# Fast Fourier Orthogonalization

**(and Applications to Lattice-Based Cryptography)**

Léo Ducas[1⋆] and Thomas Prest[2,3]

[1] Cryptology Group, CWI, Amsterdam
[2] École Normale Supérieure, Paris
[3] Thales Communications & Security, Gennevilliers
l.ducas@cwi.nl,thomas.prest@ens.fr

**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 a circulant matrix. We show that, when $n$ is composite, it is possible to proceed to the orthogonalization in an inductive way, leading to a structured Gram-Schmidt decomposition. In turn, this structured Gram-Schmidt decomposition accelerates a cornerstone lattice algorithm: the Nearest Plane algorithm. The 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

The nearest plane algorithm [Bab86] is a central algorithm over lattices. It allows, 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 [LLL82], and can be used for error correction over analogical noisy channels. It also found applications in lattice-based cryptography as a decryption algorithm, and a randomized variant (called discrete Gaussian sampling) [Kle00,GPV08] provides secure trapdoor functions based on lattice problems. This leads to cryptosystems (identity-based and attribute-based encryption) with fine-grained access control [CHKP10,MP12,Boy13,GVW13].

Given a basis $\mathbf{B}$ of a lattice $L \subset \mathbb{R}^d$, this algorithm reduces a target vector modulo $L$. The results belongs to a fundamental domain centered

⋆ Supported by an NWO Free Competition Grant.

in 0, whose shape is the cuboid defined by $\tilde{\mathbf{B}}$, the Gram-Schmidt orthogonalization (GSO) of $\mathbf{B}$. Equivalently, it finds the unique lattice point in this cuboid centered at the target. This algorithm requires a quadratic number of arithmetic operation in the dimension $d$. The orthogonalization process itself is required as a precomputation, and costlier as it performs a cubic number of arithmetic operations.

When it comes to using this algorithm for practical cryptography, a quadratic cost is rather prohibitive, considering the lattices at hand have dimensions ranging in the hundreds. For efficiency purposes, many cryptosystems (such as [HPS98,LMPR08,LPR10] to name a few) chose to rely on lattices with some algebraic structure, improving time and memory requirements by a factor quasi-linear in the dimension. This is sometimes referred as lattice-based cryptography in the *ring-setting*. The core of this optimization is the Fast-Fourier Transform (FFT) [CT65,GS66,HJB84,Nus12] allowing fast multiplication of polynomials. But this improvement doesn't apply in the case of the nearest plane algorithm or its randomized variant [Kle00,GPV08]: Gram-Schmidt orthogonalization *seems* to break the algebraic structure.

To circumvent this issue, Peikert [Pei10] proposed to switch to a simpler algorithm, the so-called round-off algorithm [Len82,Bab86], and using a perturbation technique, properly randomized it to obtain a secure discrete Gaussian sampler. This simpler algorithm is compatible with FFT-based acceleration techniques, but this comes at another price: the *quality* of the solution is affected. In short, instead of reducing the target to the cuboid associated with $\tilde{\mathbf{B}}$, this algorithm instead provides an output in the parallelepiped associated with $\mathbf{B}$: the fundamental domain has been skewed. This is pictured in Figure 1.

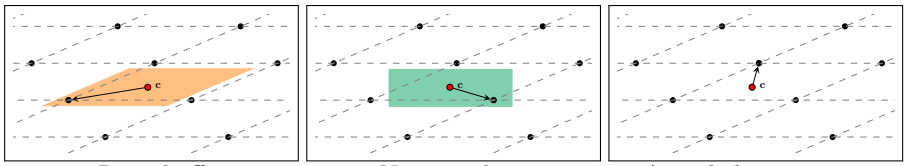



**Fig. 1.** Round-Off and Nearest Plane, and their fundamental domains.

*Quality in the ring setting.* For this discussion, let us focus on a simple case: let $\mathbf{B} \in \mathbb{R}^{d \times d}$ be a *circulant* matrix of first row vector $\mathbf{b}_1 = (f_0, \ldots, f_{d-1})$. Algebraically, it is the matrix associated to the multiplication by the

element $f = \sum f_i \cdot x^i$ of the *circular convolution ring* $\mathcal{R} = \mathbb{R}[x]/(x^d - 1)$. The typical case encountered in cryptography deals with *cyclotomic rings* instead—and involves matrices with several blocks of this form—we will address this point later.

The question is to quantify the loss of quality when switching from the slow nearest plane to the fast round-off algorithm and their respective randomized variants, Klein's sampler and Peikert's sampler. We measure this as the average euclidean length of its output: the lower the better. This is summarized in Table 1. The factor $s_1(\mathbf{B})$ denotes the largest singular

| Algorithm | Nearest plane | Round-off | Klein | Peikert |
|---|---|---|---|---|
| Quality | $\sqrt{\frac{1}{12}\sum_i \|\tilde{\mathbf{b}}_i\|^2}$ | $\sqrt{\frac{d}{12}} \cdot \|\mathbf{b}_1\|$ | $\sqrt{d}\,\|\mathbf{b}_1\|\eta_\epsilon(\mathbb{Z})$ | $\sqrt{d}\,s_1(\mathbf{B})\eta_\epsilon(\mathbb{Z})$ |

**Table 1.** Average output length for random target ($\mathbf{B}$ circulant of first row $\mathbf{b}_1$).

value of $\mathbf{B}$, which is also its spectral norm: $s_1(\mathbf{B}) = \max \|\mathbf{xB}\|/\|\mathbf{x}\|$. The factor $\eta_\epsilon$ is called the smoothing parameter [MR07], and some effort has been done to tackle it [DLP14,BLP+13,BLL+15], but the details are out of the scope of this paper. For our matter, note that we have the inequalities:

$$\sqrt{\sum_i \|\tilde{\mathbf{b}}_i\|^2} \leqslant \sqrt{d} \cdot \|\mathbf{b}_1\| \leqslant s_1(\mathbf{B}),$$

and equalities holds if and only if the matrix $\mathbf{B}$ is orthogonal. We call orthogonality defect the ratio $\delta(\mathbf{B}) = s_1(\mathbf{B})/\sqrt{d} \cdot \|\mathbf{b}_1\|$—alternatively one can consider $\delta'(\mathbf{B}) = \sqrt{d}\|\mathbf{b}_1\|/\sqrt{\sum_i \|\tilde{\mathbf{b}}_i\|^2}$.

The theory of random matrices (see [Ver10, Thm. 5.32] and [BY93]) predicts that the orthogonality defect $\delta(\mathbf{B})$ should tend to $\sqrt{2}$ as the dimension $d$ grows, if the entries of $\mathbf{B}$ are independent gaussians. But should it be the case for circulant matrices ? A simple experiment (see Fig. 2) shows it does not seem likely!

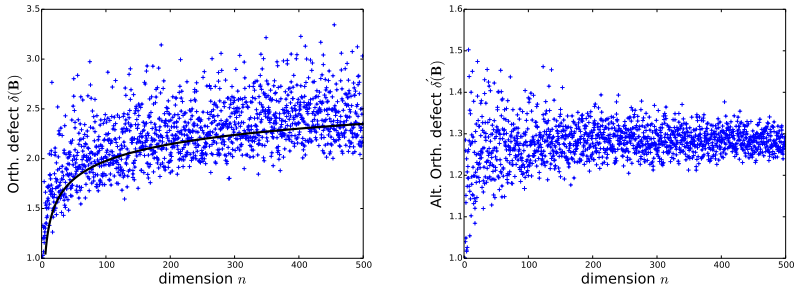



**Fig. 2.** Experimental orthogonality defect of circulant matrices, and prediction (1)

This phenomenon is easily explained by switching to an algebraic point of view. Indeed, the singular values of a circulant matrix $\mathbf{B}$ are exactly the magnitudes of the FFT coefficients of $f$: the FFT diagonalizes the matrix $\mathbf{B}$, and this is the very reason why FFT allows fast multiplication in those rings. Also, the FFT is a scaled isometry: if the coefficients of $f$ are independent gaussians of variance $\sigma^2$, then the FFT coefficients $\hat{f}_i = \sum_j f_j \zeta^{ij}$ have both their real and imaginary parts being independent gaussian as well with variance $d\sigma^2/2$—except for $\hat{f}_0$. Their magnitudes $|\hat{f}_i|$ follow a Rayleigh distribution of parameter $\varsigma = \sqrt{d\sigma^2}$, whose density over $[0, \infty)$ is given by $\frac{x}{\varsigma^2} \exp\left(\frac{-x^2}{2\varsigma^2}\right)$. Out of $d/2$ (half may be ignored as conjugates of the other half) such samples, we expect the largest one to be as large as $\approx \varsigma\sqrt{\log d/2}$. Hence

$$\delta(\mathbf{B}) \approx \sqrt{\log(d/2)}, \tag{1}$$

and this closely matches our experiment.

*Contribution.* In this work, we discover chimeric algorithms, obtained by crossing Cooley-Tukey's [CT65] Fast-Fourier Transform algorithm together with an orthogonalization and nearest plane algorithms (not exactly Gram-Schmidt orthogonalization, but rather LDL decomposition). Precisely, we show that the nearest plane algorithm can be performed using quasi-linear arithmetic operations for matrices with the appropriate algebraic structures, coming from the $d$-th circular convolution ring when $d$ is composed of small factors. The precomputed orthogonalization can also be done using quasi-linear many operations and produces the LDL decomposition in a special compact format, requiring $O(d \log d)$ complex numbers.

At the core of this technique is the realization that the circulant representation of matrices is not the appropriate one. Instead, switching to (mixed-)digit-reversal-order allows to represent the matrix $\mathbf{L}$ of the Gram-Schmidt decomposition in a compact inductive format, namely a tree of structured matrices, that get smaller and smaller as one reaches the leaves. Once this hidden structure is unveiled (Theorem 1), the algorithmic implications follow quite naturally. As a demonstration of the simplicity of those algorithms, we also propose an implementation in `python` in the power-of-two case. The code is strikingly short.

Most of the material of this paper is very well known, but the authors are unaware of any work proposing such a combination. We chose to go very thoroughly through the details of both orthogonalization and Fast-Fourier Transform, to observe how nicely those two operations fit together.

*Gaussian Sampling.* We will only present our acceleration for the (deterministic) nearest plane algorithm. Generalizing our technique to the randomized algorithm of Klein is just a matter of replacing each call to the rounding function $\lfloor \cdot \rceil$, by an appropriately randomized rounding function—following a discrete Gaussian distribution over the integers.

*Impact for lattice-based cryptography.* The structure of matrices found for lattice-based cryptography follow from cyclotomic rings $\mathcal{F}_d = \mathbb{R}[x]/(\phi_d(x))$, and often $d$ is chosen as power of 2. In fact, our algorithm can be generalized to handle such a case, because there is a ring morphism from $\mathcal{F}_d$ to $\mathcal{R}_d$ that is also an isometry. We merely chose the circulant-convolution ring as it makes our exposition significantly simpler.

Our work makes the nearest plane algorithm reach the same time complexity as the round-off algorithm for circulant matrices. The same phenomenon occurs to their randomized variants, the Gaussian samplers.

For samplers, the factor $\sqrt{\log(d/2)}$ to be saved may seem small, but it turns out that in practice the security of a cryptosystem is extremely sensitive to the quality of the Gaussian sampler. A recent work [DP15] illustrated this phenomenon: for the same parameters, the signature scheme studied would have about 128 bits of security using Peikert's sampler, against up to 192 bits of security using Klein's sampler, and a fast hybrid sampler was proposed, reaching 160 bits of security. In this study case, the seemingly anecdotal factor corresponded to 64 bits of security.

*Related work.* The first Gaussian sampler is due to Klein [Kle00], as a tool to solve the bounded-distance-decoding problem. Its use as a secure

trapdoor algorithm was discovered and analyzed in the seminal work of Gentry, Peikert and Vaikuntanathan [GPV08], opening new horizons for lattice-based cryptography. In [LP15] it is showed how to decrease the pre-computation from cubic to quadratic in the ring setting, and how to store this pre-computation compactly without slowing the nearest plane algorithm. Alternatively, a FFT-compatible Gaussian Sampler was proposed by Peikert [Pei10] with quasilinear complexity but decreased quality. A trade-off between Klein and Peikert algorithms was described in [DP15], together with a study case of its impact for signature schemes based on NTRU [HPS98] trapdoors. Optimization of the trapdoor basis generation (with a security reduction to a worst-case lattice problem [Ajt96,Reg05], unlike the trapdoors of [HPS98]) can be found in [AP11,MP12]. The question of fixed-point/floating-point precision was studied in [DN12,Duc13], and optimized subroutines for sampling in 1-dimensional lattices were developed in [DN12,DDLL13,Kar13,Duc13,BCG$^+$14,RVV14,DG14,Lep14]. Improvements on the smoothing parameter using statistical tools can be found in [DLP14,BLL$^+$15], and a dedicated algorithm was developed in [BLP$^+$13].

*Complexity.* All the complexities are expressed in terms of number of arithmetic operation over the reals. Determining the required precision for a floating-point approximation and therefore the bit-complexity of our algorithm is left for future works as this issue seems rather orthogonal to our improvements. As a starting point in that direction, one might remark that the asymptotic relative error growth of the FFT of a degree-$d$ polynomial is $O(\log d)$ in the worst case and $O(\sqrt{\log d})$ in the average case [GS66,Sch96]. As our algorithms are structurally very close to the FFT, it would be interesting to see if they benefit from the same error growth.

*Open Problems.* On the theoretical side, a interesting problem would be to apply this trick to the LLL [LLL82] algorithm, even if it weakens a bit the notion of reduction achieved. On the practical side, it would of course be nice to see a secure and fast implementation of lattice trapdoors based on this algorithm, combining the technique of [BLP$^+$13,BLL$^+$15] to tackle the smoothing parameter. This would have a wide impact, since many advanced cryptosystems as ABEs [CHKP10,MP12,Boy13,GVW13] become rather easy to implement once this primitive is provided.

*Organization.* The paper is organized as follows. First, Section 2 introduces the mathematical tools that we will use through this paper. Section 3

shows that for matrices over convolution rings, the LDL decomposition can be expressed in a compact, factorized form, and gives a "Fast Fourier" 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 a "Fast Fourier" structure. Section 5 extends all our previous results from convolution rings to cyclotomic rings, by reducing the latters to the formers. As a conclusion, Section 6 demonstrates the practical feasibility of our algorithms by presenting 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 $\mathcal{R}$. Elements of $\mathcal{R}$ will be usually noted in plain letters (such as $a, b$), vectors with coefficients in $\mathcal{R}$ will be noted in bold letters (such as $\mathbf{a}, \mathbf{b}$) and matrices with coefficients in $\mathcal{R}$ will be noted in capital bold letters (such as $\mathbf{A}, \mathbf{B}$). Vectors are mostly in row notation, and as a consequence vectors-matrix product is done in this order when not stated otherwise. $(a_1, ..., a_n)$ denotes the row vector formed of the $a_i$'s, whereas $[\mathbf{a}_1, ..., \mathbf{a}_n]$ denotes the matrix whose rows are the $\mathbf{a}_i$'s.

*Notation Summary.* This paper contains a fairly large number of notations, even after the preliminaries. To make its comprehension easier to the reader, Table 2 summarizes all the notations we use.

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

**Definition 1.** *For any $d \in \mathbb{N}^\star$, we note $\mathcal{R}_d$ 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 [CT65].

We equip the ring $\mathcal{R}_d$ with a conjugation operation as well as an inner product, giving it an inner product space structure over $\mathbb{R}$. The definitions that we give also encompass other types of rings that will be used in later sections of this paper.

**Definition 2.** *Let $h \in \mathbb{Q}[x]$ 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 conjugate of $a$ the unique element of $\mathcal{R}$ such that for any root $\zeta$ of $h$, $a^\star(\zeta) = \overline{a(\zeta)}$, where $\bar{\cdot}$ is the usual complex conjugation over $\mathbb{C}$.*

**Table 2.** Summary of notations and abbreviations

| Notation | Brief definition | 1st occurence |
|---|---|---|
| log | The natural logarithm | Introduction |
| $\mathcal{R}_d$ | The convolution ring $\mathbb{R}[x]/(x^d-1)$. | Def. 1 |
| $a^\star$ | The conjugate (transpose) of $a$.$^\dagger$ | Def. 2 |
| $c(a)$ | The coefficients' vector of $a$.$^\dagger$ | Def. 4 |
| $\mathcal{C}(a)$ | The circulant matrix of $a$.$^\dagger$ | Def. 4 |
| $\mathrm{gpd}(d)$ | The greatest proper divisor of $d$. | Def. 5 |
| $\mathsf{V}_{d\backslash d'}(a)$ | The vectorization operator.<br>"Breaks" $a \in \mathcal{R}_d$ into a vector in $\mathcal{R}_{d'}^{d/d'}$.$^\dagger$ | Def. 6 |
| $\mathsf{M}_{d\backslash d'}(a)$ | The matrixification operator.<br>"Breaks" $a \in \mathcal{R}_d$ into a matrix in $\mathcal{R}_{d'}^{(d/d')\times(d/d')}$.$^\dagger$ | Def. 7 |
| LTU | Lower triangular unit matrix. | Def. 8 |
| FRG | Full-rank Gram matrix. | Def. 9 |
| $\mathcal{P}(\mathbf{B})$ | The fundamental parallelepiped $[-1/2, 1/2]^n \cdot \mathbf{B}$. | Def. 10 |
| $\mathcal{Z}_d$ | The ring $\mathbb{Z}[x]/(x^d-1)$. | Def. 11 |
| $\Omega_d$ | The set of $d$-th primitive roots of unity. | Def. 12 |
| $\zeta_d$ | A $d$-th primitive root of unity (e.g. $e^{\frac{2i\pi}{d}}$). | Def. 12 |
| $\mathbb{Z}_d^\times$ | The invertible elements of the ring $\mathbb{Z}_d \triangleq \mathbb{Z}/d\mathbb{Z}$. | Def. 12 |
| $\phi_d$ | The cyclotomic polynomial $\prod_{\zeta\in\Omega_d}(x-\zeta)$. | Def. 12 |
| $\psi_d$ | The polynomial $\prod_{\zeta^d=1,\zeta\notin\Omega_d}(x-\zeta)$. | Def. 12 |
| $\varphi(d)$ | Euler's totient function on $d$: $\lvert\mathbb{Z}_d^\times\rvert$. | Def. 13 |
| $\mathcal{F}_d$ | The cyclotomic ring $\mathbb{R}[x]/(\phi_d(x))$. | Def. 13 |
| $\iota_d$ | An inner-product preserving embedding of $\mathcal{F}_d$ in $\mathcal{R}_d$. | Def. 14 |

$^\dagger$This operation extends to vectors and/or matrices.

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

One can check that if $a(x) = \sum_{d\in\mathbb{Z}_d} a_i x^i \in \mathcal{R}_d$, then

$$a^\star(x) = a(1/x) \mod (x^d-1) = \sum_{d\in\mathbb{Z}_d} a_i x^{d-i}$$

We extend the conjugation to matrices: if $\mathbf{B} = (b_{ij})_{i,j} \in \mathcal{R}^{n\times m}$, then the conjugate transpose of $\mathbf{B}$ is noted $\mathbf{B}^\star \in \mathcal{R}^{m\times n}$ and is the transpose of the coefficient-wise conjugation 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 $f = \sum_{0\leqslant i<d} f_i x^i \in \mathcal{R}_d$, the vector $(f(\zeta))_{\{\zeta^d=1\}}$ can be obtained from the coefficients' vector $(f_i)_{0\leqslant i<d}$ by multiplying it by the

Vandermonde matrix $\mathbf{V}_d = (\zeta_d^{ij})_{0 \leqslant i,j < d}$. $\mathbf{V}_d$ verifies $\mathbf{V}_d \mathbf{V}_d^\star = d \cdot \mathbf{I}_d$ and as an immediate consequence: $\langle f, g \rangle = d \cdot \langle f, g \rangle_2$.

**Definition 3.** *Let $m \geqslant n$ and $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathcal{R}_d^{n \times m}$. 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 $a_i \in \mathcal{R}_d$, we have the equivalence $(\sum_i a_i \mathbf{b}_i = 0) \iff (\forall i, a_i = 0)$.*

We note that since $\mathcal{R}_d$ is 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 $\{\mathbf{b}_1, ...\mathbf{b}_n\} \in (\mathcal{R}^m)^n$, or the full-rank matrix $\mathbf{B} \in \mathcal{R}^{n \times m}$ whose rows are the $\mathbf{b}_i$'s.

### 2.2 Coefficient Vectors and Circulant Matrices

**Definition 4.** *We define the coefficient vector $c : \mathcal{R}_d^m \to \mathbb{R}^{dm}$ and the circulant matrix $\mathcal{C} : \mathcal{R}_d^m \to \mathbb{R}^{dn \times dm}$ as follows. For any $a = \sum_{0 \leqslant i < d} a_i x^i \in \mathcal{R}_d$ where each $a_i \in \mathbb{R}$:*

1. *The coefficients' vector of $a$ is $c(a) = (a_0, ..., a_{d-1}) \in \mathbb{R}^d$.*
2. *The circulant matrix of $a$ is*

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

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

We give a few properties which are either folklore or trivial to verify.

**Proposition 1.** *The coefficients' vector and the circulant matrix verify the following properties:*

1. *$\mathcal{C}$ is a ring isomorphism onto its image. In particular $\mathcal{C}(a)\mathcal{C}(b) = \mathcal{C}(ab)$.*
2. *$c(a)\mathcal{C}(b) = c(ab)$.*
3. *$\mathcal{C}(a)^\star = \mathcal{C}(a^\star)$.*

### 2.3 Vectorize and Matrixify Operators

In this section, we introduce the "vectorize" and "matrixify" functions. Informally, they "break" elements of a convolutional ring $\mathcal{R}_d$ into many elements of a smaller ring $\mathcal{R}_{d'}$, with $d'|d$. From a matricial point of view, they can also be seen as permuting rows and columns of a circulant matrix to turn it into a concatenation of smaller circulant matrices.

**Definition 5.** *Let $d \in \mathbb{N}^\star$ be a product of $h$ (not necessarily distinct) primes. We note $\mathrm{gpd}(d)$ the greatest proper divisor of $d$. When clear from context, we also note $h$ the number of prime divisors of $d$ (counted with multiplicity), $d_h \stackrel{\Delta}{=} d$ and for $i \in [\![1, h]\!]$, $d_{i-1} \stackrel{\Delta}{=} d_i/\mathrm{gpd}(d_i)$ and $k_i \stackrel{\Delta}{=} d_i/d_{i-1}$, so that $1 = d_0|d_1|...|d_h = d$ and $\prod_{j \leqslant i} k_j = d_i$.*

The $d_i$'s defined in Definition 5 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 5.

**Definition 6.** *Let $d, d' \in \mathbb{N}^\star$ such that $d'|d$. We define the vectorization $\mathsf{V}_{d\backslash d'} : \mathcal{R}_d^{n\times m} \to \mathcal{R}_d^{n\times m(d/d')}$ inductively as follows:*

1. *Let $k = d/\mathrm{gpd}(d)$. For $d' = \mathrm{gpd}(d)$ and a single element $a \in \mathcal{R}_d$, $a = \sum_{0\leqslant i<k_d} x^i a_i(x^k)$ where $a_i \in \mathcal{R}_{d'}$ for each $i$. Then*

$$\mathsf{V}_{d\backslash d'}(a) \stackrel{\Delta}{=} (a_0, ..., a_{k-1}) \in \mathcal{R}_{d'}^k$$

   *In other words, $\mathsf{V}_{d\backslash d'}(a)$ is the row vector whose coefficients are the $(a_i)_{0\leqslant i<k_d}$.*
2. *For a vector $\mathbf{v} \in \mathcal{R}_d^m$ or a matrix $\mathbf{B} \in \mathcal{R}_d^{n\times m}$, $\mathsf{V}_{d\backslash d'}(\mathbf{v}) \in \mathcal{R}_d^{(d/d')m}$ and $\mathsf{V}_{d\backslash d'}(\mathbf{B}) \in \mathcal{R}_d^{n\times(d/d')m}$ are the componentwise applications of $\mathsf{V}_{d\backslash d'}$.*
3. *For $d''|d'|d$ and any element $a \in \mathcal{R}_d$,*

$$\mathsf{V}_{d\backslash d''}(a) \stackrel{\Delta}{=} \mathsf{V}_{d'\backslash d''} \circ \mathsf{V}_{d\backslash d'}(a) \in \mathcal{R}_{d''}^{d/d''}$$

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

**Interpretation.** In practice, an element $a \in \mathcal{R}_d$ is represented by a vector of $d$ elements corresponding to the $d$ coefficients of $a$. In this context, the vectorization operation simply permutes coefficients. As highlighted by Figure 3, when $d = 2^h$ is a power of two, $\mathsf{V}_{d\backslash 1}$ permutes the coefficients according to the bit-reversal order,[1] which appears in the radix-2 Fast Fourier transform (FFT). More generally, one can show that for an arbitrary $d$, $\mathsf{V}_{d\backslash 1}$ permutes the coefficient according to the more general mixed-radix digit reversal order, which appears in the mixed-radix Cooley-Tukey FFT.

[1] `https://oeis.org/A030109`


The vectorization operation "breaks" the element $0 + x + ... + 7x^7$ of $\mathcal{R}_8$ into two elements $0 + 2x + 4x^2 + 6x^3$ and $1 + 3x + 5x^2 + 7x^3$ of $\mathcal{R}_8$, then into four elements of $\mathcal{R}_2$.

**Fig. 3.** Vectorizations of $a = 0 + x + 2x^2 + \dots 7x^7$

Similarly, one can define an operation which we call "matrixification". Like the vectorization breaks any element of $\mathcal{R}_d$ into a vector, the matrixification breaks it into a matrix.

**Definition 7.** *Following the notations of Definition 6, we define the matrixification* $\mathsf{M}_{d\backslash d'} : \mathcal{R}_d^{n\times m} \to \mathcal{R}_{d'}^{n(d/d')\times m(d/d')}$ *as follows:*

1. *For* $d' = \mathrm{gpd}(d), k = d/d'$ *and a single element* $a = \sum_{0\leqslant i<k} x^i a_i(x^k)$ *where each* $a_i \in \mathcal{R}_{d'}$*:*

$$\mathsf{M}_{d\backslash d'}(a) \triangleq \begin{bmatrix} a_0 & a_1 & \ddots & a_{k-1} \\ xa_{k-1} & a_0 & \ddots & a_{k-2} \\ \ddots & \ddots & \ddots & \ddots \\ xa_1 & xa_2 & \dots & a_0 \end{bmatrix} = \begin{bmatrix} \mathsf{V}_{d\backslash d'}(a) \\ \mathsf{V}_{d\backslash d'}(x^k a) \\ \vdots \\ \mathsf{V}_{d\backslash d'}(x^{(d'-1)k} a) \end{bmatrix} \in \mathcal{R}_{d'}^{nk\times mk}$$

   *In particular, if* $d$ *is prime, then* $\mathsf{M}_{d\backslash 1}(a) \in \mathbb{R}^{d\times d}$ *is exactly the circulant matrix* $\mathcal{C}(a)$*.*
2. *For a vector* $\mathbf{v} \in \mathcal{R}_d^m$ *or a matrix* $\mathbf{B} \in \mathcal{R}_d^{n\times m}$, $\mathsf{M}_{d\backslash d'}(\mathbf{v}) \in \mathcal{R}_d^{(d/d'')\times(d/d')m}$ *and* $\mathsf{M}_{d\backslash d'}(\mathbf{B}) \in \mathcal{R}_d^{(d/d')n\times(d/d')m}$ *are the componentwise applications of* $\mathsf{M}_{d\backslash d'}$*.*
3. *For any element* $a \in \mathcal{R}_d$*,*

$$\mathsf{M}_{d\backslash d''}(a) \triangleq \mathsf{M}_{d'\backslash d''} \circ \mathsf{M}_{d\backslash d'}(a) \in \mathcal{R}_{d''}^{(d/d'')\times(d/d'')}$$

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

**Proposition 2.** *Let* $d \in \mathbb{N}^\star$*,* $a, b$ *(resp.* $\mathbf{a}, \mathbf{b}$*, resp.* $\mathbf{A}, \mathbf{B}$*) be arbitrary scalars (resp. vectors, resp. matrices) over* $\mathcal{R}_d$*, and* $d'|d$*. For concision, we note* $\mathsf{V} \triangleq \mathsf{V}_{d\backslash d'}$ *and* $\mathsf{M} \triangleq \mathsf{M}_{d\backslash d'}$*. The vectorization and matrixification verify the following properties:*

1. *The matrixification is an algebra isomorphism onto its image, and in particular* $\mathsf{M}(\mathbf{A} \cdot \mathbf{B}) = \mathsf{M}(\mathbf{A}) \cdot \mathsf{M}(\mathbf{B})$
2. *The vectorization is a vector space isomorphism onto its image.*
3. $\mathsf{V}(ab) = \mathsf{V}(a) \cdot \mathsf{M}(b)$
4. *The vectorization is an isometry for the canonical coefficient-wise scalar product* $\langle \cdot, \cdot \rangle_2$*:*

$$\langle \mathsf{V}(\mathbf{a}), \mathsf{V}(\mathbf{b}) \rangle_2 = \langle \mathbf{a}, \mathbf{b} \rangle_2$$

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 leave them in Appendix A.

**Interpretation.** In lattice-based cryptography, cryptosystems using ring lattices rely on the hardness of problems on lattices over convolution rings.[2] In this setting, the basis is constituted of the rows of (concatenations of) circulant matrices. A prevalent example are the NTRU lattices

$$\mathbf{B}_{f,g,F,G} = \left[\begin{array}{c|c} \mathcal{C}(f) & \mathcal{C}(g) \\ \hline \mathcal{C}(F) & \mathcal{C}(G) \end{array}\right]$$

If we identify an element $a \in \mathcal{R}_d$ with its circulant matrix $\mathcal{C}(a)$, all the matrixification does is permute rows and columns of $\mathcal{C}(a)$. Permuting the rows clearly leaves invariant the lattice generated by the matrix. On the other hand, permuting the columns changes the lattice generated, but since it preserves the scalar product, the geometry of the lattice isn't affected.

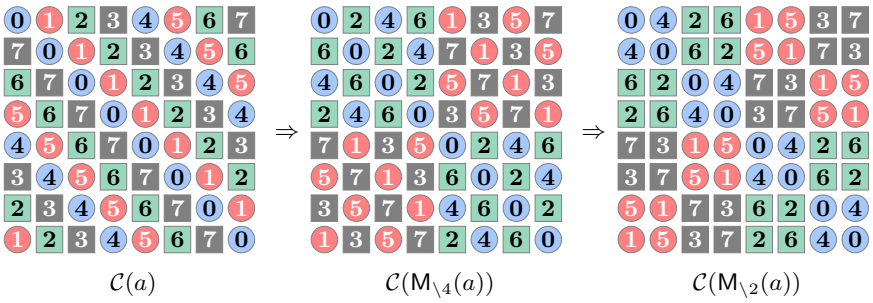



**Fig. 4.** Matrixifications of $a = 0 + x + 2x^2 + \ldots 7x^7$

[2] In reality, the underlying rings used are more often cyclotomic rings. However, we reduce the cyclotomic ring case to the convolution ring case in the ulterior Section 5.

**Computing the Vectorization and Matrixification in the Fourier Domain.** An interesting observation about the "vectorize" and "matrixify" operators that we defined is that they can be computed very efficiently when an element $a \in \mathcal{R}_d$ is represented by its coefficients but also when it is represented in the *Fourier domain*. In the first case, it is obvious that since these operations permutes coefficients of $a$, they can both be performed in time $O(d)$.

If $a$ is represented in FFT form, computing its vectorization and matrixification over $\mathcal{R}_{d'}$ in FFT form can naively be done in time $O(d \log d)$ by computing its reverse FFT, permuting its coefficients and computing $d/d'$ FFT's over $R_{d'}$. However, we observe that it can be done faster, since it is essentially equivalent to a step 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-Tukey's FFT.

**Lemma 1 ([CT65], adapted).** *Let $d \geqslant 2, d' = \mathrm{gpd}(d), k = d/d'$ and $a \in \mathcal{R}_d$ be uniquely written as $a = \sum_{0 \leqslant i < k} x^i a_i(x^k)$ where each $a_i \in \mathcal{R}_{d'}$. One can compute the FFT of $a$ from the FFT's of the $a_i$'s (and reciprocally) in time $O(dk)$. In particular, $\mathsf{V}_{d \backslash d'}(a)$, its inverse and $\mathsf{M}^{-1}_{d \backslash d'}((a_i)_i)$ (resp. $\mathsf{M}_{d \backslash d'}(a)$) can be computed in FFT form in time $O(kd)$ (resp. $O(k^2 d)$).*

*Proof.* In [CT65, equations 7 and 8], Cooley and Tukey show that one can switch from the FFT of $a$ to the FFT of the $a_i$'s (and conversely) in time $O(kd)$. Since the $a_i$'s are the coefficients of $\mathsf{V}_{d \backslash d'}(a)$ and $\mathsf{M}_{d \backslash d'}(a)$, the result follows. □

Lemma 1 allows us to gain a factor $O(\log d)$ when computing the vectorization and matrixification of $a$, compared to a naive 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.

### 2.4 The Gram-Schmidt and LDL Decompositions

In this section, $\mathcal{R} \triangleq \mathcal{R}_d$ for some $d \geqslant 1$. We recall the Gram-Schmidt and LDL decompositions, which are widespread tools inside and outside the scope of lattice-based cryptography. First, we define the notion of lower triangular unit matrix, which will be prevalent in the rest this paper.

**Definition 8.** *We say that a matrix $\mathbf{L} \in \mathcal{R}^{n \times n}$ is lower triangular unit (or LTU) if it is lower triangular and has only $1$'s on its diagonal.*

The Gram-Schmidt orthogonalization gives, for any basis $\mathbf{B} \in \mathcal{R}^{n \times m}$, an orthogonal basis $\tilde{\mathbf{B}}$ which spans the same space as a $\mathcal{R}$-module. In cryptography, it is very useful for lattice reduction, and to find or sample lattice points close to an arbitrary point.

**Algorithm 1** GramSchmidt($\mathbf{B}$)

**Require:** Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \in \mathcal{R}^{n \times m}$
**Ensure:** Decomposition $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$, where $\tilde{\mathbf{B}} \in \mathcal{R}^{n \times m}$ is orthogonal and $\mathbf{L} \in \mathcal{R}^{n \times n}$ is LTU

$\mathbf{L} = \mathbf{0}^{n \times n}$
**for** $i = 1, ..., n$ **do**
  $\tilde{\mathbf{b}}_i \leftarrow \mathbf{b}_i$
  **for** $j = 1, ..., i-1$ **do**
    $L_{i,j} = \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2}$
    $\tilde{\mathbf{b}}_i \leftarrow \tilde{\mathbf{b}}_i - L_{i,j} \tilde{\mathbf{b}}_j$
  **end for**
**end for**
**return** $(\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}, \mathbf{L} = (L_{ij})_{1 \leqslant i,j \leqslant n})$

A decomposition that is complementary to the Gram-Schmidt decomposition is the LDL decomposition. It writes any symmetric definite positive matrix (that is, a symmetric matrix $\mathbf{G} \in \mathcal{R}^{n \times n}$ such that for any $\mathbf{x} \neq \mathbf{0}$, $\mathbf{x}\mathbf{G}\mathbf{x}^\star >_{\mathcal{R}} \mathbf{0}$) as a product $\mathbf{L}\mathbf{D}\mathbf{L}^\star$, where $\mathbf{L} \in \mathcal{R}^{n \times n}$ is lower triangular with 1's on the diagonal, and $\mathbf{D} \in \mathcal{R}^{n \times n}$ is diagonal.

To properly define the notion of definite positiveness, we would need to define a semi-order $\geqslant_{\mathcal{R}}$ over elements of $\mathcal{R}$.[3] To avoid it, we resort to the notion of full-rank Gram matrix, which is essentially equivalent to definite positiveness.

**Definition 9.** *We say that a matrix $\mathbf{G} \in \mathcal{R}^{n \times n}$ is full-rank Gram (or FRG) if it is full-rank and there exists $m \geqslant n$ and $\mathbf{B} \in \mathcal{R}^{n \times m}$ such that $\mathbf{G} = \mathbf{B}\mathbf{B}^\star$.*

One can show that a matrix is FRG if and only if it is symmetric definite positive. However, the former notion requires less definitions and is much simpler to manipulate from an algorithmic viewpoint, so we will rely on it instead of the latter.

[3] The usual semi-order over $\mathcal{R} = \mathbb{R}[x]/(h(x))$ is:
$(a \geqslant_{\mathcal{R}} b) \Leftrightarrow (|a(\zeta)| \geqslant |b(\zeta)|$ for any root $\zeta$ of $h)$.

**Algorithm 2** $\mathsf{LDL}_{\mathcal{R}}(\mathbf{G})$

**Require:** A full-rank Gram matrix $\mathbf{G} = (G_{ij}) \in \mathcal{R}^{n \times n}$ defined over a ring $\mathcal{R}$
**Ensure:** The decomposition $\mathbf{G} = \mathbf{L}\mathbf{D}\mathbf{L}^\star$ over $\mathcal{R}$, where $\mathbf{L}$ is LTU and $\mathbf{D}$ is diagonal
1: $\mathbf{L}, \mathbf{D} \leftarrow \mathbf{0}^{\,n \times n}$
2: **for** $i$ from 1 to $n$ **do**
3: $\quad L_{ii} \leftarrow 1$
4: $\quad D_i \leftarrow G_{ii} - \sum_{j<i} L_{ij} L_{ij}^\star D_j$
5: $\quad$ **for** $j$ from 1 to $i-1$ **do**
6: $\quad\quad L_{ij} \leftarrow \frac{1}{D_j}\left(G_{ij} - \sum_{k<j} L_{ik} L_{jk}^\star D_k\right)$
7: $\quad$ **end for**
8: **end for**
9: **return** $((L_{ij}), \mathrm{Diag}(D_i))$

We now explicit the relation between both decompositions. For a basis $\mathbf{B}$, there exists a unique Gram-Schmidt decomposition $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$ and for a FRG matrix $\mathbf{G}$, there exists a unique LDL decomposition $\mathbf{G} = \mathbf{L}\mathbf{D}\mathbf{L}^\star$. If $\mathbf{G} = \mathbf{B}\mathbf{B}^\star$, then $\mathbf{G}$ is FRG and one can check that $\mathbf{G} = \mathbf{L} \cdot (\tilde{\mathbf{B}}\tilde{\mathbf{B}}^\star) \cdot \mathbf{L}^\star$ is a valid LDL decomposition of $\mathbf{G}$. As both decompositions are unique, the matrices $\mathbf{L}$ in both decompositions are actually the same.

### 2.5 Babai's Nearest Plane Algorithm and Klein's Sampler

**Definition 10.** *Let* $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ *be a basis. We call fundamental parallelepiped generated by* $\mathbf{B}$ *and note* $\mathcal{P}(\mathbf{B})$ *the set*

$$\sum_{1 \leqslant j \leqslant n} \left[-\frac{1}{2}, \frac{1}{2}\right) \mathbf{b}_j = \left[-\frac{1}{2}, \frac{1}{2}\right)^n \cdot \mathbf{B}$$

**Algorithm 3** $\mathsf{NearestPlane}_{\mathcal{R}}(\mathbf{B}, \mathbf{L}, \mathbf{D}, \mathbf{c})$

**Require:** The decomposition $\mathbf{B} = \mathbf{L} \cdot \tilde{\mathbf{B}}$ over $\mathcal{R}$, a vector $\mathbf{c} \in \mathrm{Span}_{\mathbb{R}}(\mathbf{B})$
**Ensure:** A vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $\mathbf{c} - \mathbf{v} \in \mathcal{P}(\tilde{\mathbf{B}})$
1: $\mathbf{t} \leftarrow \mathbf{c} \cdot \mathbf{B}^{-1}$
2: **for** $j = n, ..., 1$ **do**
3: $\quad \bar{t}_j \leftarrow t_j + \sum_{i>j} (t_i - z_i) L_{ij}$
4: $\quad z_j \leftarrow \lfloor \bar{t}_j \rceil$
5: **end for**
6: **return** $\mathbf{v} \leftarrow \mathbf{z} \cdot \mathbf{B}$

**Proposition 3.** *Algorithm 3 outputs* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *such that* $\mathbf{c} - \mathbf{v} \in \mathcal{P}(\tilde{\mathbf{B}})$.

The proof of Proposition 3 is standard and deferred to Appendix B.

## 3 Fast Fourier LDL Decomposition

### 3.1 A Compact Representation for the LDL Decomposition

**Theorem 1.** *Let $d \in \mathbb{N}$ and $1 = d_0 < d_1 < ... < d_h = d$ be a tower of proper divisors of $d$. Let $\mathbf{b} \in \mathcal{R}_d^m$ be a full-rank vector. There exists a Gram-Schmidt decomposition of $\mathsf{M}_{d\backslash 1}(\mathbf{b})$ as follows:*

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

*where $\mathbf{B}_0 \in \mathbb{R}^{d \times dm}$ is orthogonal and each $\mathbf{L}_i \in \mathcal{R}_{d_i}^{(d/d_i)\times(d/d_i)}$ is a block-diagonal matrix whose $(d/d_{i+1})$ diagonal blocks are lower triangular unit (LTU) matrices of $\mathcal{R}_{d_i}^{(d_{i+1}/d_i)\times(d_{i+1}/d_i)}$.*[4]

As a toy example, the matrix $\mathbf{L}$ of the Gram-Schmidt decomposition of $\mathsf{M}_{4\backslash 1}(\mathbf{a})$ for an element $\mathbf{a} \in \mathcal{R}_4^m$ would look like this:

$$\left[\begin{array}{cc|c|c} 1 & & & \\ & 1 & & \\ \hline a & b & 1 & \\ b & a & & 1 \end{array}\right] \cdot \left[\begin{array}{cc|c|c} 1 & & & \\ c & 1 & & \\ \hline & & 1 & \\ \hline & & & 1 \\ & & d & 1 \end{array}\right].$$

*Proof.* If $d$ is prime, the theorem is trivial. We suppose that $d$ is composite and that the theorem is true for any $\mathcal{R}_i$ with $i < d$. By Proposition 2, item 5, the matrix $\mathbf{B}_{h-1} \stackrel{\Delta}{=} \mathsf{M}_{d\backslash d_{h-1}}(\mathbf{b})$ is full-rank too. We can therefore decompose it as $\mathbf{B}_{h-1} = \mathbf{L}_{h-1}\tilde{\mathbf{B}}$, where $\mathbf{L}_{h-1} \in \mathcal{R}_{d_{h-1}}^{k_d \times k_d}, \tilde{\mathbf{B}} \in \mathcal{R}_{d_{h-1}}^{k_d \times m k_d}$ and $k_d \stackrel{\Delta}{=} d/\operatorname{gpd}(d)$. $\mathbf{L}_{h-1}$ is LTU and $\tilde{\mathbf{B}}$ is orthogonal. Noting $\tilde{\mathbf{B}} = [\mathbf{b}_1, ..., \mathbf{b}_{k_d}]$, each vector $\mathbf{b}_j$ is full-rank and orthogonal to the other $\mathbf{b}_{j'}$'s. By inductive hypothesis, they can be decomposed as follows:

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

Where each $\tilde{\mathbf{B}}_j \in \mathbb{R}^{d_{h-1} \times m d_{h-1}}$ 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 whose $(d_{h-1}/d_{i+1})$ diagonal blocks are lower triangular unit (LTU) matrices of $\mathcal{R}_{d_i}^{(d_{i+1}/d_i)\times(d_{i+1}/d_i)}$. For concision, we now note $\mathsf{M} \stackrel{\Delta}{=} \mathsf{M}_{d_{h-1}\backslash 1}$ and $\mathsf{V} \stackrel{\Delta}{=}$

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

$\mathsf{V}_{d_{h-1}\backslash 1}$. We have:

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

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

- Since each $\mathbf{L}_{i,j}$ is block diagonal with $(d_{h-1}/d_{i+1})$ LTU diagonal blocks, $\mathbf{L}_i$ is block diagonal with $k_d(d_{h-1}/d_{i+1}) = (d/d_{i+1})$ LTU diagonal blocks.
- We also need to show that $\mathbf{B}_0$ is orthogonal. Each submatrix $\tilde{\mathbf{B}}_j$ of $\mathbf{B}_0$ is the orthogonalization of $\mathsf{M}(\mathbf{b}_j)$ by induction hypothesis. Therefore, for two distinct rows $\mathbf{u}, \mathbf{v}$ of $\mathbf{B}_0$:
  - 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 $a_\ell = \mathsf{V}^{-1}(\mathbf{a}_\ell)$, we have:

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

    Where the second equality comes from Proposition 2, item 3, the third one from the fact that $\mathsf{V}$ is a scaled isometry and the fourth one from the fact that $\mathbf{b}_j, \mathbf{b}_\ell$ are orthogonal.

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

□

The theorem we stated gives the Gram-Schmidt decomposition of $\mathsf{M}_{d\backslash 1}(\mathbf{b})$ 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 $1 = d_0 < d_1 < ... < d_h = d$ 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 $(\mathbf{L}_i)_{0 \leqslant i \leqslant h}$ where:*

- *$\mathbf{L}_h \in \mathcal{R}_d^{n \times n}$ is LTU.*
- *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 LTU matrices of $\mathcal{R}_{d_i}^{(d_{i+1}/d_i) \times (d_{i+1}/d_i)}$.*

*Such that, if we note $\mathbf{L} = \left(\prod_{i=0}^{h} \mathsf{M}_{d_i \backslash 1}(\mathbf{L}_i)\right)$:*

1. *The Gram-Schmidt decomposition of $\mathsf{M}_{d\backslash 1}(\mathbf{B})$ is: $\mathsf{M}_{d\backslash 1}(\mathbf{B}) = \mathbf{L} \cdot \mathbf{B}_0$ where $\mathbf{B}_0 \triangleq \mathbf{L}^{-1} \cdot \mathsf{M}_{d\backslash 1}(\mathbf{B})$.*
2. *The LDL decomposition of $\mathsf{M}_{d\backslash 1}(\mathbf{B}\mathbf{B}^\star)$ is: $\mathsf{M}_{d\backslash 1}(\mathbf{B}) = \mathbf{L} \cdot (\mathbf{B}_0 \mathbf{B}_0^t) \cdot \mathbf{L}^t$.*

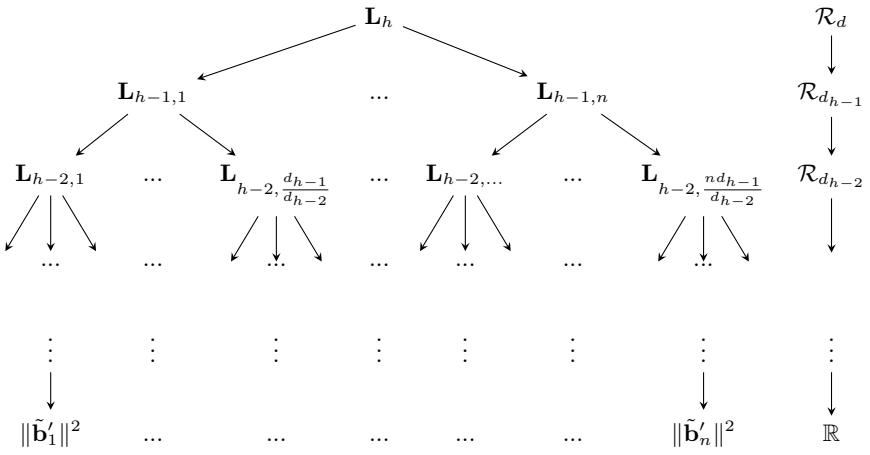
*Proof.* We have $\mathbf{B} = \mathbf{L}_h \mathbf{B}'$, where $\mathbf{L}_h$ is given by either the Gram-Schmidt or LDL decomposition algorithm. $\mathbf{B}' = \{\mathbf{b}'_1, ..., \mathbf{b}'_n\}$ is orthogonal. Applying Theorem 1 to each row vector $\mathbf{b}'_j$ of $\mathbf{B}'$ yields $n$ decompositions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

The LDL decomposition is then given "for free" by the equivalence between it and the Gram-Schmidt decomposition, and indeed, one can check that since $\mathbf{B}_0$ is orthogonal, $(\mathbf{B}_0 \mathbf{B}_0^t)$ is diagonal. □

The Theorem 1 and the Corollary 1 state that for any full-rank matrix $\mathbf{B} \in \mathcal{R}^{n \times m}$, the $\mathbf{L}$ matrix in its Gram-Schmidt and LDL decompositions can be represented in a factorized form, where each of the factors $\mathbf{L}_i$ is a sparse, block-diagonal matrix. Figure 5 synthetizes this factorization.

### 3.2 A Fast Algorithm for the Compact LDL Decomposition

Theorem 1 and Corollary 1 are constructive: more precisely, their proofs give an algorithm to compute the compact factorized form of $\mathbf{L}$ quickly. Algorithm 4 performs the compact LDL decomposition in the form of the tree given in Figure 5.




$\mathbf{L}_h \in \mathcal{R}_d^{n\times n}$ is a LTU matrix, and for each $i < h$, every $\mathbf{L}_{i,j}$ is a lower triangular unit matrix in $\mathcal{R}_{d_i}^{(d_{i+1}/d_i)\times(d_{i+1}/d_i)}$. Each of the matrices $\mathbf{L}_i$ mentioned in Theorem 1 is the block-diagonal matrix whose blocks are the $(\mathbf{L}_{i,j})_{1\leqslant j\leqslant nd/d_{i+1}}$.

**Fig. 5.** Tree $\mathfrak{L}$ of precomputed matrices $\mathbf{L}_{i,j}$ such that $\mathbf{L} = \prod_i \mathsf{M}_{d_i\setminus 1}(\mathrm{Diag}_j(\mathbf{L}_{i,j}))$

**Algorithm 4** $\mathsf{ffLDL}_{\mathcal{R}_d}(\mathbf{G})$

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

Algorithm 4 computes a "Fast Fourier LDL", instead of the "Fast Fourier Gram-Schmidt" hinted at in Theorem 1 and Corollary 1. The reason why we favor this approach is because it allows a complexity gain. This gain already happens in the classic versions of the aforementioned algorithms.

As a simple example, consider the $\mathbf{L}$ in the Gram-Schmidt decomposition of $\mathbf{B} \in \mathcal{R}_d^{2\times m}$, which is exactly the $\mathbf{L}$ in the LDL decomposition of $\mathbf{BB}^\star \in \mathcal{R}_d^{2\times 2}$. Computing it with the LDL algorithm is then $O(m)$ times

faster than with the Gram-Schmidt algorithm. The same phenomenon happens with their recursive variants.

**Lemma 2.** *Let $d \in \mathbb{N}$ and $1 = d_0 < d_1 < ... < d_h = d$ be a tower of proper divisors of $d$, and for $i \in [\![1, h]\!]$, let $k_i \stackrel{\Delta}{=} d_i/d_{i-1}$. Let $\mathbf{G} \in \mathcal{R}_d^{n \times n}$ be a full-rank Gram matrix. Then Algorithm 4 computes the LDL decomposition tree of $\mathbf{G}$ in FFT form in time*

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

*In particular, if all the $k_i$ are bounded by a small constant $k$, then the complexity of Algorithm 4 is upper bounded by $O(n^3 d + n^2 d \log d)$.*

*Proof.* Let $\mathrm{C}(k, d)$ denote the complexity of Algorithm 4 over a matrix $\mathbf{G} \in \mathbb{R}_d^{k \times k}$. We have the following recursion formula:

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

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

$$\begin{aligned} \mathrm{C}(k_h, d_{h-1}) &= \sum_{1 \leqslant i \leqslant h} \tfrac{d}{d_i} O(d_{i-1} k_i^3) + \tfrac{d}{d_1} \mathrm{C}(k_1, d_0) \\ &= O(d) \sum_{1 \leqslant i \leqslant h} k_i^2 \end{aligned} \tag{4}$$

Where the first equality is shown by induction using equation 3, *except* the first term $O(n^2 d \log d)$ which is no longer relevant since we are already in the Fourier domain. Combining equations 3 and 4, we conclude that the complexity of the total algorithm is

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

□

## 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 as its classical, iterative counterpart, but runs $\tilde{O}(d)$ times faster.

**Definition 11.** *We note $\mathcal{Z}_d$ the ring $\mathbb{Z}[x]/(x^d-1)$ of elements of $\mathcal{R}_d$ with integer coefficients.*

**Algorithm 5** $\mathsf{ffNearestPlane}_{\mathcal{R}_d}(\mathbf{t}, \mathfrak{L})$

**Require:** $\mathbf{t} \in \mathcal{R}_d^n$, a precomputed tree $\mathfrak{L}$, (implicitely) 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
1: **if** $\mathbf{t}$ is a 1-dimensional vector in $\mathbb{R}$ **then**
2: **return** $\lfloor \mathbf{t} \rceil$
3: **end if**
4: $\mathbf{L} \leftarrow \mathfrak{L}.\mathsf{Node}()$
5: **for** $j = n, ..., 1$ **do**
6: $\overline{\mathbf{t}}_j \leftarrow \mathbf{t}_j + \sum_{i>j}(\mathbf{t}_i - \mathbf{z}_i)L_{ij}$
7: $\mathbf{z}_j \leftarrow \mathsf{V}_{d\backslash \mathrm{gpd}(d)}^{-1} \left[\mathsf{ffNearestPlane}_{\mathcal{R}_{\mathrm{gpd}(d)}}(\mathsf{V}_{d\backslash \mathrm{gpd}(d)}(\overline{\mathbf{t}}_j), \mathfrak{L}.\mathsf{Child}(j))\right]$
8: **end for**
9: **return** $\mathbf{z}$

Thomas! Thomas: J'ai corrig une erreur du FFNP original (il y avait $\mathsf{M}(\overline{\mathbf{t}})$ au lieu de $\mathsf{V}(\overline{\mathbf{t}})$)...

**Lemma 3.** *Let $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ and $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$ be its GSO in $\mathcal{R}$. The vectors $\mathbf{z} = (\mathbf{z}_1, ..., \mathbf{z}_n)$ and $\overline{\mathbf{t}} = (\overline{\mathbf{t}}_1, ..., \overline{\mathbf{t}}_n)$ in Algorithm 5 verify*

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

*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:

$$\begin{aligned}
(\mathbf{z} - \overline{\mathbf{t}}) \cdot \tilde{\mathbf{B}} &= \textstyle\sum_{j=1...n}(\mathbf{z}_j - \overline{\mathbf{t}}_j) \cdot \tilde{\mathbf{b}}_j \\
&= \textstyle\sum_{j=1...n}\left[(\mathbf{z}_j - \mathbf{t}_j) + \sum_{i>j}(\mathbf{z}_i - \mathbf{t}_i) \cdot L_{ij}\right] \cdot \tilde{\mathbf{b}}_j \\
&= \textstyle\sum_{1 \leqslant i \leqslant j \leqslant n}(\mathbf{z}_i - \mathbf{t}_i) \cdot L_{ij} \cdot \tilde{\mathbf{b}}_j \\
&= \textstyle\sum_{i=1...n}(\mathbf{z}_i - \mathbf{t}_i) \cdot \mathbf{b}_i \\
&= (\mathbf{z} - \mathbf{t}) \cdot \mathbf{B}
\end{aligned} \tag{5}$$

The first and last equalities are trivial, the second one replaces the $\overline{\mathbf{t}}_j$'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 $\mathsf{M} \triangleq \mathsf{M}_{d\backslash 1}$ and $\mathsf{V} \triangleq \mathsf{V}_{d\backslash 1}$. Algorithm 5 outputs $\mathbf{z} \in \mathcal{Z}^n$ such that $\mathsf{V}((\mathbf{z} - \mathbf{t})\mathbf{B}) \in \mathcal{P}(\mathbf{B}_0)$, where $\mathbf{B}_0$ is the orthogonalization of $\mathsf{M}(\mathbf{B})$.*

*Proof.* The result is trivially true if $n = d = 1$. We prove it in the general case. By definition, each $\mathfrak{L}.\mathsf{Child}(j)$ is the LDL decomposition tree of $\mathsf{M}_{d\setminus \mathrm{gpd}(d)}(\tilde{\mathbf{b}}_j)$. 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 $\mathbf{B}_j \stackrel{\Delta}{=} \mathsf{M}(\tilde{\mathbf{b}}_j)$. From Lemma 3, 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 $\mathsf{V}((\mathbf{z} - \mathbf{t})\mathbf{B}) \in \mathcal{P}([\tilde{\mathbf{B}}_1, ..., \tilde{\mathbf{B}}_n])$. Now, from the proof of Corollary 1, we know that $\mathbf{B}_0 = [\tilde{\mathbf{B}}_1, ..., \tilde{\mathbf{B}}_n]$ is actually the orthogonalization of $\mathsf{M}(\mathbf{B})$, which concludes the proof. □

Equivalently, Theorem 2 states that Algorithm 5 outputs $\mathbf{z}$ such that $(\mathbf{z} - \mathbf{t})\mathbf{B} \in \mathsf{V}^{-1}\mathcal{P}(\widetilde{\mathsf{M}(\mathbf{B})})$. This behavior is slightly different from the usual nearest plane algorithm. When instanciated for maximal efficiency from a quality point of view (therefore completely ignoring the ring structure), the latter outputs $\mathbf{z}$ such that $(\mathbf{z} - \mathbf{t})\mathbf{B} \in c^{-1}\mathcal{P}(\widetilde{\mathcal{C}(\mathbf{B})})$.

Although $(\mathbf{z}-\mathbf{t})\mathbf{B}$ is in a different cuboid in each case, both algorithms are essentially identical in terms of quality. Ignoring the action of $\mathsf{V}^{-1}, c^{-1}$ (since they are isometries), the cuboids $\mathcal{P}(\widetilde{\mathsf{M}(\mathbf{B})})$ and $\mathcal{P}(\widetilde{\mathcal{C}(\mathbf{B})})$ predictably have the same volume. But more importantly, they are generated by $nd$ orthogonal vectors, and one can easily show that the length of the longest one is the same in both cases. Therefore, Algorithm 5 is essentially as good as the nearest plane algorithm in terms of output quality. In addition, it can be run significantly faster over convolutional ring lattices, as demonstrated in Lemma 4.

**Lemma 4.** *Let $d \in \mathbb{N}$ and $1 = d_0 < d_1 < ... < d_h = d$ be a tower of proper divisors of $d$, and for $i \in [\![1, h]\!]$, let $k_i \stackrel{\Delta}{=} d_i/d_{i-1}$. Let $\mathbf{B} \in \mathcal{R}_d^{n\times m}$ and $\mathfrak{L}$ be its LDL decomposition tree. The complexity of Algorithm 5 is upper bounded by:*

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

*In particular, if all the $k_i$ are bounded by a small constant $k$, then the complexity of Algorithm 5 is upper bounded by $O(n^2 d + nd \log d)$.*

*Proof.* Let $\mathrm{C}(k, d)$ denote the complexity of Algorithm 5 over input $\mathbf{t} \in \mathbb{R}_d^k$. We have this recursion formula:

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

Where the first term corresponds to computing the FFT of the $n$ coefficients of $\mathbf{t}$, the second term to performing computing the $\overline{\mathbf{t}}_j$'s (step 6) in FFT form, the third one to the $n$ calls to $\mathsf{V}^{-1}_{d\setminus \mathrm{gpd}(d)}, \mathsf{M}_{d\setminus \mathrm{gpd}(d)}$ (see Lemma 1) and the fourth one to the $n$ recursive call to itself. We have:

$$\begin{aligned} \mathrm{C}(k_h, d_{h-1}) &= \sum_{1 \leqslant i \leqslant h} \tfrac{d}{d_i} O(d_{i-1} k_i^3) + \tfrac{d}{d_1} \mathrm{C}(k_1, d_0) \\ &= O(d) \sum_{1 \leqslant i \leqslant h} k_i^2 \end{aligned} \tag{6}$$

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

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

□

## 5 Extending the Results to Cyclotomic Rings

In this section we argue that our result also applies in the cyclotomic case. 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. $\mathsf{M}'$ is a ring morphism.
4. $\mathsf{V}'$ is a scaled linear isometry.
5. $\mathsf{V}'(ab) = \mathsf{M}'(a)\mathsf{V}'(b)$.
6. Computing $\mathsf{V}'$, $\mathsf{V}'^{-1}$, $\mathsf{M}'$ and $\mathsf{M}'^{-1}$ takes time $O(dk)$.

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 $\mathcal{R}_d$.

### 5.1 Cyclotomic Rings

**Definition 12.** *Let $d \in \mathbb{N}^\star$ and $\zeta_d$ denote an arbitrary primitive $d$-th root of unity in $\mathbb{C}$, per example $\zeta_d = e^{\frac{2i\pi}{d}}$. Let $\Omega_d = \{\zeta_d^k | k \in \mathbb{Z}_d^\times\}$ be the set of primitive $d$-th roots of unity. We note $\phi_d$ and call $d$-th cyclotomic polynomial the polynomial in $\mathbb{Z}[x]$ defined by*

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

*In addition, we define the polynomial $\psi_d$ as follows*

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

When $d$ is a power of prime – which is a prevalent case in lattice-based cryptography –, there exist explicit formulae for computing $\phi_d$:

- For a prime $p$, $\phi_p(x) = \sum_{k=0}^{p-1} x^k$
- For a prime power $d = p^k$, $\phi_d(x) = \phi_p(x^{m/p})$

It is immediate that for any $d$, the degree of $\phi_d$ is $\varphi(d)$, where $\varphi(d) \stackrel{\Delta}{=} |\mathbb{Z}_d^\times|$ is Euler's totient function. One can also check that $\phi_d(x) \cdot \psi_d(x) = x^d - 1$.

**Definition 13.** *For $d \in \mathbb{N}^\star$, we note $\mathcal{F}_d$ the cyclotomic ring $\mathbb{R}[x]/(\phi_d(x))$.*

### 5.2 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 14.** *Let $e_d$ be the unique element in $\mathcal{R}_d$ such that $e_d = 1 \mod \phi_d$ and $e_d = 0 \mod \psi_d$. We define the embedding $\iota_d$ from $\mathcal{F}_d$ into $\mathcal{R}_d$ as follows:*

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

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

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

$$\iota(f)(\zeta) = \begin{cases} f(\zeta) & \text{if } \phi_d(\zeta) = 0 \\ 0 & \text{if } \psi_d(\zeta) = 0 \end{cases} \tag{7}$$

**Proposition 4.** *Let $d \in \mathbb{N}^\star$ and $\iota = \iota_d$. The embedding $\iota$:*

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

*Proof.* Let us first prove item 1. The element $e_d$ is idempotent in $\iota(\mathcal{F}_d)$: $e_d^2 = e_d$. From this, one can easily show that $\iota$ is a ring homomorphism. In addition, for any element $g \in \iota(\mathcal{F}_d)$, $g \mod \phi_d$ is the unique antecedent of $g$ with respect to $\iota$, so $\iota$ is bijective and $\iota^{-1}(g) = g \mod \phi_d$, which proves the point 1. Items 2. and 3. follows from equation (7). □

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

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

For readability, the proof of Lemma 5 is left in Appendix C.

### 5.3 Conclusion for Cyclotomic Rings

We now check that the 6 conditions enounced at the beginning of Section 5 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 $\iota_d$ trivialize the construction of maps $\mathsf{M}'$ and $\mathsf{V}'$ from $\mathcal{F}_d$ to $\mathcal{F}_{d'}$:

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

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 $\iota_{d'}^{-1}$: $\mathsf{V}'$ is well defined, and similarly for $\mathsf{M}'$. Conditions 3 and 5 follow from the fact that $\iota_d, \iota_{d'}$ are ring morphisms and that similar properties hold for $\mathsf{M}_{d\backslash d'}$ and $\mathsf{V}_{d\backslash d'}$. Condition 4 is true because $\iota_d$, $\mathsf{V}_{d\backslash d'}$ and $\iota_{d'}$ are isometries. Finally, condition 6 holds in the FFT representation, from Lemma 1 and the from the fact that $\iota$ in the Fourier domain simply consist of inserting some 0 at appropriate positions.

## 6 Implementation in Python

In this final section, we give the core of the python implementation of our algorithm when $d$ is a power of 2. This complete, public-domain implementation can be found at:

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

It includes a verifier `verif.py`, that is based on the (slow) Gram-Schmidt algorithm. The file `ffo.py` is the full version of the simplified algorithms given below. The file `ffo_NaN.py` is a slightly more evolved version that also handles the non-full rank case. The file `cyclo.py` implement the embedding technique of Section 5 to apply our algorithms to the cyclotomic ring setting. The file `test.py` runs test in the rings $\mathcal{R}_{64}$ and $\mathcal{F}_{64}$.

*Conventions.* In `python.numpy`, the arithmetic operations `+`,`-`,`*`,`/` on arrays denotes coefficient-wise operations. The functions `fft` and its inverse `ifft` are built in. The symbol `j` denotes the imaginary unit.

---

```
# Modified extract of ffo.py
from numpy import *
```

```
# Inverse vectorize operation 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 type
    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

# Vectorize 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)

# 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

# ffBabai alg., i/o in base B, fft format
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)

# ffBabai alg., 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)
```

## References


[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[AP11] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, 2011.

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. Preliminary version in STACS 1985.

[BCG+14] Johannes Buchmann, Daniel Cabarcas, Florian Göpfert, Andreas Hülsing, and Patrick Weiden. Discrete ziggurat: A time-memory trade-off for sampling from a gaussian distribution over the integers. In Lange et al. [LLL14], pages 402–417.

[BLL+15] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: using the rényi divergence rather than the statistical distance. 2015.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Boneh et al. [BRF13], pages 575–584.

[Boy13] Xavier Boyen. Attribute-based functional encryption on lattices. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 122–142. Springer, Heidelberg, March 2013.

[BRF13] Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors. *45th ACM STOC*. ACM Press, June 2013.

[BY93] Z Bai and YQ Yin. Limit of the smallest eigenvalue of a large dimensional sample covariance matrix. *The annals of Probability*, 21:1276–1294, 1993.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May 2010.

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Heidelberg, August 2013.

[DG14] Nagarjun C Dwarakanath and Steven D Galbraith. Sampling from discrete gaussians for lattice-based cryptography on a constrained device. *Applicable Algebra in Engineering, Communication and Computing*, 25(3):159–180, 2014.

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 22–41. Springer, Heidelberg, December 2014.

[DN12] Léo Ducas and Phong Q. Nguyen. Faster gaussian lattice sampling using lazy floating-point arithmetic. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 415–432. Springer, Heidelberg, December 2012.

[DP15] Léo Ducas and Thomas Prest. A hybrid gaussian sampler for lattices over rings. Cryptology ePrint Archive, Report 2015/660, 2015. `http://eprint.iacr.org/2015/660`.

[Duc13] Léo Ducas. *Lattice Based Signatures: Attacks, Analysis and Optimization*. These, Ecole Normale Supérieure de Paris - ENS Paris, 2013.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[GS66] W Morven Gentleman and Gordon Sande. Fast fourier transforms: for fun and profit. In *Proceedings of the November 7-10, 1966, fall joint computer conference*, pages 563–578. ACM, 1966.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Boneh et al. [BRF13], pages 545–554.

[HJB84] Michael T Heideman, Don H Johnson, and C Sidney Burrus. Gauss and the history of the fast fourier transform. *ASSP Magazine, IEEE*, 1(4):14–21, 1984.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

[Kar13] Charles FF Karney. Sampling exactly from the normal distribution. *arXiv preprint arXiv:1303.6257*, 2013.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941, 2000.

[Len82] Arjen Klaas Lenstra. Factoring polynomials over algebraic number fields. *Stichting Mathematisch Centrum. Informatica*, (IW 213/82):1–21, 1982.

[Lep14] Tancrède Lepoint. *Design and Implementation of Lattice-Based Cryptography*. Theses, Ecole Normale Supérieure de Paris - ENS Paris, June 2014.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982.

[LLL14] Tanja Lange, Kristin Lauter, and Petr Lisonek, editors. *SAC 2013*, volume 8282 of *LNCS*. Springer, Heidelberg, August 2014.

[LMPR08] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 54–72. Springer, Heidelberg, February 2008.

[LP15] Vadim Lyubashevsky and Thomas Prest. Quadratic time, linear space algorithms for Gram-Schmidt orthogonalization and gaussian sampling in structured lattices. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 789–815. Springer, Heidelberg, April 2015.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[Nus12] Henri J Nussbaumer. *Fast Fourier transform and convolution algorithms*, volume 2. Springer Science & Business Media, 2012.

[Pei10] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Heidelberg, August 2010.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[RVV14] Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. High precision discrete gaussian sampling on FPGAs. In Lange et al. [LLL14], pages 383–401.

[Sch96] James C. Schatzman. Accuracy of the discrete fourier transform and the fast fourier transform. *SIAM J. Scientific Computing*, 17(5):1150–1166, 1996.

[Ver10] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

## A Proof of Proposition 2

*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 $\mathsf{M}$ is a homomorphism are trivial except for the fact that it is multiplicative. First, one can check from Definition 7 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 $d'' | d$:

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

To show injectivity, it suffices to see that if $d' = \mathrm{gpd}(d)$, then $(\mathsf{M}_{d\backslash d'}(a) = 0) \Leftrightarrow (a = 0)$. 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 $\mathsf{M}$ 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 $\forall i, a_i = \sum_{0 \leqslant j < i} a_{i,j} x^j$ and $b_i = \sum_{0 \leqslant j < i} b_{i,j} x^j$. Then

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

5. We have:

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

The first and last equivalences are simply the definition, the second and fourth uses the fact that $\mathsf{V}$ is a vector space isomorphism and the third one uses Proposition 2, item 3.

□

## B Proof of Proposition 3

*Proof.* Let $\mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^\star$ be the LDL decomposition of $\mathbf{BB}^\star$. We have:

$$\begin{aligned} (\mathbf{v} - \mathbf{c}) \cdot \tilde{\mathbf{B}}^\star &= (\mathbf{z} - \mathbf{t})\mathbf{B} \cdot \tilde{\mathbf{B}}^\star \\ &= (\mathbf{z} - \mathbf{t})\mathbf{LD} \end{aligned} \tag{8}$$

One can check that for any $j \in [\![1, n]\!]$,

$$((\mathbf{z} - \mathbf{t})\mathbf{L})_j = \sum_{i \geqslant j} (z_i - t_i) L_{ij} = \bar{t}_j - z_j \in \left[ -\frac{1}{2}, \frac{1}{2} \right] \tag{9}$$

where the second (resp. third) equality comes from the way the $\bar{t}_j$'s (resp. $z_j$'s) are computed in Algorithm 3. We note $\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n$ the row vectors

of $\tilde{\mathbf{B}}$. Combining equations 8 and 9 yields $|\langle \mathbf{v}-\mathbf{c}, \tilde{\mathbf{b}}_j \rangle| \leqslant \frac{1}{2}\|\tilde{\mathbf{b}}_j\|^2$. Since the vectors $\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n$ are orthogonal and $(\mathbf{v}-\mathbf{c}) \in \text{Span}(\tilde{\mathbf{B}})$, we can write

$$\begin{aligned} \mathbf{v}-\mathbf{c} &= \sum_{1 \leqslant j \leqslant n} \frac{\langle \mathbf{v}-\mathbf{c}, \tilde{\mathbf{b}}_j \rangle}{\|\tilde{\mathbf{b}}_j\|^2} \tilde{\mathbf{b}}_j \\ &\in \sum_{1 \leqslant j \leqslant n} \left[-\tfrac{1}{2}, \tfrac{1}{2}\right] \tilde{\mathbf{b}}_j \\ &\in \mathcal{P}(\tilde{\mathbf{B}}) \end{aligned}$$

Where the second equality comes from the fact that $|\langle \mathbf{v}-\mathbf{c}, \tilde{\mathbf{b}}_j \rangle| \leqslant \frac{1}{2}\|\tilde{\mathbf{b}}_j\|^2$, and the third one from the definition of $\mathcal{P}$. This concludes the proof. □

## C Proof of Lemma 5

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

$$(\zeta \in U_d \backslash \Omega_d) \Leftrightarrow (\zeta^k \in U_{d'} \backslash \Omega_{d'}) \tag{10}$$

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

$$a(\zeta_d^j) = \sum_{0 \leqslant i < k} \zeta_d^{ij} a_i(\zeta_d^{kj}) = \sum_{0 \leqslant i < k} \zeta_d^{ij} a_i(\zeta_{d'}^j) \tag{11}$$

As a step of the FFT (see Lemma 1), the system 11 is invertible. In addition, one can check in equation 10 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 $a(E) \stackrel{\Delta}{=} \{a(e) | e \in E\}$:

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

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. □