# An LLL Algorithm for Module Lattices

Changmin Lee[1], Alice Pellet-Mary[1], Damien Stehlé[1], and Alexandre Wallet[2]

[1] Univ. Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342 Lyon Cedex 07, France
[2] NTT Secure Platform Laboratories, Tokyo, Japan

**Abstract.** The LLL algorithm takes as input a basis of a Euclidean lattice, and, within a polynomial number of operations, it outputs another basis of the same lattice but consisting of rather short vectors. We provide a generalization to $R$-modules contained in $K^n$ for arbitrary number fields $K$ and dimension $n$, with $R$ denoting the ring of integers of $K$. Concretely, we introduce an algorithm that efficiently finds short vectors in rank-$n$ modules when given access to an oracle that finds short vectors in rank-2 modules, and an algorithm that efficiently finds short vectors in rank-2 modules given access to a Closest Vector Problem oracle for a lattice that depends only on $K$. The second algorithm relies on quantum computations and its analysis is heuristic.

## 1 Introduction

The NTRU [HPS98], RingSIS [LM06, PR06], RingLWE [SSTX09, LPR10], ModuleSIS and ModuleLWE [BGV14, LS15] problems and their variants serve as security foundations of numerous cryptographic protocols. Their main advantages are their presumed quantum hardness, their flexibility for realizing advanced cryptographic functionalities, and their efficiency compared to their SIS and LWE counterparts [Ajt96, Reg09]. As an illustration of their popularity for cryptographic design, we note that 11 out of the 26 candidates at Round 2 of the NIST standardization process for post-quantum cryptography rely on these problems or variants thereof.[3] From a hardness perspective, these problems are best viewed as standard problems on Euclidean lattices, restricted to random lattices corresponding to modules over the rings of integers of number fields. Further, for some parametrizations, there exist reductions from and to standard worst-case problems for such module lattices [LS15, AD17, RSW18].

Let $K$ be a number field and $R$ its ring of integers. In this introduction, we will use the power-of-2 cyclotomic fields $K = \mathbb{Q}[x]/(x^d + 1)$ and their rings of integers $R = \mathbb{Z}[x]/(x^d + 1)$ as a running example (with $d$ a power of 2). An $R$-module $M \subset K^n$ is a finitely generated subset of vectors in $K^n$ that is stable under addition and multiplication by elements of $R$. As an example, if we consider $h \in R/qR$ for some integer $q$, the set $\{(f, g)^T \in R^2 : fh = g \bmod q\}$ is a module. If $h$ is an NTRU public key, the corresponding secret key is a vector in that module, and its coefficients are small. Note that for $K = \mathbb{Q}$ and $R = \mathbb{Z}$,

[3] See https://csrc.nist.gov/projects/post-quantum-cryptography

we recover Euclidean lattices in $\mathbb{Q}^n$. A first difficulty for handling modules compared to lattices is that $R$ may not be a Euclidean domain, and, as a result, a module $M$ may not be of the form $M = \sum_i R\mathbf{b}_i$ for some linearly independent $\mathbf{b}_i$'s in $M$. However, as $R$ is a Dedekind domain, for every module $M$, there exist $K$-linearly independent $\mathbf{b}_i$'s and fractional ideals $I_i$ such that $M = \sum I_i\mathbf{b}_i$ (see, e.g., [O'M63, Th. 81:3]). The set $((I_i, \mathbf{b}_i))_i$ is called a pseudo-basis of $M$. A module in $K^n$ can always be viewed as a lattice in $\mathbb{C}^{nd}$ by mapping elements of $K$ to $\mathbb{C}^d$ via the canonical embedding map (for our running example, it is equivalent to mapping a polynomial of degree $< d$ to the vector of its coefficients).

Standard lattice problems, such as finding a full-rank set of linearly independent short vectors in a given lattice, are presumed difficult, even in the context of quantum computations. In order to assess the security of cryptographic schemes based on NTRU/RingSIS/etc, an essential question is whether the restriction to module lattices brings vulnerabilities. Putting aside small polynomial speed-ups relying on the field automorphisms (multiplication by $x$ in our running example), the cryptanalytic state of the art is to view the modules as arbitrary lattices, i.e., forgetting the module structure.

LLL [LLL82] is the central algorithm to manipulate lattice bases. It takes as input a basis of a given lattice, progressively updates it, and eventually outputs another basis of the same lattice that is made of relatively short vectors. Its run-time is polynomial in the input bit-length. For cryptanalysis, one typically relies on BKZ [SE94] which extends this principle to find shorter vectors at a higher cost. Finding an analogue of LLL for module lattices has been an elusive goal for at least two decades, a difficulty being to even define what that would be. Informally, it should:

- work at the field level (in particular, it should not forget the module structure and view the module just as a lattice);
- it should find relatively short module pseudo-bases by progressively updating the input pseudo-basis;
- it should run in polynomial-time with respect to the module rank $n$ and the bit-lengths of the norms of the input vectors and ideals.

The state of the art is far from these goals. Napias [Nap96] proposed such an algorithm for fields whose rings of integers are norm-Euclidean, i.e., Euclidean for the algebraic norm. In our running example, this restricts the applicability to $d \leq 4$ (see [Cer05,Lez14] for other families of fields). Fieker and Pohst [FP96] proposed a general-purpose algorithm. However, it was not proved to provide pseudo-bases consisting of short module vectors, and a cost analysis was provided only for free modules over totally real fields. Fieker [Fie97, p. 47] suggested to use rank-2 module reduction to achieve rank-$n$ module reduction, but there was no follow-up on this approach. Gan, Ling and Mow [GLM09] described and analyzed an LLL algorithm for Gauss integers (i.e., our running example instantiated to $d = 2$). Fieker and Stehlé [FS10] proposed to apply the LLL algorithm on the lattice corresponding to the module to find short vectors in polynomial time and reconstruct a short pseudo-basis afterwards. More recently, Kim and Lee [KL17] described such an LLL algorithm for biquadratic fields whose

rings of integers are norm-Euclidean, and provided analyses for the shortness of the output and the run-time. They also proposed an extension to arbitrary norm-Euclidean rings, still with a run-time analysis but only conjecturing and experimentally supporting the output quality.

The rank-2 restriction already captures a fundamental obstacle. The LLL algorithm for 2-dimensional lattices (which is essentially Gauss' algorithm) is a succession of divide-and-swap steps. Given two vectors $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{Q}^2$, the 'division' consists in shortening $\mathbf{b}_2$ by an integer multiple of $\mathbf{b}_1$. This integer $k$ is the quotient of the Euclidean division of $\langle \mathbf{b}_1, \mathbf{b}_2 \rangle$ by $\|\mathbf{b}_1\|^2$. This leads to a vector $\mathbf{b}_2'$. If the latter is shorter than $\mathbf{b}_1$, then $\mathbf{b}_1$ and $\mathbf{b}_2$ are swapped and a new iteration starts. Crucial to this procedure is the fact that if the projection of $\mathbf{b}_2$ orthogonally to $\mathbf{b}_1$ is very small compared to $\|\mathbf{b}_1\|$, then the division will provide a vector $\mathbf{b}_2'$ that is shorter than $\mathbf{b}_1$. When a swap cannot be made, it means that the projection of $\mathbf{b}_2$ orthogonally to $\mathbf{b}_1$ is not too small, and hence the basis is of good quality, i.e., somewhat orthogonal and hence made of somewhat short vectors. What provides the convergence to a short basis is the Euclideanity of $\mathbb{Z}$. This is why prior works focused on this setup. Put differently, the crucial property is the fact that the covering radius of the $\mathbb{Z}$ lattice is smaller than 1: this makes it possible to shorten a vector $\mathbf{b}_2$ whose projection is sufficiently small by an appropriate integer multiple such that $\mathbf{b}_2'$ becomes smaller than $\mathbf{b}_1$. When we extend to modules, the corresponding lattice becomes $R$, and its covering radius has no a priori reason to be smaller than 1 (for our running example, it is $\sqrt{d}/2$). Even if we allow an infinite amount of time to find an optimal $k \in R$, the resulting $\mathbf{b}_2 - k\mathbf{b}_1$ may still be longer than $\mathbf{b}_1$, even if $\mathbf{b}_2$ is in the $K$-span of $\mathbf{b}_1$. This leads us to the following question: does there exist a lattice $L$ depending only on $K$ such that being able to solve the Closest Vector Problem (CVP) with respect to $L$ allows to find short bases of modules in $K^2$?

Contributions. The LLL algorithm for Euclidean lattices can be viewed as a way to leverage the ability of Gauss' algorithm to reduce 2-dimensional lattice bases, to reduce $n$-dimensional lattice bases for any $n \geq 2$. We propose extensions to modules of both Gauss' algorithm and of its LLL leveraging from 2 to $n$ dimensions, hence providing a full-fledged framework for LLL-like reduction of module pseudo-bases.

Our first contribution is an oracle-based algorithm which takes as input a pseudo-basis of a module $M \subset K^n$ over the ring of integers $R$ of an arbitrary number field $K$, updates it progressively in a fashion similar to the LLL algorithm, and outputs a pseudo-basis of $M$. The first output vector is short, and the algorithm runs in time polynomial in $n$ and the bit-lengths of the norms of the input vectors and ideals. It makes a polynomial number of calls to an oracle that finds short vectors in rank-2 modules. This oracle-based LLL-like algorithm for modules allows us to obtain the following result for our running example (see Theorem 3.9 for a general statement).

**Theorem 1.1.** *Let $K = \mathbb{Q}[x]/(x^d+1)$ and $R = \mathbb{Z}[x]/(x^d+1)$, for $d$ a power of 2. There is a polynomial-time reduction from finding a $(2\gamma\sqrt{d})^{2n-1}$-approximation to a shortest non-zero vector in modules in $K^n$ (with respect to the Euclidean*

*norm inherited from mapping an element of $K^n$ to the concatenation of its $n$ coefficient vectors) to finding a $\gamma$-approximation to a shortest non-zero vector in modules in $K^2$.*

For example, if $n$ is constant, then the reduction allows to obtain polynomial approximation factors in modules in $K^n$ from polynomial approximation factors in modules in $K^2$.

Our second contribution is a heuristic algorithm to find a very short non-zero vector in an arbitrary module in $K^2$, given access to a CVP oracle with respect to a lattice depending only on $K$. We obtain the following result for our running example (combine Corollary 4.10 with Lemma 2.3 for a general statement).

**Theorem 1.2 (Heuristic).** *There exists a sequence of lattices $L_d$ and an algorithm $\mathcal{A}$ such that the following holds. Algorithm $\mathcal{A}$ takes as input a pseudo-basis of a rank-2 module $M \subset (\mathbb{Q}/(x^d+1))^2$, and outputs a vector $\mathbf{v} \in M \setminus \{0\}$ that is no more than $2^{(\log d)^{O(1)}}$ longer than a shortest non-zero vector of $M$. If given access to an oracle solving CVP in $L_d$ in polynomial time, then it runs in quantum polynomial time. Finally, for any $\eta > 0$, the lattice $L_d$ can be chosen of dimension $O(d^{2+\eta})$.*

The quantum component of the algorithm is the decomposition of an ideal as the product of a subset of fixed ideals and a principal ideal with a generator [BS16]. By relying on [BEF+17] instead, one can obtain a dequantized variant of Theorem 1.2 relying on more heuristics and in which the algorithm runs in $2^{\widetilde{O}(\sqrt{d})}$ classical time.

We insist that the result relies on heuristics. Some are inherited from prior works (such as [PHS19]) and one is new (Heuristic 1 in Section 4). The new heuristic quantifies the distance to $L_d$ of vectors in the real span of $L_d$ that satisfy some properties. This heuristic is difficult to prove as the lattice $L_d$ involves other lattices that are not very well understood (the log-unit lattice and the lattice of class group relations between ideals of small algebraic norms). We justify this heuristic by informal counting arguments and by some experiments in small dimensions.

Finally, we note that the dimension of $L_d$ is near-quadratic in the degree $d$ of the field. This is much more than the lattice dimension $d$ of $R$, but we do not know how to use a CVP oracle for $R$ to obtain such an algorithm to find short vectors in rank-2 modules. An alternative approach to obtain a similar reduction from finding short non-zero vectors in rank-2 modules to CVP with preprocessing would be as follows: to reach the goal, it suffices to find a short non-zero vector in a $(2d)$-dimensional lattice; by using the LLL algorithm and numerical approximations (see, e.g., [SMSV14]), it is possible to further assume that the bit-size of the inputs is polynomial in $d$; by Kannan's search-to-decision reduction for the shortest vector problem [Kan87], it suffices to obtain an algorithm that decides whether or not a lattice contains a non-zero vector of norm below 1; the latter task can be expressed as an instance of 3SAT, as the corresponding language belongs to NP; finally, 3SAT reduces to CVP with preprocessing [Mic01]. Over-

all, this gives an alternative to Theorem 1.2 without heuristics, but lattices $L_d$ of much higher dimensions (which still grow polynomially in $d$).

Technical overview. One of the technical difficulties of extending LLL to modules is the fact that the absolute value $|\cdot|$ over $\mathbb{Q}$ has two canonical generalizations over $K$: the trace norm and the algebraic norm. Let $(\sigma_i)_{i\leq d}$ denote the embedding of $K$ into $\mathbb{C}$. The trace norm and algebraic norm of $x \in K$ are respectively defined as $(\sum_i |\sigma_i(x)|^2)^{1/2}$ and $\prod_i \sigma_i(x)$. When $K = \mathbb{Q}$, the only embedding is the identity map, and both the trace norm and the absolute value of the algebraic norm collapse to the absolute value. When the field degree is greater than 1, they do not collapse, and are convenient for diverse properties. For instance, the trace norm is convenient to measure smallness of a vector over $K^n$. A nice property is that the bit-size of an element of $R$ is polynomially bounded in the bit-size of the trace norm (for a fixed field $K$). Oppositely, an element in $R$ may have algebraic norm 1 (in this case, it is called a unit), but can have arbitrarily large bit-size. On the other hand, the algebraic norm is multiplicative, which interacts well with determinants. For example, the determinant of the lattice corresponding to a diagonal matrix over $K$ is simply the product of the algebraic norms of the diagonal entries (up to a scalar depending only on the field $K$). LLL relies on all these properties, that are conveniently satisfied by the absolute value.

In our first contribution, i.e., the LLL-like algorithm to reduce module pseudo-bases, we crucially rely on the algebraic norm. Indeed, the progress made by the LLL algorithm is measured by the so-called potential function, which is a product of determinants. As observed in prior works [FP96, KL17], using the algebraic norm allows for a direct generalization of this potential function to module lattices. What allowed us to go beyond norm-Euclidean number fields is the black-box handling of rank-2 modules. By not considering this difficult component, we can make do with the algebraic norm for the most important parts of the algorithm. The trace norm is still used to control the bit-sizes of the module pseudo-bases occurring during the algorithm, allowing to extend the so-called size-reduction process within LLL, but is not used to "make progress". The black-boxing of the rank-2 modules requires the introduction of a modified condition for deciding which 2-dimensional pseudo-basis to consider to "make progress" on the $n$-dimensional pseudo-basis being reduced. This condition is expressed as the ratio between 2-determinants, which is compatible with the exclusive use of the algebraic norm to measure progress. It involves the coefficient ideals, which was unnecessary in prior works handling norm-Euclidean fields, as for such fields, all modules can be generated by a basis instead of a pseudo-basis.

Our algorithm for finding short non-zero vectors in rank-2 modules iterates divide-and-swap steps like 2-dimensional LLL (or Gauss' algorithm). The crucial component is the generalization of the Euclidean division, from $\mathbb{Z}$ to $R$. We are given $a \in K \setminus \{0\}$ and $b \in K$, and we would like to shorten $b$ using $R$-multiples of $a$. In the context of $a \in \mathbb{Q}\setminus\{0\}$ and $b \in \mathbb{Q}$, a Euclidean division provides us with $u \in \mathbb{Z}$ such that $|b+ua| \leq |a|/2$. We would like to have an analogous division in $R$. However, the ring $R$ may not be Euclidean. Moreover, the covering radius of the

ring $R$ (viewed as a lattice) can be larger than 1, and hence, in most cases, there will not even exist an element $u \in R$ such that $\|b+au\| \leq \|a\|$ (here $\|\cdot\|$ refers to the trace norm). In order to shorten $b$ using $a$, we also allow $b$ to be multiplied by some element $v \in R$. For this extension to be non-trivial (and useful), we require that $v$ is not too large (otherwise, one can always take $u = b$ and $v = -a$ for instance, if $a, b \in R$, and extend this approach for general $a, b \in K$). Hence, we are interested in finding $u, v$ such that $\|ua + vb\| \leq \varepsilon\|a\|$ and $\|v\| \leq C$ for some $\varepsilon < 1$ and $C$ to be determined later. Intuitively, if we allow for a large number of such multiples $v$ (proportional to $1/\varepsilon$ and to the determinant of the lattice corresponding to $R$, i.e., the square root of the field discriminant), there should be one such $v$ such that there exists $u \in R$ with $\|vb + au\| \leq \varepsilon\|a\|$. We do not know how to achieve the result with this heuristically optimal number of $v$'s and use potentially larger $v$'s. The astute reader will note that if we use such a $v$ inside a divide-and-swap algorithm, we may end up computing short vectors in sub-modules of the input modules. We prevent this from happening by using the module Hermite Normal Form [BP91, Coh96, BFH17].

To find $u, v$ such that $\|vb+au\|$ is small, we use the logarithm map Log over $K$. For this discussion, we do not need to explain how it is defined, but only that it "works" like the logarithm map log over $\mathbb{R}_{>0}$. In particular if $x \approx y$, then $\operatorname{Log} x \approx \operatorname{Log} y$. We would actually prefer to have the converse property, but it does not hold for the standard Log over $K$. In Subsection 4.1, we propose an extension $\overline{\operatorname{Log}}$ such that $\overline{\operatorname{Log}}\, x \approx \overline{\operatorname{Log}}\, y$ implies that $x \approx y$. In our context, this means that we want to find $u, v$ such that $\overline{\operatorname{Log}}\, v - \overline{\operatorname{Log}}\, u \approx \overline{\operatorname{Log}}(b) - \overline{\operatorname{Log}}(a)$. To achieve this, we will essentially look for such $u$ and $v$ that are product combinations of fixed small elements in $R$. When applying the $\overline{\operatorname{Log}}$ function, the product combinations become integer combinations of the $\overline{\operatorname{Log}}$'s of the fixed elements. This gives us our CVP instance: the lattice is defined using the $\overline{\operatorname{Log}}$'s of the fixed elements and the target is defined using $\overline{\operatorname{Log}}(b) - \overline{\operatorname{Log}}(a)$. This description is only to provide intuition, as reality is more technical: we use the log-unit lattice and small-norm ideals rather than small-norm elements.

One advantage of using the $\overline{\operatorname{Log}}$ map is that the multiplicative structure of $K$ is mapped to an additive structure, hence leading to a CVP instance. On the downside, one needs extreme closeness in the $\overline{\operatorname{Log}}$ space to obtain useful closeness in $K$ (in this direction, we apply an exponential function). Put differently, we need the lattice to be very dense so that there is a lattice vector that is very close to the target vector. This is the fundamental reason why we end up with a large lattice dimension: we add a large number of $\overline{\operatorname{Log}}$'s of small-norm ideals to densify the lattice. This makes the analysis of the distance to the lattice quite cumbersome, as the Gaussian heuristic gives too crude estimates. For our running example, we have a lattice of dimension $\approx d^2$ and determinant $\approx 1$, hence we would expect a 'random' target vector to be at distance $\approx d$ from the lattice. We argue for a distance of at most $\approx \sqrt{d}$ for 'specific' target vectors. Finally, we note that the lattice and its analysis share similarities with the Schnorr-Adleman lattice that Ajtai used to prove NP-hardness of SVP under randomized reductions [Ajt98, MG02] (but we do not know if there is a connection).

Impact. Recent works have showed that lattice problems restricted to ideals of some cyclotomic number fields can be quantumly solved faster than for arbitrary lattices, for some ranges of parameters [CDW17], and for all number fields with not too large discriminant, if allowing preprocessing that depends only on the field [PHS19]. Recall that ideal lattices are rank-1 module lattices. Our work can be viewed as a step towards assessing the existence of such weaknesses for modules of larger rank, which are those that appear when trying to cryptanalyze cryptosystems based on the NTRU, RingSIS, RingLWE, ModuleSIS and ModuleLWE problems and their variants.

Similarly to [CDW17, PHS19], our results use CVP oracles for lattices defined in terms of the number field only (i.e., defined independently of the input module). In [CDW17, PHS19], the weaknesses of rank-1 modules stemmed from two properties of these CVP instances: the lattices had dimension quasi-linear in the log-discriminant (quasi-linear in the field degree, for our running example), and either the CVP instances were easy to solve [CDW17], or approximate solutions sufficed [PHS19] and one could rely on Laarhoven's CVP with preprocessing algorithm [Laa16]. In our case, we need (almost) exact solutions to CVP instances for which we could not find any efficient algorithm, and the invariant lattice has a dimension that is more than quadratic in the log-discriminant (in the field degree, for our running example). It is not ruled out that there could be efficient CVP algorithms for such lattices, maybe for some fields, but we do not have any lead to obtain them.

As explained earlier, CVP with preprocessing is known to be NP-complete, so there always exists a fixed lattice allowing to solve the shortest vector problem in lattices of a target dimension. However, the dimension of that fixed lattice grows as a high degree polynomial in the target dimension. The fact that we only need near-quadratic dimensions (when the log-discriminant is quasi-linear in the field degree) may be viewed as a hint that finding short non-zero vectors in rank-2 modules might be easier than finding short non-zero vectors in arbitrary lattices of the same dimension.

Finally, our first result shows the generality of rank-2 modules towards finding short vectors in rank-$n$ modules for any $n \geq 2$. The reduction allows to stay in the realm of polynomial approximation factors (with respect to the field degree) for any constant $n$. This tends to back the conjecture that there might be a hardness gap between rank-1 and rank-2 modules, and then a smoother transition for higher rank modules.

Notations. For two real valued functions $f$ and $g$, we write $f(x) = \tilde{O}(g(x))$ if and only if there exists some constant $c > 0$ such that $f(x) = O(g(x) \cdot |\log g(x)|^c)$. By abuse of notations, we write $O(x^\alpha \mathrm{poly}(\log x))$ as $\widetilde{O}(x^\alpha)$ even if $\alpha = 0$. We let $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$ denote the sets of integers, rational, real, and complex numbers, respectively. For $x \in \mathbb{C}$, we let $\bar{x}$ denote its complex conjugate. We use lower-case (resp. upper-case) bold letters for vectors (resp. matrices). For vectors $\mathbf{x}_i = (x_{ij})_j$ for $i \leq k$, we write $(\mathbf{x}_1 \| \ldots \| \mathbf{x}_k)$ to denote the vector obtained by concatenation. By default, the matrices are written with column vectors.

For a vector $\mathbf{x} = (x_i)_i \in \mathbb{C}^n$, we write $\|\mathbf{x}\|_i$ for $i \in \{1, 2, \infty\}$ to denote $\ell_i$-norm, and we typically omit the subscript when $i = 2$. For a lattice $\Lambda \subset \mathbb{R}^n$, we let $\rho(\Lambda)$ denote the covering radius with respect to Euclidean norm.

Supplementary material. Due to lack of space, some material is provided only in the full version [LPSW19]. This includes: background on computational aspects on number fields, several proofs, and reports on experiments backing the heuristic claims.

## 2 Preliminaries

In this section, we first recall some necessary algebraic number theory background and discuss some computational aspects. We then extend Gram-Schmidt orthogonalization to matrices over number fields. In this section, we assume that the reader is somehow familiar with the algebraic notions used in this article and in previous works. For more details on these mathematical objects, we refer the reader to [Neu99, Chapter 1] for algebraic number theory questions, to [Hop98] for anything related to modules and to [PHS19] where the same techniques were used in a simpler setting.

### 2.1 Algebraic background

Number fields. We let $K$ be a number field of degree $d$ and $K_{\mathbb{R}} = K \otimes_{\mathbb{Q}} \mathbb{R}$. A number field comes with $r_1$ real embeddings and $2r_2$ complex embeddings $\sigma_i$'s, where $r_1 + 2r_2 = d$. The field norm is defined as $\mathcal{N}(x) = \prod_{i \leq d} \sigma_i(x)$ and the field trace is $\mathrm{Tr}(x) = \sum_{i \leq d} \sigma_i(x)$. The canonical embedding of $K$ is then defined as $\sigma(x) \in \mathbb{R}^{r_1} \times \mathbb{C}^{2r_2}$, where $\sigma_{r_1+i}(x) = \overline{\sigma_{r_1+r_2+i}(x)}$ for $1 \leq i \leq r_2$. The field trace then induces a Hermitian inner product over $K_{\mathbb{R}}$ whose associated Euclidean norm is $\|x\| = (\sum_{1 \leq i \leq d} |\sigma_i(x)|^2)^{1/2}$ for $x \in K_{\mathbb{R}}$. We also define $\|x\|_\infty = \max_{i \in [d]} |\sigma_i(x)|$.

In this work, elements of $K$ are identified to their canonical embeddings. From this perspective, the set $K_{\mathbb{R}}$ is also identified to $\{\mathbf{y} \in \mathbb{R}^{r_1} \times \mathbb{C}^{2r_2} : \forall i \leq r_2\,,\, \overline{y_{r_1+r_2+i}} = y_{r_1+i}\}$ (the embedding map $\sigma$ provides a ring isomorphism between $K_{\mathbb{R}}$ and the latter subspace of $\mathbb{R}^{r_1} \times \mathbb{C}^{2r_2}$). We write $K_{\mathbb{R}}^\times$ for the subset of vectors in $K_{\mathbb{R}}$ with non-zero entries (it forms a group, for component-wise multiplication). We also write $K_{\mathbb{R}}^+$ for the subset of vectors in $K_{\mathbb{R}}$ with non-negative (real) coefficients. For $x \in K_{\mathbb{R}}$, we let $\bar{x}$ refer to the element of $K_{\mathbb{R}}$ obtained by complex conjugation of every coordinate.[4] We can also define a square-root $\sqrt{\cdot} : K_{\mathbb{R}}^+ \to K_{\mathbb{R}}^+$ by taking coordinate-wise square roots.

We let $R$ be the ring of integers of $K$. It is a free $\mathbb{Z}$-module of rank $d$, and can be seen as a lattice via the canonical embedding. The discriminant $\Delta_K$ of $K$

[4] Observe that even if complex conjugation might not be well defined over $K$ (i.e., the element $\bar{x}$ might not be in $K$ even if $x$ is), it is however always defined over $K_{\mathbb{R}}$. In this article, complex conjugation will only be used on elements of $K_{\mathbb{R}}$, and we make no assumption that $K$ should be stable by conjugation.

is then the squared volume of $R$, i.e., $\Delta_K = \det((\sigma_i(x_j))_{ij})^2$ for any $\mathbb{Z}$-basis $(x_i)_{i\le d}$ of $R$. We will often use the inequality $\log \Delta_K \ge \Omega(d)$ to simplify cost estimates.

We let $R^\times = \{u \in R \,|\, \exists v \in R : uv = 1\}$ denote the group of units of $R$. Dirichlet's unit theorem states that $R^\times$ is isomorphic to the Cartesian product of a finite cyclic group (formed by the roots of unity contained in $K$) with the additive group $\mathbb{Z}^{r_1+r_2-1}$. We define $\mathrm{Log} : K_{\mathbb{R}}^\times \to \mathbb{R}^d$ by $\mathrm{Log}(x) = (\log(|\sigma_1(x)|), \ldots, \log(|\sigma_d(x)|))^T$. Let $E = \{x \in \mathbb{R}^d \mid \forall r_1 \le i \le r_2 : x_i = x_{i+r_2}\}$. We have $\mathrm{Log}(K_{\mathbb{R}}^\times) \subseteq E$. We also define $H = \{x \in \mathbb{R}^d : \sum_{i\in[d]} x_i = 0\}$ and $\mathbf{1} = (1, \ldots, 1)^T$, which is orthogonal to $H$ in $\mathbb{R}^d$. The set $\Lambda = \{\mathrm{Log}(u) : u \in R^\times\}$ is a lattice, called "log-unit" lattice. It has rank $r_1 + r_2 - 1$, by Dirichlet's units theorem and is full rank in $E \cap H$. Further, its minimum satisfies $\lambda_1(\Lambda) \ge (\ln d)/(6d^2)$ (see [FP06, Cor. 2]).

Ideals. A fractional ideal $I$ of $K$ is an additive subgroup of $K$ which is also stable by multiplication by any element of $R$, and such that $xI \subseteq R$ for some $x \in \mathbb{Z}\backslash\{0\}$. Any non-zero fractional ideal is also a free $\mathbb{Z}$-module of rank $d$, and can therefore be seen as a lattice in $K_{\mathbb{R}}$ using the canonical embedding: such lattices are called ideal lattices. The product $IJ$ of two fractional ideals $I$ and $J$ is the fractional ideal generated by all elements $xy$ with $x \in I$ and $y \in J$. Any non-zero fractional ideal $I$ is invertible, i.e., there exists a unique ideal $I^{-1} = \{x \in K : xI \subseteq R\}$ such that $II^{-1} = R$. When $I \subseteq R$, it is said to be an integral ideal. An integral ideal $\mathfrak{p}$ is said to be prime if whenever $\mathfrak{p} = IJ$ with $I$ and $J$ integral, then either $I = \mathfrak{p}$ or $J = \mathfrak{p}$. For any $g \in K$, we write $\langle g \rangle = gR$ the smallest fractional ideal containing $g$, and we say that it is a principal ideal. The quotient of the group of non-zero fractional ideals (for ideal multiplication) by the subgroup consisting in principal ideals is the class group $\mathcal{C}l_K$. Its cardinality $h_K$ is called the class number. Under the GRH, there is a set of cardinality $\le \log h_K = \widetilde{O}(\log \Delta_K)$ of prime ideals of norms $\le 12 \log^2 \Delta_K$ that generates $\mathcal{C}l_K$ (see, e.g., [PHS19, Se. 2.3]). We also will use the bound $h_K \cdot (\det \Lambda) \le 2^{\widetilde{O}(\log \Delta_K)}$ (see, e.g., [PHS19, Se. 2.4]).

The algebraic norm $\mathcal{N}(I)$ of an integral ideal $I$ is its index as a subgroup of $R$, and is equal to $\det(\sigma(I))/\Delta_K^{1/2}$. The algebraic norm of a prime ideal is a power of a prime number. For a principal ideal, we also have $\mathcal{N}(\langle g \rangle) = |\mathcal{N}(g)|$. The norm extends to fractional ideals using $\mathcal{N}(I) = \mathcal{N}(xI)/|\mathcal{N}(x)|$, for any $x \in R \setminus \{0\}$ such that $xI \subseteq R$. We have $\mathcal{N}(IJ) = \mathcal{N}(I)\mathcal{N}(J)$ for all fractional ideals $I, J$.

**Lemma 2.1 ( [BS96, Th. 8.7.4]).** *Assume the GRH. Let $\pi_K(x)$ be the number of prime integral ideals of $K$ of norm $\le x$. Then there exists an absolute constant $C$ (independent of $K$ and $x$) such that $|\pi_K(x) - \mathrm{li}(x)| \le C \cdot \sqrt{x}(d \log x + \log \Delta_K)$, where $\mathrm{li}(x) = \int_2^x \frac{\mathrm{d}t}{\ln t} \sim \frac{x}{\ln x}$.*

Module lattices and their geometry. In this work, we call ($R$-)module any set of the form $M = I_1\mathbf{b}_1 + \ldots + I_n\mathbf{b}_n$, where the $I_j$'s are non-zero fractional ideals of $R$ and the $\mathbf{b}_j$'s are $K_{\mathbb{R}}$-linearly independent[5] vectors in $K_{\mathbb{R}}^m$, for some $m > 0$.

[5] The vectors $\mathbf{b}_j$'s are said to be $K_{\mathbb{R}}$-linearly independent if and only if there is no non-trivial ways to write the zero vector as a $K_{\mathbb{R}}$-linear combination of the $\mathbf{b}_j$'s.

The tuple of pairs $((I_1, \mathbf{b}_1), \ldots, (I_n, \mathbf{b}_n))$ is called a pseudo-basis of $M$, and $n$ is its rank. Note that the notion of rank of a module is usually only defined when the module has a basis (i.e., is of the form $M = R\mathbf{b}_1 + \ldots + R\mathbf{b}_n$, with all the ideals equal to $R$). In this article, we consider an extension of the definition of rank, defined even if the module does not have a basis, as long as it has a pseudo-basis. In particular, fractional ideals are rank-1 modules contained in $K$, and sets of the form $\alpha \cdot I$ for $\alpha \in K_{\mathbb{R}}^{\times}$ and a non-zero fractional ideal $I$ are rank-1 modules in $K_{\mathbb{R}}$. We refer to [Hop98] for a thorough study of $R$-modules, and concentrate here on the background necessary to the present work.

Two pseudo-bases $((I_1, \mathbf{b}_1), \ldots, (I_n, \mathbf{b}_n))$ and $((J_1, \mathbf{c}_1), \ldots, (J_n, \mathbf{c}_n))$ represent the same module if and only if there exists $\mathbf{U} = (u_{ij})_{i,j} \in K^{n \times n}$ invertible such that $\mathbf{C} = \mathbf{B} \cdot \mathbf{U}$; we have $u_{ij} \in I_i J_j^{-1}$ and $u'_{ij} \in J_i I_j^{-1}$ for all $i, j$ and for $\mathbf{U}' = (u'_{ij})_{i,j} := \mathbf{U}^{-1}$. Here, the matrix $\mathbf{B}$ is the concatenation of the column vectors $\mathbf{b}_i$ (and similarly for $\mathbf{C}$). If $n > 0$, we define $\det_{K_{\mathbb{R}}} M = \det(\overline{\mathbf{B}}^{\top}\mathbf{B})^{1/2} \cdot \prod_i I_i$. It is an $R$-module in $K_{\mathbb{R}}$. Note that it is a module invariant, i.e., it is identical for all pseudo-bases of $M$.

We extend the canonical embedding to vectors $\mathbf{v} = (v_1, \ldots, v_m)^T \in K_{\mathbb{R}}^m$ by defining $\sigma(\mathbf{v})$ as the vector of $\mathbb{R}^{dm}$ obtained by concatenating the canonical embeddings of the $v_i$'s. This extension of the canonical embedding maps any module $M$ of rank $n$ to a $(dn)$-dimensional lattice in $\mathbb{R}^{dm}$. We abuse notation and use $M$ to refer to both the module and the lattice obtained by applying the canonical embedding.

The determinant of a module $M$ seen as a lattice is $\det M = \Delta_K^{n/2} \cdot \mathcal{N}(\det_{K_{\mathbb{R}}} M)$. This matches with the module determinant definition from [FS10, Se. 2.3]. Since $\det(M) \neq 0$, this shows in particular that the diagonal coefficients $r_{ii}$ of the R-factor are invertible in $K_{\mathbb{R}}$ (otherwise, one of their embedding would be 0 and so would be their norm).

We consider the following inner products for $\mathbf{a}, \mathbf{b} \in K_{\mathbb{R}}^m$:

$$\langle \mathbf{a}, \mathbf{b} \rangle_{K_{\mathbb{R}}} = \sum_{i \in [m]} a_i \bar{b}_i \in K_{\mathbb{R}} \quad \text{and} \quad \langle \mathbf{a}, \mathbf{b} \rangle = \mathrm{Tr}\Big( \sum_{i \in [m]} a_i \bar{b}_i \Big) \in \mathbb{C}.$$

Note that we have $\langle \mathbf{v}, \mathbf{v} \rangle_{K_{\mathbb{R}}} \in K_{\mathbb{R}}^+$, as all $\sigma_i(\langle \mathbf{v}, \mathbf{v} \rangle_{K_{\mathbb{R}}})$'s are non-negative. For $\mathbf{v} \in K_{\mathbb{R}}^m$, we define $\|\mathbf{v}\|_{K_{\mathbb{R}}} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{K_{\mathbb{R}}}}$ and $\|\mathbf{v}\| = \sqrt{\mathrm{Tr}(\langle \mathbf{v}, \mathbf{v} \rangle_{K_{\mathbb{R}}})} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$. Observe that $\|\mathbf{v}\|$ correspond to the Euclidean norm of $\mathbf{v}$ when seen as a vector of dimension $dm$ via the canonical embedding. We extend the infinity norm to vectors $\mathbf{v} \in K_{\mathbb{R}}^m$ by $\|\mathbf{v}\|_{\infty} = \max_{i \in [m]} \|v_i\|_{\infty}$, where $\mathbf{v} = (v_1, \ldots, v_m)$. We also extend the algebraic norm to vectors $\mathbf{v} \in K_{\mathbb{R}}^m$ by setting $\mathcal{N}(\mathbf{v}) := \mathcal{N}(\|\mathbf{v}\|_{K_{\mathbb{R}}})$. For $m = 1$, we see that $\mathcal{N}(\mathbf{v}) = |\mathcal{N}(\mathbf{v})|$. By the arithmetic-geometric inequality, we have $\sqrt{d} \cdot \mathcal{N}(\mathbf{a})^{1/d} \leq \|\mathbf{a}\|$ for $\mathbf{a} \in K_{\mathbb{R}}^m$. Observe also that for any vector $\mathbf{v} = (v_1, \ldots, v_m)^T \in K_{\mathbb{R}}$, we have $\mathcal{N}(\mathbf{v}) \geq \max_i(\mathcal{N}(v_i))$, because for any embedding $\sigma_j$, it holds that $|\sigma_j(v_1\overline{v_1} + \cdots + v_m\overline{v_m})| = |\sigma_j(v_1)|^2 + \cdots + |\sigma_j(v_m)|^2 \geq \max_i |\sigma_j(v_i)|^2$.

Because $K_{\mathbb{R}}$ is a ring and not a field, this definition is stronger than requiring that none of the $\mathbf{b}_j$'s is in the span of the others.

We define the module minimum $\lambda_1(M)$ as the norm of a shortest non-zero element of $M$ with respect to $\|\cdot\|$. Our module-LLL algorithm will rely on the algebraic norm rather than the Euclidean norm. For this reason, we will also be interested to the minimum $\lambda_1^{\mathcal{N}}(M) = \inf(\mathcal{N}(\mathbf{v}) : \mathbf{v} \in M \setminus \{\mathbf{0}\})$. We do not know if this minimum is always reached for some vector $\mathbf{v} \in M$, but we can find an element of $M$ whose algebraic norm is arbitrarily close to $\lambda_1^{\mathcal{N}}(M)$. The following lemma provides relationships between $\lambda_1(M)$ and $\lambda_1^{\mathcal{N}}(M)$.

**Lemma 2.2.** *For any rank-$n$ module $M$, we have:*

$$d^{-d/2}\lambda_1(M)^d \Delta_K^{-1/2} \leq \lambda_1^{\mathcal{N}}(M) \leq d^{-d/2}\lambda_1(M)^d \leq n^{d/2}\Delta_K^{1/2}\mathcal{N}(\det_{K_{\mathbb{R}}} M)^{1/n}.$$

### 2.2 Computing over rings

Background on field and ideal arithmetic is provided in the full version [LPSW19].

Computations with an oracle. In Section 4, we will assume that we have access to an oracle for the Closest Vector Problem, for lattices related to $K$. For example, we will assume that we can solve CVP for the lattice corresponding to $R$, with respect to $\|\cdot\|$. This lattice has dimension $d$.

In a similar vein, we will use the following adaptation from [PHS19, Th. 3.4], to find short elements in rank-1 modules.

**Lemma 2.3 (Heuristic).** *There exists a lattice $L_K$ (that only depends on $K$ and has dimension $\widetilde{O}(\log \Delta_K)$) such that, given an oracle access to an algorithm that solves* CVP *for $L_K$, the following holds. There exists a heuristic quantum polynomial-time algorithm that takes as input an ideal $I$ of $K$ and any $\alpha \in K_{\mathbb{R}}^{\times}$, and outputs $x \in \alpha I \setminus \{0\}$ such that*

$$\|x\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(\alpha)|^{1/d} \cdot \mathcal{N}(I)^{1/d},$$

*where $\mathrm{c} = 2^{\widetilde{O}(\log|\Delta|)/d}$. In particular, we have $\|x\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(x)|^{1/d}$.*

The result assumes GRH and Heuristic 4 from [PHS19]. The quantum computation performed by the algorithm derives from [BS16] and consists in computing the log-unit lattice, finding a small generating set $([\mathfrak{p}_i])_i$ of the class group $\mathcal{C}l_K$ of $K$, and decomposing the class $[I]$ of $I$ in $\mathcal{C}l_K$ in terms of that generating set. These quantum computations can be replaced by classical ones (e.g., [BF14, BEF+17]), at the expense of increased run-times and additional heuristic assumptions.

The lemma can be derived from [PHS19, Th. 3.4] by replacing Laarhoven's CVPP algorithm [Laa16] by an exact CVPP oracle. In [PHS19], the CVPP algorithm is used with a target vector $\mathbf{t}$ derived from the decomposition of $[I]$ on the $[\mathfrak{p}_i]$'s and the logarithm $\mathrm{Log}(g)$ of an element $g \in K$. To obtain the statement above, we replace $\mathrm{Log}(g)$ by $\mathrm{Log}(g \cdot \alpha) = \mathrm{Log}(g) + \mathrm{Log}(\alpha)$. The last lemma statement $\|x\|_\infty \leq \mathrm{c}|\mathcal{N}(x)|^{1/d}$ comes from the observation that $|\mathcal{N}(x)| \geq \mathcal{N}(\alpha) \cdot \mathcal{N}(I)$ (which holds because $x$ belongs to $\alpha I \setminus \{0\}$).

### 2.3 Gram-Schmidt orthogonalization

We extend Gram-Schmidt Orthogonalization from matrices over the real numbers to matrices over $K_{\mathbb{R}}^m$. For $(\mathbf{b}_1, \ldots, \mathbf{b}_n) \in K_{\mathbb{R}}^{m \times n}$ such that $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are $K_{\mathbb{R}}$-linearly independent, we define $\mathbf{b}_1^* = \mathbf{b}_1$ and, for $1 < i \leq n$:

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j<i} \mu_{ij} \mathbf{b}_j^* \quad \text{with} \quad \forall j < i: \ \mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle_{K_{\mathbb{R}}}}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle_{K_{\mathbb{R}}}}.$$

It may be checked that $\langle \mathbf{b}_i^*, \mathbf{b}_j^* \rangle = 0$ for $i \neq j$, and that $\mathbf{b}_i^* = \text{argmin}(\|\mathbf{b}_i - \sum_{j<i} y_j \mathbf{b}_j\| \mid \forall j : y_j \in K_{\mathbb{R}})$.

We also extend the QR-factorization to matrices over $K_{\mathbb{R}}$. We define $r_{ii} = \|\mathbf{b}_i^*\|_{K_{\mathbb{R}}}$ for $i \leq n$, $r_{ij} = \mu_{ji} r_{ii}$ when $i < j$, and $r_{ij} = 0$ when $i > j$. We then have $\mathbf{B} = \mathbf{Q} \cdot \mathbf{R}$, where $\mathbf{Q} \in K_{\mathbb{R}}^{m \times n}$ is the matrix whose columns are the $\mathbf{b}_i^* / \|\mathbf{b}_i^*\|_{K_{\mathbb{R}}}$'s and $\mathbf{R} = (r_{ij})_{ij}$. Note that $\overline{\mathbf{Q}}^T \mathbf{Q} = \mathbf{Id}$ and that $\mathbf{R}$ is upper-triangular with diagonal coefficients in $K_{\mathbb{R}}^+$.

The following lemma provides relationships between some module invariants and the QR-factorization.

**Lemma 2.4.** *Let $M \subset K_{\mathbb{R}}^m$ be a module with pseudo-basis $((I_i, \mathbf{b}_i))_{i \leq n}$. Let $\mathbf{R}$ be the R-factor of $\mathbf{B}$. Then, we have $\det_{K_{\mathbb{R}}} M = \prod_i r_{ii} I_i$ and $\det M = \Delta_K^{n/2} \prod_i \mathcal{N}(r_{ii} I_i)$. Further, for any vector $\mathbf{v} \in K_{\mathbb{R}}^m$ and fractional ideal $I \subset K$ such that $0 \subsetneq \mathbf{v}I \subseteq M$, it holds that $\mathcal{N}(\mathbf{v}) \cdot \mathcal{N}(I) \geq \min_i \mathcal{N}(r_{ii} I_i)$. This implies in particular that $\lambda_1^{\mathcal{N}}(M) = \inf_{\mathbf{s} \in M \setminus \{\mathbf{0}\}} \mathcal{N}(\mathbf{s}) \geq \min_i \mathcal{N}(r_{ii} I_i)$.*

In this work, we will mostly rely on QR-factorization. It carries the same information as Gram-Schmidt orthogonalization, but allows for simpler explanations. However, from a computational perspective, the R-factor may be difficult to represent exactly even for modules contained in $K^m$, because of the square roots appearing in its definition. This difficulty is circumvented by computing the Gram-Schmidt orthogonalization instead, and using it as a means to represent the R-factor. In the full version, we explain how to efficiently compute Gram-Schmidt orthogonalizations.

For lattices, if we have a basis and a full-rank family of short vectors, then we can efficiently obtain a basis of the lattice whose Gram-Schmidt vectors are no longer than those of the full-rank family of short vectors. This was generalized to modules in [FS10], relying on the extension to modules of the Hermite Normal Form [BP91, Coh96, BFH17].

**Lemma 2.5 ( [FS10, Th. 4]).** *There exists an algorithm that takes as inputs a pseudo-basis $((I_i, \mathbf{b}_i))_{i \leq n}$ of a module $M \subset K_{\mathbb{R}}^m$ and a full-rank set of vectors $(\mathbf{s}_i)_{i \leq n}$ of $M$ and outputs a pseudo-basis $((J_i, \mathbf{c}_i))_{i \leq n}$ such that $\mathbf{c}_i \in M$ and $\mathbf{c}_i^* = \mathbf{s}_i^*$ for all $i$. If $M \subset K^m$, then it terminates in polynomial-time.*

Note that the condition that $\mathbf{c}_i \in M$ implies that $R \subseteq J_i$, for all $i$.

## 3 LLL-reduction of module pseudo-bases

LLL-reduction of lattice bases is defined in terms of Gram-Schmidt orthogonalization (or, equivalently, QR-factorization). A basis is said LLL-reduced if two conditions are satisfied. The first one, often referred to as size-reduction condition, states that any off-diagonal coefficients $r_{ij}$ of the R-factor should have a small magnitude compared to the diagonal coefficient $r_{ii}$ on the same row. The second one, often referred to as Lovász' condition, states that the 2-dimensional vector $(r_{i,i}, 0)^T$ is no more than $1/\delta$ times longer than $(r_{i,i+1}, r_{i+1,i+1})^T$, for some parameter $\delta < 1$. The size-reduction condition allows to ensure that the norms of the vectors during the LLL execution and at its completion stay bounded. More importantly, in combination with Lovász' condition, it makes it impossible for $r_{i+1,i+1}/r_{i,i}$ to be arbitrarily small (for an LLL-reduced basis). The latter is the crux of both the LLL output quality and its fast termination.

### 3.1 An LLL algorithm for module lattices

When extending to rings, the purpose of the size-reduction condition is better expressed in terms of the Euclidean norm $\|\cdot\|$, whereas the bounded decrease of the $r_{ii}$'s is better quantified in terms of the algebraic norm $\mathcal{N}(\cdot)$. This discrepancy makes the definition of a LLL-reduction algorithm for modules difficult. In this section, we circumvent this difficulty by directly focusing on the decrease of the $r_{ii}$'s, deferring to later sections the handling of the rank-2 modules of pseudo-bases $((I_i, (r_{i,i}, 0)^T), (I_{i+1}, (r_{i,i+1}, r_{i+1,i+1})^T))$. We also defer to later the bounding of bit-sizes.

**Definition 3.1 (LLL-reducedness of a pseudo-basis).** *A module pseudo-basis $((I_i, \mathbf{b}_i))_{i \leq n}$ is called LLL-reduced with respect to a parameter $\alpha_K \geq 1$ if, for all $i < n$, we have:*

$$\mathcal{N}(r_{i+1,i+1} I_{i+1}) \geq \frac{1}{\alpha_K} \cdot \mathcal{N}(r_{i,i} I_i), \tag{3.1}$$

*where $\mathbf{R} = (r_{i,j})_{i,j}$ refers to the R-factor of the matrix basis $\mathbf{B}$.*

We first explain that LLL-reduced pseudo-bases are of interest, and we will later discuss their computation (for some value of $\alpha_K$).

**Lemma 3.2.** *Assume that $((I_i, \mathbf{b}_i))_{i \leq n}$ is an LLL-reduced pseudo-basis of a module $M$. Then:*

$$\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq \alpha_K^{(n-1)/2} \cdot (\mathcal{N}(\det_{K_\mathbb{R}} M))^{1/n},$$
$$\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq \alpha_K^{n-1} \cdot \lambda_1^{\mathcal{N}}(M).$$

Our LLL algorithm for modules is very similar to the one over the integers.

The algorithm proceeds by finding an approximation to a shortest non-zero element in a rank-2 module, with respect to the algebraic norm. Using

**Algorithm 3.1** LLL-reduction over $K$

**Input:** A pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$ of a module $M \subset K^m$.

**Output:** An LLL-reduced pseudo-basis of $M$.

1: **while** there exists $i < n$ such that $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1}I_{i+1}) < \mathcal{N}(r_{i,i}I_i)$ **do**
2: Define $M_i$ as the rank-2 module spanned by $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$, with $\mathbf{a}_i = (r_{ii}, 0)^T$ and $\mathbf{a}_{i+1} = (r_{i,i+1}, r_{i+1,i+1})^T$;
3: Find $\mathbf{s}_i \in M_i \setminus \{\mathbf{0}\}$ such that $\mathcal{N}(\mathbf{s}_i) \leq \gamma^d \cdot \lambda_1^{\mathcal{N}}(M_i)$;
4: Set $\mathbf{s}_{i+1} = \mathbf{a}_i$ if it is linearly independent with $\mathbf{s}_i$, and $\mathbf{s}_{i+1} = \mathbf{a}_{i+1}$ otherwise;
5: Call the algorithm of Lemma 2.5 with $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ and $(\mathbf{s}_i, \mathbf{s}_{i+1})$ as inputs, and let $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ denote the output;
6: Update $I_i := I'_i$, $I_{i+1} := I'_{i+1}$ and $[\mathbf{b}_i|\mathbf{b}_{i+1}] := [\mathbf{b}_i|\mathbf{b}_{i+1}] \cdot \mathbf{A}^{-1} \cdot \mathbf{A}'$ (where $\mathbf{A} = [\mathbf{a}_i|\mathbf{a}_{i+1}]$ and $\mathbf{A}' = [\mathbf{a}'_i|\mathbf{a}'_{i+1}]$).
7: **end while**
8: **return** $((I_i, \mathbf{b}_i))_{i\leq n}$.

Lemma 2.2, we obtain a sufficient condition on $\alpha_K$ such that Algorithm 3.1 terminates. In particular, if $\alpha_K$ is sufficiently large, then $\mathcal{N}(r_{i+1,i+1}I_{i+1}) < \frac{1}{\alpha_K}\mathcal{N}(r_{i,i}I_i)$ implies that there is a vector $\mathbf{s}$ in the local projected rank-2 module of norm significantly less than $\mathcal{N}(r_{i,i}I_i)$.

**Lemma 3.3.** *Take the notations of Algorithm 3.1, and consider an index $i < n$ such that $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1}I_{i+1}) < \mathcal{N}(r_{i,i}I_i)$. We have $\mathcal{N}(\mathbf{s}_i) \leq \gamma^d \sqrt{\frac{2^d \Delta_K}{\alpha_K}} \mathcal{N}(r_{i,i}I_i)$.*

We are now ready to prove the main result of this section.

**Theorem 3.4.** *Assume that Step 3 of Algorithm 3.1 is implemented with some algorithm $\mathcal{O}$ for some parameter $\gamma$. Assume that $\alpha_K > \gamma^{2d} 2^d \Delta_K$. Then Algorithm 3.1 terminates and outputs an LLL-reduced pseudo-basis of $M$. Further, the number of loop iterations is bounded by*

$$\frac{n(n+1)}{\log(\alpha_K/(\gamma^{2d}2^d\Delta_K))} \cdot \log \frac{\max \mathcal{N}(r_{ii}I_i)}{\min \mathcal{N}(r_{ii}I_i)},$$

*where the $I_i$'s and $r_{ii}$'s are those of the input pseudo-basis.*

*Proof.* We first show that at every stage of the algorithm, the current pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$ is a pseudo-basis of $M$. For this, it suffices to show that the operations performed on it at Step 6 preserve this property. This is provided by the fact that $\mathbf{A}^{-1} \cdot \mathbf{A}'$ maps the pseudo-basis $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ into the pseudo-basis $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ of the same rank-2 module (by Lemma 2.5). Applying the same transformation to $((I_i, \mathbf{b}_i), (I_{i+1}, \mathbf{b}_{i+1}))$ preserves the spanned rank-2 module. The correctness of Algorithm 3.1 is implied by termination and the above.

We now prove a bound on the number of loop iterations, which will in particular imply termination. Consider the quantity

$$\Pi := \prod_{i\leq n} \mathcal{N}(r_{ii}I_i)^{n-i+1}.$$

This quantity if bounded from above by $\max \mathcal{N}(r_{ii}I_i)^{n(n+1)/2}$ and from below by $\min \mathcal{N}(r_{ii}I_i)^{n(n+1)/2}$. Below, we show that $\Pi$ never increases during the execution of the algorithm, and that at every iteration of the while loop, it decreases by a factor $\geq \sqrt{\alpha_K/(\gamma^{2d}2^d\Delta_K)}$. We also show that the quantity $\min \mathcal{N}(r_{ii}I_i)^{n(n+1)/2}$ can only increase during the execution of the algorithm, hence the lower bound above holds with respect to the input $r_{ii}$ and $I_i$ at every step of the algorithm. Combining the decrease rate with the above upper and lower bounds, this implies that the number of loop iterations is bounded by

$$\frac{n(n+1)}{\log(\alpha_K/(\gamma^{2d}2^d\Delta_K))} \cdot \log \frac{\max \mathcal{N}(r_{ii}I_i)}{\min \mathcal{N}(r_{ii}I_i)},$$

where the $I_i$'s and $r_{ii}$'s are those of the input pseudo-basis.

Consider an iteration of the while loop, working at index $i$. We have $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1}I_{i+1}) < \mathcal{N}(r_{i,i}I_i)$. Step 6 is the only one that may change $\Pi$. Observe that we have

$$\Pi = \prod_{j \leq n} \mathcal{N}\left(\det_{K_\mathbb{R}}\left(((I_i, \mathbf{b}_i))_{i \leq j}\right)\right).$$

During the loop iteration, none of the $n$ modules in the expression above changes, except possibly the $i$-th one. Now, note that

$$\mathcal{N}\left(\det_{K_\mathbb{R}}\left(((I_k, \mathbf{b}_k))_{k \leq i}\right)\right) = \prod_{k \leq i} \mathcal{N}(r_{kk}I_k).$$

During the loop iteration under scope, only the $i$-th term in this product may change. At Step 6, it is updated from $\mathcal{N}(r_{ii}I_i)$ to $\mathcal{N}(I_i')\mathcal{N}(\mathbf{a}_i')$. By Lemma 2.5, we have $\mathcal{N}(I_i') \leq 1$ and $\mathbf{a}_i' = \mathbf{s}_i$. Now, by Lemma 3.3, we have that $\mathcal{N}(\mathbf{s}_i) \leq \gamma^d \sqrt{\frac{2^d\Delta_K}{\alpha_K}} \mathcal{N}(r_{ii}I_i)$. Overall, this gives that $\mathcal{N}(r_{ii}I_i)$ and hence $\Pi$ decrease by a factor $\geq \sqrt{\alpha_K/(\gamma^{2d}2^d\Delta_K)}$.

To show that $\min \mathcal{N}(r_{ii}I_i)$ can only increase during the execution of the algorithm, observe that, during a loop iteration, only $\mathcal{N}(r_{ii}I_i)$ and $\mathcal{N}(r_{i+1,i+1}I_{i+1})$ may be modified. Let us call $\mathcal{N}(r_{ii}'I_i')$ and $\mathcal{N}(r_{i+1,i+1}'I_{i+1}')$ the corresponding values at the end of the iteration. We have seen above that $\mathcal{N}(r_{ii}'I_i') \leq \mathcal{N}(r_{ii}I_i)$, which implies that $\mathcal{N}(r_{ii}'I_i') \leq \max(\mathcal{N}(r_{ii}I_i), \mathcal{N}(r_{i+1,i+1}I_{i+1}))$. We also know from Lemma 2.4 that $\mathcal{N}(r_{ii}'I_i') \geq \min(\mathcal{N}(r_{ii}I_i), \mathcal{N}(r_{i+1,i+1}I_{i+1}))$. As the determinant of $M_i$ is constant, we have

$$\mathcal{N}(r_{ii}'I_i') \cdot \mathcal{N}(r_{i+1,i+1}'I_{i+1}') = \mathcal{N}(r_{ii}I_i) \cdot \mathcal{N}(r_{i+1,i+1}I_{i+1}).$$

This implies that $\mathcal{N}(r_{i+1,i+1}'I_{i+1}') \geq \min(\mathcal{N}(r_{ii}I_i), \mathcal{N}(r_{i+1,i+1}I_{i+1}))$. Overall, we have that $\mathcal{N}(r_{ii}'I_i'), \mathcal{N}(r_{i+1,i+1}'I_{i+1}') \geq \min(\mathcal{N}(r_{ii}I_i), \mathcal{N}(r_{i+1,i+1}I_{i+1}))$. □

### 3.2 Handling bit-sizes

In terms of bit-sizes of the diverse quantities manipulated during the execution of the algorithm, there can be several sources of bit-size growth. Like in

the classical LLL-algorithm, the Euclidean norms of off-diagonal coefficients $r_{ij}$ for $i < j$ could grow during the execution. We handle this using a generalized size-reduction algorithm. Other annoyances are specific to the number field setup. There is too much freedom in representing a rank-1 module $I\mathbf{v}$: scaling the ideal $I$ by some $x \in K$ and dividing $\mathbf{v}$ by the same $x$ preserves the module. In the extreme case, it could cost an arbitrarily large amount of space, even to store a trivial rank-1 module such as $R \cdot (1, 0, \ldots, 0)^T$, if such a bad scaling is used (e.g., using such an $x$ with large algebraic norm). Finally, even if the ideal $I$ is "scaled", we can still multiply $\mathbf{v}$ by a unit: this preserves the rank-1 module, but makes its representation longer.[6]

**Definition 3.5.** *A pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$, with $I_i \subset K$ and $\mathbf{b}_i \in K_{\mathbb{R}}^m$ for all $i \leq n$, is said scaled if, for all $i \leq n$,*

$$R \subseteq I_i, \quad \mathcal{N}(I_i) \geq 2^{-d^2} \Delta_K^{-1/2} \text{ and } \|r_{ii}\| \leq 2^d \Delta_K^{1/(2d)} \mathcal{N}(r_{ii} I_i)^{1/d}.$$

*It is said size-reduced if $\|r_{ij}/r_{ii}\| \leq (4d)^d \Delta_K^{1/2}$ for all $i < j \leq n$.*

Note that if $((I_i, \mathbf{b}_i))_{i\leq n}$ is scaled, then $\mathcal{N}(I_i) \leq 1$ for all $i \leq n$. Further, if the spanned module is contained in $R^m$, then $\mathbf{b}_i \in R^m$ for all $i \leq n$. Algorithm 3.2 transforms any pseudo-basis into a scaled pseudo-basis of the same module.

**Algorithm 3.2** Scaling the ideals.

**Input:** A pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$ of a module $M$.
**Output:** A scaled pseudo-basis $((I'_i, \mathbf{b}'_i))_{i\leq n}$ of $M$.

1: **for** $i = 1$ **to** $n$ **do**
2: Use LLL to find $s_i \in r_{ii} \cdot I_i \setminus \{0\}$ such that $\|s_i\| \leq 2^d \Delta_K^{1/(2d)} \mathcal{N}(r_{ii} I_i)^{1/d}$;
3: Write $s_i = r_{ii} \cdot x_i$, with $x_i \in I_i$;
4: Define $I'_i = I_i \cdot \langle x_i \rangle^{-1}$ and $\mathbf{b}'_i = x_i \mathbf{b}_i$.
5: **end for**
6: **return** $((I'_i, \mathbf{b}'_i))_{i\leq n}$.

**Lemma 3.6.** *Algorithm 3.2 outputs a scaled pseudo-basis of the module $M$ generated by the input pseudo-basis and preserves the $\mathcal{N}(r_{ii} I_i)$'s. If $M \subseteq R^m$, then it runs in time polynomial in the input bit-length and in $\log \Delta_K$.*

Algorithm 3.3 aims at size-reducing a scaled pseudo-basis. It relies on a $\lfloor \cdot \rceil_R$ operator which takes as input a $y \in K_{\mathbb{R}}$ and rounds it to some $k \in R$ by writing $y = \sum y_i r_i$ for some $y_i$'s in $\mathbb{R}$, and rounding each $y_i$ to the nearest integer: $k = \sum k_i r_i = \sum \lfloor y_i \rceil r_i$ (remember that the $r_i$'s form an LLL-reduced basis of $R$). For computations, we will apply this operator numerically, so that we may not have $\max_i |k_i - y_i| \leq 1/2$ but, with a bounded precision computation, we can ensure that $\max_i |k_i - y_i| \leq 1$.

[6] Note that ideal scaling and size-reduction have been suggested in [FS10, Se. 4.1], but without a complexity analysis (polynomial complexity was claimed but not proved).

**Algorithm 3.3** Size-reduction.

**Input:** A scaled pseudo-basis $((I_i, \mathbf{b}_i))_{i \leq n}$ of a module $M$.
**Output:** A size-reduced pseudo-basis of $M$.

1: **for** $j = 1$ **to** $n$ **do**
2: **for** $i = j - 1$ **to** $1$ **do**
3: Compute $x_i = \lfloor r_{ij}/r_{ii} \rceil_R$;
4: $\mathbf{b}_j := \mathbf{b}_j - x_i \mathbf{b}_i$.
5: **end for**
6: **end for**
7: **return** $((I_i, \mathbf{b}_i))_{i \leq n}$.

**Lemma 3.7.** *Algorithm 3.3 outputs a scaled size-reduced pseudo-basis of the module $M$ generated by the input pseudo-basis and preserves the $\mathcal{N}(r_{ii} I_i)$'s. If $M \subseteq R^m$, then it runs in time polynomial in the input bit-length and in* $\log \Delta_K$.

We now consider Algorithm 3.4, which is a variant of Algorithm 3.1 that allows us to prove a bound on the bit cost. The only difference (Step 7) is that we call Algorithms 3.2 and 3.3 at every loop iteration of Algorithm 3.1, so that we are able to master the bit-lengths during the execution. Without loss of generality, we can assume that the pseudo-basis given as input is scaled and size-reduced: if it is not the case, we can call Algorithms 3.2 and 3.3, which will produce a pseudo-basis of the same module, whose bit-length is polynomial in the input bit-length and in $\log \Delta_K$.

**Algorithm 3.4** LLL-reduction over $K$ with controlled bit-lengths

**Input:** A scaled size-reduced pseudo-basis $((I_i, \mathbf{b}_i))_{i \leq n}$ of a module $M \subseteq R^m$.
**Output:** An LLL-reduced pseudo-basis of $M$.

1: **while** there exists $i < n$ such that $\alpha_K \cdot \mathcal{N}(r_{i+1,i+1} I_{i+1}) < \mathcal{N}(r_{i,i} I_i)$ **do**
2: Let $M_i$ be the rank-2 module spanned by the pseudo-basis $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$, with $\mathbf{a}_i = (r_{ii}, 0)^T$ and $\mathbf{a}_{i+1} = (r_{i,i+1}, r_{i+1,i+1})^T$;
3: Find $\mathbf{s}_i \in M_i \setminus \{\mathbf{0}\}$ such that $\mathcal{N}(\mathbf{s}_i) \leq \gamma^d \cdot \lambda_1^{\mathcal{N}}(M_i)$;
4: Set $\mathbf{s}_{i+1} = \mathbf{a}_i$ if it is linearly independent with $\mathbf{s}_i$, and $\mathbf{s}_{i+1} = \mathbf{a}_{i+1}$ otherwise;
5: Call the algorithm of Lemma 2.5 with $((I_i, \mathbf{a}_i), (I_{i+1}, \mathbf{a}_{i+1}))$ and $(\mathbf{s}_i, \mathbf{s}_{i+1})$ as inputs, and let $((I'_i, \mathbf{a}'_i), (I'_{i+1}, \mathbf{a}'_{i+1}))$ denote the output;
6: Update $I_i := I'_i$, $I_{i+1} := I'_{i+1}$ and $[\mathbf{b}_i | \mathbf{b}_{i+1}] := [\mathbf{b}_i | \mathbf{b}_{i+1}] \cdot \mathbf{A}^{-1} \cdot \mathbf{A}'$ (where $\mathbf{A} = [\mathbf{a}_i | \mathbf{a}_{i+1}]$ and $\mathbf{A}' = [\mathbf{a}'_i | \mathbf{a}'_{i+1}]$);
7: Update the current pseudo-basis by applying Algorithm 3.2 and then Algorithm 3.3 to it.
8: **end while**
9: **return** $((I_i, \mathbf{b}_i))_{i \leq n}$.

**Theorem 3.8.** *Assume that Step 3 of Algorithm 3.4 is implemented with some algorithm $\mathcal{O}$ for some parameter $\gamma$. Assume that $\alpha_K > \gamma^{2d} 2^d \Delta_K$. Given as input a scaled and size-reduced pseudo-basis of a module $M \subseteq R^m$, Algorithm 3.4*

*outputs an* LLL*-reduced pseudo-basis of $M$ in time polynomial in the bit-length of the input pseudo-basis,* $\log \Delta_K$ *and* $1/\log(\alpha_K/(\gamma^{2d}2^d\Delta_K))$.

### 3.3 Finding short vectors for the Euclidean norm

By Lemma 3.2 and Theorem 3.8 with $\alpha_k = (1 + c/n) \cdot \gamma^{2d}2^d\Delta_K$ for a well-chosen constant $c$, Algorithm 3.4 may be interpreted as a reduction from finding a $2 \cdot (\gamma^{2d}2^d\Delta_K)^n$ approximation to a vector reaching $\lambda_1^{\mathcal{N}}$ in rank-$n$ modules, to finding a $\gamma^d$ approximation to a vector reaching $\lambda_1^{\mathcal{N}}$ in rank-2 modules.

By using Lemma 2.2, we can extend the above to the Euclidean norm instead of the algebraic norm.

**Theorem 3.9.** *Let $\gamma \geq 1$, assume that $\log \Delta_K$ is polynomially bounded, and assume that a $\mathbb{Z}$-basis of $R$ is known. Then there exists a polynomial-time reduction from solving $SVP_{\gamma'}$ in rank-$n$ modules (with respect to $\|\cdot\|$) to solving $SVP_\gamma$ in rank-2 modules, where $\gamma' = (2\gamma\Delta_K^{1/d})^{2n-1}$.*

*Proof.* The reduction consists in first using Algorithm 3.4 with Step 3 implemented using the oracle solving $\mathrm{SVP}_\gamma$ in rank-2 modules. Using the arithmetic-geometric inequality and Lemma 2.2, one can see that a vector $\mathbf{s}$ satisfying $\|\mathbf{s}\| \leq \gamma \cdot \lambda_1(M)$ also satisfies $\mathcal{N}(\mathbf{s}) \leq \gamma^d \cdot \Delta_K^{1/2} \cdot \lambda_1^{\mathcal{N}}(M)$. Hence, we have an oracle computing a $\gamma_{\mathcal{N}} = \gamma \cdot \Delta_K^{1/(2d)}$ approximation of $\lambda_1^{\mathcal{N}}(M)$. We then run Algorithm 3.4 with this oracle by setting the parameter $\alpha_K$ to $(1+c/n)\cdot\gamma^{2d}2^d\Delta_K^2$, where $c$ is a constant such that $(1 + c/n)^{n-1} \leq 2$.

By Theorem 3.8, the reduction runs in in polynomial time. Further, by Lemma 3.2, the output pseudo-basis satisfies $\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq \alpha_K^{n-1} \cdot \lambda_1^{\mathcal{N}}(M)$. By Lemma 2.2 and by definition of $\alpha_K$, this gives:

$$\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1) \leq 2(\gamma^{2d}2^d\Delta_K^2)^{n-1} \cdot d^{-d/2}\lambda_1(M)^d.$$

Now, an $\mathrm{SVP}_\gamma$ solver for rank-2 modules directly provides an $\mathrm{SVP}_\gamma$ solver for rank-1 module. We hence use our oracle again, on $I_1\mathbf{b}_1$. This provides a non-zero vector $\mathbf{s} \in I_1\mathbf{b}_1 \subseteq M$ such that $\|\mathbf{s}\| \leq \gamma\sqrt{d}\Delta_K^{1/(2d)} \cdot (\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1))^{1/d}$, by Minkowski's theorem. Combining the latter with the above upper bound on $\mathcal{N}(I_1)\mathcal{N}(\mathbf{b}_1)$ provides the result. □

## 4 The divide-and-swap algorithm

We now focus on how to implement Step 3 of Algorithm 3.1, using a CVP oracle for a lattice depending on $K$ only. To handle projected 2-dimensional lattices, the LLL algorithm for integer lattices proceeds like the Gauss/Lagrange reduction algorithm for 2-dimensional lattices. It relies on a divide-and-swap elementary procedure: first shorten the second vector using a $\mathbb{Z}$-multiple of the first one (using a Euclidean division, or, more pedantically, a CVP solver for the trivial lattice $\mathbb{Z}$); then swap these two vectors if the second has become (significantly)

shorter than the first one. It has the effect that if this 2-dimensional basis is not reduced, then a swap occurs, and some progress is made towards reducedness of the 2-dimensional basis. This elementary step is repeated as many times as needed to achieve reduction of the lattice under scope. In this section, we generalize this process to rank-2 modules.

We first describe a lattice $L$ that depends only on $K$ and for which we will assume that we possess a CVP oracle. Then, we give an algorithm whose objective is to act as a Euclidean algorithm, i.e., enabling us to shorten an element of $K_{\mathbb{R}}$ using $R$-multiples of another. Once we have this generalization of the Euclidean algorithm, we finally describe a divide-and-swap algorithm for rank-2 modules.

### 4.1 Extending the logarithm

The lattice $L$ is defined using (among others) the log-unit lattice $\Lambda$. However, the Log function does not suffice for our needs. In particular, for $a, b \in K_{\mathbb{R}}^{\times}$, the closeness between $a$ and $b$ is not necessarily implied by the closeness of $\operatorname{Log} a$ and $\operatorname{Log} b$, because Log does not take into account the complex arguments of the entries of the canonical embeddings of $a$ and $b$. However, we will need such a property to hold. For this purpose, we hence extend the Log function. For $x \in K_{\mathbb{R}}^{\times}$, we define $\overline{\operatorname{Log}}\, x := (\theta_1, \ldots, \theta_{r_1+r_2}, \log|\sigma_1(x)|, \ldots, \log|\sigma_d(x)|)^T$, where $\sigma_i(x) = |\sigma_i(x)| \cdot \mathrm{e}^{\mathrm{I}\theta_i}$ for all $i \leq r_1 + r_2$ and $I$ is a complex root of $x^2 + 1$. The $\overline{\operatorname{Log}}$ function takes values in $(\pi\mathbb{Z}/2\pi\mathbb{Z})^{r_1} \times (\mathbb{R}/(2\pi\mathbb{Z}))^{r_2} \times \mathbb{R}^d$.

**Lemma 4.1.** *For $x, y \in K_{\mathbb{R}}^{\times}$, we have:*

$$\|x - y\|_\infty \leq \left(\mathrm{e}^{\sqrt{2}\|\overline{\operatorname{Log}}\, x - \overline{\operatorname{Log}}\, y\|_\infty} - 1\right) \cdot \min(\|x\|_\infty, \|y\|_\infty).$$

Observe that for $t \leq (\ln 2)/\sqrt{2}$, we have $\mathrm{e}^{\sqrt{2}t} - 1 \leq 2\sqrt{2}t$.

### 4.2 The lattice $L$

Let $r = \mathrm{poly}(d)$ and $\beta > 0$ be some parameter to be chosen later. Let $\Lambda$ denote the log-unit lattice. Let $\mathfrak{B}_0 = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_{r_0}\}$ be a set of cardinality $r_0 \leq \log h_K$ of prime ideals generating $\mathcal{C}l_K$, with algebraic norms $\leq 2^{\delta_0}$, with $\delta_0 = O(\log\log|\Delta|)$. We will also consider another set $\mathfrak{B} = \{\mathfrak{q}_1, \ldots, \mathfrak{q}_r\}$ of cardinality $r$, containing prime ideals (not in $\mathfrak{B}_0$) of norms $\leq 2^{\delta}$, for some parameters $r$ and $\delta \leq \delta_0$ to be chosen later. We also ask that among these ideals $\mathfrak{q}_j$, at least half of them have an algebraic norm $\geq \sqrt{2^{\delta}}$. Because we want $r$ such ideals, we should make sure that the number of prime ideals of norm bounded by $2^{\delta}$ in $R$ is larger than $r$. This will asymptotically be satisfied if $r \leq O(2^{\delta}/\delta)$ (by Lemma 2.1). The constraint that at least $r/2$ ideals should have norm larger than $\sqrt{2^{\delta}}$ is not very limiting, as we expect that roughly $2^{\delta} - \sqrt{2^{\delta}} \geq r - \sqrt{r}$ ideals should have algebraic norm between $\sqrt{2^{\delta}}$ and $2^{\delta}$ (forgetting about the $\mathrm{poly}(\delta)$ terms).

We now define $L$ as the lattice of dimension $\nu = 2(r_1 + r_2) + r_0 + r - 1$ (included in $\mathbb{R}^{\nu+1}$) spanned by the columns of the following basis matrix:

$$
\mathbf{B}_L := \begin{array}{|c|c|c|c|}
\multicolumn{2}{c}{\overleftarrow{\qquad 2(r_1+r_2)-1 \qquad}} & \multicolumn{2}{c}{\overleftarrow{\qquad\qquad r_0 + r \qquad\qquad}} \\
\hline
\multicolumn{2}{|c|}{0} & \multicolumn{2}{c|}{\beta \cdot a_{g_{2(r_1+r_2)}} \cdots \quad \beta \cdot a_{g_\nu}} \\
\hline
\beta \cdot 2\pi & & \multicolumn{2}{c|}{} \\
\quad \ddots & \beta \cdot \theta_{g_{r_1+r_2+1}} & \multicolumn{2}{c|}{\cdots \qquad \beta \cdot \theta_{g_\nu}} \\
\beta \cdot 2\pi & & \multicolumn{2}{c|}{} \\
\hline
0 & \beta \cdot \mathbf{B}_\Lambda & \multicolumn{2}{c|}{\beta \cdot \mathbf{h}_{g_{2(r_1+r_2)}} \cdots \quad \beta \cdot \mathbf{h}_{g_\nu}} \\
\hline
\multicolumn{2}{|c|}{} & \mathbf{B}_{\Lambda_0} & \mathbf{w}_{\nu-r+1} \cdots \mathbf{w}_\nu \\
\multicolumn{2}{|c|}{0} & 0 & 1 \quad \\
\multicolumn{2}{|c|}{} & & \quad \ddots \\
\multicolumn{2}{|c|}{} & & \quad 1 \\
\hline
\end{array}
\begin{array}{l}
\updownarrow 1 \\
\\
r_1 + r_2 \\
\\
r_1 + r_2 - 1 \\
\\
r_0 \\
\\
r
\end{array}
$$

where

- $\mathbf{B}_\Lambda$ is a basis of $\Lambda$, and we let $(\mathbf{h}_i)_{r_1+r_2<i<2(r_1+r_2)}$ denote its columns;
- $\mathbf{B}_{\Lambda_0}$ is a basis of the lattice $\Lambda_0 := \{(x_i)_i \in \mathbb{Z}^{r_0} : \prod_i \mathfrak{p}_i^{x_i} \text{ is principal}\}$, and we let $(\mathbf{w}_i)_{2(r_1+r_2)\leq i\leq \nu-r}$ denote its columns;
- for any $g \in K$, we have $a_g = (\log|\mathcal{N}(g)|)/\sqrt{d}$ ;
- for any $g \in K$, the vector $\theta_g$ consists of the first $r_1 + r_2$ entries of $\overline{\mathrm{Log}}(g)$;
- for any $g \in K$, we have $\mathbf{h}_g = i_{H\cap E}(\Pi_H(\mathrm{Log}(g)))$, where $\Pi_H$ is the orthogonal projection on $H$ and $i_{H\cap E}$ is an isometry mapping $H \cap E$ to $\mathbb{R}^{r_1+r_2-1}$;
- for any $i > r_1 + r_2$, if we parse the bottom $r_0 + r$ coordinates of the $i$-th column vector as $(w_{i,1}, \ldots, w_{i,r_0}, w'_{i,1}, \ldots, w'_{i,r})$, then we have that $\langle g_i \rangle = \prod_j \mathfrak{p}_j^{w_{ij}} \cdot \prod_j \mathfrak{q}_j^{w'_{ij}}$;
- the $g_i$'s for $i > r_1 + r_2$ are in $K$ and, among them, $g_{r_1+r_2+1}, \ldots g_{2(r_1+r_2)-1}$ are the units of $R$ corresponding to the columns of $\mathbf{B}_\Lambda$.

We now list a few properties satisfied by vectors in this lattice.

**Lemma 4.2.** *For every vector $(\beta a \| \beta\theta \| \beta\mathbf{h} \| \mathbf{w} \| \mathbf{w}') \in L \setminus \{\mathbf{0}\}$ (with blocks of dimensions $1, r_1 + r_2, r_1 + r_2 - 1, r_0$ and $r$), there exists $g \in K \setminus \{0\}$ with*

- $a = (\log|\mathcal{N}(g)|)/\sqrt{d}$
- $\overline{\mathrm{Log}}(g) = (\theta' \| \mathrm{Log}(g))$ *with* $\theta' = \theta \bmod 2\pi$.
- $\mathbf{h} = i_{H\cap E}(\Pi_H(\mathrm{Log}(g)))$
- $\langle g \rangle = \prod_j \mathfrak{p}_j^{w_j} \prod_j \mathfrak{q}_j^{w'_j}$, *where* $\mathbf{w} = (w_1, \cdots, w_{r_0})$ *and* $\mathbf{w}' = (w'_1, \cdots, w'_r)$.

*Further, we have that* $\| \mathrm{Log}(g)\|_2 = \|(a, \mathbf{h})\|_2$.

### 4.3 On the distance of relevant vectors to the lattice

In this section, we make a heuristic assumption on the distance between target vectors of a specific form and the lattice $L$ defined in the previous section. This heuristic is backed with a counting argument and numerical experiments (see the full version). As $L$ is not full rank, we only consider target vectors $\mathbf{t}$ lying in the span of $L$. Also, as $\mathbf{B}_L$ contains the identity matrix in its bottom right corner, we cannot hope to have a covering radius that is much smaller than $\sqrt{r}$. In our case, the lattice dimension $\nu$ will be of the order of $r$, but in our application we will need a vector of $L$ much closer to $\mathbf{t}$ than $\sqrt{r} \approx \sqrt{\nu}$. In order to go below this value, we only consider target vectors $\mathbf{t}$ whose last $r$ coordinates are almost 0.

**Heuristic 1.** Assume that there exist some integer $B \leq r$ such that $B \geq 100 \cdot (\log h_K) \cdot \delta_0/\delta$ and that

$$\alpha_0 := \sqrt{2\pi} \left( \left(\frac{2B}{r^{0.96}}\right)^B \cdot \delta B (\det \Lambda) h_K \right)^{1/d} \leq \frac{\ln d}{12 d^{2.5}}.$$

Assume that the scaling parameter $\beta$ in $\mathbf{B}_L$ is set to $\frac{1}{\alpha_0}\sqrt{\frac{0.01 \cdot B}{2d}}$. Then for any $\mathbf{t} \in \mathrm{Span}(L)$ whose last $r$ coordinates $\mathbf{w}'_t$ satisfy $\|\mathbf{w}'_t\|_2 \leq 0.01 \cdot B/\sqrt{r}$, we have $\mathrm{dist}(\mathbf{t}, L) \leq \sqrt{1.05 \cdot B}$.

*Discussion about Heuristic 1.* We provide below a counting argument to justify Heuristic 1. We consider the following set of vectors of $L$, parametrized by $B \leq r$, which we view as candidates for very close vectors to such target vectors:

$$S_B := \{\mathbf{s} = (\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L : \mathbf{w}'_s \in \{-1, 0, 1\}^r \wedge \|\mathbf{w}'_s\|_1 = B\}.$$

We argue that there is a vector in $S_B$ that is very close to $\mathbf{t}$. Our analysis is heuristic, but we justify it with both mathematical arguments and experiments. We are going to examine the vectors $\mathbf{s} \in S_B$ such that $\mathbf{s} - \mathbf{t}$ is reduced modulo $L$. Let us write $\mathbf{t} = (\beta a_t \| \beta\theta_t \| \beta \mathbf{h}_t \| \mathbf{w}_t \| \mathbf{w}'_t)^T$. We define:

$$\begin{aligned} S^{(1)}_{B,\mathbf{t}} := \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L : \; & \mathbf{w}'_s \in \{-1, 0, 1\}^r \wedge \|\mathbf{w}'_s\|_1 = B, \\ & \mathbf{w}_s - \lfloor \mathbf{w}_t \rceil \in \mathcal{V}(\Lambda_0), \\ & \mathbf{h}_t - \mathbf{h}_s \in \mathcal{V}(\Lambda), \\ & \theta_t - \theta_s \in (-\pi, \pi]^{r_1 + r_2}\}, \end{aligned}$$

where the notation $\mathcal{V}$ refers to the Voronoi cell (i.e., the set of points which are closer to 0 than to any other point of the lattice). The choice of $\mathbf{w}'_s$ fully determines $\mathbf{s} \in S^{(1)}_{B,\mathbf{t}}$, which gives the bound $|S^{(1)}_{B,\mathbf{t}}| = 2^B \cdot \binom{r}{B} \geq (2r/B)^B$.

We consider the following subset of $S^{(1)}_{B,\mathbf{t}}$:

$$S^{(2)}_{B,\mathbf{t}} = S^{(1)}_{B,\mathbf{t}} \cap \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L : \mathbf{w}_s = \lfloor \mathbf{w}_t \rceil\}.$$

We heuristically assume that when we sample a uniform vector in $S^{(1)}_{B,\mathbf{t}}$, the components $\mathbf{w}_s$ of the vectors $\mathbf{s} \in S^{(1)}_{B,\mathbf{t}}$ are uniformly distributed modulo $\Lambda_0$.

Then the proportion of those for which $\mathbf{w}_s = \lfloor \mathbf{w}_t \rceil \bmod \Lambda_0$ is $1/\det(\Lambda_0) = 1/h_K$. Hence, we expect that $|S^{(2)}_{B,\mathbf{t}}| \approx |S^{(1)}_{B,\mathbf{t}}|/h_K$.

We consider the following subset of $S^{(2)}_{B,\mathbf{t}}$, parametrized by $\alpha \leq (\ln d)/(12d^{2.5})$:

$$S^{(3)}_{B,\alpha,\mathbf{t}} = S^{(2)}_{B,\mathbf{t}} \cap \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L : \|(\theta_s \| \mathbf{h}_s) - (\theta_t \| \mathbf{h}_t)\|_\infty \leq \alpha\}.$$

We heuristically assume that when we sample a uniform vector in $S^{(2)}_{B,\mathbf{t}}$, the components $(\theta_s, \mathbf{h}_s)$ are uniformly distributed modulo $2\pi\mathbb{Z}^{r_1+r_2} \times \Lambda$. Observe that the first $r_1$ coordinates of $\theta_s$ (corresponding to real embeddings) are either 0 or $\pi$. Hence, the probability that $\theta_s = \theta_t$ on these coordinates is $2^{-r_1}$. Once these first $r_1$ coordinates are fixed, the remaining coordinates of $(\theta_s, \mathbf{h}_s)$ have no a priori reason to be bound to a sublattice of $2\pi\mathbb{Z}^{r_2} \times \Lambda$ and we heuristically assume them to be uniformly distributed in $\mathbb{R}^{r_1+2r_2-1}/(2\pi\mathbb{Z}^{r_2} \times \Lambda)$. Overall, the probability that a vector $\mathbf{s} \in S^{(2)}_{B,\mathbf{t}}$ satisfies $\|(\theta_s, \mathbf{h}_s) - (\theta_t, \mathbf{h}_t)\|_\infty \leq \alpha$ is $\approx \frac{\alpha^{r_1+2r_2-1}}{2^{r_1} \cdot (2\pi)^{r_2} \cdot \det(\Lambda)}$. Here, we used the fact that $\sqrt{r_1 + 2r_2 - 1} \cdot \alpha$ is smaller than $\lambda_1(2\pi\mathbb{Z}^{r_2} \times \Lambda)/2$ (recall from preliminaries that $\lambda_1^{(\infty)}(\Lambda) \geq (\ln d)/(6d^2)$). We conclude that

$$|S^{(3)}_{B,\alpha,\mathbf{t}}| \approx |S^{(2)}_{B,\mathbf{t}}| \frac{\alpha^{r_1+2r_2-1}}{2^{r_1} \cdot (2\pi)^{r_2} \cdot \det(\Lambda)} \geq |S^{(2)}_{B,\mathbf{t}}| \frac{\alpha^{d-1}}{(2\pi)^{d/2} \cdot \det(\Lambda)}.$$

Finally, we consider the following subset of $S^{(3)}_{B,\alpha,\mathbf{t}}$:

$$S^{(4)}_{B,\alpha,\mathbf{t}} = S^{(3)}_{B,\alpha,\mathbf{t}} \cap \{(\beta a_s \| \beta\theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s) \in L : |a_s - a_t| \leq \alpha\}.$$

We will assume that

$$|S^{(4)}_{B,\alpha,\mathbf{t}}| \geq \frac{0.344^B \cdot \alpha\sqrt{d}}{\delta B \cdot r^{0.04 \cdot B}} \cdot |S^{(3)}_{B,\alpha,\mathbf{t}}|.$$

This assumption is backed with mathematical arguments as well as numerical experiments in the full version. Overall, we obtain that

$$\begin{aligned}
|S^{(4)}_{B,\alpha,\mathbf{t}}| &\geq \frac{0.344^B \cdot \alpha\sqrt{d}}{\delta B \cdot r^{0.04 \cdot B}} \cdot \frac{\alpha^{d-1}}{(2\pi)^{d/2} \cdot \det(\Lambda)} \cdot \frac{1}{h_K} \cdot \left(\frac{2r}{B}\right)^B \\
&\geq \left(\frac{\alpha}{\sqrt{2\pi}}\right)^d \frac{1}{\delta B \cdot \det(\Lambda) \cdot h_K} \left(\frac{0.344 \cdot 2r}{B \cdot r^{0.04}}\right)^B \\
&\geq \left(\frac{\alpha}{\sqrt{2\pi}}\right)^d \frac{1}{\delta B \cdot \det(\Lambda) \cdot h_K} \left(\frac{r^{0.96}}{2B}\right)^B.
\end{aligned}$$

When the above is $\geq 1$, we expect that there exists $\mathbf{s} \in S^{(4)}_{B,\alpha,\mathbf{t}}$. If that is the case, then we have

$$\|\mathbf{s} - \mathbf{t}\|^2 \leq (\beta \cdot \sqrt{2d} \cdot \alpha)^2 + r_0 + \|\mathbf{w}'_t - \mathbf{w}'_s\|^2.$$

By condition on $B$, we know that $r_0 \leq 0.01 \cdot B$. Also, by choice of $\mathbf{w}'_t$ (and using the fact that $r \geq B$), we have that $\|\mathbf{w}'_t - \mathbf{w}'_s\|^2 \leq (\sqrt{B} + 0.01 \cdot \sqrt{B})^2 \leq 1.03 \cdot B$. Finally, choosing $\alpha$ minimal provides the result.

*Numerical experiments.* Heuristic 1 is also backed with numerical experiments. We performed the experiments with $r$ of the order of $d^2$ (looking forward, this is the value of $r$ that will be used by our algorithm). This means that our lattice $L$ has dimension roughly $d^2$, and so solving CVP in it quickly becomes impractical. We were still able to check that our heuristic seems correct in cyclotomic number fields of very small degree (up to $d = 8$). More details on these numerical experiments can be found in the full version.

### 4.4 A "Euclidean division" over $R$

We will need the following technical observation that, given $a, b \in K_{\mathbb{R}}$, it is possible to add a small multiple $ka$ of $a$ to $b$ to ensure that $\mathcal{N}(b + ka) \geq \mathcal{N}(a)$.

**Lemma 4.3.** *For any $a \in K_{\mathbb{R}}^{\times}$ and $b \in K_{\mathbb{R}}$, there exists $k \in [-d, d] \cap \mathbb{Z}$ such that $|\mathcal{N}(b + ka)| \geq |\mathcal{N}(a)|$.*

Note that an integer $k$ such as in Lemma 4.3 can be found efficiently by exhaustive search.

We can now describe our "Euclidean division" algorithm over $R$. Our algorithm takes as input a fractional ideal $\mathfrak{a}$ and two elements $a, b \in K_{\mathbb{R}}$, and outputs a pair $(u, v) \in R \times \mathfrak{a}$. The first five steps of this algorithm aim at obtaining, for any input $(a, b)$, a replacement $(a_1, b_1)$ that satisfies some conditions. Namely, we would like $a_1$ to be balanced, i.e., $\|a_1\|$ should not be significantly more than $\mathcal{N}(a_1)^{1/d}$. We also would like $b_1$ to be not much larger that $a_1$ and $\mathcal{N}(a_1/b_1)$ to be close to 1. These conditions are obtained by multiplying the element $a$ by an appropriate element of $R$, and removing a multiple of $a$ from $b$. Note that we require that the output element $v$ should not be too large. As $b$ is not multiplied by anything, these normalization steps will not impact this output property. After these first five steps, the core of the algorithm begins. It mainly consists in the creation of a good target vector $\mathbf{t}$ in $\mathbb{R}^{\nu+1}$, followed by a CVP computation in the lattice $L$.

**Theorem 4.4 (Heuristic).** *Assume that $\mathfrak{a}$ satisfies $\mathrm{c}^{-d} \leq \mathcal{N}(\mathfrak{a}) \leq \mathrm{c}^{d}$, with $\mathrm{c}$ as in Lemma 2.3. Assume also that $B$ and $r$ are chosen so that*

$$B \geq \max\left(100 \cdot d \cdot \log[(\rho(R) + d)c^4], \log h_K \cdot (103 \cdot \frac{\delta_0}{\delta})^2\right),$$
$$\alpha_0 := \sqrt{2\pi}\left((\frac{2B}{r^{0.96}})^B \cdot \delta B(\det \Lambda) h_K\right)^{1/d} \leq \frac{\varepsilon}{43 \cdot \sqrt{d} \cdot (\rho(R) + d)\mathrm{c}^4 \cdot 2^{0.55 \cdot \delta \cdot B/d}},$$

*for some $\varepsilon > 0$. Assume also that $\alpha_0 \leq (\ln d)/(12d^{2.5})$, and set the scaling parameter $\beta$ of $\mathbf{B}_L$ as in Heuristic 1. Then, under Heuristic 1 and the heuristics of Lemma 2.3, Algorithm 4.1 outputs a pair $(u, v) \in R \times \mathfrak{a}$ with*

$$\|ua + vb\|_\infty \leq \varepsilon \cdot \|a\|_\infty,$$
$$\|v\|_\infty \leq \mathrm{c} \cdot 2^{0.55 \cdot \delta \cdot B/d}.$$

**Algorithm 4.1** A Euclidean division over $R$

**Input:** A fractional ideal $\mathfrak{a}$, and two elements $a \in K_{\mathbb{R}}^{\times}$ and $b \in K_{\mathbb{R}}$.
**Output:** A pair $(u, v) \in R \times \mathfrak{a}$.

*Computing a better pair* $(a_1, b_1)$
1: Find $s \in \mathfrak{a}^{-1} \setminus \{0\}$ such that $\|s\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(\mathfrak{a}^{-1})^{1/d}$ as in Lemma 2.3.
2: Find $y \in R \setminus \{0\}$ such that $\|ya\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(a)|^{1/d}$ as in Lemma 2.3 (with ideal $\langle a \rangle$). Define $a_1 = ya$.
3: Solve CVP in $R$ to find $x \in R$ such that $\|b/(s \cdot a_1) - x\| \leq \rho(R)$.
4: Find $k \in \mathbb{Z} \cap [-d, d]$ such that $|\mathcal{N}(b - xsa_1 + ksa_1)| \geq |\mathcal{N}(sa_1)|$ (see Lemma 4.3).
5: Define $b_1 = b + (k - x)s \cdot a_1$.

*Defining the target vector and solving CVP*
6: Compute $(w_{t,j})_{j \leq r_0}$ and $g_t$ such that $\mathfrak{a}^{-1} = \prod_j \mathfrak{p}_j^{w_{t,j}} \langle g_t \rangle$. Let $\mathbf{w}_t = (w_{t,j})_{j \leq r_0}$.
7: Let $a_t = (\log \mathcal{N}|b_1/(a_1 g_t)|)/\sqrt{d}$, $\theta_t$ be the first $r_1 + r_2$ coordinates of $\overline{\mathrm{Log}}(b_1/(a_1 g_t))$ and $\mathbf{h}_t = i_{E \cap H}(\Pi_H(\mathrm{Log}(b_1/(a_1 g_t))))$.
8: Define $\mathbf{t} = (\beta a_t \| \beta \theta_t \| \beta \mathbf{h}_t \| \mathbf{w}_t \| \mathbf{0})$.
9: Solve CVP in $L$ with target vector $\mathbf{t}$, to obtain a vector $\mathbf{s}$.

*Using* $\mathbf{s}$ *to create a good ring element*
10: Write $\mathbf{s} = (\beta a_s \| \beta \theta_s \| \beta \mathbf{h}_s \| \mathbf{w}_s \| \mathbf{w}'_s)$ and let $g_s \in K^*$ be the associated element as in Lemma 4.2.
11: Define the ideal $I = \mathfrak{a} \prod_{j : w_{s,j} - w_{t,j} < 0} \mathfrak{p}_j^{w_{t,j} - w_{s,j}} \prod_{j : w'_{s,j} < 0} \mathfrak{q}_j^{-w'_{s,j}}$.
12: Find $v \in I \setminus 0$ such that $\|v\|_\infty \leq c \cdot \mathcal{N}(I)^{1/d}$ as in Lemma 2.3.
13: Define $u' = g_s \cdot g_t \cdot v$.
14: **return** $(u'y + (k - x)sy \cdot v, v)$.

*Apart from the CVP calls in $R, L_K$ and $L$, Algorithm 4.1 runs in quantum polynomial time.*

*Proof.* Throughout the proof, we keep the notations of Algorithm 4.1.

We first prove that $(u, v) \in R \times \mathfrak{a}$. As $s \in \mathfrak{a}^{-1}$ and $x, k, y \in R$, it suffices to prove that $(u', v) \in R \times \mathfrak{a}$. By definition of $g_t$ and $g_s$, we have

$$\langle g_s g_t \rangle = \mathfrak{a}^{-1} \prod_j \mathfrak{p}_j^{w_{s,j} - w_{t,j}} \prod_j \mathfrak{q}_j^{w'_{s,j}} = J \cdot I^{-1},$$

with $J = \prod_{j : w_{s,j} - w_{t,j} > 0} \mathfrak{p}_j^{w_{s,j} - w_{t,j}} \prod_{j : w'_{s,j} > 0} \mathfrak{q}_j^{w'_{s,j}}$. As the $\mathfrak{p}_j$'s and $\mathfrak{q}_j$'s are integral ideals, we see that $J \subseteq R$ and $I \subseteq \mathfrak{a}$. As $v \in I$, we obtain that $v \in \mathfrak{a}$. Since $g_s \cdot g_t \in JI^{-1}$ and $v \in I$, we also have $u' = g_s g_t v \in JI^{-1} I = J \subseteq R$. This gives our first claim.

As a preliminary step towards bounding $\|ua + bv\|_\infty = \|u'a_1 + vb_1\|_\infty$, we study the sizes of $a_1$ and $b_1$. Using the equality $b_1 = b - xsa_1 + ksa_1$, we have

$$\|b_1\|_\infty \leq (\|b/(sa_1) - x\|_\infty + |k|) \cdot \|sa_1\|_\infty \leq (\rho(R) + d) \cdot \|s\|_\infty \cdot \|a_1\|_\infty.$$

By definition of $a_1$, we have $\|a_1\|_\infty \leq \mathrm{c}\|a\|_\infty$. By assumption on $\mathfrak{a}$, we also have $\|s\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(\mathfrak{a}^{-1})^{1/d} \leq \mathrm{c}^2$. Hence, we obtain

$$\|b_1\|_\infty \leq (\rho(R) + d)\mathrm{c}^3 \|a\|_\infty.$$

Now, by definition of $a_1$, we know that $\|a_1\|_\infty \leq \mathrm{c} \cdot |\mathcal{N}(a_1)|^{1/d}$. Hence, we obtain

$$\mathrm{c}^{-1} \leq |\mathcal{N}(b_1/a_1)|^{1/d} \leq \mathrm{c} \cdot \frac{\|b_1\|_\infty}{\|a_1\|_\infty} \leq (\rho(R) + d) \cdot \mathrm{c}^3.$$

The left inequality is provided by the choice of $k$ at Step 4 (and the fact that $\mathcal{N}(s) \geq \mathcal{N}(\mathfrak{a}^{-1})$).

To bound $\|u'a_1 + vb_1\|_\infty$, we estimate the closeness of $\mathbf{t}$ and $\mathbf{s}$. If $\mathbf{t}$ was in $\mathrm{Span}(L)$, then we could apply Heuristic 1. As this is not necessarily the case, we first need to compute the distance between $\mathbf{t}$ and $\mathrm{Span}(L)$. This is done in the proof of the following lemma, which is provided in the full version.

**Lemma 4.5 (Heuristic).** *Under the assumptions of Theorem 4.4, we have* $\|\mathbf{s} - \mathbf{t}\|_2 \leq \sqrt{1.06 \cdot B}$.

This lemma implies that

$$\|(a_s\|\theta_s\|\mathbf{h}_s) - (a_t\|\theta_t\|\mathbf{h}_t)\|_2 \leq \sqrt{1.06 \cdot B}/\beta \leq 15 \cdot \sqrt{d} \cdot \alpha_0.$$

By definition of $\mathbf{t}$ and construction of $L$, this means that

$$\|\overline{\mathrm{Log}}(g_t g_s \cdot a_1/b_1)\|_2 = \|(a_s\|\theta_s\|\mathbf{h}_s) - (a_t\|\theta_t\|\mathbf{h}_t)\|_2 \leq 15 \cdot \sqrt{d} \cdot \alpha_0.$$

Recall that $u'/v = g_t g_s$. Hence we have $\|\overline{\mathrm{Log}}(u'a_1) - \overline{\mathrm{Log}}(vb_1)\|_\infty \leq 15 \cdot \sqrt{d} \cdot \alpha_0$. Using Lemma 4.1, we deduce that

$$\begin{aligned}\|u'a_1 - vb_1\|_\infty &\leq (\mathrm{e}^{15 \cdot \sqrt{2d} \cdot \alpha_0} - 1) \cdot \|b_1\|_\infty \cdot \|v\|_\infty \\ &\leq 43 \cdot \sqrt{d} \cdot \alpha_0 \cdot \|b_1\|_\infty \cdot \|v\|_\infty,\end{aligned}$$

where we used the fact that $\alpha_0 \leq (\ln d)/(12d^{2.5})$ and so the exponent should be smaller than $(\ln 2)/\sqrt{2}$ for $d$ large enough. We have already bounded $\|b_1\|_\infty$. We now bound $\|v\|_\infty$. By definition of $v$, we have $\|v\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(I)^{1/d}$. The task is then to provide an upper bound on $\mathcal{N}(I)$. As $IJ = \mathfrak{a} \cdot \prod_j \mathfrak{p}_j^{|w_{s,j} - w_{t,j}|} \cdot \prod_j \mathfrak{q}_j^{|w'_{s,j}|}$, we have:

$$\begin{aligned}\log \mathcal{N}(IJ) &= \log \mathcal{N}(\mathfrak{a}) + \sum_j |w_{s,j} - w_{t,j}| \log \mathcal{N}(\mathfrak{p}_j) + \sum_j |w'_{s,j}| \cdot \log \mathcal{N}(\mathfrak{q}_j) \\ &\leq \log \mathcal{N}(\mathfrak{a}) + \|\mathbf{w}_s - \mathbf{w}_t\|_1 \cdot \delta_0 + \|\mathbf{w}'_s\|_1 \cdot \delta\end{aligned}$$

Recall from Lemma 4.5 that we have $\|\mathbf{s} - \mathbf{t}\|_2 \leq \sqrt{1.06 \cdot B}$. This implies that $\|\mathbf{w}_s - \mathbf{w}_t\|_2, \|\mathbf{w}'_s\|_2 \leq \sqrt{1.06 \cdot B}$. Note that

$$\|\mathbf{w}_s - \mathbf{w}_t\|_1 \leq \sqrt{r_0} \cdot \|\mathbf{w}_s - \mathbf{w}_t\|_2 \leq 1.03 \cdot \sqrt{B \cdot r_0} \leq 0.01 \cdot \frac{\delta}{\delta_0} \cdot B,$$

by assumption on $B$ and the fact that $r_0 \leq \log h_K$. For $\mathbf{w}'_s$, we use the fact that it has integer coordinates, to obtain $\|\mathbf{w}'_s\|_1 \leq \|\mathbf{w}'_s\|_2^2 \leq 1.06 \cdot B$. We thus obtain

$$\log \mathcal{N}(IJ) \leq \log \mathcal{N}(\mathfrak{a}) + 1.07 \cdot \delta \cdot B.$$

As $J$ is integral, this gives an upper bound on $\mathcal{N}(I)$. However this upper bound is not sufficient for our purposes. We improve it by giving an upper bound on $\log \mathcal{N}(IJ^{-1})$, using the fact that the ideal $IJ^{-1}$ is designed to have an algebraic norm close to the one of $a_1/b_1$. Recall that $a_1$ and $b_1$ are constructed so that $\mathcal{N}(a_1/b_1)$ is close to 1, which means that $I$ and $J$ should have roughly the same norm. More precisely, it is worth recalling that $I^{-1}J = \langle g_s g_t \rangle$, and that $\|\overline{\mathrm{Log}}(g_t g_s \cdot a_1/b_1)\|_2 \leq 15 \cdot \sqrt{d} \cdot \alpha_0$. Looking at the first coordinate of the $\overline{\mathrm{Log}}$ vector and multiplying it by $\sqrt{d}$ shows that $|\log|\mathcal{N}(g_s g_t)| + \log|\mathcal{N}(a_1/b_1)|| \leq 15 \cdot d \cdot \alpha_0$. This gives us

$$\log \mathcal{N}(IJ^{-1}) \leq |\log|\mathcal{N}(a_1/b_1)|| + 15 \cdot d \cdot \alpha_0$$

Combining the bounds on $\log \mathcal{N}(IJ)$ and $\log \mathcal{N}(IJ^{-1})$, we finally obtain that

$$\log \mathcal{N}(I) \leq \frac{1}{2} \cdot |\log \mathcal{N}(\mathfrak{a})| + 0.535 \cdot \delta \cdot B + \frac{1}{2} \cdot |\log|\mathcal{N}(a_1/b_1)|| + 7.5 \cdot d \cdot \alpha_0.$$

We have seen that $\mathrm{c}^{-1} \leq |\mathcal{N}(b_1/a_1)|^{1/d} \leq (\rho(R) + d) \cdot \mathrm{c}^3$. Finally, recall that $\mathrm{c}^{-d} \leq \mathcal{N}(\mathfrak{a}) \leq \mathrm{c}^d$. Hence, we conclude that $|\log|\mathcal{N}(a_1/b_1)|| + |\log \mathcal{N}(\mathfrak{a})| \leq d \cdot \log((\rho(R) + d) \cdot \mathrm{c}^4) \leq 0.01 \cdot B$ by assumption on $B$. Recall that we assumed that $\alpha_0 \leq (\ln d)/(12d^{2.5}) \leq 1/d$. Hence, we have $d \cdot \alpha_0 \leq 1$. Using the fact that $B \geq 750$ (which is implied by the second term in the max), we obtain $7.5 \cdot d \cdot \alpha_0 \leq 0.01 \cdot B$. We conclude that

$$\log \mathcal{N}(I) \leq 0.55 \cdot \delta \cdot B.$$

Collecting terms and using the assumptions, this allows us to write

$$\begin{aligned} \|u'a_1 - vb_1\|_\infty &\leq 43 \cdot \sqrt{d} \cdot \alpha_0 \cdot \|b_1\|_\infty \cdot \|v\|_\infty \\ &\leq \alpha_0 \cdot 43 \cdot \sqrt{d} \cdot 2^{0.55 \cdot \delta \cdot B/d} \cdot (\rho(R) + d)\mathrm{c}^4 \|a\|_\infty \\ &\leq \varepsilon \cdot \|a\|_\infty. \end{aligned}$$

Finally, the run-time bound follows by inspection. □

We observe that the parameters $r$ and $B$ of Theorem 4.4 can be instantiated as $B = \widetilde{O}(\log|\Delta| + d \log \rho(R))$ and $r^{0.96} = \Theta((1/\varepsilon)^{d/B} \cdot B \cdot 2^{0.55\delta})$. Thanks to the 0.55 in the exponent, this choice of $r$ is compatible with the condition $r \leq O(2^\delta/\delta)$ which was required for the construction of the lattice $L$ (recall that we want $r$ prime ideals of norm smaller than $2^\delta$). We note also that the constants 0.96 and 0.55 appearing in the exponent can be chosen as close as we want to 1 and 0.5 respectively, by adapting the argument above. Hence, assuming $(1/\varepsilon)^{d/B} = O(1)$, we expect to be able to choose $2^\delta$ as small as $B^{2+\eta}$ for any $\eta > 0$. Overall, the following corollary gives an instantiation of Theorem 4.4 with parameters that are relevant to our upcoming divide-and-swap algorithm.

**Corollary 4.6 (Heuristic).** *Let $\varepsilon = 1/2^{\widetilde{O}(\log \Delta_K)/d}$. For any $\eta > 0$, there exists a lattice $L'$ of dimension $\widetilde{O}((\log|\Delta_K| + d\log\rho(R))^{2+\eta})$, an upper bound $C = 2^{\widetilde{O}(\log|\Delta_K|+d\log\rho(R))/d}$ and an algorithm $\mathcal{A}$ that achieve the following. Under Heuristic 1 and the heuristics of Lemma 2.3, algorithm $\mathcal{A}$ takes as inputs $a \in K_{\mathbb{R}}^{\times}$, $b \in K_{\mathbb{R}}$ and an ideal $\mathfrak{a}$ satisfying $\mathrm{c}^{-d} \leq \mathcal{N}(\mathfrak{a}) \leq \mathrm{c}^{d}$, and outputs $u, v \in R \times \mathfrak{a}$ such that*

$$\|ua + bv\|_\infty \leq \varepsilon \cdot \|a\|_\infty$$
$$\|v\|_\infty \leq C.$$

*If given access to an oracle solving the closest vector problem in $L'$ in polynomial time, and when restricted to inputs $a, b$ belonging to $K$, Algorithm 4.1 runs in quantum polynomial time.*

### 4.5 The divide-and-swap algorithm

In this subsection, we describe a divide-and-swap algorithm, which takes as input a pseudo-basis of a rank-2 module and outputs a short non-zero vector of this module (for the algebraic norm). In order to do so, we will need to link the Euclidean and algebraic norms of vectors appearing during the execution, and limit the degree of freedom of the ideal coefficients. For this purpose we use a strengthening of the notion of scaled pseudo-bases from Section 3.2.

**Definition 4.7.** *A pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$, with $I_i \subset K$ and $\mathbf{b}_i \in K_{\mathbb{R}}^m$ for all $i \leq n$, is said strongly scaled if, for all $i \leq n$,*

$$R \subseteq I_i, \quad \mathcal{N}(I_i) \geq \mathrm{c}^{-d} \text{ and } \|r_{ii}\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(r_{ii}I_i)^{1/d},$$

*where* c *is as in Lemma 2.3.*

Algorithm 4.2 below strongly scales a given module pseudo-basis. It is a direct adaptation of Algorithm 3.2 in which the LLL algorithm is replaced by the algorithm from Lemma 2.3 (relying on a CVP oracle for $L_K$).

**Algorithm 4.2** Strongly scaling the ideals.

**Input:** A pseudo-basis $((I_i, \mathbf{b}_i))_{i\leq n}$ of a module $M$.
**Output:** A strongly scaled pseudo-basis $((I'_i, \mathbf{b}'_i))_{i\leq n}$ of $M$.

1: **for** $i = 1$ **to** $n$ **do**
2: Use Lemma 2.3 to find $s_i \in r_{ii} \cdot I_i \setminus \{0\}$ such that $\|s_i\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(r_{ii}I_i)^{1/d}$;
3: Write $s_i = r_{ii} \cdot x_i$, with $x_i \in I_i$;
4: Define $I'_i = I_i \cdot \langle x_i \rangle^{-1}$ and $\mathbf{b}'_i = x_i \mathbf{b}_i$.
5: **end for**
6: **return** $((I'_i, \mathbf{b}'_i))_{i\leq n}$.

**Lemma 4.8.** *Algorithm 4.2 outputs a strongly scaled pseudo-basis of the module $M$ generated by the input pseudo-basis and preserves the $\mathcal{N}(r_{ii}I_i)$'s. If given access to an oracle that solves CVP in the lattice $L_K$ of Lemma 2.3, and if $M \subseteq R^m$, then it runs quantumly in time polynomial in the input bit-length and in $\log \Delta_K$.*

We can now describe Algorithm 4.3, our divide-and-swap algorithm. During the execution of the algorithm, the R-factor of the current matrix $(\mathbf{b}_1|\mathbf{b}_2)$ is always computed. The algorithm is very similar to the LLL algorithm in dimension 2, except for Step 4, which is specific to this algorithm. This step ensures that when we swap the vectors, we still obtain a pseudo-basis of the input module. This seems necessary, as our Euclidean division over $R$ involves a multiplication of the second vector by a ring element, and hence the new vector and the second pseudo-basis vector may not span the whole module anymore. At Step 4, note that the gcd is well-defined, as $\langle u \rangle$ and $\langle v \rangle \mathfrak{a}^{-1}$ are integral ideals. As an alternative to Step 4, we could use Lemma 2.5 as in Algorithm 3.1.

**Algorithm 4.3** Divide-and-swap.

**Input:** A pseudo-basis $((\mathfrak{a}_1, \mathbf{b}_1), (\mathfrak{a}_2, \mathbf{b}_2))$ of a module $M \subset K_{\mathbb{R}}^2$.
**Output:** A vector $\mathbf{v} \in M$.

1: **while** $(\gamma/\mathrm{c})^d \mathcal{N}(r_{22}\mathfrak{a}_2) < \mathcal{N}(r_{11}\mathfrak{a}_1)$ **do**
2: Strongly scale the pseudo-basis $((\mathfrak{a}_1, \mathbf{b}_1), (\mathfrak{a}_2, \mathbf{b}_2))$ using Algorithm 4.2.
3: Apply Algorithm 4.1 to $(a, b, \mathfrak{a}) = (r_{11}, r_{12}, \mathfrak{a}_2 \cdot \mathfrak{a}_1^{-1})$ and $\varepsilon = 1/(4\mathrm{c})$. Let $(u, v)$ be the output.
4: Let $\mathfrak{b} = \gcd(\langle u \rangle, \langle v \rangle \mathfrak{a}^{-1})$, find $x \in \mathfrak{a}^{-1}\mathfrak{b}^{-1}$ and $y \in \mathfrak{b}^{-1}$ such that $uy - vx = 1$.
5: Update $(\mathbf{b}_1, \mathbf{b}_2) \leftarrow (u\mathbf{b}_1 + v\mathbf{b}_2, x\mathbf{b}_1 + y\mathbf{b}_2)$ and $(\mathfrak{a}_1, \mathfrak{a}_2) \leftarrow (\mathfrak{a}_1\mathfrak{b}^{-1}, \mathfrak{a}_2\mathfrak{b})$.
6: **end while**
7: Strongly scale the pseudo-basis $((\mathfrak{a}_1, \mathbf{b}_1), (\mathfrak{a}_2, \mathbf{b}_2))$ using Algorithm 4.2.
8: **return** $\mathbf{b}_1$

**Lemma 4.9.** *Let $\gamma \geq 4 \cdot C \cdot \mathrm{c}^2$, where $C$ is as in Corollary 4.6. Then, given as input a pseudo-basis of a rank-2 module $M \subset K_{\mathbb{R}}^2$, Algorithm 4.3 outputs a vector $\mathbf{v} \in M \setminus \{\mathbf{0}\}$ such that $\mathcal{N}(\mathbf{v}) \leq \gamma^d \lambda_1^{\mathcal{N}}(M)$. Further, if $M \subseteq R^m$ and Algorithms 4.1 and 4.2 run in polynomial time, then Algorithm 4.3 runs in time polynomial in the input bit-length and in $\log \Delta_K$.*

*Proof.* We only prove here that at each loop iteration, the value $\mathcal{N}(r_{11}\mathfrak{a}_1)$ decreases by a factor at least $2^d$. As in the LLL algorithm, this is the main technical part of the proof. The rest of the proof can be found in the full version.

Recall that at the end of Step 2, we have $\|r_{ii}\|_\infty \leq \mathrm{c} \cdot \mathcal{N}(r_{ii}\mathfrak{a}_i)^{1/d}$ for $i = 1, 2$. Recall also that Algorithm 4.1 outputs $u, v$ such that $\|ur_{11} + vr_{12}\|_\infty \leq \varepsilon \|r_{11}\|_\infty$ and $\|v\|_\infty \leq C$. The new vector $\mathbf{b}_1$ at the end of the loop iteration is $u\mathbf{b}_1 + v\mathbf{b}_2$.

We compute an upper bound on its algebraic norm:

$$\begin{aligned}\mathcal{N}(u\mathbf{b}_1 + v\mathbf{b}_2) &\leq (\sqrt{d})^{-d}\|u\mathbf{b}_1 + v\mathbf{b}_2\|^d = (\sqrt{d})^{-d}\left\|\begin{pmatrix} ur_{11} + vr_{12} \\ vr_{22}\end{pmatrix}\right\|^d \\ &\leq (\sqrt{d})^{-d}\left(\|ur_{11} + vr_{12}\| + \|vr_{22}\|\right)^d \\ &\leq \left(\|ur_{11} + vr_{12}\|_\infty + \|vr_{22}\|_\infty\right)^d \\ &\leq \left(\varepsilon\|r_{11}\|_\infty + \|v\|_\infty \cdot \|r_{22}\|_\infty\right)^d.\end{aligned}$$

Using the facts that the basis is c-strongly scaled and that the condition of Step 1 is satisfied, we have:

$$\begin{aligned}\mathcal{N}(u\mathbf{b}_1 + v\mathbf{b}_2) &\leq \mathrm{c}^d \cdot \left(\varepsilon\mathcal{N}(r_{11}\mathfrak{a}_1)^{1/d} + C \cdot \mathcal{N}(r_{22}\mathfrak{a}_2)^{1/d}\right)^d \\ &\leq \mathrm{c}^d \cdot (\varepsilon + C \cdot (\mathrm{c}/\gamma))^d \cdot \mathcal{N}(r_{11}\mathfrak{a}_1).\end{aligned}$$

Now, by choice of $\varepsilon$ and $\gamma$:

$$\mathcal{N}(u\mathbf{b}_1 + v\mathbf{b}_2) \leq \mathrm{c}^d \cdot \left(\frac{1}{4\mathrm{c}} + \frac{1}{4\mathrm{c}}\right)^d \cdot \mathcal{N}(r_{11}\mathfrak{a}_1) = 2^{-d} \cdot \mathcal{N}(r_{11}\mathfrak{a}_1).$$

Recall that $\mathfrak{a}_1$ is also updated as $\mathfrak{a}_1\mathfrak{b}^{-1}$. Hence, to conclude, we argue that $\mathcal{N}(\mathfrak{a}_1\mathfrak{b}^{-1}) \leq 1$. Note that $\mathcal{N}(\mathfrak{a}_1) \leq 1$ holds due to scaling, and that $\mathcal{N}(\mathfrak{b}) \geq 1$ holds because $\mathfrak{b}$ is integral. Overall, we obtain that $\mathcal{N}(r_{11}\mathfrak{a}_1)$ decreases by a factor $\geq 2^d$ during a loop iteration. □

Instantiating this lemma with the value of $C$ obtained in Corollary 4.6, we obtain the following corollary.

**Corollary 4.10 (Heuristic).** *For any number field $K$ and any $\eta > 0$, there exists a lattice $L'$ of dimension $\widetilde{O}((\log|\Delta_K| + d\log\rho(R))^{2+\eta})$, a choice of the approximation factor $\gamma = 2^{\widetilde{O}(\log|\Delta_K| + d\log\rho(R))/d}$ and an algorithm $\mathcal{A}$ such that the following holds. Under Heuristic 1 and the heuristics of Lemma 2.3, algorithm $\mathcal{A}$ takes as input a pseudo-basis of a rank-2 module $M \subset K_{\mathbb{R}}^2$, and outputs a vector $\mathbf{v} \in M$ such that $\mathcal{N}(\mathbf{v}) \leq \gamma^d\lambda_1^{\mathcal{N}}(M)$. If given access to an oracle solving the closest vector problem in $L'$ in polynomial time, and when restricted to modules contained in $K^2$, Algorithm $\mathcal{A}$ runs in quantum polynomial time.*

**Acknowledgments.** We thank Léo Ducas for helpful discussions. This work was supported in part by BPI-France in the context of the national project RISQ (P141580), by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701) and by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

## References


AD17. M. R. Albrecht and A. Deo. Large Modulus Ring-LWE $\geq$ Module-LWE. In *ASIACRYPT*, 2017.

Ajt96. M. Ajtai. Generating hard instances of lattice problems. In *STOC*, 1996.

Ajt98. M. Ajtai. The shortest vector problem in $l_2$ is NP-hard for randomized reductions. In *STOC*, 1998.

BEF+17. J.-F. Biasse, T. Espitau, P.-A. Fouque, A. Gélin, and P. Kirchner. Computing generator in cyclotomic integer rings. In *EUROCRYPT*, 2017.

BF14. J.-F. Biasse and C. Fieker. Subexponential class group and unit group computation in large degree number fields. *LMS J Comput Math*, 2014.

BFH17. J.-F. Biasse, C. Fieker, and T. Hofmann. On the computation of the HNF of a module over the ring of integers of a number field. *J Symb Comput*, 2017.

BGV14. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ToCT*, 2014.

BP91. W. Bosma and M. Pohst. Computations with finitely generated modules over Dedekind domains. In *ISSAC*, 1991.

BS96. E. Bach and J. O. Shallit. *Algorithmic Number Theory: Efficient Algorithms*. MIT Press, 1996.

BS16. J.-F. Biasse and F. Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In *SODA*, 2016.

CDW17. R. Cramer, L. Ducas, and B. Wesolowski. Short Stickelberger class relations and application to ideal-SVP. In *EUROCRYPT*, 2017.

Cer05. J.-P. Cerri. *Spectres euclidiens et inhomogènes des corps de nombres*. PhD thesis, Université Henri Poincaré, Nancy, 2005.

Coh96. H. Cohen. Hermite and Smith normal form algorithms over Dedekind domains. *Math Comp*, 1996.

Fie97. C. Fieker. *Über relative Normgleichungen in älgebraischen Zahlkörpern*. PhD thesis, TU Berlin, 1997.

FP96. C. Fieker and M. E. Pohst. Lattices over number fields. In *ANTS*, 1996.

FP06. C. Fieker and M. E. Pohst. Dependency of units in number fields. *Math Comp*, 2006.

FS10. C. Fieker and D. Stehlé. Short bases of lattices over number fields. In *ANTS*, 2010.

GLM09. Y. H. Gan, C. Ling, and W. H. Mow. Complex lattice reduction algorithm for low-complexity full-diversity MIMO detection. *IEEE Trans Signal Processing*, 2009.

Hop98. A. Hoppe. *Normal forms over Dedekind domains, efficient implementation in the computer algebra system KANT*. PhD thesis, TU Berlin, 1998.

HPS98. J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: a ring based public key cryptosystem. In *ANTS*, 1998.

Kan87. R. Kannan. Minkowski's convex body theorem and integer programming. *Math Oper Res*, 1987.

KL17. T. Kim and C. Lee. Lattice reductions over euclidean rings with applications to cryptanalysis. In *IMACC*, 2017.

Laa16. T. Laarhoven. Sieving for closest lattice vectors (with preprocessing). In *SAC*, 2016.

Lez14. P. Lezowski. Computation of the euclidean minimum of algebraic number fields. *Math Comp*, 83(287):1397–1426, 2014.
LLL82. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math Ann*, 1982.
LM06. V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP*, 2006.
LPR10. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.
LPSW19. C. Lee, A. Pellet-Mary, D. Stehlé, and A. Wallet. An LLL algorithm for module lattices (full version). Cryptology ePrint Archive, 2019.
LS15. A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. *Des Codes Cryptography*, 2015.
MG02. D. Micciancio and S. Goldwasser. *Complexity of lattice problems: a cryptographic perspective*. Kluwer Academic Press, 2002.
Mic01. D. Micciancio. The hardness of the closest vector problem with preprocessing. *Trans Inf Theory*, 2001.
Nap96. H. Napias. A generalization of the LLL-algorithm over Euclidean rings or orders. *J théorie des nombres de Bordeaux*, 1996.
Neu99. J. Neukirch. Algebraic number theory. In *Grundlehren der Mathematischen Wissenschaften*, volume 322. Springer, 1999.
O'M63. O. T. O'Meara. *Introduction to Quadratic Forms*. Springer, 1963.
PHS19. A. Pellet-Mary, G. Hanrot, and D. Stehlé. Approx-SVP in ideal lattices with pre-processing. In *EUROCRYPT*, 2019.
PR06. C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, 2006.
Reg09. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J ACM*, 2009.
RSW18. M. Rosca, D. Stehlé, and A. Wallet. On the Ring-LWE and Polynomial-LWE problems. In *EUROCRYPT*, 2018.
SE94. C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Math Progr*, 1994.
SMSV14. Saruchi, I. Morel, D. Stehlé, and G. Villard. LLL reducing with the most significant bits. In *ISSAC*, 2014.
SSTX09. D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, 2009.