Fast Fourier Orthogonalization

Léo Ducas*
Cryptology Group, CWI
Amsterdam, The Netherlands
ducas@cwi.nl

Thomas Prest†
Thales Communications & Security
Gennevilliers, France
thomas.prest@thalesgroup.com

ABSTRACT

The classical fast Fourier transform (FFT) allows to compute in quasi-linear time the product of two polynomials, in the *circular convolution ring* $\mathbb{R}[x]/(x^d - 1)$ — a task that naively requires quadratic time. Equivalently, it allows to accelerate matrix-vector products when the matrix is *circulant*.

In this work, we discover that the ideas of the FFT can be applied to speed up the orthogonalization process of matrices with circulant blocks of size $d \times d$. We show that, when d is composite, it is possible to proceed to the orthogonalization in an inductive way —up to an appropriate re-indexation of rows and columns. This leads to a structured Gram-Schmidt decomposition. In turn, this structured Gram-Schmidt decomposition accelerates a cornerstone lattice algorithm: the nearest plane algorithm. The complexity of both algorithms may be brought down to $\Theta(d \log d)$.

Our results easily extend to *cyclotomic rings*, and can be adapted to Gaussian samplers. This finds applications in lattice-based cryptography, improving the performances of trapdoor functions.

Keywords. Fast Fourier transform, Gram-Schmidt orthogonalization, nearest plane algorithm, lattice algorithms, lattice trapdoor functions.

1. INTRODUCTION

When operations involving linear algebra are to be performed over structured matrices, a natural problem is to accelerate them by exploiting the structure. The most classical example is the fast Fourier transform [2, 8], which allows to perform matrix-vector multiplication in quasilinear time when the matrix is circulant. This is achieved by exploiting the isomorphism between the ring of $d \times d$ circulant matrices and the *circular convolution ring* $\mathcal{R}_d = \mathbb{R}[x]/(x^d - 1)$.

A widely studied and more involved question is matrix decomposition [20] for structured matrices, in particular Gram-Schmidt orthgonalization (GSO). In this work, we are interested in the GSO of *circulant matrices*, and more generally of matrices with circulant blocks. Our main motivation is to accelerate lattice algorithms for lattices that admit a basis with circulant blocks. This use case allows a helpful extra degree of freedom: one may permute rows and columns of the lattice basis since this leaves the generated lattice unchanged —up to an isometry.

As we will show, a proper re-indexation of these matrices highlights an inductive structure, with a fast Fourier flavor. This leads to accelerations of the orthogonalization process —and of the related nearest plane algorithm— down to quasilinear time and space.

The Nearest Plane Algorithm, Lattices and Cryptography

The nearest plane algorithm [1] is a central algorithm over lattices. It allows, after precomputation of the GSO and using a quadratic number of real operations, to find a relatively close point in a lattice to an arbitrary target. It is a core subroutine of LLL [13], and can be used for error correction over analogical noisy channels. It has also found applications in lattice-based cryptography as a decryption algorithm, and a randomized variant (called discrete Gaussian sampling) [11, 5] provides secure trapdoor functions based on lattice problems. This leads to cryptosystems (attribute-based encryption) with fine-grained access control, as [18, 6] to name a few.

Given a basis $\mathbf{B}$ of a lattice $L \subset \mathbb{R}^d$ and a target vector $\mathbf{c}$, the nearest plane algorithm finds a lattice point somewhat close to $\mathbf{c}$. The result belongs to a fundamental domain centered in $\mathbf{c}$, whose shape is the cuboid defined by $\tilde{\mathbf{B}}$, the GSO of $\mathbf{B}$ (see Figure 1). This algorithm performs $\Theta(d^2)$ real operations. The GSO itself is required as a precomputation.

Structured lattices in cryptography.

When it comes to practical lattice-based cryptography, a quadratic cost in the dimension is rather prohibitive considering the lattices at hand have dimensions ranging in the hundreds, or even thousands. For efficiency purposes, many cryptosystems (such as [10, 15, 16] to name a few) chose to rely on lattices with some algebraic structure, improving time and memory requirements to quasilinear in the dimension. This is sometimes referred as lattice-based cryptography in

---

*This work has been supported by a grant from CWI from budget for public-private-partnerships and in part by a grant from NXP Semiconductors. Part of this work was also supported by an NWO Free Competition Grant.

†Part of this work was done when the author was at the École Normale Supérieure, Paris.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

the *ring setting*. Technically, the chosen rings typically are *cyclotomic rings*, but those are closely related to the *convolution rings* discussed so far. The core of this optimization is the fast Fourier transform (FFT) [2, 4, 8, 19] allowing fast multiplication of polynomials. But this improvement did not apply in the case of the nearest plane algorithm or its randomized variant [11, 5]: naïve GSO do not take the algebraic structure into account.

One possible work-around [9, 22] consist of using the round-off algorithm [1] instead of the nearest-plane algorithm. However, this simpler algorithm outputs further vectors, both in the average and worst cases, weakening those cryptosystems.

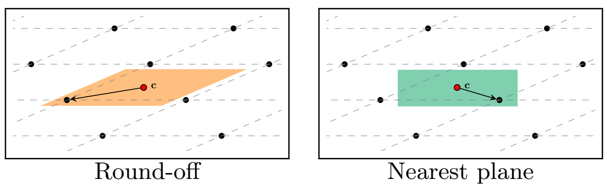


Figure 1: Round-off and nearest plane algorithms, and their associated fundamental domains.

Our contribution

In this work, we discover new algorithms, obtained by crossing Cooley-Tukey's [2] fast Fourier transform algorithm together with the orthogonalization and nearest plane algorithms (not exactly the GSO, but the closely related LDL* decomposition). Precisely, we show that, up to a re-indexation of rows and columns, the orthogonalization of matrices composed of $d \times d$-circulant blocks can be done in time $\Theta(d \log d)$ when the prime factors of d are bounded. Our algorithm produces the LDL* decomposition in a special compact format, requiring $\Theta(d \log d)$ complex numbers to represent.

From this compact representation, the nearest plane algorithm can also be performed using $\Theta(d \log d)$ real operations.[1] As a demonstration of the simplicity of our algorithms, we propose an implementation in `python` for $d \times d$-circulant matrices, when d is a power of 2.

Computational model and Parallelism.

Our algorithms and their complexity are described for Random Access Machines with unitary arithmetic operations on *real numbers*, i.e. we do not study the issue related to floating-point approximations and numerical stability.

Regarding parallelism, we note that our Fast-Fourier LDL* algorithm is almost fully parallelizable: viewed as an arithmetic circuit on real numbers, it has depth $O(\log(n))$ and width $O(n)$. This is unfortunately not the case of the Nearest Plane algorithm nor our Fast-Fourier variant, for which the n rounding steps are inherently sequential.

Techniques.

At the core of our techniques is the realization that representing elements of the convolution ring $\mathcal{R}_d = \mathbb{R}[x]/(x^d - 1)$ as circulant matrices is not the appropriate choice. To allow an induction similar to Cooley-Tukey's FFT, our representation must follow the *tower of rings* $\mathbb{R} \subset \mathcal{R}_{d_1} \subset \cdots \subset \mathcal{R}_{d_{i-1}} \subset \mathcal{R}_d$, for some chain of divisors $1 | d_1 | \dots | d_{i-1} | d$.

Such a representation is obtained by applying the (mixed-radix) digit-reversal order to the indexation of the rows and column of the circulant blocks, as pictured in Figure 2.

We show that this alternative indexation allows to represent the matrix $\mathbf{L}$ of the GSO in a factorized form: a product of $\Theta(\log d)$ (sparse) structured matrices, each of which can be stored in space $O(d)$. An example is given in Figure 3.

Once this hidden structure is unveiled (Theorem 1), the algorithmic implications follow quite naturally: one first easily derive an algorithm in time $O(n \log^2 n)$ —matching previous and more general results [21]— but noting that the splitting step may be performed directly in the Fourier domain allows us to save another $\log n$ factor. For easier algorithmic manipulations, the factorization of $\mathbf{L}$ is represented using a tree.

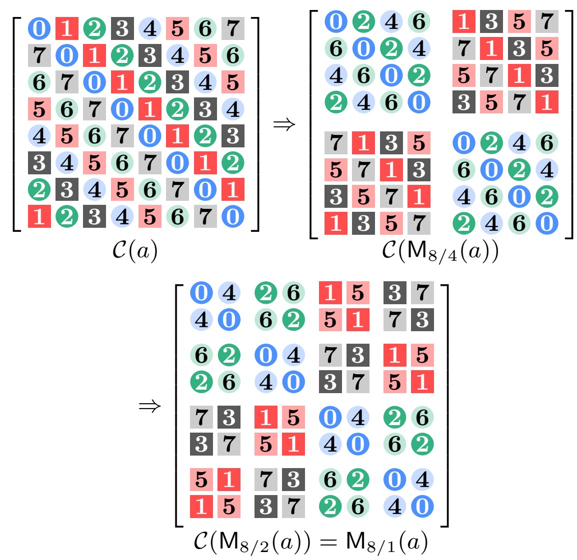


Figure 2: Re-indexing the transformation matrix of $f \in \mathcal{R}_8 \mapsto fx$ $(a = 0 + x + 2x^2 + \cdots + 7x^7 \in \mathcal{R}_8)$.

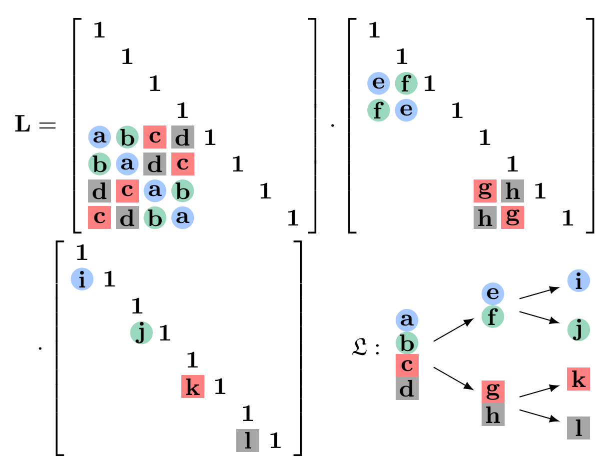


Figure 3: Factorization of L in the LDL* decomposition of $\mathsf{M}_{8/1}(a)$, and its tree representation $\mathfrak{L}$.

---

[1] If the number $n \times m$ of $d \times d$ circulant blocks is not constant, they contribute a factor $O(n^3)$ to the runtime of the fast-Fourier LDL* algorithm, $O(n^2)$ to its output size, and $O(n^2)$ to the run-time of the fast-Fourier nearest plane algorithm.

Related Works.

There exist many works related to the orthogonalization of structured bases. For Toeplitz matrices, Sweet [23] introduced an algorithm faster than the naive orthogonalization by a linear factor. Gragg [7] has shown that for Krylov bases – which are bases of the form $\{\mathbf{b}, r(\mathbf{b}), ..., r^{d-1}(\mathbf{b})\}$ –, the Levinson recursion [14, 3] allows, when r is an isometric operator, to perform orthogonalization in time $\Theta(d^2)$ instead of $\Theta(d^3)$.

There also exists superfast (running time $O(n \log^2 n)$) algorithms for the orthogonalization of Toeplitz-like matrices, for example by Olshevsky and Pan [21], and those are already based on a structure-preserving induction. In this light, one may interpret our result as a Fourier-compatible version of these superfast algorithms.

The question of improving the nearest-plane algorithm for structured matrices seems less studied. As far as we know, the state of the art consist of a single result [17], applying the Levinson recursion [14] to reduce by a linear factor its *space* complexity. Alternatively, a work-around of lesser quality was proposed for the NTRU signature scheme [9, 22].

Outline.

Section 2 introduces the mathematical tools that we will use through this paper. Section 3 presents our main result, namely the existence of a compact, factorized representation for the GSO and LDL* decomposition, and gives a fast Fourier flavored algorithm for computing it in this form. This compact LDL* decomposition is further exploited in Section 4, which presents a nearest plane algorithm that also has an FFT flavor. Appendix A extends all our results from convolution rings to cyclotomic rings, by reducing the latters to the formers. Appendix B demonstrates the practical feasibility of our algorithms by presenting a rather succinct `python` implementations of them in the case where d is a power of two.

2. PRELIMINARIES

For any ring $\mathcal{R}$, $\mathcal{R}[x]$ will denote the ring of univariate polynomials over $\mathcal{R}$. Scalars (which includes elements of $\mathcal{R}$) will usually be noted in plain letters (such as a, b), vectors will be noted in bold letters (such as $\mathbf{a}, \mathbf{b}$) and matrices will be noted in capital bold letters (such as $\mathbf{A}, \mathbf{B}$). Vectors are mostly in row notation, and as a consequence vector-matrix products are done in this order unless stated otherwise. $(a_1, \ldots, a_n)$ denotes the row vector formed of the a_i's, whereas $[\mathbf{a}_1, \ldots, \mathbf{a}_n]$ denotes the matrix whose rows are the $\mathbf{a}_i$'s. $\mathbb{N}$ denotes the set of non-negative integers, and $\mathbb{N}^*$ the set $\mathbb{N}\backslash\{0\}$. For $i, j \in \mathbb{Z}$, $[\![i, j]\!]$ will denote the set $\{i, i+1, \ldots, j-1, j\}$.

2.1 The Convolution Ring $\mathcal{R}_d$

Definition 1. *For any $d \in \mathbb{N}^*$, let $\mathcal{R}_d$ denote the ring $\mathbb{R}[x]/(x^d - 1)$, also known as circular convolution ring, or simply convolution ring.*

When d is highly composite, elementary operations in $\mathcal{R}_d$ can be performed in time $\Theta(d \log d)$ using the fast Fourier transform [2].

We equip the ring $\mathcal{R}_d$ with a conjugation operation as well as an inner product, making it an Hermitian inner product space. The definitions that we give also encompass other types of rings that will be used in Appendix A.

Definition 2. *Let $h \in \mathbb{R}[x]$ be a monic polynomial with distinct roots over $\mathbb{C}$, $\mathcal{R} \triangleq \mathbb{R}[x]/(h(x))$ and a, b be arbitrary elements of $\mathcal{R}$.*

- *We note $a^\star$ and call (Hermitian) adjoint of a the unique element of $\mathcal{R}$ such that for any root ζ of h, $a^\star(\zeta) = \overline{a(\zeta)}$, where $\bar{\cdot}$ is the usual complex conjugation over $\mathbb{C}$.*
- *The inner product over $\mathcal{R}$ is $\langle a, b\rangle \triangleq \sum_{h(\zeta)=0} a(\zeta) \cdot \overline{b(\zeta)}$, and the associated norm is $\|a\| \triangleq \sqrt{\langle a, a\rangle}$.*

In the particular case of convolution rings, one can check that if $a(x) = \sum_{i=0}^{d-1} a_i x^i \in \mathcal{R}_d$, then

$$a^\star(x) = a(1/x) \bmod (x^d - 1) = a_0 + \sum_{i=1}^{d-1} a_i x^{d-i}.$$

The (Hermitian) adjoint $\mathbf{B}^\star$ of a matrix $\mathbf{B} \in \mathcal{R}^{n\times m}$ is the transpose of the coefficient-wise adjoint of $\mathbf{B}$.

While the inner product $\langle\cdot,\cdot\rangle$ (resp. the associated norm $\|\cdot\|$) is not to be mistaken with the canonical coefficient-wise dot product $\langle\cdot,\cdot\rangle_2$ (resp. the associated norm $\|\cdot\|_2$), they are closely related. One can easily check that for any $f = \sum_{i\in\mathbb{Z}_d} f_i x^i \in \mathcal{R}_d$, the vector $(f(\zeta))_{\{\zeta^d=1\}}$ can be obtained from the coefficient vector $(f_i)_{0\leqslant i<d}$ by multiplying it by the Vandermonde matrix $\mathbf{V}_d = (\zeta_d^{ij})_{0\leqslant i,j<d}$, where ζ_d denotes an arbitrary d-th primitive root of unity. The matrix $\mathbf{V}_d$ satisfies $\mathbf{V}_d\mathbf{V}_d^\star = d \cdot \mathbf{I}_d$ and as an immediate consequence: $\langle f, g\rangle = d \cdot \langle f, g\rangle_2$.

Definition 3. *Let $m \geqslant n$ and $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathcal{R}^{n\times m}$. We say that $\mathbf{B}$ is full-rank (or is a basis) if for any linear combination $\sum_{1\leqslant i\leqslant n} a_i\mathbf{b}_i$ with $a_i \in \mathcal{R}$, we have the equivalence $(\sum_i a_i\mathbf{b}_i = \mathbf{0}) \iff (\forall i, a_i = 0)$.*

We note that since $\mathcal{R}$ is generally not an integral domain, a set formed of a single nonzero vector is not necessarily full-rank. In the rest of the paper, a basis will either denote a set of independent vectors $\{\mathbf{b}_1, \ldots \mathbf{b}_n\} \in (\mathcal{R}^m)^n$, or the full-rank matrix $\mathbf{B} \in \mathcal{R}^{n\times m}$ whose rows are the $\mathbf{b}_i$'s.

2.2 The GSO and LDL* Decomposition

In this section, $\mathcal{R} = \mathbb{R}[x]/(h(x))$ as in Definition 2. We first recall a few standard definitions. A matrix $\mathbf{L} \in \mathcal{R}^{n\times n}$ is unit lower triangular if it is lower triangular and has only 1's on its diagonal.

We say that a self-adjoint matrix $\mathbf{G} \in \mathcal{R}^{n\times n}$ is full-rank Gram (or FRG) if there exist $m \geqslant n$ and a full-rank matrix $\mathbf{B} \in \mathcal{R}^{n\times m}$ such that $\mathbf{G} = \mathbf{B}\mathbf{B}^\star$. This generalizes the notion of positive definiteness for symmetric real matrices.

We now recall the GSO and LDL* decomposition. The GSO decomposes any full-rank matrix as the product of a unit lower triangular matrix and an orthogonal matrix.

Proposition 1. *Let $\mathbf{B} \in \mathcal{R}^{n\times m}$ be a full-rank matrix. $\mathbf{B}$ can be uniquely decomposed as*

$$\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}, \qquad (1)$$

where $\mathbf{L}$ is unit lower triangular, and the rows of $\tilde{\mathbf{B}}$ are pairwise orthogonal.

When $\mathcal{R}$ is replaced by $\mathbb{R}$ or a number field, Proposition 1 is standard. In our case, a proof can be found in Appendix C.

The LDL* decomposition writes any positive definite matrix as a product $\mathbf{LDL}^\star$, where $\mathbf{L} \in \mathcal{R}^{n\times n}$ is unit lower triangular with 1's on the diagonal, and $\mathbf{D} \in \mathcal{R}^{n\times n}$ is diagonal. It is related to the GSO as for a basis $\mathbf{B}$, there exists a unique GSO $\mathbf{B} = \mathbf{L}\cdot\tilde{\mathbf{B}}$ and for an FRG matrix $\mathbf{G}$, there exists a unique LDL* decomposition $\mathbf{G} = \mathbf{LDL}^\star$. If $\mathbf{G} = \mathbf{BB}^\star$, then $\mathbf{G} = \mathbf{L}\cdot(\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\star)\cdot\mathbf{L}^\star$ is a valid LDL* decomposition of $\mathbf{G}$. As both decompositions are unique, the matrices $\mathbf{L}$ in both cases are actually the same. In a nutshell:

$$\begin{aligned} & \mathbf{L}\cdot\tilde{\mathbf{B}} \text{ is the GSO of } \mathbf{B} \\ \Leftrightarrow\quad & \mathbf{L}\cdot(\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\star)\cdot\mathbf{L}^\star \text{ is the LDL}^\star \text{ decomposition of } (\mathbf{BB}^\star). \end{aligned}$$

Algorithm 1 $\mathsf{LDL}^\star_{\mathcal{R}}(\mathbf{G})$

Require: A full-rank Gram matrix $\mathbf{G} = (G_{ij}) \in \mathcal{R}^{n\times n}$.
Ensure: The decomposition $\mathbf{G} = \mathbf{LDL}^\star$ over $\mathcal{R}$, where $\mathbf{L}$ is unit lower triangular and $\mathbf{D}$ is diagonal.

```
L, D ← 0^{n×n}
for i from 1 to n do
    L_ii ← 1
    D_i ← G_ii − Σ_{j<i} L_ij L*_ij D_j
    for j from 1 to i − 1 do
        L_ij ← (1/D_j) (G_ij − Σ_{k<j} L_ik L*_jk D_k)
    end for
end for
return ((L_ij), Diag(D_i))
```

Algorithm 1 computes the LDL* decomposition. When $\mathcal{R}$ is replaced by $\mathbb{R}$, the decomposition is noted LDL* rather than LDL* and it is well-known that it terminates without encountering divisions by 0 . In our case, we prove that it terminates correctly in Appendix C.

2.3 Babai's Nearest Plane Algorithm

The nearest plane algorithm allows to find a lattice close to an arbitrary target in the ambient vector space. Precisely, it ensures that the difference between the target and the output lies in the fundamental parallelepiped spanned by the GSO $\tilde{\mathbf{B}}$ of a given lattice basis $\mathbf{B}$, as depicted on Figure 1.

Definition 4. *Let $\mathbf{B} = \{\mathbf{b}_1,\dots,\mathbf{b}_n\}$ be a real basis. We call fundamental parallelepiped generated by $\mathbf{B}$ and note $\mathcal{P}(\mathbf{B})$ the set* $\sum\limits_{1\leqslant j\leqslant n}\left[-\frac{1}{2},\frac{1}{2}\right]\mathbf{b}_j = \left[-\frac{1}{2},\frac{1}{2}\right]^n\cdot\mathbf{B}$.

Algorithm 2 $\mathsf{NP}_{\mathbb{R}}(\mathbf{t},\mathbf{L})$

Require: The decomposition $\mathbf{B} = \mathbf{L}\cdot\tilde{\mathbf{B}}$ of $\mathbf{B} \in \mathbb{R}^{n\times m}$, a vector $\mathbf{t} \in \mathbb{R}^n$
Ensure: A vector $\mathbf{z} \in \mathbb{Z}^n$ such that $(\mathbf{t}-\mathbf{z})\mathbf{B} \in \mathcal{P}(\tilde{\mathbf{B}})$

```
z ← 0
for j = n, ..., 1 do
    t̄_j ← t_j + Σ_{i>j} (t_i − z_i) L_ij
    z_j ← ⌊t̄_j⌉
end for
return z
```

Proposition 2 (From [1, 13]). *Algorithm 2 outputs an integer vector $\mathbf{z}$ ($\mathbf{zB} \in \mathcal{L}(\mathbf{B})$) such that $(\mathbf{t}-\mathbf{z})\mathbf{B} \in \mathcal{P}(\tilde{\mathbf{B}})$.*

2.4 Coefficient Vectors and Circulant Matrices

Definition 5. *We define linear maps $c : \mathcal{R}_d^m \to \mathbb{R}^{dm}$ and $\mathcal{C} : \mathcal{R}_d^{n\times m} \to \mathbb{R}^{dn\times dm}$ as follows. For any $a = \sum_{i\in\mathbb{Z}_d} a_i x^i \in \mathcal{R}_d$ where each $a_i \in \mathbb{R}$:*

1. *The coefficient vector of a is $c(a) = (a_0,\dots,a_{d-1})$.*
2. *The circulant matrix of a is*

$$\mathcal{C}(a) \triangleq \begin{bmatrix} a_0 & a_1 & \cdots & a_{d-1} \\ a_{d-1} & a_0 & \cdots & a_{d-2} \\ \ddots & \ddots & \ddots & \ddots \\ a_1 & a_2 & \dots & a_0 \end{bmatrix} = \begin{bmatrix} c(a) \\ c(xa) \\ \vdots \\ c(x^{d-1}a) \end{bmatrix} \in \mathbb{R}^{d\times d}.$$

3. *c and $\mathcal{C}$ generalize to vectors and matrices in a coefficient-wise manner.*

Proposition 3. *The maps c and $\mathcal{C}$ satisfy the following properties:*

1. *$\mathcal{C}$ is an injective algebra morphism. In particular, $\mathcal{C}(a)\mathcal{C}(b) = \mathcal{C}(ab)$.*
2. *$c(a)\mathcal{C}(b) = c(ab)$.*
3. *$\mathcal{C}(a)^\star = \mathcal{C}(a^\star)$.*

Proposition 3, item 2 illustrates the following fact: the maps c and $\mathcal{C}$ are complementary. Indeed, if we consider the ordered $\mathbb{R}$-basis $\mathbf{B} = \{1, x, \dots, x^{d-1}\}$ of $\mathcal{R}_d$, then c is the map that writes any $a \in \mathcal{R}_d$ as its unique decomposition according to $\mathbf{B}$. Similarly, for the same basis, $\mathcal{C}(a)$ is the transformation matrix of the injective map $f \in \mathcal{R}_d \mapsto fa$. This is the intrinsic reason that makes Proposition 3, item 2 true.

In Section 2.5, we will define maps V, M which will be complementary in the same way as $c, \mathcal{C}$ are.

2.5 Linearization Operators

In this section, we introduce two partial linearization operators, a vectorial one denoted by $\mathsf{V}_{d/d'}$ (or V), and a matricial one denoted by $\mathsf{M}_{d/d'}$ (or M).

They are very similar to $c, \mathcal{C}$. Indeed, for any $d'|d$ and $a \in \mathcal{R}_d$, $\mathsf{V}_{d/d'}(a)$ will write a as a vector of the $\mathcal{R}_{d'}$-module $\mathcal{R}_{d'}^{d/d'}$, and $\mathsf{M}_{d/d'}(a)$ will be the matrix associated to the injective map $f \in \mathcal{R}_d \mapsto fa$. They are in fact generalizations of the maps from Section 2.4:

1. $\mathcal{R}_d$ may not only be seen as a d-dimensional $\mathbb{R}$-vector space, but also as a d/d'-dimensional $\mathcal{R}_{d'}$-module for any $d'|d$.

2. When writing $\mathcal{R}_d$ as a d-dimensional $\mathbb{R}$-vector space, we will not use $\{1, x, \dots, x^{d-1}\}$ as our canonical basis, but a permutation of this basis

Definition 6. *Let $d \in \mathbb{N}^*$ be a product of h (not necessarily distinct) primes. Let $\mathrm{gpd}(d)$ denote the greatest proper divisor of d. When clear from context, we also note h the number of prime divisors of d (counted with multiplicity), $d_h \triangleq d$ and for $i \in [\![1, h]\!]$, $d_{i-1}\ \mathrm{gpd}(d_i)$ and $k_i \triangleq d_i/d_{i-1}$, so that $1 = d_0|d_1|\dots|d_h = d$ and $\prod_{j\leqslant i} k_j = d_i$.*

The d_i's defined in Definition 6 form a tower of proper divisors of d. For any composite d, there exist multiple towers of proper divisors: per example , 1|6, 1|2|6 and 1|3|6 for $d = 6$. In this paper, each time we mention a tower of proper divisors of d it will refer to the unique one induced by Definition 6 (e.g. for $d = 6$, that tower is 1|3|6).

Definition 7. *Let $d, d' \in \mathbb{N}^*$ such that d' is in the tower of proper divisors of d, and let $k = d/d'$. We denote by x the indeterminate of the polynomial ring $\mathcal{R}_d = \mathbb{R}[x]/(x^d - 1)$ and by $y = x^k$ the indeterminate of $\mathcal{R}_{d'} = \mathbb{R}[y]/(y^{d'} - 1)$. We define the partial linearization $\mathsf{V}_{d/d'} : \mathcal{R}_d^m \to \mathcal{R}_{d'}^{km}$ recursively as follows:*

1. *For $d = d' = 1$, $\mathsf{V}_{d/d'}$ is the identity.*
2. *For $d' = \operatorname{gpd}(d)$ and a single element $a \in \mathcal{R}_d$, let $a = \sum_{i \in \mathbb{Z}_k} x^i a_i(y)$ where $a_i \in \mathcal{R}_{d'}$ for each i. Then*
$$\mathsf{V}_{d/d'}(a) \triangleq (a_0, \ldots, a_{k-1}).$$
In other words, $\mathsf{V}_{d/d'}(a) \in \mathcal{R}_{d'}^k$ is the row vector whose coefficients are the $(a_i)_{i \in \mathbb{Z}_k}$.
3. *For a vector $\mathbf{v} \in \mathcal{R}_d^m$, $\mathsf{V}_{d/d'}(\mathbf{v}) \in \mathcal{R}_{d'}^{km}$ is the component-wise applications of $\mathsf{V}_{d/d'}$.*
4. *For $d''|d'|d$ and any vector $\mathbf{v} \in \mathcal{R}_d^m$,*
$$\mathsf{V}_{d/d''}(\mathbf{v}) \triangleq \mathsf{V}_{d'/d''} \circ \mathsf{V}_{d/d'}(\mathbf{v}) \in \mathcal{R}_{d''}^{(d/d'')m}.$$

When d is clear from context, we simply note $\mathsf{V}_{d/d'} = \mathsf{V}_{/d'}$.

Interpretation.

In practice, an element $a \in \mathcal{R}_d$ is represented by a vector of d real elements corresponding to the d coefficients of a. In this context, the operator V simply permutes coefficients. As highlighted by Figure 4, when $d = 2^h$ is a power of two, $\mathsf{V}_{d/1}$ permutes the coefficients according to the bit-reversal order[2], which appears in the radix-2 fast Fourier transform (FFT). More generally, one can show that for an arbitrary d, $\mathsf{V}_{d/1}$ permutes the coefficient according to the general mixed-radix digit reversal order, which appears in the mixed-radix Cooley-Tukey FFT [2].

0 1 2 3 4 5 6 7 $\Rightarrow$ 0 2 4 6 1 3 5 7
$c(a)$ $c(\mathsf{V}_{/4}(a))$

$\Rightarrow$ 0 4 2 6 1 5 3 7
$c(\mathsf{V}_{/2}(a)) = \mathsf{V}_{/1}(a)$

Figure 4: Partial vectorial linearizations.

We now move to the matrix representation M compatible with V.

Definition 8. *Following the notations of Definition 7, we define the operator $\mathsf{M}_{d/d'} : \mathcal{R}_d^{n \times m} \to \mathcal{R}_{d'}^{kn \times km}$ as follows:*

1. *For $d = d' = 1$, $\mathsf{M}_{d/d'}$ is the identity.*
2. *For $d' = \operatorname{gpd}(d), k = d/d'$ and a single element $a = \sum_{i \in \mathbb{Z}_k} x^i a_i(y)$ where each $a_i \in \mathcal{R}_{d'}$, $\mathsf{M}_{d/d'}(a)$ is the following matrix of $\in \mathcal{R}_{d'}^{k \times k}$:*
$$\begin{bmatrix} a_0 & a_1 & \cdots & a_{k-1} \\ ya_{k-1} & a_0 & \cdots & a_{k-2} \\ \ddots & \ddots & \ddots & \ddots \\ ya_1 & ya_2 & \cdots & a_0 \end{bmatrix} = \begin{bmatrix} \mathsf{V}_{d/d'}(a) \\ \mathsf{V}_{d/d'}(x^k a) \\ \vdots \\ \mathsf{V}_{d/d'}(x^{(d'-1)k} a) \end{bmatrix}.$$
In particular, if d is prime, then $\mathsf{M}_{d/1}(a) \in \mathbb{R}^{d \times d}$ is exactly the circulant matrix $\mathcal{C}(a)$.
3. *For a vector $\mathbf{v} \in \mathcal{R}_d^m$ or a matrix $\mathbf{B} \in \mathcal{R}_d^{n \times m}$, $\mathsf{M}_{d/d'}(\mathbf{v}) \in \mathcal{R}_{d'}^{k \times km}$ and $\mathsf{M}_{d/d'}(\mathbf{B}) \in \mathcal{R}_{d'}^{kn \times km}$ are the component-wise applications of $\mathsf{M}_{d/d'}$.*
4. *For $d''|d'|d$ and any element $a \in \mathcal{R}_d$,*
$$\mathsf{M}_{d/d''}(a) \triangleq \mathsf{M}_{d'/d''} \circ \mathsf{M}_{d/d'}(a) \in \mathcal{R}_{d''}^{(d/d'') \times (d/d'')}.$$

When d is clear from context, we simply note $\mathsf{M}_{d/d'} = \mathsf{M}_{/d'}$.

[2] https://oeis.org/A030109

Interpretation.

As for the operator V, the steps of applying M are depicted in Section 1 Figure 2. The linearization $\mathsf{M}_{d/d'}(a)$ writes the transformation matrix of the map $f \in \mathcal{R}_d \mapsto fa$ using the same basis as the partial linearization $\mathsf{V}_{d/d'}$. As a result, both operators are compatible: $\forall a, b \in \mathcal{R}_d, \mathsf{V}_{d/d'}(ab) = \mathsf{V}_{d/d'}(a) \cdot \mathsf{M}_{d/d'}(b)$. More properties follows.

Proposition 4. *Let $d \in \mathbb{N}^*$, a, b (resp. $\mathbf{a}, \mathbf{b}$, resp. $\mathbf{A}, \mathbf{B}$) be arbitrary scalars (resp. vectors, resp. matrices) over $\mathcal{R}_d$, and $d'|d$. To be concise, we note $\mathsf{V} \triangleq \mathsf{V}_{d/d'}$ and $\mathsf{M} \triangleq \mathsf{M}_{d/d'}$. The maps V and M satisfy the following properties:*

1. *M is an injective algebra morphism, and in particular:*
$$\mathsf{M}(\mathbf{A} \cdot \mathbf{B}) = \mathsf{M}(\mathbf{A}) \cdot \mathsf{M}(\mathbf{B}).$$
2. *V is an injective linear map.*
3. *$\mathsf{V}(ab) = \mathsf{V}(a) \cdot \mathsf{M}(b)$.*
4. *V is an isometry:*
$$\langle \mathsf{V}(\mathbf{a}), \mathsf{V}(\mathbf{b}) \rangle_2 = \langle \mathbf{a}, \mathbf{b} \rangle_2.$$
5. *$\mathbf{B}$ is full-rank if and only if $\mathsf{M}(\mathbf{B})$ is full-rank.*

Since the proofs are rather straightforward to check from the definitions, we defer them to Appendix D.

Computing V and M in the Fourier Domain.

The operators V and M that we defined can be computed very efficiently when an element $a \in \mathcal{R}_d$ is represented by its coefficients but also when represented in the *Fourier domain*. In the first case, it is obvious that they can both be performed in time $\Theta(d)$ as they (symbolically) permute coefficients of a.

If a is represented in FFT form —that is, by the vector $(a(\zeta_d^i))_{i \in \mathbb{Z}_d}$ in $\mathbb{C}^d$— then computing $\mathsf{V}_{d/d'}(a)$ and $\mathsf{M}_{d/d'}(a)$ in FFT form can naively be done in time $\Theta(d \log d)$ by computing its inverse FFT, permuting its coefficients, and computing d/d' FFT's over $\mathcal{R}_{d'}$. However, it can be done in time $\Theta(d)$, as it is a single step – also known as *butterfly* – of the original fast Fourier transform. This is formalized in Lemma 1, a reformulation of a simple lemma that is at the heart of Cooley and Tukey's FFT.

Lemma 1 ([2], adapted). *Let $d \geqslant 2, d' = \operatorname{gpd}(d)$ and $k = d/d'$. Let $\mathsf{V} \triangleq \mathsf{V}_{d/d'}, \mathsf{M} \triangleq \mathsf{M}_{d/d'}$. For any $a \in \mathcal{R}_d$ (resp. $\mathbf{a} \in \mathsf{V}(\mathcal{R}_d)$, resp. $\mathbf{A} \in \mathsf{M}(\mathcal{R}_d)$):*

- *$\mathsf{V}(a)$, $\mathsf{V}^{-1}(\mathbf{a})$, $\mathsf{M}^{-1}(\mathbf{A})$ can be computed in FFT form in time $\Theta(kd)$.*
- *$\mathsf{M}(a)$ can be computed in FFT form in time $\Theta(k^2 d)$.*

Proof. Let $a \in \mathcal{R}_d$ be uniquely written $a = \sum_{i \in \mathbb{Z}_k} x^i a_i(x^k)$, where each $a_i \in \mathcal{R}_{d'}$. Cooley and Tukey show in [2] (equations 7, 8) that we can switch from the FFT of a to the FFT of all the a_i's (and conversely) in time $\Theta(kd)$. As the a_i's are the coefficients of $\mathsf{V}(a)$ and $\mathsf{M}(a)$, the result follows. □

In Sections 3 and 4, we will use the operators V and M to speed up the orthogonalization and nearest plane algorithms. The core idea is that these operators allow to "batch" orthogonalization operations, resulting in a $\Theta(d/\log(d))$ speedup.

3. FAST FOURIER LDL DECOMPOSITION

This section presents our main result. We present the existence of a compact representation in Section 3.1, and then derive a fast algorithm to compute it in Section 3.2.

3.1 A Compact LDL* Decomposition

THEOREM 1. *Let $d \in \mathbb{N}$ and $1 = d_0|d_1|\dots|d_h = d$ be a tower of proper divisors of d. Let $\mathbf{b} \in \mathcal{R}_d^m$ such that $\mathsf{M}_{d/1}(\mathbf{b})$ is full-rank. There exists a GSO of $\mathsf{M}_{d/1}(\mathbf{b})$ as follows:*

$$\mathsf{M}_{d/1}(\mathbf{b}) = \left(\prod_{i=0}^{h-1} \mathsf{M}_{d_i/1}(\mathbf{L}_i)\right) \cdot \tilde{\mathbf{B}}_0$$

where $\tilde{\mathbf{B}}_0 \in \mathbb{R}^{d\times dm}$ is orthogonal, and each $\mathbf{L}_i \in \mathcal{R}_{d_i}^{(d/d_i)\times(d/d_i)}$ is a block-diagonal matrix with unit lower triangular matrices of $\mathcal{R}_{d_i}^{(d_{i+1}/d_i)\times(d_{i+1}/d_i)}$ as its d/d_{i+1} diagonal blocks.[3]

As an example, the matrix $\mathbf{L}$ of the GSO of $\mathsf{M}_{8/1}(a)$ for some $a \in \mathcal{R}_8$ is depicted in Section 1 Figure 3.

PROOF. If d is prime, the theorem is trivial as it is exactly the GSO. We suppose that d is composite and that the theorem is true for any $\mathcal{R}_i$ with $i < d$. By Proposition 4, item 5, the matrix $\mathbf{B}_{h-1} \triangleq \mathsf{M}_{d/d_{h-1}}(\mathbf{b})$ is full-rank too. Using the classical GSO, we can therefore decompose it as $\mathbf{B}_{h-1} = \mathbf{L}_{h-1}\tilde{\mathbf{B}}$, where $\mathbf{L}_{h-1} \in \mathcal{R}_{d_{h-1}}^{k_h\times k_h}, \tilde{\mathbf{B}} \in \mathcal{R}_{d_{h-1}}^{k_h\times mk_h}$ and $k_h \triangleq d/\operatorname{gpd}(d)$. $\mathbf{L}_{h-1}$ is unit lower triangular and $\tilde{\mathbf{B}}$ is orthogonal. Noting $\tilde{\mathbf{B}} = [\mathbf{b}_1, \dots, \mathbf{b}_{k_h}]$, all the $\mathbf{b}_j$'s are pairwise orthogonal and each $\mathsf{M}_{/1}(\mathbf{b}_j)$ is full-rank. By inductive hypothesis, they can be decomposed as follows:

$$\forall j \in [\![1, k_h]\!], \mathsf{M}_{d_{h-1}/1}(\mathbf{b}_j) = \left(\prod_{i=0}^{h-2} \mathsf{M}_{d_i/1}(\mathbf{L}_{i,j})\right) \cdot \tilde{\mathbf{B}}_j, \quad (2)$$

where each $\tilde{\mathbf{B}}_j \in \mathbb{R}^{d_{h-1}\times md_{h-1}}$ is full-rank orthogonal and for $i < h-1$, each $\mathbf{L}_{i,j} \in \mathcal{R}_{d_i}^{(d_{h-1}/d_i)\times(d_{h-1}/d_i)}$ is a block-diagonal matrix with unit lower triangular matrices of $\mathcal{R}_{d_i}^{(d_{i+1}/d_i)\times(d_{i+1}/d_i)}$ as its d_{h-1}/d_{i+1} diagonal blocks. To be concise, we now note $\mathsf{M} \triangleq \mathsf{M}_{d_{h-1}/1}$ and $\mathsf{V} \triangleq \mathsf{V}_{d_{h-1}/1}$. We have:

$$\begin{aligned}
\mathsf{M}_{d/1}(\mathbf{b}) &= \mathsf{M}(\mathbf{L}_{h-1}) \cdot \mathsf{M}(\tilde{\mathbf{B}}_{h-1}) \\
&= \mathsf{M}(\mathbf{L}_{h-1}) \cdot \mathsf{M}[\mathbf{b}_1, \dots, \mathbf{b}_{k_h}] \\
&= \mathsf{M}(\mathbf{L}_{h-1}) \cdot [\mathsf{M}(\mathbf{b}_1), \dots, \mathsf{M}(\mathbf{b}_{k_h})] \\
&= \mathsf{M}(\mathbf{L}_{h-1}) \cdot \operatorname{Diag}\left(\prod_{i=0}^{h-2} \mathsf{M}_{d_i/1}(\mathbf{L}_{i,j})\right) \cdot \tilde{\mathbf{B}}_0 \\
&= \mathsf{M}(\mathbf{L}_{h-1}) \cdot \left(\prod_{i=0}^{h-2} \mathsf{M}_{d_i/1}(\mathbf{L}_i)\right) \cdot \tilde{\mathbf{B}}_0 \\
&= \left(\prod_{i=0}^{h-1} \mathsf{M}_{d_i/1}(\mathbf{L}_i)\right) \cdot \tilde{\mathbf{B}}_0,
\end{aligned}$$

where $\tilde{\mathbf{B}}_0 = [\tilde{\mathbf{B}}_1, \dots, \tilde{\mathbf{B}}_{k_h}]$. The first equality simply uses the fact that M is a ring homomorphism (Proposition 4, item 1). The second and third ones are immediate from the definitions. The fourth one uses the inductive hypothesis (equation 2) on each $\mathbf{b}_j$ and take $\tilde{\mathbf{B}}_0 \triangleq [\tilde{\mathbf{B}}_1, \dots, \tilde{\mathbf{B}}_{k_h}]$. In the fifth equality, we take $\mathbf{L}_i \triangleq \operatorname{Diag}(\mathbf{L}_{i,1}, \dots, \mathbf{L}_{i,k_h})$ and just need to check that $\tilde{\mathbf{B}}_0$ and $\mathbf{L}$ are as stated by the theorem:

[3]Indexed products are to be read $\prod_{i=0}^k \alpha_i = \alpha_k\alpha_{k-1}\cdots\alpha_0$.

- Since each $\mathbf{L}_{i,j}$ is block diagonal with d_{h-1}/d_{i+1} unit lower triangular diagonal blocks, $\mathbf{L}_i$ is block diagonal with $k_h(d_{h-1}/d_{i+1}) = d/d_{i+1}$ unit lower triangular diagonal blocks.
- We also need to show that $\tilde{\mathbf{B}}_0$ is orthogonal. Each submatrix $\tilde{\mathbf{B}}_j$ of $\tilde{\mathbf{B}}_0$ is the orthogonalization of $\mathsf{M}(\mathbf{b}_j)$ by induction hypothesis. Therefore, for two distinct rows $\mathbf{u}, \mathbf{v}$ of $\tilde{\mathbf{B}}_0$:
  - If they belong to the same submatrix $\tilde{\mathbf{B}}_j$, they are orthogonal by induction hypothesis.
  - Suppose they belong to different submatrices: $\mathbf{u} \in \tilde{\mathbf{B}}_j, \mathbf{v} \in \tilde{\mathbf{B}}_\ell$ and $j \neq \ell$. Then $\mathbf{u}$ (resp. $\mathbf{v}$) is a linear combination of rows of $\mathsf{M}(\mathbf{b}_j)$ (resp. $\mathsf{M}(\mathbf{b}_\ell)$): $\mathbf{v} = \mathbf{a}_j \cdot \mathsf{M}(\mathbf{b}_j)$ and $\mathbf{v} = \mathbf{a}_\ell \cdot \mathsf{M}(\mathbf{b}_j)$ for some $\mathbf{a}_j, \mathbf{a}_\ell$ in $\mathbb{R}^{d_{h-1}}$. Noting $a_j = \mathsf{V}^{-1}(\mathbf{a}_j)$ and $a_\ell = \mathsf{V}^{-1}(\mathbf{a}_\ell)$:

    $$\begin{aligned}
    \langle \mathbf{u}, \mathbf{v}\rangle_2 &= \langle \mathsf{V}(a_j)\mathsf{M}(\mathbf{b}_j), \mathsf{V}(a_\ell)\mathsf{M}(\mathbf{b}_\ell)\rangle_2 \\
    &= \langle \mathsf{V}(a_j\mathbf{b}_j), \mathsf{V}(a_\ell\mathbf{b}_\ell)\rangle_2 \\
    &= \langle a_j\mathbf{b}_j, a_\ell\mathbf{b}_\ell\rangle_2 = 0
    \end{aligned}$$

    Where the second equality comes from Proposition 4, item 3, the third one from the fact that V is a scaled isometry (Proposition 4, item 4) and the fourth one from $\mathbf{b}_j, \mathbf{b}_\ell$ being orthogonal.

Therefore $\tilde{\mathbf{B}}_0$ is orthogonal. □

The theorem we stated gives the GSO of $\mathsf{M}_{d/1}(\mathbf{b})$ for a vector $\mathbf{b} \in \mathcal{R}_d^m$, but can be easily generalized from a vector $\mathbf{b}$ to a matrix $\mathbf{B}$, and also yields a compact LDL* decomposition.

COROLLARY 1. *Let $d \in \mathbb{N}$ and $1 = d_0|d_1|\dots|d_h = d$ be a tower of proper divisors of d. Let $\mathbf{B} \in \mathcal{R}_d^{n\times m}$ be a full-rank matrix. There exist $h+1$ matrices $(\mathbf{L}_i)_{0\leqslant i\leqslant h}$ such that:*

- *$\mathbf{L}_h \in \mathcal{R}_d^{n\times n}$ is unit lower triangular.*
- *For each $i < h$, $\mathbf{L}_i \in \mathcal{R}_{d_i}^{n(d/d_i)\times n(d/d_i)}$ is a block-diagonal matrix whose $n(d/d_{i+1})$ diagonal blocks are unit lower triangular matrices of $\mathcal{R}_{d_i}^{(d_{i+1}/d_i)\times(d_{i+1}/d_i)}$.*

Furthermore, if we note $\mathbf{L} = \left(\prod_{i=0}^h \mathsf{M}_{d_i/1}(\mathbf{L}_i)\right)$ and $\tilde{\mathbf{B}}_0 \triangleq \mathbf{L}^{-1} \cdot \mathsf{M}_{d/1}(\mathbf{B})$, then:

1. *The GSO of $\mathsf{M}_{d/1}(\mathbf{B})$ is $\mathsf{M}_{d/1}(\mathbf{B}) = \mathbf{L} \cdot \tilde{\mathbf{B}}_0$.*
2. *The LDL* decomposition of $\mathsf{M}_{d/1}(\mathbf{B}\mathbf{B}^\star)$ is*

$$\mathsf{M}_{d/1}(\mathbf{B}) = \mathbf{L} \cdot (\tilde{\mathbf{B}}_0\tilde{\mathbf{B}}_0^t) \cdot \mathbf{L}^t.$$

PROOF. We have $\mathbf{B} = \mathbf{L}_h\mathbf{B}'$, where $\mathbf{L}_h$ is given by either the GSO or LDL* decomposition algorithm. $\mathbf{B}' = \{\mathbf{b}'_1, \dots, \mathbf{b}'_n\}$ is orthogonal. Applying Theorem 1 to each row vector $\mathbf{b}'_j$ of $\mathbf{B}'$ yields n decompositions $(\mathbf{L}_{i,j})_{0\leqslant i<h}$ and n orthogonal matrices $\tilde{\mathbf{B}}_j$, each spanning the same space as $\mathbf{B}_j \triangleq \mathsf{M}_{d/1}(\mathbf{b}'_j)$. Taking $\mathbf{L}_i \triangleq \operatorname{Diag}(\mathbf{L}_{i,j})$ and $\tilde{\mathbf{B}}_0 \triangleq [\tilde{\mathbf{B}}_1, \dots, \tilde{\mathbf{B}}_n]$ yields the GSO.

The LDL* decomposition is then given "for free" by its equivalence with the GSO, and indeed, one can check that since $\tilde{\mathbf{B}}_0$ is orthogonal, $\tilde{\mathbf{B}}_0\tilde{\mathbf{B}}_0^t$ is diagonal. □

Theorem 1 and Corollary 1 state that for any full-rank matrix $\mathbf{B} \in \mathcal{R}_d^{n\times m}$, the $\mathbf{L}$ matrix in the GSO (resp. LDL* decomposition) of $\mathsf{M}_{d/1}(\mathbf{B})$ (resp. $\mathsf{M}_{d/1}(\mathbf{B}\mathbf{B}^\star)$) can be represented in a factorized form, where each of the factors $\mathbf{L}_i$ is a sparse, block-diagonal matrix.

3.2 A Fast Algorithm for the Compact LDL Decomposition

Theorem 1 and Corollary 1 are constructive: more precisely, their proofs give a fast algorithm for computing a compact factorized form of $\mathbf{L}$ quickly. Algorithm 3 computes a compact LDL* decomposition in the form of the tree $\mathfrak{L}$, which nodes are labeled by structured matrices of various sizes. We note that this decomposition depends of the tower of proper divisors chosen. Algorithm 3 uses the unique one induced by Definition 6.

Algorithm 3 $\mathsf{ffLDL}^\star_{\mathcal{R}_d}(\mathbf{G})$

Require: A full-rank Gram matrix $\mathbf{G} \in \mathcal{R}_d^{n\times n}$
Ensure: The compact LDL* decomposition of $\mathbf{G}$ in FFT form
1: $(\mathbf{L}, \mathbf{D}) \leftarrow \mathsf{LDL}^\star_{\mathcal{R}_d}(\mathbf{G})$
2: **if** $d = 1$ **then**
3:    **return** $(\mathbf{L}, \mathbf{D})$
4: **end if**
5: $d' \leftarrow \mathrm{gpd}(d)$
6: **for** $i = 1, \ldots, n$ **do**
7:    $\mathfrak{L}_i \leftarrow \mathsf{ffLDL}^\star_{\mathcal{R}_{d'}}(\mathsf{M}_{d/d'}(\mathbf{D}_{ii}))$
8: **end for**
9: **return** $(\mathbf{L}, (\mathfrak{L}_i)_{1\leqslant i\leqslant n})$

Algorithm 3 computes a "fast Fourier LDL*", instead of the "fast Fourier GSO" hinted at in the proof of Corollary 1. The reason why we favor this approach is because it allows a complexity gain. This gain can already be observed in the classic versions of the aforementioned algorithms. Indeed, consider the $\mathbf{L}$ in the GSO of $\mathbf{B} \in \mathcal{R}_d^{2\times m}$, which is exactly the $\mathbf{L}$ in the LDL* decomposition of $\mathbf{B}\mathbf{B}^\star \in \mathcal{R}_d^{2\times 2}$. Computing it with the LDL* algorithm is then $\Theta(m)$ times faster than with the Gram-Schmidt process. The same phenomenon happens with their recursive variants.

Lemma 2. *Let $d \in \mathbb{N}$ and $1 = d_0|d_1|\ldots|d_h = d$ be the tower of proper divisors of d given by the successive* gpd*'s, and for $i \in [\![1, h]\!]$, let $k_i \triangleq d_i/d_{i-1}$. Let $\mathbf{G} \in \mathcal{R}_d^{n\times n}$ be a full-rank Gram matrix. Then Algorithm 3 computes the LDL* decomposition tree of $\mathbf{G}$ in FFT form in time*

$$\Theta(n^2 d \log d) + \Theta(n^3 d) + \Theta(nd) \sum_{1\leqslant i\leqslant h} k_i^2.$$

In particular, if all the k_i are bounded by a small constant k, then the complexity of Algorithm 3 is upper bounded by $\Theta(n^3 d + n^2 d \log d)$.

The proof of Lemma 2 is left in Appendix E. We note that Algorithm 3 is parallelizable to up to d processes,: step 1 relies on operations on polynomials which are parallelizable and all the iterations of step 7 are independent.

4. FAST FOURIER NEAREST PLANE

In this section, we show how to exploit further the compact form of the LDL* decomposition to have a fast Fourier variant of the nearest plane algorithm. It outputs vectors of the same quality (*ie* as close to the target vector) as its classical iterative counterpart Algorithm 2, but runs $\tilde{\Theta}(d)$ times faster.

Definition 9. *Let $\mathcal{Z}_d$ denote the ring $\mathbb{Z}[x]/(x^d - 1)$ of elements of $\mathcal{R}_d$ with integer coefficients.*

Algorithm 4 $\mathsf{ffNP}_{\mathcal{R}_d}(\mathbf{t}, \mathfrak{L})$

Require: $\mathbf{t} \in \mathcal{R}_d^n$, a precomputed tree $\mathfrak{L}$, (implicitly) a matrix $\mathbf{B} \in \mathcal{R}_d^{n\times m}$ such that $\mathfrak{L}$ is the compact LDL* decomposition tree of $\mathbf{B}\mathbf{B}^\star$.
Ensure: $\mathbf{z} \in \mathcal{Z}_d^n$ such that $\mathsf{V}((\mathbf{z} - \mathbf{t})\mathbf{B}) \in \mathcal{P}(\tilde{\mathbf{B}}_0)$, where $\tilde{\mathbf{B}}_0$ is the orthogonalization of $\mathsf{M}(\mathbf{B})$.
if $d = 1$ **then**
   $(\mathbf{L}, \mathbf{D}) \leftarrow \mathfrak{L}$
   return $\mathsf{NP}_{\mathbb{R}}(\mathbf{L}, \mathbf{t})$
end if
$(\mathbf{L}, (\mathfrak{L}_i)_{1\leqslant j\leqslant n}) \leftarrow \mathfrak{L}$
$d' \leftarrow \mathrm{gpd}(d)$
for $j = n, \ldots, 1$ **do**
   $\bar{\mathbf{t}}_j \leftarrow \mathbf{t}_j + \sum_{i>j}(\mathbf{t}_i - \mathbf{z}_i) L_{ij}$
   $\mathbf{z}_j \leftarrow \mathsf{V}^{-1}_{d/d'}\left[\mathsf{ffNP}_{\mathcal{R}_{d'}}(\mathsf{V}_{d/d'}(\bar{\mathbf{t}}_j), \mathfrak{L}_j)\right]$
end for
return $\mathbf{z} = (\mathbf{z}_i, \ldots, \mathbf{z}_n)$

Lemma 3. *Let $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathcal{R}_d^{n\times m}$ and $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n\}$ be its Gram-Schmidt orthogonalization in $\mathcal{R}_d$. The vectors $\mathbf{z}$ and $\bar{\mathbf{t}} = (\bar{\mathbf{t}}_1, \ldots, \bar{\mathbf{t}}_n)$ in Algorithm 4 satisfy*

$$(\mathbf{z} - \mathbf{t}) \cdot \mathbf{B} = (\mathbf{z} - \bar{\mathbf{t}}) \cdot \tilde{\mathbf{B}}.$$

Proof. We recall that for each $i \in [\![1, n]\!]$, $\tilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{j<i} L_{ij}\tilde{\mathbf{b}}_j$. We have:

$$\begin{aligned}
(\mathbf{z} - \bar{\mathbf{t}}) \cdot \tilde{\mathbf{B}} &= \textstyle\sum_{1\leqslant j\leqslant n}(\mathbf{z}_j - \bar{\mathbf{t}}_j)\tilde{\mathbf{b}}_j \\
&= \textstyle\sum_{1\leqslant j\leqslant n}\left[(\mathbf{z}_j - \mathbf{t}_j) + \sum_{i>j} L_{ij}(\mathbf{z}_i - \mathbf{t}_i)\right]\tilde{\mathbf{b}}_j \\
&= \textstyle\sum_{1\leqslant i\leqslant j\leqslant n}(\mathbf{z}_i - \mathbf{t}_i) L_{ij}\tilde{\mathbf{b}}_j \\
&= \textstyle\sum_{1\leqslant i\leqslant n}(\mathbf{z}_i - \mathbf{t}_i)\mathbf{b}_i \\
&= (\mathbf{z} - \mathbf{t}) \cdot \mathbf{B}.
\end{aligned} \tag{3}$$

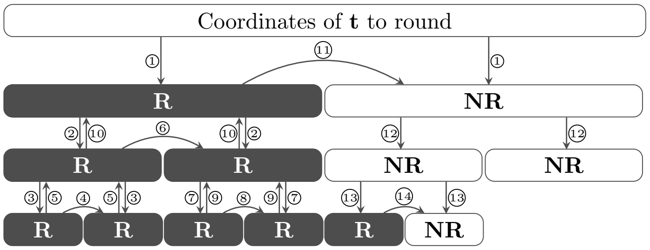
The first and last equalities are trivial, the second one replaces the $\bar{\mathbf{t}}_j$'s by their definitions, the third one just simplifies the sum and the fourth one is another way of saying that $\mathbf{L} \cdot \tilde{\mathbf{B}} = \mathbf{B}$. □

Theorem 2. *Let $\mathsf{M} \triangleq \mathsf{M}_{d/1}$ and $\mathsf{V} \triangleq \mathsf{V}_{d/1}$. Algorithm 4 outputs $\mathbf{z} \in \mathcal{Z}^n$ such that $\mathsf{V}((\mathbf{z} - \mathbf{t})\mathbf{B}) \in \mathcal{P}(\tilde{\mathbf{B}}_0)$, where $\tilde{\mathbf{B}}_0$ is the orthogonalization of $\mathsf{M}(\mathbf{B})$ over $\mathbb{R}$.*

Proof. The result is trivially true if $d = 1$. We prove it in the general case. By definition, each subtree $\mathfrak{L}_j$ is the LDL* decomposition tree of $\mathsf{M}_{d/\,\mathrm{gpd}(d)}(\tilde{\mathbf{b}}_j)$. By induction hypothesis, we therefore know that $\mathsf{V}((\mathbf{z}_j - \bar{\mathbf{t}}_j)\tilde{\mathbf{b}}_j) \in \mathcal{P}(\tilde{\mathbf{B}}_j)$, where $\tilde{\mathbf{B}}_j$ is the orthogonalization of $\mathbf{B}_j \triangleq \mathsf{M}(\tilde{\mathbf{b}}_j)$. From Lemma 3, we have

$$(\mathbf{z} - \mathbf{t})\mathbf{B} = \sum_{j=1,\ldots,n} (\mathbf{z}_j - \bar{\mathbf{t}}_j) \cdot \tilde{\mathbf{b}}_j,$$

so $\mathsf{V}((\mathbf{z} - \mathbf{t})\mathbf{B}) \in \mathcal{P}([\tilde{\mathbf{B}}_1, \ldots, \tilde{\mathbf{B}}_n])$. Now, from the proof of Corollary 1, we know that $\tilde{\mathbf{B}}_0 = [\tilde{\mathbf{B}}_1, \ldots, \tilde{\mathbf{B}}_n]$ is actually the orthogonalization of $\mathsf{M}(\mathbf{B})$, which concludes the proof. □



An ongoing execution of Algorithm 4 for a single $t \in \mathcal{R}_8$.

1. Arrows are labeled in their order of execution. Downward ($\downarrow \downarrow$) (resp. upward ($\uparrow \uparrow$)) arrows correspond to computing $\mathsf{V}_{d/d'}(\bar{\mathbf{t}}_j)$ (resp. $\mathsf{V}^{-1}_{d/d'}(\dots)$) at step 9. Transverse ($\curvearrowright$) arrows correspond to step 8.

2. Cells labeled with **R** (as in **R**ounded) correspond to already completed subcalls of Algorithm 4, as opposed to those labeled with **NR** (as in **N**ot **R**ounded).

Figure 5: High-level execution of the fast Fourier nearest plane algorithm

Unlike the fast Fourier transform, Algorithm 4 is not fully parallelizable, due to step 8 ($\curvearrowright$ arrows in Figure 5). However, its complexity in d is $\Theta(d \log d)$: informally, this is because each arrow $\downarrow, \uparrow$ or $\curvearrowright$ has a linear complexity in the size of the cells it connects. A more precise statement follows.

Lemma 4. *Let $d \in \mathbb{N}$ and $1 = d_0 | d_1 | \dots | d_h = d$ be be the tower of proper divisors of d given by the successive* gpd*s, and for $i \in [\![1, h]\!]$, let $k_i \triangleq d_i / d_{i-1}$. Let $\mathbf{B} \in \mathcal{R}_d^{n \times m}$ and $\mathfrak{L}$ be its LDL* decomposition tree. The complexity of Algorithm 4 is given by:*

$$\Theta(nd \log d) + \Theta(n^2 d) + \Theta(nd) \sum_{1 \leqslant i \leqslant h} k_i^2.$$

In particular, if all the k_i are bounded by a constant, then the complexity of Algorithm 4 is $\Theta(n^2 d + nd \log d)$.

The proof of Lemma 4 is deferred to Appendix F.

Acknowledgements

The authors would like to thank the anonymous reviewers of ISSAC and EUROCRYPT for their diligent comments, significantly contributing to improve the presentation of this article.

5. REFERENCES

[1] Babai, L. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica 6*, 1 (1986), 1–13. Preliminary version in STACS 1985.

[2] Cooley, J. W., and Tukey, J. W. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation 19*, 90 (1965), 297–301.

[3] Durbin, J. The fitting of time-series models. *Review of the International Statistical Institute 28*, 3 (1960), pp. 233–244.

[4] Gentleman, W. M., and Sande, G. Fast Fourier transforms: for fun and profit. In *Proceedings of the November 7-10, 1966, fall joint computer conference* (1966), ACM, pp. 563–578.

[5] Gentry, C., Peikert, C., and Vaikuntanathan, V. Trapdoors for hard lattices and new cryptographic constructions. In *STOC (2008)*.

[6] Gorbunov, S., Vaikuntanathan, V., and Wee, H. Attribute-based encryption for circuits. In *45th ACM STOC* (June 2013), D. Boneh, T. Roughgarden, and J. Feigenbaum, Eds., ACM Press, pp. 545–554.

[7] Gragg, W. B. Positive definite toeplitz matrices, the arnoldi process for isometric operators, and gaussian quadrature on the unit circle. *Journal of Computational and Applied Mathematics 46*, 1-2 (1993), 183 – 198.

[8] Heideman, M. T., Johnson, D. H., and Burrus, C. S. Gauss and the history of the fast Fourier transform. *ASSP Magazine, IEEE 1*, 4 (1984), 14–21.

[9] Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J. H., and Whyte, W. NTRUSIGN: Digital signatures using the NTRU lattice. In *CT-RSA (2003)*.

[10] Hoffstein, J., Pipher, J., and Silverman, J. H. NTRU: A ring-based public key cryptosystem. In *ANTS* (1998), pp. 267–288.

[11] Klein, P. N. Finding the closest lattice vector when it's unusually close. In *SODA* (2000), pp. 937–941.

[12] Lang, S. *Algebraic number theory*, 3 ed. 1995.

[13] Lenstra, A. K., Lenstra, Jr., H. W., and Lovász, L. Factoring polynomials with rational coefficients. *Mathematische Annalen 261*, 4 (December 1982), 515–534.

[14] Levinson, N. The Wiener RMS (root mean square) error criterion in filter design and prediction. *J. Math. Phys. Mass. Inst. Tech. 25* (1947), 261–278.

[15] Lyubashevsky, V., Micciancio, D., Peikert, C., and Rosen, A. SWIFFT: A modest proposal for FFT hashing. In *FSE (2008)*.

[16] Lyubashevsky, V., Peikert, C., and Regev, O. On ideal lattices and learning with errors over rings. In *EUROCRYPT* (2010), pp. 1–23.

[17] Lyubashevsky, V., and Prest, T. Quadratic time, linear space algorithms for Gram-Schmidt orthogonalization and Gaussian sampling in structured lattices. In *EUROCRYPT 2015*.

[18] Micciancio, D., and Peikert, C. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT (2012)*.

[19] Nussbaumer, H. J. *Fast Fourier transform and convolution algorithms*, vol. 2. Springer Science & Business Media, 2012.

[20] Olshevsky, V. *Fast Algorithms for Structured Matrices: Theory and Applications*, vol. 323. American Mathematical Soc., 2003.

[21] Olshevsky, V., and Pan, V. Y. A unified superfast algorithm for boundary rational tangential interpolation problems and for inversion and factorization of dense structured matrices. In *FOCS (1998)*.

[22] Peikert, C. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO* (2010), pp. 80–97.

[23] Sweet, D. R. Fast toeplitz orthogonalization. *Numerische Mathematik 43*, 1 (1984), 1–21.

APPENDIX

A. EXTENDING THE RESULTS TO CYCLOTOMIC RINGS

In this section we argue that our results hold in the cyclotomic case as well. It turns out that all the previous arguments can be made more general. The required ingredients are the following:

1. A tower of unitary rings endowed with inner products onto $\mathbb{R}$.
2. For any rings $\mathcal{S}, \mathcal{T}$ of the tower, injective maps $\mathsf{M}' : \mathcal{S} \to \mathcal{T}^{k \times k}$ and $\mathsf{V}' : \mathcal{S} \to \mathcal{T}^k$, with $\mathcal{S}$ of rank d over $\mathbb{R}$, and $\mathcal{T}$ of rank d/k over $\mathbb{R}$.
3. M' is a ring morphism.
4. V' is a scaled linear isometry.
5. $\mathsf{V}'(ab) = \mathsf{M}'(a)\mathsf{V}'(b)$.
6. Computing V', V'^{-1}, M' and M'^{-1} takes time $\Theta(dk)$.

It remains to prove the existence of such maps for towers of cyclotomic rings. We give explicit constructions in this section, using both our maps from the previous sections and a generic embedding from cyclotomic rings $\mathcal{F}_d$ to convolution rings $\mathcal{R}_d$.

A.1 Cyclotomic Rings

We give brief reminders about cyclotomic polynomials and rings. For $d \in \mathbb{N}^*$, ζ_d denotes an arbitrary primitive d-th root of unity in $\mathbb{C}$, for example $\zeta_d = e^{\frac{2i\pi}{d}}$. $\Omega_d = \{\zeta_d^k | k \in \mathbb{Z}_d^\times\}$ denotes the set of primitive d-th roots of unity. Let

$$\phi_d(x) = \prod_{\zeta \in \Omega_d} (x - \zeta) = \prod_{k \in \mathbb{Z}_d^\times} (x - \zeta_d^k).$$

This polynomial in $\mathbb{Z}[x]$ is called the d-th cyclotomic polynomial. In addition, we define the polynomial ψ_d as follows:

$$\psi_d(x) = \prod_{\zeta^d = 1, \zeta \notin \Omega_d} (x - \zeta) = \prod_{k \in \mathbb{Z}_d \setminus \mathbb{Z}_d^\times} (x - \zeta_d^k).$$

It is immediate that for any d, the degree of ϕ_d is $\varphi(d)$, where $\varphi(d) \triangleq |\mathbb{Z}_d^\times|$ is Euler's totient function. One can also check that $\phi_d(x) \cdot \psi_d(x) = x^d - 1$. To conclude, let $\mathcal{F}_d$ denote the cyclotomic ring $\mathbb{R}[x]/(\phi_d(x))$.

For additional documentation about cyclotomic polynomials, rings and fields, the readers can refer to e.g. [12], Chapter IV.

A.2 Embedding the Ring $\mathcal{F}_d$ in the Ring $\mathcal{R}_d$

We now explicit an embedding of $\mathcal{F}_d$ into $\mathcal{R}_d$.

Definition 10. *Let e_d be the unique element in $\mathcal{R}_d$ such that $e_d = 1 \bmod \phi_d$ and $e_d = 0 \bmod \psi_d$. We define the embedding ι_d from $\mathcal{F}_d$ into $\mathcal{R}_d$ as follows:*

$$\begin{array}{cccc} \iota_d & : & \mathcal{F}_d & \to & \mathcal{R}_d \\ & & f & \mapsto & f \cdot e_d. \end{array}$$

When clear from context, we simply note $\iota = \iota_d$.

Equivalently, $\iota(f)$ is the only element in $\mathcal{R}_d$ satisfying:

$$\iota(f)(\zeta) = \begin{cases} f(\zeta) & \text{if } \phi_d(\zeta) = 0 \\ 0 & \text{if } \psi_d(\zeta) = 0 \end{cases} \tag{4}$$

Proposition 5. *Let $d \in \mathbb{N}^*$ and $\iota = \iota_d$. The embedding ι:*

1. *is an injective ring morphism.*
2. *is an isometry : for any $f, g \in \mathcal{F}_d$, $\langle \iota(f), \iota(g) \rangle = \langle f, g \rangle$.*

Proof. Item 1 follows from the fact that e_d is idempotent $e_d^2 = e_d$. Indeed this implies that $e_d(a + bc) = e_d a + e_d bc = e_d a + e_d^2 bc = e_d a + (e_d b)(e_d c)$. In addition, for any element $g \in \iota(\mathcal{F}_d)$, $g \bmod \phi_d$ is the unique antecedent of g with respect to ι, so ι is bijective and $\iota^{-1}(g) = g \bmod \phi_d$, which proves the point 1. Item 2 follows from equation (4). □

Lemma 5. *Let $d \geqslant 2, d' | d, k = d/d'$ and $a \in \mathcal{R}_d$. Then*

$$(a \in \iota(\mathcal{F}_d)) \Leftrightarrow \mathsf{V}_{d/d'}(a) \in \iota(\mathcal{F}_{d'})^k$$

Proof. We prove the lemma for $d' = \mathrm{gpd}(d)$, extension to the general case is straightforward. a can be uniquely written as $a = \sum_{0 \leqslant i < k} x^i a_i(x^k)$ where each $a_i \in \mathcal{R}_{d'}$. Let ζ_d be an arbitrary d-th primitive root of unity. We recall that $\Omega_d = \{\zeta_d^j | j \in \mathbb{Z}_d^\times\}$ and note $U_d \triangleq \{\zeta \in \mathbb{C} | \zeta^d = 1\} = \{\zeta_d^j | j \in \mathbb{Z}_d\}$. One can check that

$$(\zeta \in U_d \backslash \Omega_d) \Leftrightarrow (\zeta^k \in U_{d'} \backslash \Omega_{d'}), \tag{5}$$

which is immediate by writing $\zeta = \zeta_d^j$, with $j \in \mathbb{Z}_d \backslash \mathbb{Z}_d^\times$. We recall that evaluating a on each $\zeta_d^j \in U_d$ yields the linear system

$$a(\zeta_d^j) = \sum_{0 \leqslant i < k} \zeta_d^{ij} a_i(\zeta_d^{kj}) = \sum_{0 \leqslant i < k} \zeta_d^{ij} a_i(\zeta_{d'}^j). \tag{6}$$

As a step of the FFT (see Lemma 1), the system 6 is invertible. In addition, one can check in equation 5 that if $\zeta \in U_d \backslash \Omega_d$, then $a(\zeta)$ depends only of the $a_i(\zeta')$ for $\zeta' \in U_{d'} \backslash \Omega_{d'}$. Similarly, if $\zeta \in \Omega_d$, then $a(\zeta)$ depends only of the $a_i(\zeta')$ for $\zeta' \in \Omega_{d'}$. So the linear system can be separated in two independent systems. Noting $a(E) \triangleq \{a(e) | e \in E\}$:

$$\begin{aligned} &\left[\begin{array}{c|c} a(\Omega_d) & a(U_d \backslash \Omega_d) \end{array} \right] = \\ &\left[\begin{array}{c|c} (a_i(\Omega_{d'}))_{0 \leqslant i < k} & (a_i(U_{d'} \backslash \Omega_{d'}))_{0 \leqslant i < k} \end{array} \right] \left[\begin{array}{c|c} \mathbf{M}_1 & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{M}_2 \end{array} \right]. \end{aligned} \tag{7}$$

Since the whole system is invertible, both matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ are invertible too. We can conclude that $a(U_{d'} \backslash \Omega_{d'}) = \mathbf{0}^{d - \varphi(d)}$ iff all the $a_i(U_{d'} \backslash \Omega_{d'})$'s are zero too. This is equivalent to saying that $a \in \iota(\mathcal{F}_d)$ iff $\forall i, a_i \in \iota(\mathcal{F}_{d'})$, which proves the lemma. □

A.3 Conclusion for Cyclotomic Rings

We now check that the 6 conditions enounced at the beginning of Section A are verified. For $d' | d$, $\mathcal{F}_{d'}$ and $\mathcal{F}_{d'}$ are unitary rings endowed with the dot product defined in Definition 2, which gives the condition 1. The embeddings ι_d trivialize the construction of maps M' and V' from $\mathcal{F}_d$ to $\mathcal{F}_{d'}$:

$$\mathsf{V}' = \iota_{d'}^{-1} \circ \mathsf{V}_{d/d'} \circ \iota_d \qquad \mathsf{M}' = \iota_{d'}^{-1} \circ \mathsf{M}_{d/d'} \circ \iota_d.$$

This gives the condition 2. Lemma 5 allows to argue that the image of $\mathsf{V}_{d/d'} \circ \iota_d$ is in the definition domain of $\iota_{d'}^{-1}$: V' is well defined, and similarly for M'. Conditions 3 and 5 follow from the fact that $\iota_d, \iota_{d'}$ are ring morphisms and that similar properties hold for $\mathsf{M}_{d/d'}$ and $\mathsf{V}_{d/d'}$. Condition 4 is true because ι_d, $\mathsf{V}_{d/d'}$ and $\iota_{d'}$ are isometries. Finally, condition 6 holds in the FFT representation, from Lemma 1 and from the fact that ι in the Fourier domain simply consist of inserting some zeros at appropriate positions.

B. IMPLEMENTATION IN PYTHON

In this section, we give the core of the Python implementation of our algorithms when d is a power of 2. The full implementation, including correctness tests, is in the Public Domain and is available online at the folliwng link:

`https://github.com/lducas/ffo.py`

Conventions.

In `python.numpy`, the arithmetic operations `+`,`-`,`*`,`/` on arrays denote coefficient-wise operations. The functions `fft` and its inverse `ifft` are built in. The symbol `j` denotes the imaginary unit. The primitive `zeros(d)` creates the `d`-dimensional zero vector.

```python
# Simplified extract of ffo.py
from numpy import *

# Linearize operation V, i/o in fft format
def ffsplit(F):
    d = len(F)
    winv = exp(2j*pi / d)
    Winv = array([winv**i for i in range(d/2)])
    F1 = .5* (F[:d/2] + F[d/2:])
    F2 = .5* (F[:d/2] - F[d/2:]) * Winv
    return (F1,F2)

# Inverse linearize V^-1, i/o in fft format
def ffmerge(F1,F2):
    d = 2*len(F1)
    F = 0.j*zeros(d)     # Force F to complex float
    w = exp(-2j*pi / d)
    W = array([w**i for i in range(d/2)])
    F[:d/2] = F1 + W * F2
    F[d/2:] = F1 - W * F2
    return F

# ffLDL alg., i/o in fft format
# Outputs an L-Tree (sec 3.2)
def ffLDL(G):
    d = len(G)
    if d==1:
        return (G,[])
    (G1,G2) = ffsplit(G)
    L  = G2 / G1
    D1 = G1
    D2 = G1 - L * G1 * conjugate(L)
    return (L, [ffLDL(D1),ffLDL(D2)] )

# ffLQ, i/o in fft format
# outputs an L-Tree (sec 3.2)
def ffLQ(f):
    F = fft(f)
    G = F*conjugate(F)
    T = ffLDL(G)
    return T

# ffNearestPlane, i/o in base B, fft format (sec. 4)
def ffBabai_aux(T,t):
    if len(t)==1:
        return array([round(t.real)])
    (t1,t2) = ffsplit(t)
    (L,[T1,T2]) = T
    z2  = ffBabai_aux(T2,t2)
    tb1 = t1 + (t2-z2) * conjugate(L)
    z1  = ffBabai_aux(T1,tb1)
    return ffmerge(z1,z2)

# ffNearestPlane, i/o in canonical base, coef. format
def ffBabai(f,T,c):
    F = fft(f)
    t = fft(c) / F
    z = ffBabai_aux(T,t)
    return ifft(z * F)
```

C. PROOF OF PROPOSITION 1

PROOF. For any $\mathbf{x}, \mathbf{y} \in \mathcal{R}^m$, let $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{R}} \triangleq \mathbf{x} \cdot \mathbf{y}^\star$. One can check that $\langle \cdot, \cdot \rangle_{\mathcal{R}}$ is a Hermitian inner product. In particular,

$$(\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{R}} = 0) \Leftrightarrow (\langle \mathbf{x}, \mathbf{x} \rangle = 0) \Leftrightarrow (\mathbf{x} = \mathbf{0}).$$

The decomposition $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$ can be computed using the Gram-Schmidt process (Algorithm 5).

Algorithm 5 $\mathsf{GramSchmidt}_{\mathcal{R}}(\mathbf{B})$

1: **for** $i = 1, \ldots, n$ **do**
2: $\quad\tilde{\mathbf{b}}_i \leftarrow \mathbf{b}_i$
3: $\quad$**for** $j = 1, \ldots, i-1$ **do**
4: $\quad\quad L_{i,j} = \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle_{\mathcal{R}}}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle_{\mathcal{R}}}$
5: $\quad\quad \tilde{\mathbf{b}}_i \leftarrow \tilde{\mathbf{b}}_i - L_{i,j} \tilde{\mathbf{b}}_j$
6: $\quad$**end for**
7: **end for**
8: **return** $(\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n\}, \mathbf{L} = (L_{ij})_{1 \leqslant i,j \leqslant n})$

If we replace $\mathcal{R}$ with $\mathbb{R}$ or a number field, it is well-known that Algorithm 5 terminates whenever $\mathbf{B}$ is full-rank, and outputs $(\tilde{\mathbf{B}}, \mathbf{L})$ satisfying equation 1. However, it is less obvious in our case, since $\mathcal{R}$ is no longer a field and the division by $\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle_{\mathcal{R}}$ in step 4 might be problematic.

To show that the output of Algorithm 5 satisfies equation 1 when $\mathcal{R} = \mathbb{R}[x]/(h(x))$, it suffices to show that for any $j \in [\![1, n]\!]$, $\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle_{\mathcal{R}}$ is invertible. Suppose that it is not the case, then there exists $j \in [\![1, n]\!]$, and $a \in \mathcal{R} \backslash \{0\}$ such that $a \langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle_{\mathcal{R}} = 0$. By linearity, $\langle a\tilde{\mathbf{b}}_j, a\tilde{\mathbf{b}}_j \rangle_{\mathcal{R}} = 0$ and therefore $a\tilde{\mathbf{b}}_j = 0$. Since $\tilde{\mathbf{b}}_j = \mathbf{b}_j - \sum_{i<j} L_{i,j} \tilde{\mathbf{b}}_j$, this means that there exists a nonzero linear combination $\sum_{i \leqslant j} a_i \mathbf{b}_i$ equal to zero. Therefore $\mathbf{B}$ is not full-rank, which contradicts the hypothesis of Proposition 1.

Unicity of the decomposition follows from the unicity of the orthogonal projection of a vector onto a $\mathcal{R}$-module. $\square$

Our arguments seamlessly transfer to the termination of Algorithm 1, as the elements D_j in Algorithm 1 are exactly the elements $\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle_{\mathcal{R}}$ in Algorithm 5.

D. PROOF OF PROPOSITION 4

PROOF. We show the properties separately:

1. We first prove this statement for $d' = \mathrm{gpd}(d)$ and for elements $a, b \in \mathcal{R}_d$. All the requirements for showing that M is a homomorphism are trivial, except for the fact that it is multiplicative. First, one can check from Definition 8 that $\mathsf{M}(ab) = \mathsf{M}(a) \cdot \mathsf{M}(b)$. Let $\mathbf{A} = (a_{ij}) \in \mathcal{R}_d^{n\times p}$ and $\mathbf{B} = (b_{ij}) \in \mathcal{R}_d^{p\times m}$. Since $\mathbf{AB} \triangleq (\sum_{1\leqslant k\leqslant p} a_{ik}b_{kj})_{1\leqslant i\leqslant n, 1\leqslant j\leqslant m}$, we have

$$\begin{aligned} \mathsf{M}(\mathbf{AB}) &= \mathsf{M}\left(\left(\textstyle\sum_{1\leqslant k\leqslant p} a_{ik}b_{kj}\right)_{i,j}\right) \\ &= \left(\textstyle\sum_{1\leqslant k\leqslant p} \mathsf{M}(a_{ik})\mathsf{M}(b_{kj})\right)_{i,j} \\ &= \mathsf{M}(\mathbf{A})\mathsf{M}(\mathbf{B}) \end{aligned}.$$

Multiplicity then seamlessly transfers to any $d''|d$:

$$\begin{aligned} \mathsf{M}_{d/d''}(\mathbf{A}\cdot\mathbf{B}) &= \mathsf{M}_{d'/d''} \circ \mathsf{M}_{d/d'}(\mathbf{A}\cdot\mathbf{B}) \\ &= \mathsf{M}_{d'/d''}(\mathsf{M}_{d/d'}(\mathbf{A}) \cdot \mathsf{M}_{d/d'}(\mathbf{B})) \\ &= \mathsf{M}_{d'/d''} \circ \mathsf{M}_{d/d'}(\mathbf{A}) \cdot \mathsf{M}_{d'/d''} \circ \mathsf{M}_{d/d'}(\mathbf{B}) \\ &= \mathsf{M}_{d/d''}(\mathbf{A}) \cdot \mathsf{M}_{d'/d''}(\mathbf{B}). \end{aligned}$$

To show injectivity, it suffices to see that if $d' = \mathrm{gpd}(d)$, then $(\mathsf{M}_{d/d'}(a) = 0) \Leftrightarrow (a = 0)$. From the definition, this property seamlessly transfers to any $d'|d$ and any matrix $\mathbf{A}$.

2. This item is immediate from the definition.

3. It suffices to notice that for any a, $\mathsf{V}(a)$ is the first line of $\mathsf{M}(a)$. As M is a multiplicative homomorphism, the result follows.

4. It suffices to prove it for elements $a, b \in \mathcal{R}_d$ (instead of vectors) and for $d' = \mathrm{gpd}(d)$, the generalization to vectors and to arbitrary values of d' is then immediate. Let $a = \sum_i x^i a_i(x^{\mathrm{gpd}(d)})$, $b = \sum_i x^i b_i(x^{\mathrm{gpd}(d)})$, where $\forall i, a_i = \sum_{j\in\mathbb{Z}_{\mathrm{gpd}(d)}} a_{i,j}x^j$ and $b_i = \sum_{j\in\mathbb{Z}_{\mathrm{gpd}(d)}} b_{i,j}x^j$. Then

$$\langle a, b\rangle_2 \triangleq \sum_{i,j} \langle a_{i,j}, b_{i,j}\rangle_2 = \sum_i \langle a_i, b_i\rangle_2 \triangleq \langle \mathsf{V}(a), \mathsf{V}(b)\rangle_2.$$

5. We have:

$$\begin{aligned} \mathbf{B} \text{ full-rank} &\Leftrightarrow \forall \mathbf{a}, \mathbf{aB} = 0 \text{ iff } \mathbf{a} = 0 \\ &\Leftrightarrow \forall \mathbf{a}, \mathsf{V}(\mathbf{aB}) = 0 \text{ iff } \mathsf{V}(\mathbf{a}) = 0 \\ &\Leftrightarrow \forall \mathbf{a}', \mathsf{V}(\mathbf{a}')\mathsf{M}(\mathbf{B}) = 0 \text{ iff } \mathsf{V}(\mathbf{a}') = 0 \\ &\Leftrightarrow \forall \mathbf{a}', \mathbf{a}'\mathsf{M}(\mathbf{B}) = 0 \text{ iff } \mathbf{a}' = 0 \\ &\Leftrightarrow \mathsf{M}(\mathbf{B}) \text{ full-rank} \end{aligned}$$

The first and last equivalences are simply the definition, the second and fourth uses the fact that V is a vector space isomorphism and the third one uses Proposition 4, item 3.

□

E. PROOF OF LEMMA 2

Proof. Let $\mathrm{C}(k, d)$ denote the complexity of Algorithm 3 over a matrix $\mathbf{G} \in \mathcal{R}_d^{k\times k}$. We have the following recursion formula:

$$\mathrm{C}(n, d) = \Theta(n^2 d \log d) + \Theta(n^3 d) + \Theta(dk_h^2) + n\mathrm{C}(k_h, d_{h-1}), \tag{8}$$

where the first term corresponds to computing the FFT of the n^2 coefficients of $\mathbf{G}$, and the second term to performing $(\mathbf{L}, \mathbf{D}) \leftarrow \mathsf{LDL}^t_{\mathcal{R}_d}(\mathbf{G})$ in FFT form. For each $i \in [\![1, n]\!]$, we know from Lemma 1 that $\mathsf{M}_{d/\mathrm{gpd}(d)}(\mathbf{D}_{ii})$ can be computed in time $\Theta(dk_h^2)$, hence the third term. The last one is for the n recursive calls to itself. We then have

$$\begin{aligned} \mathrm{C}(k_h, d_{h-1}) &= \sum_{1\leqslant i\leqslant h} \tfrac{d}{d_i}\Theta(d_{i-1}k_i^3) + \tfrac{d}{d_1}\mathrm{C}(k_1, d_0) \\ &= \Theta(d) \sum_{1\leqslant i\leqslant h} k_i^2, \end{aligned} \tag{9}$$

where the first equality is shown by induction using equation 8, *except* the first term $\Theta(n^2 d \log d)$ which is no longer relevant since we are already in the Fourier domain. Combining equations 8 and 9, we conclude that the complexity of the whole algorithm is

$$\begin{aligned} \mathrm{C}(n, d) &= \Theta(n^2 d \log d) + \Theta(n^3 d) + n\mathrm{C}(k_h, d_{h-1}) \\ &= \Theta(n^2 d \log d) + \Theta(n^3 d) + \Theta(nd) \sum_{1\leqslant i\leqslant h} k_i^2. \end{aligned}$$

□

F. PROOF OF LEMMA 4

Proof. Let $\mathrm{C}(k, d)$ denote the complexity of Algorithm 4 over input $\mathbf{t} \in \mathcal{R}_d^k$. We have this recursion formula:

$$\mathrm{C}(n, d) = \Theta(nd \log d) + \Theta(n^2 d) + \Theta(ndk_h^2) + n\mathrm{C}(k_h, d_{h-1}),$$

where the first term corresponds to computing the FFT of the n coefficients of $\mathbf{t}$, the second term to performing computing the $\bar{\mathbf{t}}_j$'s (step 8) in FFT form, the third one to the n calls to $\mathsf{V}^{-1}_{d/\mathrm{gpd}(d)}, \mathsf{M}_{d/\mathrm{gpd}(d)}$ (see Lemma 1) and the fourth one to the n recursive calls to itself. We have

$$\begin{aligned} \mathrm{C}(k_h, d_{h-1}) &= \sum_{1\leqslant i\leqslant h} \tfrac{d}{d_i}\Theta(d_{i-1}k_i^3) + \tfrac{d}{d_1}\mathrm{C}(k_1, d_0) \\ &= \Theta(d) \sum_{1\leqslant i\leqslant h} k_i^2, \end{aligned} \tag{10}$$

where the equalities are obtained using the same reasoning as in the proof of Lemma 2. Similarly, we can then conclude that:

$$\mathrm{C}(n, d) = \Theta(nd \log d) + \Theta(n^2 d) + \Theta(nd) \sum_{1\leqslant i\leqslant h} k_i^2.$$

□