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, which allows to perform matrix-vector multiplication in quasi-linear 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 \( \mathbb{R}_d = \mathbb{R}[x]/(x^d - 1) \).

A widely studied and more involved question is matrix decomposition \([2]\) for structured matrices, in particular Gram-Schmidt orthogonalization. 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 arithmetic 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) \([11, 6]\) provides secure trapdoor functions based on lattice problems. This leads to cryptosystems (attribute-based encryption) with fine-grained access control, as \([18, 7]\) 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 \( \mathbf{B} \). The GSO of \( \mathbf{B} \) (see Figure 1). This algorithm requires \( \Theta(d^2) \) arithmetic operations. The orthogonalization process 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 quasi-linear in the dimension.
This is sometimes referred as lattice-based cryptography in 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) allowing fast multiplication of polynomials. But this improvement did not apply in the case of the nearest plane algorithm or its randomized variant: naive GSO seems to break the algebraic structure.

One can circumvent this issue by using the round-off 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 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 $\Theta(d \log d)$ arithmetic operations. 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.

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 $\mathcal{R} \subset \mathcal{R}_{d_1} \subset \cdots \subset \mathcal{R}_{d_{r-1}} \subset \mathcal{R}_d$, for some chain of divisors $1|d_1|\ldots|d_{r-1}|d$.

Such a representation is obtained by applying the (mixed-digit) 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 $L$ 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. For easier algorithmic manipulations, the factorization of $L$ is represented using a tree.

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

Figure 3: Factorization of $L$ in the LDL decomposision of $M_{d/2}(a)$, and its tree representation $\mathcal{L}$.

Related Works.

There exist several works related to the orthogonalization of structured bases. For Toeplitz matrices, Sweet introduced an algorithm faster than the naive orthogonalization by a linear factor. Gragg has shown that for Krylov bases which are bases of the form \{b, r(b), \ldots, r^{d-1}(b)\} – the Levinson recursion allows, when r is an isometric operator, to perform orthogonalization in time $\Theta(d^2)$ instead of $\Theta(d^3)$. This fact was used by Lyubashovsky and Prest to reduce by a linear factor the space complexity of the nearest plane algorithm.

To our knowledge, the only result related to orthogonalization breaking the $\Theta(d^2)$ barrier is the work of van Barel, Heining and Kravanja, which solves Toeplitz linear least squares problems in time $\Theta(d \log^2(d))$. Yet this task (equivalent to a single projection on a subspace) is weaker than the nearest plane problem.
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 previous results from convolution rings to cyclotomic rings, by reducing the latter to the former. Appendix B demonstrates the practical feasibility of our algorithms by presenting a 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 $[a_1, \ldots, a_n]$ denotes the matrix whose rows are the $a_i$'s. $\mathbb{N}$ denotes the set of non-negative integers, and $\mathbb{N}^*$ the set $\mathbb{N}\backslash\{0\}$.

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 $O(d \log d)$ using the fast Fourier transform [3].

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 $a^\ast$ and call conjugate of $a$ the unique element of $\mathcal{R}$ such that for any root $\zeta$ of $h$, $a^\ast(\zeta) = \overline{a(\zeta)}$, where $\overline{\cdot}$ is the usual complex conjugation over $\mathbb{C}$.

- The inner product over $\mathcal{R}$ is $(a, b) \triangleq \sum_{\zeta} h(\zeta) = 0 a(\zeta) \cdot \overline{b(\zeta)}$, and the associated norm is $\|a\| \triangleq \sqrt{(a, a)}$.

We note that the conjugation $\ast$ is well-defined. Indeed, let $\overline{\cdot}$ denote the coefficient-wise conjugation over $\mathbb{C}[x]$. For any $a \in \mathcal{R}$, $\overline{a} - a$ is the null polynomial for any root $\zeta$ of $h$, $\overline{\overline{\sigma(\zeta)} - a^\ast(\zeta)} = \overline{\overline{\sigma(\zeta)}} - \overline{\overline{a(\zeta)}} = (a - \overline{a})(\overline{\zeta}) = 0$. Therefore $a^\ast - a^\ast$ is also the null polynomial so $a^\ast$ is real. In particular, one can check that if $a(x) = \sum_{i \in \mathbb{Z}} a_i x^i \in \mathcal{R}_d$, then $a^\ast(x) = a(1/x) \mod (x^d - 1) = \sum_{i \in \mathbb{Z}} a_i x^{d-i}$.

We extend the conjugation to matrices: if $\mathbf{B} = (b_{ij})_{i,j} \in \mathbb{R}^{n \times m}$, then the conjugate transpose of $\mathbf{B}$ is noted $\mathbf{B}^\ast \in \mathbb{R}^{m \times n}$ and is the transpose of the coefficient-wise conjugation of $\mathbf{B}$.

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

**Definition 3.** Let $m \geq n$ and $\mathbf{B} = (b_{ij})_{i,j} \in \mathbb{R}^{n \times m}$. We say that $\mathbf{B}$ is full-rank (or is a basis) if for any linear combination $\sum_{i \leq n} a_i b_i$, with $a_i \in \mathcal{R}$, we have the equivalence $\left(\sum_i a_i b_i = 0\right) \iff (\forall i, a_i = 0)$.

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 $(b_1, \ldots, b_n) \in (\mathbb{R}^n)^n$, or the full-rank matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ whose rows are the $b_i$'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 $\mathbf{L} \in \mathbb{C}^{n \times n}$ is unit lower triangular if it is lower triangular and has only 1’s on its diagonal. The usual semi-order over $\mathcal{R}$ is as follows: $(a \geq_R b) \iff ([a(\zeta)] \geq_R [b(\zeta)])$ for any root $\zeta$ of $h$.

A matrix $\mathbf{G} \in \mathbb{R}^{n \times n}$ is said to be Hermitian if $\mathbf{G}^\ast = \mathbf{G}$. A Hermitian matrix $\mathbf{G}$ is said to be positive definite if for any $\mathbf{x} \neq 0$, $\mathbf{x} \mathbf{G}^\ast \mathbf{x} > \mathbf{0}$. We say that a matrix $\mathbf{G} \in \mathbb{R}^{n \times n}$ is full-rank Gram (or FRG) if it is full-rank and there exist $m \geq n$ and $\mathbf{B} \in \mathbb{R}^{n \times m}$ such that $\mathbf{G} = \mathbf{B}^\ast \mathbf{B}$. One can show that a matrix is positive definite if and only if it is FRG. However, we believe that the latter notion is more convenient to manipulate from an algorithmic viewpoint, so we rely on it instead of the former.

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 $\mathbf{B} \in \mathbb{R}^{n \times m}$ be a full-rank matrix. $\mathbf{B}$ can be uniquely decomposed as follows:

$$\mathbf{B} = \mathbf{L} \cdot \mathbf{B},$$

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 $\mathbf{L} \mathbf{D} \mathbf{L}^\ast$, where $\mathbf{L} \in \mathbb{R}^{n \times n}$ is unit lower triangular with 1’s on the diagonal, and $\mathbf{D} \in \mathbb{R}^{n \times n}$ is diagonal. It is related to the GSO as for a basis $\mathbf{B}$, there exists a unique GSO decomposition $\mathbf{B} = \mathbf{R} \mathbf{L} \mathbf{D} \mathbf{L}^\ast \mathbf{B}^\ast$. If $\mathbf{G} = \mathbf{B}^\ast \mathbf{B}$, then $\mathbf{G} = \mathbf{L} \cdot (\mathbf{B}^\ast \mathbf{B}) \cdot \mathbf{L}^\ast$ 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:

$$\mathbf{L} \cdot \mathbf{B} \text{ is the GSO of } \mathbf{B} \iff \mathbf{L} \cdot (\mathbf{B}^\ast \mathbf{B}) \cdot \mathbf{L}^\ast \text{ is the LDL decomposition of } \mathbf{B}^\ast \mathbf{B}. $$
Algorithm 1 LDL\(_R\)(G)

**Require:** A full-rank Gram matrix \( G = (G_{ij}) \in \mathbb{R}^{n \times n} \).

**Ensure:** The decomposition \( G = LDL^* \) over \( \mathcal{R} \), where \( L \) is unit lower triangular and \( D \) is diagonal.

1. \( L, D \leftarrow 0_{n \times n} \)
2. for \( i \) from 1 to \( n \) do
3. \( L_{ii} \leftarrow 1 \)
4. \( D_i \leftarrow G_{ii} \sum_{j<i} L_{ij} L^*_j D_j \)
5. for \( j \) from 1 to \( i-1 \) do
6. \( L_{ij} \leftarrow \frac{1}{D_{ii}} \left( G_{ij} - \sum_{k<i} L_{ik} L^*_k D_k \right) \)
7. end for
8. end for
9. return \( ((L_{ij}), \text{Diag}(D_{ii})) \)

Algorithm 1 computes the LDL decomposition. It is well-known that it terminates without encountering divisions by 0 when \( \mathcal{R} \) is replaced by \( \mathbb{R} \). In our case, we prove that all the relevant elements will be invertible 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 \( B \) of a given lattice basis \( B \), as depicted on Figure 1.

**Definition 4.** Let \( B = \{b_1, \ldots, b_n\} \) be a real basis. We call fundamental parallelepiped generated by \( B \) and note \( \mathcal{P}(B) \) the set \( \sum_{1 \leq j \leq n} \left[ -\frac{1}{2}, \frac{1}{2} \right] b_j = \left[ -\frac{1}{2}, \frac{1}{2} \right]^n \cdot B \).

Algorithm 2 NearestPlanes\(_R\)(t, L)

**Require:** The decomposition \( B = L \cdot B \) of \( B \in \mathbb{R}^{n \times m} \), a vector \( t \in \mathbb{R}^n \)

**Ensure:** A vector \( z \in \mathbb{Z}^n \) such that \((t - z)B \in \mathcal{P}(B)\)

1. \( z \leftarrow 0 \)
2. for \( j = n, \ldots, 1 \) do
3. \( t_j \leftarrow t_j + \sum_{i>j} (t_i - z_i) L_{ij} \)
4. \( z_j \leftarrow \lfloor t_j \rfloor \)
5. end for
6. return \( z \)

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

### 2.4 Coefficient Vectors and Circulant Matrices

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

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

\[
C(a) \triangleq \begin{bmatrix}
a_0 & a_1 & \cdots & a_{d-1} \\
a_{d-1} & a_0 & \cdots & a_{d-2} \\
\vdots & \vdots & \ddots & \vdots \\
a_1 & a_2 & \cdots & a_0
\end{bmatrix} \in \mathbb{R}^{d \times d}.
\]

\( c \) and \( C \) generalize to vectors and matrices in a coefficient-wise manner.

**Proposition 3.** The maps \( c \) and \( C \) satisfy the following properties:

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

**Proof:**

1. \( \mathcal{R} \) may not only be seen as a \( d \)-dimensional \( \mathbb{R} \)-vector space, but also as a \( d/d' \)-dimensional \( \mathbb{R}_d \)-module for any \( d' \mid d \).
2. When writing \( c \) as a \( d \)-dimensional \( \mathbb{R} \)-vector space, we will not use \( \{1, x, \ldots, x^{d-1}\} \) 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 \( \text{gcd}(d) \) 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 \( i \in [1, h] \), \( d_{i-1} \triangleq d_i / \text{gcd}(d_i) \) and \( k_i \triangleq d_i / d_{i-1} \), so that \( 1 = d_0 d_1 \cdots d_h = d \) and \( \prod_{j \leq i} k_j = d_i \).

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: for example, \( 1 \mid 6, 1 \mid 2 \mid 6 \) and \( 1 \mid 3 \mid 6 \) 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.

**Definition 7.** Let \( d, d' \in \mathbb{N}^* \) such that \( d' \mid d \) 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 \( V_{d,d'} : \mathcal{R}^m_d \to \mathcal{R}^{km}_{d'} \) recursively as follows:

1. \( V_d = V_{d,d'} = \text{the identity.} \)
When

As highlighted by Figure 4, when

In other words, \( V_{d/d'}(a) \) is the row vector whose coefficients are the \((a_i)_{i \in \mathbb{Z}_d}\).

For a vector \( v \in \mathbb{R}^m_d \), \( V_{d/d'}(v) \in \mathbb{R}^m_d \) is the component-wise applications of \( V_{d/d'} \).

4. For \( d''|d|d \) and any vector \( v \in \mathbb{R}^m_d \),

\[
V_{d/d'}(v) \xrightarrow{\Delta} V_{d''/d'} \circ V_{d/d'}(v) \in \mathbb{R}^{(d''/d')m}_d.
\]

When \( d \) is clear from context, we simply note \( V_{d/d'} = V_{d'} \).

**Interpretation.**

In practice, an element \( a \in \mathbb{R}_d \) 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 \( d = 2^k \) is a power of two, \( V_{d/1} \) permutes the coefficients according to the bit-reversal order \( \pi \) which appears in the radix-2 fast Fourier transform (FFT). More generally, one can show that for an arbitrary \( d \), \( V_{d/1} \) permutes the coefficient according to the general mixed-radix digit reversal order, which appears in the mixed-radix Cooley-Tukey FFT [3].

**Figure 4: Partial vectorial linearizations.**

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

**Definition 8. Following the notations of Definition [4], we define the operator \( M_{d/d'} : \mathbb{R}^{km}_d \rightarrow \mathbb{R}^{km}_d \) as follows:**

1. For \( d = d' = 1 \), \( M_{d/d'} \) is the identity.
2. For \( d' = \gcd(d) \), \( k = d/d' \), and a single element \( a = \sum_{i \in \mathbb{Z}_d} x^i a_i(y) \) where each \( a_i \in \mathbb{R}_d' \), \( M_{d/d'}(a) \) is the following matrix of \( \mathbb{R}^{k \times d}_d' \):

\[
\begin{bmatrix}
    a_0 & a_1 & \cdots & a_{k-1} \\
    y a_1 & a_0 & \cdots & a_{k-2} \\
    y a_2 & y a_1 & \cdots & \vdots \\
    \vdots & \vdots & \ddots & \ddots \\
    y a_{k-1} & y a_{k-2} & \cdots & a_0
\end{bmatrix}
\]

\[
\Rightarrow \begin{bmatrix}
    \mathbb{V}_{d/d'}(a) \\
    \mathbb{V}_{d/d'}(x^k a) \\
    \vdots \\
\end{bmatrix}.
\]

In particular, if \( d \) is prime, then \( M_{d/1}(a) \in \mathbb{R}^{d \times d} \) is exactly the circulant matrix \( C(a) \).

3. For a vector \( v \in \mathbb{R}^m_d \) or a matrix \( B \in \mathbb{R}^{m \times m}_d \), \( M_{d/d'}(v) \in \mathbb{R}^{m \times km}_d \) and \( M_{d/d'}(B) \in \mathbb{R}^{km \times km}_d \) are the component-wise applications of \( M_{d/d'} \).

4. For \( d''|d|d' \) and any element \( a \in \mathbb{R}_d \),

\[
M_{d/d'}(a) \xrightarrow{\Delta} M_{d''/d'} \circ M_{d/d'}(a) \in \mathbb{R}^{(d''/d') \times (d/d')}.
\]

When \( d \) is clear from context, we simply note \( M_{d/d'} = M_{d'} \).

**Interpretation.**

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

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

1. \( M \) is an injective algebra morphism, and in particular:

\[
M(A \cdot B) = M(A) \cdot M(B).
\]

2. \( V \) is an injective linear map.

3. \( V(ab) = V(a) \cdot V(b) \).

4. \( V \) is an isometry:

\[
\langle V(a), V(b) \rangle = \langle a, b \rangle.
\]

5. \( B \) is full-rank if and only if \( M(B) \) 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 \( a \in \mathbb{R}_d \) 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 \((a(\zeta_i))_{i \in \mathbb{Z}_d} \in \mathbb{C}^d \)— then computing \( V_{d/d'}(a) \) and \( M_{d/d'}(a) \) in FFT form can naively be done in time \( \Theta(d \log d) \) by computing its inverse FFT, permuting its coefficients, and computing \( d'/d \) FFTs over \( \mathbb{R}_d' \). However, it can be done faster as it is a single step — also known as butterfly — of the original fast Fourier transform. This is formalized in Lemma [4] a reformulation of a simple lemma that is at the heart of Cooley and Tukey’s FFT.

**Lemma 1** (Adapted). Let \( d \geq 2, d' = \gcd(d) \) and \( k = d/d' \). Let \( V \xrightarrow{\Delta} V_{d/d'} \), \( M \xrightarrow{\Delta} M_{d/d'} \). For any \( a \in \mathbb{R}_d \) (resp. \( A \in \mathbb{M}(\mathbb{R}_d) \)):

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

**Proof.** Let \( a \in \mathbb{R}_d \) be uniquely written \( a = \sum_{i \in \mathbb{Z}_d} x^i a_i(x^k) \), where each \( a_i \in \mathbb{R}_{d'} \). Cooley and Tukey show in [3] (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 \( V(a) \) and \( M(a) \), the result follows. □

Lemma [1] allows us to gain a factor \( \Theta(\log d) \) when computing \( V_{d/d'}(a) \) and \( M_{d/d'}(a) \), compared to a naïve approach. In Sections [3] and [4], we will define algorithms which heavily rely on these operators, and will therefore benefit from this speedup as well.
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 Representation for the LDL Decomposition

Theorem 1. Let $d \in \mathbb{N}$ and $l = d_0 | d_1 | \ldots | d_h = d$ be a tower of proper divisors of $d$. Let $b \in \mathbb{R}^n$ be a full-rank vector. There exists a GSO of $M_{d/l}(b)$ as follows:

$$M_{d/l}(b) = \left( \prod_{i=0}^{h-1} M_{d_{i+1}/d_i}(L_i) \right) \cdot B_0$$

where $B_0 \in \mathbb{R}^{d \times dm}$ is orthogonal, and each $L_i \in \mathbb{R}^{d_{i+1}/d_i \times (d_{i+1}/d_i)}$ is a block-diagonal matrix with unit lower triangular matrices of $\mathbb{R}^{d_{i+1}/d_i \times (d_{i+1}/d_i)}$ as its $d_{i+1}/d_i$ diagonal blocks.

As an example, the matrix $L$ of the GSO of $M_8(b)$ for some $a \in \mathbb{R}^8$ 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 $\mathbb{R}$, with $i < d$. By Proposition item 3, the matrix $B_{h-1} \cong M_{d/d_{h-1}}(b)$ is full-rank too. Using the classical Gram-Schmidt decomposition, we can therefore decompose it as $B_{h-1} = L_{h-1} B_h$.

Let $L_{h-1} \in \mathbb{R}^{d_{i+1}/d_i \times d_{i+1}/d_i}$ and $B_h \in \mathbb{R}^{d_{i+1}/d_i \times k_h}$ and $k_h = \begin{cases} \frac{d}{\gcd(d)} & \text{if } i = 0 \\ \frac{d}{\gcd(k_h, d_{i+1})} & \text{else} \end{cases}$.

$L_{h-1}$ is unit lower triangular and $B_h$ is orthogonal. Noting $B = [B_1, \ldots, B_{k_h}]$, each vector $b_j$ is full-rank and orthogonal to the other $b_j$'s. By inductive hypothesis, they can be decomposed as follows:

$$\forall j \in [1, k_h], M_{d_{h-1}/d_i}(b_j) = \left( \prod_{i=0}^{h-2} M_{d_{i+1}/d_i}(L_{i,j}) \right) \cdot B_j,$$

where each $B_j \in \mathbb{R}^{d_{i+1}/d_i \times d_{i+1}/d_i}$ is unit lower triangular and for $i < h-1$, each $L_i \in \mathbb{R}^{d_{i+1}/d_i \times (d_{i+1}/d_i)}$ is a block-diagonal matrix with unit lower triangular matrices of $\mathbb{R}^{d_{i+1}/d_i \times (d_{i+1}/d_i)}$ as its $d_{i+1}/d_i$ diagonal blocks. To be concise, we now note $M \cong M_{d_{h-1}/d_i}$ and $V \cong V_{d_{h-1}/d_i}$.

We have:

$$M_{d/l}(b) = \left( \prod_{i=0}^{h-1} M_{d_{i+1}/d_i}(L_i) \right) \cdot B_0$$

where $B_0 = [B_1, \ldots, B_{k_h}]$. The first equality simply uses the fact that $M$ is a ring homomorphism (Proposition item 1). The second and third ones are immediate from the definitions. The fourth one uses the inductive hypothesis (equation 2) on each $b_j$ and take $B_0 \cong [B_1, \ldots, B_{k_h}]$. In the fifth equality, we take $L_i \cong \text{Diag}(L_{i,1}, \ldots, L_{i,k_h})$ and just need to check that $B_0$ and $L$ are as stated by the theorem:

- Since each $L_{i,j}$ is block diagonal with $d_{h-1}/d_i$ unit lower triangular diagonal blocks, $L_i$ is block diagonal with $k_h(d_{h-1}/d_i) = d/d_i$ unit lower triangular diagonal blocks.

We also need to show that $B_0$ is orthogonal. Each submatrix $B_j$ of $B_0$ is the orthogonalization of $M(b_j)$ by induction hypothesis. Therefore, for two distinct rows $u, v$ of $B_0$:

$$\langle u, v \rangle = \langle V(u_j), V(v_j) \rangle = \langle V(a_j b_j), V(a_k b_k) \rangle = \langle a_j b_j, a_k b_k \rangle = 0$$

where the second equality comes from Proposition item 3, the third one from the fact that $V$ is a scaled isometry (Proposition item 4) and the fourth one from the fact that $b_j, b_k$ are orthogonal.

Therefore $B_0$ is orthogonal. \qed

The theorem we stated gives the GSO of $M_{d/l}(b)$ for a vector $b \in \mathbb{R}^d$, but can be easily generalized from a vector $b$ to a matrix $B$, and also yields a compact LDL decomposition.

Corollary 1. Let $d \in \mathbb{N}$ and $l = d_0 | d_1 | \ldots | d_h = d$ be a tower of proper divisors of $d$. Let $B \in \mathbb{R}^{n \times m}$ be a full-rank matrix. There exist $h$ matrices $L_i$ of $L_{i+1}$ and $V_{i+1}$ such that:

$$L_h \in \mathbb{R}^{n \times m} \text{ is unit lower triangular.}$$

For each $i < h$, $L_i \in \mathbb{R}^{d_i/d_{i+1} \times n \times m}$ is a block-diagonal matrix whose $n/d_{i+1}$ diagonal blocks are unit lower triangular matrices of $\mathbb{R}^{d_i/d_{i+1} \times d_i/d_{i+1}}$.

Furthermore, if we note $L = \left( \prod_{i=0}^{h} M_{d_{i+1}/d_i}(L_i) \right)$ and $B_0 \cong L^{-1} \cdot M_{d/l}(B_1)$, then:

1. The GSO of $M_{d/l}(B)$ is $L \cdot B_0$.
2. The LDL decomposition of $M_{d/l}(B^*)$ is $M_{d/l}(B) = L \cdot (B_0 B_0^*) \cdot L^*$.\n
Proof. We have $B = L_h B'$, where $L_h$ is given by either the GSO or LDL decomposition algorithm. $B' = \{b_1, \ldots, b_n\}$ is orthogonal. Applying Theorem 1 to each row vector $b_j$ of $B'$ yields $n$ decompositions $L_i = \{L_{i,j}\}_{0 \leq j < h}$ and $n$ orthogonal matrices $B_i$, each spanning the same space as $B_j \cong M_{d/l}(b_j)$. Taking $L_i \cong \text{Diag}(L_{i,j})$ and $B_0 \cong [B_1, \ldots, B_{k_h,0}]$ yields the GSO.

The LDL decomposition is then given “for free” by its equivalence with the GSO, and indeed, one can check that since $B_0$ is orthogonal, $B_0 B_0^*$ is diagonal. \qed
Theorem [3] and Corollary [3] state that for any full-rank matrix $B \in \mathbb{R}^{n \times m}$, the $L$ matrix in the GSO (resp. LDL decomposition) of $M_{d}(B)$ (resp. $M_{d}(BB^*)$) can be represented in a factored form, where each of the factors $L_i$ is a sparse, block-diagonal matrix.

3.2 A Fast Algorithm for the Compact LDL Decomposition

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

Algorithm 3 \texttt{ffLDL}_{\mathcal{R}_d}(G)

Require: A full-rank Gram matrix $G \in \mathbb{R}^{n \times n}$
Ensure: The compact LDL decomposition of $G$
1: $(L, D) \leftarrow \text{LDL}_{\mathcal{R}_d}(G)$
2: if $d = 1$ then
3: return $(L, D)$
4: end if
5: $d' \leftarrow \text{gpd}(d)$
6: for $i = 1, \ldots, n$ do
7: $L_i \leftarrow \text{ffLDL}_{\mathcal{R}_{d'}}(M_{d'/d'}(D_{1i}))$
8: end for
9: return $(L, (L_i)_{1 \leq i \leq n})$


The reason why we favor this approach is because it allows a complexity gain. This gain can already be observed in the form of the LDL decomposition to have a fast Fourier variant. The vectors $\mathbf{z}$ and $\mathcal{R}$ in Algorithm [6] satisfy

$$(\mathbf{z} - \mathbf{t}) \cdot \mathbf{B} = (\mathbf{z} - \mathcal{R}) \cdot \mathbf{B}.$$ (3)

The first and last equalities are trivial, the second one replaces the $\mathcal{R}$'s by their definitions, the third one just simplifies the sum and the fourth one is another way of saying that $L \cdot B = \mathcal{R}$. □

**Lemma 3.** Let $B = \{b_1, \ldots, b_n\} \in \mathbb{R}^{n \times m}$ and $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be its Gram-Schmidt orthogonalization in $\mathbb{R}^d$. The vectors $\mathbf{z}$ and $\mathcal{R}$ in Algorithm [6] satisfy

$$(\mathbf{z} - \mathbf{t}) \cdot \mathbf{B} = (\mathbf{z} - \mathcal{R}) \cdot \mathbf{B}.$$ (3)

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

$$(\mathbf{z} - \mathcal{R}) \cdot \mathbf{B} = \sum_{1 \leq i \leq n}(\mathbf{z}_i - \mathcal{R}_i) \mathbf{b}_j$$

$$= \sum_{1 \leq i \leq n} \left[ (\mathbf{z}_i - \mathbf{t}_i) + \sum_{j > i} L_{ij} (\mathbf{z}_i - \mathbf{t}_i) \right] \mathbf{b}_j$$

$$= \sum_{1 \leq i \leq n} (\mathbf{z}_i - \mathbf{t}_i) \mathbf{b}_j$$

$$= \sum_{1 \leq i \leq n} (\mathbf{z}_i - \mathbf{t}_i) \mathbf{b}_i$$

$$= (\mathbf{z} - \mathbf{t}) \cdot \mathbf{B}. \quad \text{(3)}$$

**Theorem 2.** Let $M \overset{\Delta}{=} M_{d(1)}$ and $V \overset{\Delta}{=} V_{d(1)}$. Algorithm [6] outputs $\mathbf{z} \in \mathbb{Z}^n$ such that $V((\mathbf{z} - \mathbf{t}) \cdot B) \in \mathcal{P}(B_0)$, where $B_0$ is the orthogonalization of $M(B)$ over $\mathbb{R}$.

Proof. The result is trivially true if $d = 1$. We prove it in the general case. By definition, each subtree $L_j$ is the LDL decomposition tree of $M_{d/d'(j)}$ by induction hypothesis, we therefore know that $V((\mathbf{z}_j - \mathcal{R}_j) \mathbf{b}_j) \in \mathcal{P}(B_j)$, where $B_j$ is the orthogonalization of $B_j \overset{\Delta}{=} M(B_j)$. From Lemma [3] we have

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

so $V((\mathbf{z} - \mathbf{t}) \cdot \mathbf{B}) \in \mathcal{P}([B_1, \ldots, B_n])$. Now, from the proof of Corollary [3], we know that $B_0 = [B_1, \ldots, B_n]$ is actually the orthogonalization of $\mathbf{M}(\mathbf{B})$, which concludes the proof. □

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

**Algorithm 4** \texttt{ffNearestPlane}_{\mathcal{R}_d}(t, L)

Require: $t \in \mathbb{R}^{d}$, a precomputed tree $L$, (implicitly) a matrix $B \in \mathbb{R}^{n \times m}$ such that $L$ is the compact LDL decomposition tree of $BB^*$
Ensure: $\mathbf{z} \in Z_d^n$ such that $V((\mathbf{z} - \mathbf{t}) \cdot B) \in \mathcal{P}(B_0)$, where $B_0$ is the orthogonalization of $M(B)$
1: if $d = 1$ then
2: $(L, D) \leftarrow L$
3: return \texttt{NearestPlane}_{\mathcal{R}_d}(L, t)
4: end if
5: $(L, (L_i)_{1 \leq i \leq n}) \leftarrow L$
6: $d' \leftarrow \text{gpd}(d)$
7: for $j = n, \ldots, 1$ do
8: $\mathbf{r}_j \leftarrow \mathbf{t}_j + \sum_{i > j} (\mathbf{t}_j - \mathbf{z}_i) L_{ij}$
9: $\mathbf{z}_j \leftarrow V_{d'/d'}^{-1}([\texttt{ffNearestPlane}_{\mathcal{R}_{d'}}(V_{d'/d'}(\mathcal{R}_j), L))]$
10: end for
11: return $\mathbf{z} = (z_1, \ldots, z_n)$

The last lines are the classical iterative counterpart Algorithm [2], but runs $\Theta(d)$ times faster.
An ongoing execution of Algorithm 4 for a single \( t \in \mathcal{R}_s \).

1. Arrows are labeled in their order of execution. Downward (\( \downarrow \)) and upward (\( \uparrow \)) arrows correspond to the recursive calls (step 9). Transverse (\( \curvearrowright \)) arrows correspond to step 8.

2. Cells labeled with \( R \) (as in \( \text{Rounded} \)) correspond to already completed subcalls of Algorithm 4 as opposed to those labeled with \( NR \) (as in \( \text{Not Rounded} \)).

**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 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 formal proof is given in Lemma 4.

**Lemma 4.** 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, and for \( i \in [1, h] \), let \( k_i = \frac{d_i}{d_{i-1}} \). Let \( B \in \mathcal{R}_{m \times m}^d \) and \( \Sigma \) be its LDL decomposition tree. The complexity of Algorithm 4 is given by:

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

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.

## 5. REFERENCES


## 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 $S, T$ of the tower, injective maps $M' : S \to T^k \times k$ and $V' : S \to T^k$, with $S$ of rank $d$ over $\mathbb{R}$, and $T$ of rank $d/k$ over $\mathbb{R}$.
3. $M'$ is a ring morphism.
4. $V'$ is a scaled linear isometry.
5. $V'(ab) = M'(a)V'(b)$.
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 $F_d$ to convolution rings $R_d$.

### A.1 Cyclotomic Rings

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

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

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

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

It is immediate that for any $d$, the degree of $\phi_d$ is $\varphi(d)$, where $\varphi(d) = |\mathbb{Z}_d^\ast|$ is Euler's totient function. One can also check that $\phi_d(x) \cdot \psi_d(x) = x^d - 1$. To conclude, let $F_d$ 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 $F_d$ in the Ring $R_d$

We now explicit an embedding of $F_d$ into $R_d$.

**Definition 10.** Let $c_d$ be the unique element in $R_d$ such that $c_d = 1 \mod \phi_d$ and $c_d = 0 \mod \psi_d$. We define the embedding $\iota_d$ from $F_d$ into $R_d$ as follows:

$$\iota_d : F_d \to R_d, \quad f \mapsto f \cdot c_d.$$ 

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

Equivalently, $\iota(f)$ is the only element in $R_d$ satisfying:

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

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

1. is an injective ring morphism.
2. is an isometry: for any $f, g \in F_d$, $\langle \iota(f), \iota(g) \rangle = \langle f, g \rangle$.

**Proof.** Item 1 follows from the fact that $c_d$ is idempotent $c_d^2 = c_d$. Indeed this implies that $e_d(a + bc) = e_d(a + c_d bc) = e_d(a + c_d a + (a c_d)(b c_d))$. In addition, for any element $g \in \mathbb{C} F_d$, mod $\phi_d$ is the unique antecedent of $g$ with respect to $\iota$, so $\iota$ is injective and $\iota^{-1}(g) = g \mod \phi_d$, which proves the point [1]. 

**Lemma 2** follows from equation (4).

**Lemma 5.** Let $d \geq 2, d | d', k = d/d'$ and $a \in R_d$. Then

$$\langle (a \in \iota(F_d)) \mathfrak{V}_{d/d'}(a) \in \iota(F_{d'}) \rangle^k$$

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

$$(\zeta \in U_d \setminus \Omega_d) \Leftrightarrow (\zeta^d \in U_{d'} \setminus \Omega_{d'})$$

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

$$a(\zeta_d^j) = \sum_{0 \leq k < d} \zeta_d^j \mathfrak{a}_k(\zeta_d^d) = \sum_{0 \leq k < d'} \zeta_d^j \mathfrak{a}_k(\zeta_d^d).$$

As a step of the FFT (see Lemma 1), the system 6 is lent to saying that $\zeta_d^j$ are independent systems. Noting $\mathfrak{a}(E) \triangleq \{ \mathfrak{a}(e) | e \in E \}$:

$$\begin{bmatrix} a(\Omega_d) \\ a(U_d \setminus \Omega_d) \end{bmatrix} = \begin{bmatrix} \mathfrak{a}(U_{d'} \setminus \Omega_{d'}) & a(\Omega_d) \\ a(U_d \setminus \Omega_d) & a(U_{d'} \setminus \Omega_{d'}) \end{bmatrix} \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}.$$ (7)

Since the whole system is invertible, both matrices $M_1$ and $M_2$ are invertible too. We can conclude that $a(\Omega_{d'} \setminus \Omega_d) = 0^{d' - d}$ if all the $a_k(U_{d'} \setminus \Omega_d)'s$ are zero too. This is equivalent to saying that $a \in \mathbb{C} F_d$ if $\forall i, a_i \in \mathbb{C} F_{d_i}'$, 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 $d | d'$, $F_{d'}$ and $F_d$ are unitary rings endowed with the dot product defined in Definition 2 which gives the condition [1]. The embeddings $\iota_d$ trivialize the construction of maps $M'$ and $V'$ from $F_d$ to $F_{d'}$:

$$V' = \iota_{d'}^{-1} \circ V_{d/d'} \circ \iota_d, \quad M' = \iota_{d'}^{-1} \circ M_{d/d'} \circ \iota_d.$$ 

This gives the condition [2]. Lemma [3] allows to argue that the image of $V_{d/d'} \circ \iota_d$ is in the definition domain of $\iota_{d'}^{-1} \circ V'$ is well defined, and similarly for $M'$. Conditions [5] and [6] follow from the fact that $\iota_d, \iota_{d'}$ are ring morphisms and that similar properties hold for $M_{d/d'}$ and $V_{d/d'}$. Condition [4] is true because $\iota_d, V_{d/d'}$ and $\iota_{d'}$ are isometries. Finally, condition [2] holds in the FFT representation, from Lemma [4] and from the fact that $\zeta$ 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, will be made available on-line and placed in the Public Domain.

Conventions.

In python, the arithmetic operations +,−,*,/ on arrays 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
def ffLDL(G):
    if d=1:
        return (G,[])
    (G1,G2) = ffsplit(G)
    L = G2 / G1
    D1 = G1
    D2 = G1 - L * G1
    return (L, [ffLDL(D1),ffLDL(D2)] )

# ffLQ, i/o in fft format
# outputs an L-Tree (sec 3.2)
def ffLQ(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)
    tbl = t1 + (t2*z2) * conjugate(L)
    z1 = ffBabai_aux(T1,tbl)
    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 \( x,y \in \mathbb{R}^m \), let \( \langle x,y \rangle_R \triangleq x \cdot y^* \). One can check that \( \langle \cdot, \cdot \rangle_R \) is a Hermitian inner product. In particular, \( \langle x,x \rangle_R = 0 \Leftrightarrow \langle x \rangle = 0 \Leftrightarrow \langle x \rangle = 0 \).

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

Algorithm 5 GramSchmidt_R(B)

1: for i = 1,...,n do
2: \( b_i \leftarrow b_i \)
3: for j = 1,...,i-1 do
4: \( L_{i,j} = (b_i,b_j)_R \)
5: \( b_i \leftarrow b_i - L_{i,j}b_j \)
6: end for
7: end for
8: return \( B = \{b_1,\ldots,b_n\}, L = (L_{i,j})_{1 \leq i,j \leq n} \)

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

To show that the output of Algorithm 5 satisfies equation 1.

When \( \mathbb{R} = \mathbb{R}[x]/(h(x)) \), it suffices to show that for any \( j \in [1,n], \langle b_i, b_j \rangle_R \) is invertible. Suppose that it is not the case, then there exists \( j \in [1,n] \), and \( a \in \mathbb{R} \) such that \( a\langle b_j, b_i \rangle_R = 0 \). By linearity, \( \langle ab_j, a\bar{b}_j \rangle_R = 0 \) and therefore \( a\bar{b}_j = 0 \). Since \( b_j = b_j - \sum_{i \neq j} L_{i,j}b_j \), this means that there exists a nonzero linear combination \( \sum_{i \neq j} a_i b_i \) equal to zero. Therefore \( B \) is not full-rank, which contradicts the hypothesis of Proposition 5.

Unicity of the decomposition follows from the unicity of the orthogonal projection of a vector onto a \( \mathbb{R} \)-module.

D. PROOF OF PROPOSITION 4

Proof. We show the properties separately:

1. We first prove this statement for \( d' = \text{gpd}(d) \) and for elements \( a,b \in \mathbb{R}_d \). 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 5 that \( M(ab) = M(a) \cdot M(b) \). Let \( A = (a_{ij}) \in \mathbb{R}_d^{\times n} \) and \( B = (b_{ij}) \in \mathbb{R}_d^{p \times m} \). Since
where the first equality is shown by induction using equation \[8\] except the first term \(\Theta(n^2d\log d)\) 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

\[
C(n, d) = \Theta(n^2d\log d) + \Theta(n^3d) + nC(k_h, d_{h-1})
\]

\[
= \Theta(n^2d\log d) + \Theta(n^3d) + \Theta(nd) \sum_{1\leq i\leq h} k_i^2.
\]

\[
\square
\]

### F. PROOF OF LEMMA \[4\]

**Proof.** Let \(C(k, d)\) denote the complexity of Algorithm \[4\] over input \(t \in \mathbb{R}_{\mathbb{R}}^{k}\). We have this recursion formula:

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

where the first term corresponds to computing the FFT of the \(n\) coefficients of \(t\), the second term to performing the \(t_i\)'s \(\ell\)'s \(8\) in FFT form, the third one to the \(n\) calls to \(V_d^{(j)}{\mathfrak{gpd}}(d)\), the fourth one to the \(n\) recursive calls to itself. We have

\[
C(k_h, d_{h-1}) = \sum_{1\leq i\leq h} \frac{d}{d_i}{\mathfrak{gpd}}(d_{i-1}k_i^2) + \frac{d}{d_i}C(k_1, d_0)
\]

\[
= \Theta(d) \sum_{1\leq i\leq h} k_i^2,
\]

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

\[
C(n, d) = \Theta(nd\log d) + \Theta(n^2d) + \Theta(nd) \sum_{1\leq i\leq h} k_i^2.
\]

\[
\square
\]