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

Jingwei Chen†‡, Damien Stehlé, Gilles Villard
†Chengdu Institute of Computer Application, CAS
‡CNRS, ENS de Lyon, Inria
UCBL, Université de Lyon
Laboratoire LIP
damien.stehle@gmail.com
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 \( m \in \mathbb{Z}^n \setminus \{0\} \) is called an integer relation for \( x \in \mathbb{R}^n \) if \( x \cdot m^T = 0 \). The HJLS algorithm [7, Sec. 3], proposed by Hästad, 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 \( x \in \mathbb{R}^n \), 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 \( E = \text{Span}(x)^\perp \subseteq \mathbb{R}^n \). It proceeds as follows. (1) It first projects the rows of the identity matrix (which forms a basis of \( \Lambda \)) 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 \( n \) 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 \( x \). 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 \textbf{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 \( \mathcal{S} \) of any fgas \( S \subseteq \mathbb{R}^m \) is the orthogonal sum of a unique lattice component \( \Lambda \) and a unique vector subspace component \( E \), i.e., \( \mathcal{S} = \Lambda \oplus E \). The \textbf{Decomp} problem takes as input an fgas \( S \) described by a generating set and returns bases of \( \Lambda \) and \( E \). We exhibit a duality relationship between the \textbf{Intersect} and \textbf{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 \textbf{Decomp}_{HJLS}, for decomposing fgas’s. Prior to this work, only special cases...
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 $O(r^3 + r^2 \log \frac{r}{N(A)})$, where $r$ is the rank of the input fgas, $X$ is an upper bound on the euclidean norms of the input generators and $\lambda_1(A)$ is the minimum of the lattice component $A$. For an fgas $S \subseteq \mathbb{R}^m$ with $n$ generators, an iteration consumes $O(nm^2)$ 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. 252]. 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 $(e_{-1}.I_n|A)$, 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 $b$ is a vector, then $|b|$ denotes its euclidan norm. We let $(b, c)$ denote the usual inner product between two real vectors $b$ and $c$ sharing the same dimension. If $b \in \mathbb{R}^n$ is a vector and $E \subseteq \mathbb{R}^n$ is a vector space, we let $\pi(b, E)$ denote the orthogonal projection of $b$ 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 $Q \in \mathbb{R}^{n \times m}$ has orthonormal rows (i.e., $Q^T \cdot I_r$), and the L-factor $L \in \mathbb{R}^{r \times r}$ satisfies the following property: there exist diagonal indices $1 \leq k_1 < \ldots < k_r \leq n$, such that $l_{i,j} = 0$ for all $i < k_j$, and $l_{k_j,j} > 0$ for all $j \leq r$ (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 $L = (l_{i,j}) \in \mathbb{R}^{r \times r}$ be a lower trapezoidal matrix with rank $r$ and diagonal indices $k_1 < \ldots < k_r$. We say $L$ is size-reduced if $|l_{i,j}| \leq \frac{1}{2} |l_{k_j,j}|$ holds for $i > k_j$.

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

Lattices. A euclidean lattice $A \subseteq \mathbb{R}^n$ is a discrete (additive) subgroup of $\mathbb{R}^n$. A basis of $A$ consists of $n$ linearly independent vectors $b_1, \ldots, b_n \in \mathbb{R}^n$ such that $A = \sum Zb_i$. We say that $B = (b'_1, \ldots, b'_n)^T \in \mathbb{R}^{n \times m}$ is a basis matrix of $A$. The integer $n$ is called the dimension of $A$. If $n \geq 2$, then $A$ has infinitely many bases, that exactly consist in the rows of $U \cdot B$ where $B$ is an arbitrary basis matrix of $A$ and $U$ ranges over $GL_n(\mathbb{Z})$. The $i$-th successive minimum $\lambda_i(A)$ for $i \leq n$ is defined as the radius of the smallest ball that contains $i$ linearly independent vectors of $A$. The dual lattice $A^*$ of $A$ is defined as $A^* = \{(x \in \mathbb{R}^n) : (b, x) \in A, (b, x) \in \mathbb{Z}\}$. If $B$ is a basis matrix of $A$, then $(BB^T)^{-1}B$ is a basis of $A$, 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 $B = (b'_1, \ldots, b'_n)^T \in \mathbb{R}^{n \times m}$ be the basis matrix of a lattice $A$, and $L = (l_{i,j})$ its L-factor. We say the basis $b_1, \ldots, b_n$ is weakly-reduced with parameters $\gamma > 2/\sqrt{3}$ and $C \geq 1$ if $L$ is size-reduced and satisfies the (generalized) Schönhage condition $l_{i,j} \leq C \cdot \gamma^{i-j} l_{i,i}$ for $1 \leq j \leq i \leq n$. Note that a LLL-reduced basis is always weakly-reduced, with $C = 1$. If a lattice basis is weakly-reduced, then

$$\|b_i\| \leq \sqrt{n}C \gamma^{i-1} l_{i,i},$$

$$(\sqrt{n}C^2 \gamma^{2i})^{-1} \cdot \lambda_i(A) \leq \|b_i\| \leq \sqrt{n}C^2 \gamma^{2i} \cdot \lambda_i(A).$$

HJLS-PSLQ. We recall HJLS [7, Sec. 3] using the PSLQ setting [6]. We call the resulting algorithm HJLS-PSLQ (Algorithm 1). Given $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, HJLS-PSLQ either returns an integer relation for $x$, or gives a lower bound on $\lambda_1(A_x)$, where $A_x$ is the lattice of all integer relations for $x$. The updates of $U$ and $Q$ at Steps 1b, 2a, 2b and 2c are implemented so as to maintain the relationship $U \cdot L = LQ$ 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: $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ with $x_i \neq 0$ for $i \leq n$, $M > 0$ and $\gamma > 2/\sqrt{3}$.
Output: Either return an integer relation for $x$, or claim that $\lambda_1(A_x) > M$.
1. (a) Normalize $x$, i.e., set $x := x/\|x\|$, set $U := I_n$ and $Q = I_{n-1}$.
(b) Compute the Q-factor $(x^T | L_x)^T$ of $(x|I_n)^T$; set $L := L_x$; size-reduce $L$ and update $U$.
2. While $l_{n-1,n-1} \neq 0$ and max $l_{i,i} \geq 1/M$ do
(a) Choose $k$ such that $\gamma^{k-1} l_{k,k} = \max_{j \leq n} \gamma^{j-1} l_{j,j}$; 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 $\lambda_1(A_x) > M$. Else return the last column of $U^{-1}$.

For the proof of termination, it suffices to enforce the partial size-reduction condition $|l_{k+1,k}| \leq \frac{1}{2} l_{k,k}$ 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 \((\mathbf{x} \cdot U^{-1})\), for some \(j < n\), 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 \([10, \text{Sec. 2.3.1}]\) (see also \([2, \text{App. B, Th. 7}]\)). Full size-reduction is essentially irrelevant in the exact real number model. Indeed, HJLS works correctly and consumes \(O(n^3 + n^2 \lambda_i(A_j))\) 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 \(\gamma\) 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 \((\text{Decomp})\) and the problem of computing the intersection of a lattice and a vector subspace \((\text{Intersect})\), assuming exact computations over the reals.

3.1 FGAS of a euclidean space

Given \(a_1, \ldots, a_n \in \mathbb{R}^m\), the \textit{finitely generated additive subgroup} (fgas for short) spanned by the \(a_i\)'s is the set of all integral linear combinations of the \(a_i\)'s:

\[
S = \sum_{i=1}^{n} z_i a_i = \left\{ \sum_{i=1}^{n} z_i a_i : z_i \in \mathbb{Z} \right\} \subset \mathbb{R}^m. \tag{3.1}
\]

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

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

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 \textit{finite sum of lattices}.

Fgas's can also be viewed as \textit{orthogonal projections of lattices onto vector subspaces}. Let \(A = \sum_{i=1}^{n} \mathbb{Z} b_i \subset \mathbb{R}^m\) be a lattice and \(E \subset \mathbb{R}^m\) be a vector subspace. The \textit{orthogonal projection} of \(A\) onto \(E\), i.e., the set \(\pi(A, E) = \{ v_1 \in E : \exists v_2 \in E^\perp, v_1 + v_2 \in A \}\), is an fgas of \(\mathbb{R}^m\); it is spanned by the projections of the \(b_i\)'s. Conversely, given an fgas \(\mathcal{S}\) with generating matrix \(A \in \mathbb{R}^{n \times m}\), let \(\mathcal{A} \subset \mathbb{R}^{n \times m}\) be the lattice generated by the rows of \((I_n | A)\) and \(E = \text{Span}(I_n | 0)^\perp \subset \mathbb{R}^{n \times m}\). Then \(\mathcal{S} = \pi(\mathcal{A}, E)\).

3.2 The Decomp and Intersect problems

Consider the topological closure \(\overline{\mathcal{S}}\) of an fgas \(\mathcal{S} \subset \mathbb{R}^m\), 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 \([9, \text{Th. 1.1.2}]\) (see also \([3, \text{Chap. VII, Th. 2}]\)), there exists a unique lattice \(\Lambda \subset \mathbb{R}^m\) and a unique vector subspace \(E \subset \mathbb{R}^m\) such that their direct sum is \(\overline{\mathcal{S}}\), and the vector space \(\text{Span}(\Lambda)\) spanned by \(\Lambda\) is orthogonal to \(E\). We denote the latter decomposition by \(\overline{\mathcal{S}} = \Lambda \oplus E\). More explicitly, if \(\text{rank} \mathcal{S} = \text{dim} (\text{Span}(\mathcal{S})) = r \leq m\), then there exist \(0 \leq d \leq r\), \((b_i)_{i \leq d}\), and \((e_i)_{i \leq r-d}\) in \(\mathbb{R}^m\) such that:

- the \(r\) vectors \(b_i\) \((i \leq d)\) and \(e_i\) \((i > r-d)\) are linearly independent;
- for any \(i \leq d\) and \(j > r-d\), we have \((b_i, e_j) = 0\).

Then \((b_i)_{i \leq d}\) and \((e_i)_{i \geq r-d}\) are bases of \(\Lambda\) and \(E\), respectively. We call \(\Lambda\) and \(E\) the \textit{lattice} and \textit{vector space components} of \(\mathcal{S}\), respectively; and define the \(\Lambda E\) decomposition of \(\mathcal{S}\) as \((\Lambda, E)\). The \textit{Decomp} problem is the associated computational task.

\begin{definition}
3.1. The \textit{Decomp} problem is as follows: Given as input a finite generating set of an fgas \(\mathcal{S}\), the goal is to compute its \(\Lambda E\) decomposition, i.e., find bases for the lattice and vector space components \(\Lambda\) and \(E\).
\end{definition}

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

\begin{lemma}
3.2. Let \((a_i)_{i \leq n}\) be a generating set of an fgas \(\mathcal{S}\) with \(\Lambda E\) decomposition \(\mathcal{S} = \Lambda \oplus E\). Define \(a'_i\) as the projection of \(a_i\) orthogonally to \(\text{Span}(a_j)_{j \leq n-k}\), for \(n-k+1 \leq j \leq n\) and assume the \(a'_i\)'s are linearly independent. Then \(a'_{n-k+1}, \ldots, a'_{n}\) form a basis of a projection of \(\Lambda\) and \(E \subseteq \text{Span}(a'_{n-k+1}, \ldots, a'_{n})\). Further, if \(k = \text{dim} \Lambda\), then \(\Lambda = \sum_{n-k+1 \leq i \leq n} \mathbb{Z} a'_i\) and \(E = \text{Span}(a'_{n-k})\).
\end{lemma}

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

We now introduce another problem, \textit{Intersect}, which generalizes the integer relation finding problem.

\begin{definition}
3.3. The \textit{Intersect} problem is as follows: Given as inputs a basis of a lattice \(\Lambda\) and a basis of a vector subspace \(E\), the goal is to find a basis of the lattice \(\Lambda \cap E\).
\end{definition}

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

3.3 Relationship between the problems

The \textit{Decomp} and \textit{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 $\widehat{S}$ of an fgas $S$ is defined as $\widehat{S} = \{x \in \text{Span}(S) : \forall b \in S, \langle x, b \rangle \in \mathbb{Z}\}$.

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

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

Lemma 3.5. Let $S$ be an fgas and $A$ its lattice component. Then $\widehat{A} = \widehat{S}$.

Proof. Let $\widehat{S} = A \bigoplus E$ be the LAE decomposition of $S$. Recall that for any $x \in \mathbb{S}$, there exist unique $x_A \in A$ and $x_E \in E$ such that $x = x_A + x_E$ and $\langle x_A, x_E \rangle = 0$.

We first prove that $\widehat{A} \subseteq \widehat{S}$. For all $\hat{x} \in \widehat{A}$ and all $x \in \mathbb{S}$, we have $\langle \hat{x}, x \rangle = \langle \hat{x}, x_A \rangle + \langle \hat{x}, x_E \rangle = \langle \hat{x}, x_A \rangle \in \mathbb{Z}$, where the second equality follows from the orthogonality between the vector subspaces $E$ and $\text{Span}(A)$, and $\langle \hat{x}, x_A \rangle \in \mathbb{Z}$ derives from the definition of $\widehat{A}$.

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

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

Lemma 3.6. Let $S$ be an fgas and $A$ its lattice component. Then $\Lambda = \widehat{S}$.

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

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

Proof. Let $b \in \Lambda \cap E$ and $y \in \pi(\Lambda, E)$. There exist $b \in \Lambda$ and $y' \in E$ such that $b = y + y'$. Then $\langle b, y \rangle = \langle b, \hat{b} \rangle - \langle b, y' \rangle = \langle b, \hat{b} \rangle \in \mathbb{Z}$.

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

Now, let $b \in \pi(\Lambda, E)$. By definition, we have $b \in \text{Span}(\pi(\Lambda, E)) \subseteq E$.

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

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

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

Reducing $\text{Decomp}$ to $\text{Intersect}$. Suppose we are given a generating set $a_1, \cdots, a_n \in \mathbb{R}^m$ of an fgas $S$. Our goal is to find the LAE decomposition of $S$, using an oracle that solves $\text{Intersect}$. It suffices to find a basis of the lattice component $\Lambda$, which, by Lemma 3.6, satisfies $\Lambda = \widehat{S}$.

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

Reducing $\text{Intersect}$ to $\text{Decomp}$. Assume we are given a basis $(b_i)$ of a lattice $\Lambda' \subseteq \mathbb{R}^m$ and a basis $(e_i)$, 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 $\text{Decomp}$.

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

4. A NEW VIEW ON HJLSPSLQ

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

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

$$(x^T | I_n) = \begin{pmatrix} 1 & x^T L_x \\ I_n & L_x \end{pmatrix}^T.$$

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 $(x^T | I_n)^T$. It is worth noting 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 $\pi_x = I_n - x^T x = L_x L_x^T$ corresponds to the orthogonal projection that maps $\mathbb{R}^n$ to $\text{Span}(x)^\perp$. Therefore the rows of $L_x$ are the coordinate vectors of the rows of $\pi_x$ with respect to the normalized orthogonal basis of $\text{Span}(x)^\perp$ given by the $n - 1$ $-$ $1$ rows of $L_x^T$. Overall, we obtain that $(0 | L_x) \cdot Q_x$ is a generating matrix of the fgas $S_x = \pi(\mathbb{Z}^n, \text{Span}(x)^\perp)$.

Step 2 revisited: A partial solution to $\text{Decomp}$. Since $(0 | L_x) \cdot Q_x$ is a generating matrix of the fgas $S_x$, the whole loop of HJLSPSLQ only considers this fgas (in fact, HJLSPSLQ 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 $\text{Decomp}$ problem. By Lemma 3.7,
finding a basis of the lattice component of $S_x$ suffices to find all integer relations of $x$: 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 $A'$ of $S_x$. 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 $\text{Decomp}$ on input $S_x$. It gets the full solution only when $\dim(\mathbb{Z}^n \cap \text{Span}(x)^\perp) = \dim(A') = 1$.

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 $A'$, orthogonally to the first vectors of that basis of $A'$. This vector is:

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

Its dual, when considered as a basis, is

$$\hat{b} = b/\|b\|^2 = (0|l_{n-1,n-1}) \cdot \text{diag}(1, Q) \cdot Q_x.$$

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

$$\hat{b} = (0|l_{n-1,n-1}) \cdot \left( \frac{1}{Q} \right) \cdot \left( \frac{x}{L_x} \right) = (0|l_{n-1,n-1}) \cdot \left( \frac{x}{QL_x} \right).$$

Now, as $UL_x = LQ$, we obtain that $\hat{b} = (0|l_{n-1,n-1}) \cdot (\mathbb{x}^T)(U^{-1}L)^T = (0|1)U^{-T}$. 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 $x = \left( \frac{1}{\sqrt{2}}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right)$. At the beginning, we have

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

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

$$\Lambda = \begin{pmatrix}
15 & -10\sqrt{2} & 0 \\
-2(41 + 29\sqrt{2}) & 35 & 0 \\
41\sqrt{2} & 58 & 1
\end{pmatrix}.$$

$$U = \begin{pmatrix}
-2 & -3 & -4 \\
5 & 7 & 10 \\
-1 & -2 & -3
\end{pmatrix}, \quad Q = \begin{pmatrix}
\frac{\sqrt{2}}{35} & -\frac{\sqrt{2}}{35} & -\frac{\sqrt{2}}{35} \\
\frac{\sqrt{2}}{\sqrt{35}} & \frac{\sqrt{2}}{\sqrt{35}} & \frac{\sqrt{2}}{\sqrt{35}} \\
\frac{\sqrt{2}}{\sqrt{35}} & \frac{\sqrt{2}}{\sqrt{35}} & \frac{\sqrt{2}}{\sqrt{35}}
\end{pmatrix}.$$

Thanks to the shape of $L$, the AE decomposition $\overline{\Lambda} = \Lambda \cap \mathbb{Z}$ can be derived from $(0|L)$. In this precise case, HJLS-PSLQ discloses the full lattice component. Thanks to Lemma 3.7, we have $\Lambda = \hat{\Lambda}_x$, and hence $\dim(\Lambda) = \dim(\hat{\Lambda}_x) = \dim(A_x) = 1$ (as $x$ contains two rational entries and one irrational entry). Using the matrix factorisation above, we obtain

$$A = \mathbb{Z} \cdot \left( 0, 0, 0 / \sqrt{5} \right) \cdot \text{diag}(1, Q) \cdot Q_x = \mathbb{Z} \cdot (2/5, 0, -1/5)$$

and

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

5. SOLVING $\text{Decomp} \text{ à LA HJLS}$

Let $A \in \mathbb{R}^{n \times n}$ be a generating matrix of an fgs $S$ and $\mathcal{S} = \Lambda \cap \mathbb{Z}$ be the AE decomposition of $S$ with $\dim(\Lambda) = d$. In this section, we present and analyze an algorithm, named $\text{Decomp}_{\text{HJLS}}$, for solving the $\text{Decomp}$ problem.

Note that $\text{Decomp}_{\text{HJLS}}$ 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 $x \in \mathbb{R}^n$. Computing the dimension of the lattice component of an fgs would allow us to solve that decision problem.

5.1 The $\text{Decomp}_{\text{HJLS}}$ algorithm

$\text{Decomp}_{\text{HJLS}}$, given as Algorithm 2, is a full fgs 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 $k'$ we select, hereafter at Step 2c of Algorithm 2, may differ from $k + 1$.) 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 $0 \leq \ell \leq r$. If a lower trapezoidal matrix $L \in \mathbb{R}^{n \times \ell}$ can be written as

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

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

$\text{Decomp}_{\text{HJLS}}$ takes as input an fgs 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 $\text{Decomp}_{\text{HJLS}}$ is to apply unimodular transformations (namely, size-reductions and swaps) to a current generating matrix $L \cdot Q$ of the input fgs, 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).
Algorithm 2 (Decomp_HJLS).

Input: A generating matrix $A = (a_1, \ldots, a_r)^T \in \mathbb{R}^{n \times m}$ of an fgas $S$ with $\max_{i \leq m} \|a_i\|_2 \leq X$; a positive integer $d$ as the dimension of the lattice component $A$ of $S$; a parameter $\gamma > 2/\sqrt{3}$.

Output: A basis matrix of $A$.

1. (a) Compute $r = \text{rank}(A)$. If $d = r$, then return $a_1, \ldots, a_r$. Else, using row pivoting, ensure that the first $r$ rows of $A$ are linearly independent.
2. While $l_{r-d+1, r-\ell+1} \neq 0$ do
   (a) Choose $\kappa$ such that $\gamma^{\kappa} \cdot (l_{r, n}) = \max_{\kappa \leq l \leq r} \gamma^\kappa \cdot (l_{r, n})$.
   (b) If $\kappa < r - \ell$, then swap the $\kappa$-th and the $(\kappa + 1)$-th rows of $L$; compute the LQ decomposition of $L$; replace $L$ by its L-factor and update $Q$.
3. Return $(0_{d \times (r-\ell)})(l_{i,j})_{i \in [n-d+1, n], j \in [r-d+1, r]} \cdot Q$.

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 $A$ be the lattice component of the input fgas, and $d = \dim(A) \geq 1$. Then, for any $t \in [1, \tau]$, we have
$$\lambda_1(A) \leq \max_{i \leq r-t} \gamma^t \cdot l_{(i,t)}.$$ 

Proof. The matrix $L^{(r+1)}$ has shape Trap($d$), and
$$\left(0^{r-d}, \gamma^t \cdot l_{(r-d+1, r+d)}, 0^{d-1}\right) \cdot Q^{(r+1)}$$
belongs to $A$ (by Lemma 3.2). As the matrix $Q^{(r+1)}$ is orthogonal, it has norm $I^{(r+1)}_{n-d+1, r-d+1}$. We thus have $\lambda_1(A) \leq I^{(r+1)}_{n-d+1, r-d+1}$. Now, as $\tau$ 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:
$$\lambda_1(A) \leq \max_{i \leq r-d} \gamma^t \cdot l_{(i,t)} \leq \max_{i \leq r-t} \gamma^t \cdot l_{(i,t)}.$$ 

The last inequality follows by 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 $S$ as input and the dimension $d$ of its lattice component, it returns a weakly-reduced basis, with parameters $\gamma$ and $C = \gamma^{r-d}$, of the lattice component of $S$.

Proof. At the end of the while loop in Decomp_HJLS, the L-factor $L^{(r+1)}$ has shape Trap($d$), where $d = \dim(A)$. As we only apply unimodular operations to the row vectors, the fgas $\mathbb{Z}^n \cdot L^{(r+1)} \cdot Q^{(r+1)}$ matches the input fgas $\mathbb{Z}^n \cdot A$. Let $A' = \mathbb{Z}^d \cdot \left(0_{d \times (r-\ell)}(l_{(i,j)})_{i \in [n-d+1, n], j \in [r-d+1, r]} \cdot Q^{(r+1)}\right)$ denote the output of Decomp_HJLS. By Lemma 3.2, the lattice $A'$ is exactly the lattice component $A$. Let $L' \in \mathbb{R}^{d \times d}$ be the matrix corresponding to the bottom right $d \times d$ columns of $L$. We now check that $L'$ is size-reduced and satisfies the Schönage conditions. Thanks to the size-reductions of Steps 1c and 2d, the whole matrix $L^{(r+1)}$ is size-reduced. It remains to show that $l'_{j,j} \leq \gamma^{r-d+i}$, $l'_{j,i}$ for all $1 \leq j < i \leq d$. 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:
$$l'_{i,i} = l^{(r+1)}_{n-d+i, r-d+i} = l^{(t_i)}_{n-d+i, r-d+i},$$
$$l'_{j,j} = l^{(r+1)}_{n,d+1, r+d+1} = l^{(t_j)}_{n,d+1, r+d+1}.$$ 

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 $\kappa(t_i - 1) = r - d + i$.
1. The last item follows from max(we also have γ x decide whether there exists an integer relation for a given under the exact real arithmetic model, it is impossible to formation on the set of solutions. As mentioned previously, Π (1) follows from proofs of [7, Th. 3.2] and [6, Lem. 9]. We omit it here.

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 ≥ 1, we define

\[ \pi_j^{(t)} := \begin{cases} \ell_j^{(t)} & \text{if } \ell_j^{(t)} ≠ 0, \\ \ell_{n-r+j,j}^{(t)} & \text{if } \ell_j^{(t)} = 0 \end{cases} \]

and

\[ \Pi(t) := \prod_{i=1}^{r-1} \prod_{j=1}^{n} \max \left( \pi_j^{(t)}, \gamma^{r-1} \cdot \lambda_i(A) \right). \]

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

**Lemma 5.5.** Let β = 1/\( \sqrt{1/\gamma^2 + 1/4} \) > 1. Then for any loop iteration t ∈ [1, τ], we have Π(t) ≥ β \( \Pi(t+1) \). Further, we also have Π(1) ≤ X \( \gamma^{r-1} \) with X = max ≤ |a_i| and Π(r + 1) ≥ \( \gamma^{r+1} \) \( \gamma^{r-1} \) \( \lambda_1(A) \). If τ > 1, then \( \Pi(t) \) decreases significantly.

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 Π(1) follows from

\[ \gamma^{r-1} \cdot \lambda_1(A) ≤ \lambda_1(A) ≤ \max_{s \leq r} \ell_i^{(t)} ≤ \max_i |a_i| = X, \]

where the second inequality follows from Lemma 5.3 with t = 1. The last item follows from max(\( \gamma^{r+1} \), \( \gamma^{r-1} \cdot \lambda_1(A) \) ≥ \( \gamma^{r-1} \cdot \lambda_1(A) \).
there exists a threshold $c_0 > 0$ such that $2 \frac{\log n}{n} \cdot \lambda_{n-d}(A_c) < \lambda_{\ell}(A)$ holds for any $c > c_0$. Finally, Algorithm 3 consumes $O(n^2 \log(cX))$ LLL swaps, where $X = \max \|a_i\|$. 

Proof. Since $\dim(A) = d$, there is a unimodular matrix $U$ such that the L-factor of $UA$ has shape $\text{Trap}(d)$ (see Definition 5.1) and the first $n - d$ vectors of $UA$ have norms $\leq 2 \frac{\log n}{n} \cdot \lambda_{\ell}(A)$. For example, we can use the while loop in Decom_LHJS to generate such a $U$ that makes the M-part small enough (using the notation from Definition 5.1). Then, choosing $c > \max_{i \leq n-d} (\frac{\log n}{n} \cdot \|a_i\|) / \lambda_{\ell}(A)$ implies that $\lambda_{n-d}(A_c) < 2 \frac{\log n}{n} \cdot \lambda_{\ell}(A)$, where $u_i$ is the $i$-th row of $U$.

Write $A_c = (a_1^c, \ldots, a_n^c)^\top$. Since the basis $a_1^c, \ldots, a_n^c$ is LLL-reduced, it follows that

$$\forall i \leq n - d : \|a_i^c\| \leq 2^{(n-1)/2} \cdot \lambda_{\ell}(A_c) \leq 2^{(n-1)/2} \cdot \lambda_{n-d}(A_c).$$

Hence the condition on $c$ implies that $\|a_i^c\| < \lambda_{\ell}(A)$ for $1 \leq i \leq n - d$. Let $\pi_m(a_i^c)$ denote the vector in $\mathbb{R}^m$ consisting in keeping only the last $m$ components of $a_i^c$. Then for $i > n - d$, it follows that $\pi_m(a_i^c) \in S$ and $\|\pi_m(a_i^c)\| < \lambda_{\ell}(A)$. Thus $\pi_m(a_i^c)$ is in $E$, where $E$ is the vector space component of $\mathbb{R}^n \setminus \Lambda \oplus E$. Since $\dim(A) = d$, it follows from Lemma 3.2 that $E = \operatorname{Span}_{c < n-d}(\pi_m(a_i^c))$, and that the output is exactly a basis of the lattice component $\Lambda$. Recall that in the classical LLL analysis for integral inputs, the number of iterations is at most $O(n^2 \log K)$, 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 $cX$. \qed

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

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

with $c_0$ a large irrational number. Its lattice component is $\mathbb{Z} \cdot (0,1,0)$. 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 $(c^{-1}UUA)$ will be $(1/c_0, 0)$ and $(0,1)$. In this case, Decom_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: Decom_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 these experimental observations, nor bit-complexity analysis.

An intriguing aspect of HJLS-PSLQ is that it solves (a variant of) Intersect via a reduction to Decom and (partially) solving Decom. 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, NNSF (11001040, 11171053) and NKBRC (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