c_0 tends to infinity, the required parameter c will be arbitrary large, even for bounded input norms: `Decomp_LLL` may hide singularities when appending the scaled identity matrix.

7. OPEN PROBLEMS

We restricted ourselves to describing and analyzing algorithms with exact real arithmetic operations, and we did not focus on lowering the cost bounds. A natural research direction is to analyze the numerical behavior of these algorithms when using floating-point arithmetic and to bound their bit-complexities. It has been experimentally observed (see, e.g., [5]) that the underlying QR-factorisation algorithm and the choice of full size-reduction impact the numerical behavior. However, to the best of our knowledge, there is no theoretical study of those experimental observations, nor bit-complexity analysis.

An intriguing aspect of HJLS-PSLQ is that it solves (a variant of) `Intersect` via a reduction to `Decomp` and (partially) solving `Decomp`. Designing a more direct approach for `Intersect` is an interesting open problem.

Acknowledgments. We thank D. Dadush, G. Hanrot and G. Lecerf for helpful discussions. We also thank the reviewers for helpful comments, and for pointing out the result of Babai et al. [1] on the impossibility of deciding whether there exists an integer relation among real numbers. This work was partly supported by the ANR HPAC project, the CAS-CNRS Joint Doctoral Promotion Programme, NSFC (11001040, 11171053) and NKBRPC (2011CB302400). Part of this research was undertaken while the first author was visiting École Normale Supérieure de Lyon, whose hospitality is gratefully acknowledged.

8. REFERENCES

[1] L. Babai, B. Just, and F. Meyer auf der Heide. On the limits of computations with the floor function. *Inf. Comput.*, 78(2):99–107, 1988.

[2] P. Borwein. *Computational Excursions in Analysis and Number Theory*. Springer, New York, 2002.

[3] N. Bourbaki. *Elements of Mathematics: General Topology, Part II*. Addison-Wesley, Massachusetts, 1967. A translation of *Éléments de Mathématique : Topologie Générale*, Hermann, Paris, 1966.

[4] D. Dadush and O. Regev. Lattices, convexity and algorithms: Dual lattices and lattice membership problems, 2013. Notes of the second lecture of a course taught at New York University. Available from `http://cs.nyu.edu/~dadush/`.

[5] H. Ferguson and D. Bailey. A polynomial time, numerically stable integer relation algorithm. Technical Report RNR-91-032, SRC-TR-92-066, NAS Applied Research Branch, NASA Ames Research Center, July 1992.

[6] H. Ferguson, D. Bailey, and S. Arno. Analysis of PSLQ, an integer relation finding algorithm. *Math. Comput.*, 68(225):351–369, 1999.

[7] J. Håstad, B. Just, J. Lagarias, and C. Schnorr. Polynomial time algorithms for finding integer relations among real numbers. *SIAM J. Comput.*, 18(5):859–881, 1989. Preliminary version: Proceedings of STACS'86, pp. 105–118, 1986.

[8] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

[9] J. Martinet. *Perfect Lattices in Euclidean Spaces*. Springer, Berlin, 2003.

[10] A. Meichsner. Integer Relation Algorithms and the Recognition of Numerical Constants. Master's thesis, Simon Fraser University, 2001.

[11] A. Meichsner. *The Integer Chebyshev Problem: Computational Explorations*. PhD thesis, Simon Fraser University, 2009.

[12] M. Pohst. A modification of the LLL reduction algorithm. *J. Symb. Comput.*, 4(1):123–127, 1987.

[13] O. Regev. Lattices in Computer Science, 2004. Lecture notes of a course taught at Tel Aviv University. Available from `http://www.cims.nyu.edu/~regev/`.

[14] A. Schönhage. Factorization of univariate integer polynomials by Diophantine approximation and an improved basis reduction algorithm. In *Proceedings of ICALP*, volume 172 of *LNCS*, pages 436–447. Springer, 1984.