An Exposition on Fast Fourier Orthogonalization

Pritam Chandra

Ashoka Mphasis Lab

June 2025

Abstract

In this article we will reproduce and illustrate the works of Léo Ducas and Thomas Prest in [DP16] relating to fast orthogonalization of block circulant matrices. In particular, we prove that a block circulant matrix A can be rearranged in a special way such that the unit lower triangular part L of its Gram-Schmidt Orthogonalization admits a compact factored representation. We will see a fast algorithm to compute this representation, reminiscent of Cooley-Tukey's FFT, reducing the time and space complexity of the classical algorithm from cubic to quasilinear. We will then modify a foundational tool in lattice based cryptography – Babai's Nearest Plane algorithm – to be compatible with the factored representation.

Contents

1 Introduction

With the onset of quantum computation and proposed quantum algorithms, many classical cryptographic systems stood the fear of being rendered insecure. Among other techniques, lattice based encryption and signature schemes were formulated to withstand the quantum threat. A major step in this research was developing new algorithms in $\mathbb{Z}$ modules over polynomial rings, instead of the vanilla $\mathbb{Z}^n$ lattices – see the GPV framework and NTRU lattices [HPS98, GPV08]. This was found to boost both the security and the efficiency of the cryptosystems. The work of Ducas and Prest in *Fast Fourier Orthogonalization* [DP16] is in a similar vein.

A key component of a lattice based constructions is a problem called the closest vector problem (CVP) [Reg05]. Lattice based signature schemes roughly follow this outline: map documents to the ambient space, and produce its signature as its nearest lattice point, thereby making digital signatures an instance of the CVP. Babai's Nearest Plane algorithm that approximately solves CVP therefore becomes the bedrock of such cryptographic schemes. In particular, for a lattice generated by the rows of $\mathbf{A}$, the algorithm promises a lattice point in the neighborhood of the target vector defined by the rows of $\mathbf{B}$, which is the Gram Schmidt Orthogonalization of $\mathbf{A}$. The underlying ring $\mathcal{R}$, from which $\mathbf{A}$ draws its entries, determines the size of this neighborhood – simpler rings like $\mathbb{R}$ promises tighter neighborhoods. See definition 2.6.

Babai's algorithm on a lattice generated by $\mathbf{A}$ is implicitly fed the Gram-Schmidt process on $\mathbf{A}$. Consider $\mathcal{R}$, a ring of polynomials of degree $d-1$. Then the module $\mathcal{R}_d^{n\times n}$ is identified with the space $\mathbb{R}^{nd\times nd}$ in a straightforward manner. Cryptographic lattices are usually $\mathcal{Z}$-submodules of such $\mathcal{R}_d$-modules. The hardness of cracking CVP in such lattices is a blessing of the dimensionality, that is, nd must be large. If d is large and n is comparatively small, then GSO is performed quickly for matrices in $\mathcal{R}_d^{n\times n}$, however the quality of the output of Babai's algorithm is poor, thereby affecting the security of the signature. However, when d is small and n large, like $\mathbb{R}^{nd\times nd}$, even though Nearest Plane produces very close vectors, the underlying GSO has cubic time complexity over nd.

Using techniques from the well established theory of circulant matrices, Ducas and Prest propose new techniques for special rings where this tradeoff can be overcome. Recall that permuting the rows and columns of a lattice basis keeps the geometry of the lattice intact, up to an isometry. With this in mind, they show that the $2d \times 2d$ real matrix, obtained from an equivalent lattice basis matrix from $\mathcal{R}_d^{2\times 2}$, can be rearranged such that it admits a GSO in as fast as $\mathcal{O}(d \log d)$ time. Further, they propose a compatible variant of Babai's algorithm, showing that this efficiency can be achieved while maintaining outputs of the same quality as that of the Nearest Plane algorithm over $\mathbb{R}$.

2 Preliminaries

To begin with we will set forth some notational conventions.

- Vectors in this article by default are row vectors. So are the coefficient vectors of polynomials. A degree $d+1$ polynomial f is understood to be $f(x) = f_0 + f_1 x + \cdots + f_d x^d$.
- Matrices will be uppercase bold and vectors lowercase bold, whereas ring elements will be in normal text. When not defined otherwise, an $m \times n$ matrix $\mathbf{A}$ will have a row representation $\begin{bmatrix}\mathbf{a}_1 & \dots & \mathbf{a}_m\end{bmatrix}^T$ or a component representation $[a_{ij}]$.
- Similarly, a vector $\mathbf{a}$ is $\begin{bmatrix}a_1 & \dots & a_n\end{bmatrix}$. The vector $\mathbf{e}_j = \begin{bmatrix}0 & \dots & 1 & \dots & 0\end{bmatrix}$ is the j-th standard basis vector.
- The matrix $\mathbf{I}_j$ will denote the $j \times j$ identity matrix. When the dimension is clear from context, identity will only be $\mathbf{I}$.

2.1 Recap of Relevant Topics

Definition 2.1. *We define the ring of polynomials* $\mathcal{R}_d = \dfrac{\mathbb{R}[x]}{(x^d - 1)}$.

We know that one can do operations like multiplications and inverses on $\mathcal{R}_d$ in $\Theta(d \log d)$ time using the Fast Fourier Transform (FFT) [CT65]. All operations in this article will happen in such rings.

Definition 2.2. *Let* $\mathcal{R} = \mathbb{R}[x]/(p)$ *for some polynomial* p. *Then for any* $f \in \mathcal{R}$, *we call the polynomial* f^* *as the* ***adjoint*** *of* f, *if for any root* ζ *of* p, *we have* $f^*(\zeta) = \overline{f(\zeta)}$.

With adjoints one can define an inner product on $\mathcal{R}$*-modules. Consider* $\mathbf{f}, \mathbf{g} \in \mathcal{R}^m$, *then*

$$\langle \mathbf{f}, \mathbf{g} \rangle_{\mathcal{R}} := \sum_{i=1}^{m} f_i g_i^*.$$

In the scope of this article we consider the rings $\mathcal{R}_d$ where $p(x) = x^d - 1$. The roots of p are $\{e^{2\pi \frac{k}{d} i} : k = 0, \dots, d-1\}$. Then observe that for any root ζ of p, $\overline{\zeta} = \zeta^{-1}$. Since a $d-1$ degree polynomial is completely determined by its evaluations on d unique points, then for any $f \in \mathcal{R}_d$, it follows that $f^*(x) = f(x^{-1})$, as f having real coefficients ensures

$$f^*(\zeta) = f(\zeta^{-1}) = f(\overline{\zeta}) = \overline{f(\zeta)}.$$

Hence, for $f \in \mathcal{R}_d$ we get

$$\begin{aligned} f^*(x) &= f_0 + f_1 x^{-1} + \cdots + f_{n-1} x^{-(d-1)} \\ &= f_0 + f_{d-1} x + \cdots + f_1 x^{d-1}. \end{aligned}$$

Remark 2.1. *An element $f \in \mathcal{R}_d$ is bijectively associated with a circulant matrix* $\mathbf{F}$ *given by*

$$\mathbf{F} = \begin{bmatrix} f_0 & f_1 & \cdots & f_{d-1} \\ f_{d-1} & f_0 & \cdots & f_{d-2} \\ \vdots & & \ddots & \\ f_1 & f_2 & \cdots & f_0 \end{bmatrix}.$$

Then the adjoint f^* of f simply corresponds to the commonly known matrix adjoint F^T of the F. From this it is easy to see again that

$$f^*(x) = f_0 + f_{d-1}x + \cdots + f_1 x^{d-1}.$$

2.1.1 Gram-Schmidt Orthogonalization (GSO)

Once we have an inner product in a ring $\mathcal{R}$, we can use the Gram-Schmidt process to orthogonalize matrices with entries from $\mathcal{R}$. The process also furnishes a matrix decomposition that we define below.

Definition 2.3. *Let $\mathbf{A} \in \mathcal{R}^{m \times n}$ have linearly independent rows. Then we define the GSO of* $\mathbf{A}$ *as the decomposition*

$$\mathbf{A} = \mathbf{L}\mathbf{B}$$

where the rows of $\mathbf{B}$ *are mutually orthogonal, and* $\mathbf{L}$ *is a unit lower triangular $m \times m$ matrix.*

The algorithm to compute GSO is popoular, and is an essential subroutine to the Nearest Plane algorithm of Babai [Bab86]. We quickly recall the algorithm here.

Algorithm 1 $\mathsf{GSO}_{\mathcal{R}}(\mathbf{A})$

Require: $\mathbf{A} \in \mathcal{R}^{m \times n}$
Ensure: $\mathbf{L}, \mathbf{B}$ the GSO of $\mathbf{A}$ over $\mathcal{R}$

$\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_m \end{bmatrix}^T \leftarrow \mathbf{A}$
for $i = 1, \dots, m$ **do**
  for $j = 1, \dots, n$ **do**
    $L_{ij} \leftarrow \dfrac{\langle \mathbf{a}_i, \mathbf{a}_j \rangle_{\mathcal{R}}}{\langle \mathbf{a}_i, \mathbf{a}_i \rangle_{\mathcal{R}}}$
  end for
  $\mathbf{b}_i = \mathbf{a}_i - \sum_{j=1}^{i-1} L_{ij}\mathbf{b}_j$
end for
return $\mathbf{L}, \mathbf{B}$

The algorithm stores $\mathcal{O}(mn)$ ring elements, each of which are computed as inner products of n-length vectors. Thus, the algorithm makes $\mathcal{O}(mn^2)$ ring operations. Each ring operation in $\mathcal{R}_d$ under FFT can be done in $\Theta(d \log d)$. So the overall run-time of algorithm is $\mathcal{O}(mn^2 d \log d)$.

2.1.2 Babai's Nearest Plane Algorithm

Definition 2.4. *We define the ring* $\mathcal{Z}_d = \dfrac{\mathbb{Z}}{(x^d - 1)}$.
For a $m \times n$ *matrix* $\mathbf{A}$ *over* $\mathcal{R}_d$ *with full row rank, we denote the integer lattice generated by the rows of* $\mathbf{A}$ *by* $\mathcal{Z}_d^m\mathbf{A}$.

We will now recall the cornerstone lattice based cryptographic algorithm to solve the closest vector problem: Babai's Nearest Plane [Bab86]. Before proceeding, we first define the closest vector problem.

Definition 2.5. *Let* $\mathbf{A} \in \mathcal{R}_d^{m\times n}$ *have linearly independent rows. Given* $\mathbf{t} \in \mathcal{R}_d^m$, *the (approximate) Closest Vector Problem (CVP) seeks a* $\mathbf{z} \in \mathcal{Z}_d^m$ *such that* $(\mathbf{z} - \mathbf{t})\mathbf{A}$ *is small.*

Definition 2.6. *Given* $\mathbf{B} = \begin{bmatrix}\mathbf{b}_1 & \dots & \mathbf{b}_m\end{bmatrix}^T$ *of full row rank with entries from a ring* $\mathcal{R}$, *we define* $\mathcal{P}_\gamma(\mathbf{B})$ *as the parallelepiped generated by the rows of* $\mathbf{B}$ *defined as follows*

$$\mathcal{P}_\gamma(\mathbf{B}) = \{\mathbf{xB} : \mathbf{x} \in \mathcal{R}^m,\ \|\mathbf{x}\|_\infty \le \gamma\} = [-\gamma, \gamma]^m\,\mathbf{B},$$

where γ *is a constant depending on the underlying ring* $\mathcal{R}$. *When* γ *is not specified it is taken to be* $\frac{1}{2}$, *that is,* $\mathcal{P}(\mathbf{B}) = \mathcal{P}_{1/2}(\mathbf{B})$.

In particular, let $\mathcal{R}$ *be a ring and let* $\mathcal{Z}$ *be its subring of integral elements. Then*

$$\gamma = \sup_{f\in\mathcal{R}} \|f - \lfloor f \rceil\|,$$

where $\lfloor f \rceil$ *is the element of* $\mathcal{Z}$ *nearest to* f. *For example* $\gamma = 1/2$ *for* $\mathbb{R}$. *For* $\mathcal{R}_d$ *we have* $\gamma = \sqrt{d}/2$.

Now we explicitly write out Babai's algorithm, and then dive into a simple interpretation.

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

Require: $\mathbf{L}$, which is the L-part of the GSO of a full row rank $m \times n$ matrix $\mathbf{A} = \mathbf{LB}$ with entries in $\mathcal{R}_d$.
$\mathbf{t} \in \mathcal{R}_d^m$, such that the target vector is $\mathbf{tB}$

Ensure: $\mathbf{z} \in \mathcal{Z}_d^m$, such that $(\mathbf{t} - \mathbf{z})\mathbf{A} \in \mathcal{P}_\gamma(\mathbf{B})$

1: $\mathbf{z} \leftarrow \mathbf{0}$
2: **for** $j = m, \dots, 1$ **do**
3: $\quad \bar{t}_j \leftarrow t_j + \sum_{i>j}(t_i - z_i)L_{ij}$
4: $\quad z_j = \lfloor \bar{t}_j \rceil$
5: **end for**
6: **return z**

Interpretation. We will spend some time here understanding the intuition behind this algorithm. The algorithm is framed in several ways. The above framing will help us modify it into a Fourier-compatible version. However, the above framing may not immediately reflect its recursive nature. Let us try to illuminate that in the next steps.

Step 1. For $1 \leq j \leq m$, let $\mathbf{A}_j$ be the matrix of the first j rows of $\mathbf{A}$, that is, $\mathbf{A}_j = \begin{bmatrix}\mathbf{a}_1 & \dots & \mathbf{a}_j\end{bmatrix}^T$. Similarly, define $\mathbf{B}_j$. And let $\mathbf{L}_j$ be the principal $j \times j$ submatrix of $\mathbf{L}$. Then due to the iterative nature of the Gram-Schmidt process we get $\mathbf{A}_j = \mathbf{L}_j\mathbf{B}_j$ as a valid GSO of $\mathbf{A}_j$, and hence

$$\text{span}\{\mathbf{a}_1, \dots, \mathbf{a}_j\} = \text{span}\{\mathbf{b}_1, \dots, \mathbf{b}_j\} := S_j.$$

Step 2. The algorithm tries to find the solution $\mathbf{z}$ coordinate by coordinate, from z_m to z_1. Consider an intermediate, say $(m-j)$th, stage of the algorithm. Before this stage is entered we suppose that the coordinates $z_{j+1}, \dots, z_m$ are solved, and the vector $\mathbf{z}$ looks like

$$\mathbf{z} = \begin{bmatrix}0 & \dots & 0 & z_{j+1} & \dots & z_m\end{bmatrix}.$$

What this means is that at this stage the lattice $\mathbf{zA}$ is close to the target vector

$$\begin{bmatrix}0 & \dots & 0 & t_{j+1} & \dots t_m\end{bmatrix}\mathbf{A}.$$

Now, we will take this partial solution to construct the current step.

Step 3. Note that since $\mathbf{zA}$ is a lattice vector, solving CVP for $\mathbf{tA}$ is equivalent to solving CVP for $(\mathbf{t}-\mathbf{z})\mathbf{A}$. Now, here is the integral step of the algorithm. Since the last $m-j$ coordinates of $(\mathbf{t}-\mathbf{z})$ are small, we reduce this to a smaller problem. Precisely, instead of solving CVP for $(\mathbf{t}-\mathbf{z})\mathbf{A}$, we project this target vector to the space S_j, and solve CVP in smaller lattice $\mathcal{Z}_d^j\mathbf{A}_j$.

Let P be the projection onto S_j and let

$$\bar{\mathbf{t}}\mathbf{A} = P\left([\mathbf{t}-\mathbf{z}]\mathbf{A}\right).$$

From our algorithm, to find z_j, which is the first step of solving the CVP with a truncation of the vector $\bar{\mathbf{t}}$ in the smaller lattice $\mathcal{Z}_d^j\mathbf{A}_j$, all we need is $\bar{t}_j$.

Step 4. Now let $\mathbf{A}^\dagger$ be the pseudo-inverse of $\mathbf{A}$, that is, $\mathbf{A}\mathbf{A}^\dagger = \mathbf{I}$. Then

$$\bar{t}_j = P\left([\mathbf{t}-\mathbf{z}]\mathbf{A}\right)\mathbf{A}^\dagger\mathbf{e}_j^T.$$

But recall that $\mathbf{A} = \mathbf{LB}$, and further decompose $\mathbf{B} = \mathbf{\Sigma U}$, where $\mathbf{\Sigma}$ is positive diagonal matrix normalizing the rows of $\mathbf{B}$ to the orthonormal rows of $\mathbf{U}$. Then, we also have $\mathbf{B}_j = \mathbf{\Sigma}_j\mathbf{U}_j$, where $\mathbf{U}_j = \begin{bmatrix}\mathbf{u}_1 & \dots \mathbf{u}_j\end{bmatrix}^T$. And, the map P is achieved by the action of the matrix $\mathbf{U}_j^T\mathbf{U}_j$. Therefore, proceeding with $\mathbf{A}^\dagger = \mathbf{U}^T\mathbf{\Sigma}^{-1}\mathbf{L}^{-1}$,

we get

$$\begin{aligned}
\bar{t}_j &= (\mathbf{t}-\mathbf{z})\mathbf{A}\left(\mathbf{U}_j^T\mathbf{U}_j\right)\mathbf{A}^\dagger\mathbf{e}_j^T \\
&= (\mathbf{t}-\mathbf{z})\mathbf{L}\cdot\mathbf{\Sigma}\left(\mathbf{U}_j\mathbf{U}^T\right)^T\left(\mathbf{U}_j\mathbf{U}^T\right)\mathbf{\Sigma}^{-1}\cdot\mathbf{L}^{-1}\mathbf{e}_j^T \\
&= (\mathbf{t}-\mathbf{z})\mathbf{L}\cdot\mathbf{\Sigma}\begin{bmatrix}\mathbf{I}_j & \mathbf{0}\end{bmatrix}^T\begin{bmatrix}\mathbf{I}_j & \mathbf{0}\end{bmatrix}\mathbf{\Sigma}^{-1}\cdot\mathbf{L}^{-1}\mathbf{e}_j^T \\
&= (\mathbf{t}-\mathbf{z})\mathbf{L}\cdot\begin{bmatrix}\mathbf{I}_j & \\ & \mathbf{0}\end{bmatrix}\mathbf{L}^{-1}\mathbf{e}_j^T \\
&= (\mathbf{t}-\mathbf{z})\mathbf{L}\mathbf{e}_j^T \\
&= t_j + \sum_{i>j}(t_i - z_i)L_{ij},
\end{aligned}$$

where the second last equality is just the observation that the j-th column of $\mathbf{L}^{-1}$ is of the form $\begin{bmatrix}0 & \dots & 1 & *\end{bmatrix}$, since its a unit lower triangular matrix, the $*$ part is killed by the pre-multiplied matrix, leaving us with $\mathbf{e}_j^T$. This illuminates the expression in the $(m-j)$th step of the algorithm. Once the coordinate z_j is also solved, the same process can be repeated for the next coordinate.

In order to show that the algorithm provides what it promises, we will take help of the following lemma.

Lemma 2.1. *At any stage of the algorithm 2, the vectors* $\mathbf{t}, \bar{\mathbf{t}}$ *and* $\mathbf{z}$ *satisfies*

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

Proof. Let us start from the definition of $\bar{t}_j$, and follow with suitable steps.

$$\begin{aligned}
&\bar{t}_j = t_j + \sum_{i>j}(t_i - z_i)L_{ij} \\
\implies &\bar{t}_j - z_j = t_j - z_j + \sum_{i>j}(t_i - z_i)L_{ij} = (\mathbf{t}-\mathbf{z})\mathbf{L}\mathbf{e}_j^T \\
\implies &\bar{\mathbf{t}} - \mathbf{z} = (\mathbf{t}-\mathbf{z})\mathbf{L} \\
\implies &(\mathbf{z}-\bar{\mathbf{t}})\mathbf{B} = (\mathbf{z}-\mathbf{t})\mathbf{A}.
\end{aligned}$$

□

Corollary 2.1. *Algorithm 2 is correct.*

To see this, we simply note that $z_j = \lfloor \bar{t}_j \rceil$ implying that $|z_j - \bar{t}_j| \leq \gamma$. Hence $(\mathbf{z}-\mathbf{t})\mathbf{A} \in \mathcal{P}_\gamma(\mathbf{B})$.

Remark 2.2. *The subroutine that supplies the matrix* $\mathbf{L}$ *to Babai's algorithm in [DP16] is the LDL decomposition instead of the GSO decomposition. Even FALCON is implemented with LDL [PFH+19]. When a matrix is* $m \times n$*, the LDL decomposition does offer a complexity gain over GSO, since it handles diagonal matrices instead of full matrices. However, in our coming section, we focus only on* 2×2 *matrices over a ring, and hence the complexities are asymptotically equal*

for the two subroutines. However the involved constants sufficiently different for FALCON to use LDL for its fast implementation.

Theoretically, the two decompositions are equivalent. That is, $\mathbf{A} = \mathbf{LB}$ *is the GSO of* $\mathbf{A}$ *if and only if* $\mathbf{G} = \mathbf{LDL}^T$ *is the LDL decomposition of* $\mathbf{G} = \mathbf{AA}^T$*, so the two subroutines supply the same* $\mathbf{L}$ *to Babai's algorithm. Further, the illustration of the L-tree and the fast variant of Babai's algorithm remain unaffected by the choice of subroutine.*

2.2 Linearization Operators

Here onward we will only look at $\mathcal{R}_d$ where $d = 2^h$.

We make a preliminary note here: every result of this article holds, by simple extensions, when d is not a power of 2. However, we will stick to this special case for the sake of illustrations. Further, the most important application of these results – the NIST standardized signature scheme FALCON [PFH+19], only uses the special case where d is a power of 2.

Definition 2.7. Vectorize Operation. *Let* $d_i = 2^i$*, for* $i = 0, \ldots, h$*. For fixed* i, j *let*

$$\mathcal{R}_{d_i} = \mathbb{R}[x]/(x^{d_i}-1) \text{ and } \mathcal{R}_{d_j} = \mathbb{R}[y]/(x^{d_j}-1),$$

where $y = x^k$*, where* $k = {}^{d_i}/_{d_j}$*. Then the "vectorize" operator*

$$\mathcal{V}_{d_i}^{d_j} : \mathcal{R}_{d_i}^n \longrightarrow \mathcal{R}_{d_j}^{kn}$$

is defined recursively as follows.

1. *For all* i*, the operator* $\mathcal{V}_{d_i}^{d_i}$ *is identity,*
2. *For* $i > 0$ *and* $n = 1$*, let us consider an arbitrary polynomial* $f \in \mathcal{R}_{d_i}$*, and write*

$$\begin{aligned} f(x) &= f_0 + f_1 x + \cdots + f_{d_i-1} x^{d_i-1} \\ &= f_0 + f_2 x^2 + \cdots + f_{d_i-2} \left(x^2\right)^{d_{i-1}-1} \\ &\quad + x\left[f_1 + f_2 x^2 + \cdots + f_{d_i-1}\left(x^2\right)^{d_{i-1}-1}\right] \\ &:= f_o(y) + x f_e(y). \end{aligned}$$

   Then we write

$$\mathcal{V}_{d_i}^{d_{i-1}} = \begin{bmatrix} f_o & f_e \end{bmatrix}.$$

3. *For a vector* $\begin{bmatrix} f & g & \ldots \end{bmatrix}$*, the operator* $\mathcal{V}_{d_i}^{d_{i-1}}$ *acts component-wise according to bullet 2.*
4. *Generally for* $i > j$ *and a single polynomial* $f \in \mathcal{R}_{d_i}$*, we have*

$$\mathcal{V}_{d_i}^{d_j}(f) = \mathcal{V}_{d_{j+1}}^{d_j} \circ \mathcal{V}_{d_{j+2}}^{d_{j+1}} \circ \cdots \circ \mathcal{V}_{d_i}^{d_{i-1}}(f).$$

   For vectors of polynomials, this operation is applied component-wise.

5. *Finally, for $i < j$, we define*

$$\mathcal{V}_{d_i}^{d_j} = \left(\mathcal{V}_{d_j}^{d_i}\right)^{-1}.$$

For conciseness, we will use the shorthand $\mathcal{V}_{d_i}$ for the operator $\mathcal{V}_{d_i}^1$. To see this in action we look at the following example 1.

Example 1. For $d = 8$, we look at the polynomial $f(x) = x + x^2 + 2x^3 + \cdots + 7x^7$ and show how its coefficient vector is rearranged to obtain $\mathcal{V}_8(f)$.

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix} = x + x^2 + 2x^3 + \cdots + 7x^7$$

$$\Big\downarrow \mathcal{V}_8^4$$

$$\begin{bmatrix} 0 & 2 & 4 & 6 & 1 & 3 & 5 & 7 \end{bmatrix} = \begin{bmatrix} 2y + 4y^2 + 6y^3 & 1 + 3y + 5y^2 + 7y^3 \end{bmatrix}$$

$$\Big\downarrow \mathcal{V}_4^2$$

$$\begin{bmatrix} 0 & 4 & 2 & 6 & 1 & 5 & 3 & 7 \end{bmatrix} = \begin{bmatrix} 4z & 2 + 6z & 1 + 5z & 3 + 7z \end{bmatrix}$$

$$\Big\downarrow \mathcal{V}_4^2$$

$$\begin{bmatrix} 0 & 4 & 2 & 6 & 1 & 5 & 3 & 7 \end{bmatrix}$$

The vectorization is done with the subsequent ring homomorphisms $x^2 \to y$ and $y^2 \to z$.

Definition 2.8. Matrixify Operation. *Refer to the notations in definition 2.7. Here, for $i \geq j$, the "matrixify" operator*

$$\mathcal{M}_{d_i}^{d_j} : \mathcal{R}_{d_i}^{m \times n} \longrightarrow \mathcal{R}_{d_j}^{km \times kn}$$

is defined as follows.

1. *For all i, the operator $\mathcal{M}_{d_i}^{d_i}$ is identity.*

2. *For a single polynomial $f \in \mathcal{R}_{d_i}$, the operator $\mathcal{M}_{d_i}^{d_{i-1}}(f)$ is the unique matrix such that for all $g \in \mathcal{R}_{d_i}$*

$$\mathcal{V}_{d_i}^{d_{i-1}}(g)\mathcal{M}_{d_i}^{d_{i-1}}(f) = \mathcal{V}_{d_i}^{d_{i-1}}(gf).$$

If $\mathcal{V}_{d_i}^{d_{i-1}}(f) = \begin{bmatrix} f_o & f_e \end{bmatrix}$, it turns out that

$$\mathcal{M}_{d_i}^{d_{i-1}}(f) = \begin{bmatrix} f_0 & f_e \\ f_e & f_o \end{bmatrix}.$$

3. *For a matrix in* $\mathcal{R}_{d_i}^{m\times n}$*, the operator* $\mathcal{M}_{d_i}^{d_{i-1}}$ *applies component-wise.*

4. *Finally, for arbitrary* $i > j$*,*

$$\mathcal{M}_{d_i}^{d_j}(f) = \mathcal{M}_{d_{j+1}}^{d_j} \circ \mathcal{M}_{d_{j+2}}^{d_{j+1}} \circ \cdots \circ \mathcal{M}_{d_i}^{d_{i-1}}(f).$$

For conciseness, the operator $\mathcal{M}_{d_i}^1$ is written as $\mathcal{M}_{d_i}$. An example of matrixification is seen in example 3.

Proposition 2.1. *The operators* $\mathcal{V}$ *and* $\mathcal{M}$ *(with the appropriate sub and superscripts) have some nice properties as follows.*

- $\mathcal{M}$ *is multiplicative, that is* $\mathcal{M}(\mathbf{AB}) = \mathcal{M}(\mathbf{A})\mathcal{M}(\mathbf{B})$.
- $\mathcal{M}$ *and* $\mathcal{V}$ *are compatible in the sense that* $\mathcal{V}(f)\mathcal{M}(g) = \mathcal{V}(fg)$ *for polynomials* f, g.
- $\mathcal{V}$ *preserves inner product, that is* $\langle \mathcal{V}(\mathbf{a}), \mathcal{V}(\mathbf{b}))\rangle = \langle \mathbf{a}, \mathbf{b}\rangle$.
- $\mathbf{A}$ *has linearly independent rows if and only if* $\mathcal{M}(\mathbf{A})$ *has linearly independent rows.*

Remark 2.3. *For* $f \in \mathcal{R}_d$*, representations of* $\mathcal{V}_d^{d/2}(f)$ *and* $\mathcal{M}_d^{d/2}(f)$ *can be computed in* $\mathcal{O}(d)$ *both in standard and FFT format.*

To see this let's recall that $\mathcal{V}_d^{d/2}(f) = \begin{bmatrix} f_o & f_e \end{bmatrix}$ as defined in definition 2.7. As we see in example 1, the coefficients of the polynomials f_o and f_e are only subsets of the coefficients of f, and together they are only a rearrangement of f's coefficients. Thus this transformation is easily done in $\mathcal{O}(d)$.

Similarly, for matrixification of f, we note that $\mathcal{M}_d^{d/2}$ is also completely characterized by f_o and f_e. More precisely, $\mathcal{M}_d^{d/2}$ is the circulant generated by $\mathcal{V}_d^{d/2}(f)$. Hence even this operation can be computed in $\mathcal{O}(d)$ time.

The same holds for the FFT format as well where again the coefficients of f_o and f_e are linear manipulations on the coefficients of f.

3 Compact Gram-Schmidt Orthogonalization

3.1 Existence of Compact Factorization of L

Following the definitions in section 2.2, we are ready to state the central result of this article.

Theorem 1. *Let* $d_i = 2^i$ *for* $i = 0, \ldots, h$*. Let* $\mathbf{b} \in \mathcal{R}_{d_h}^m$ *such that* $\mathcal{M}_{d_h}(\mathbf{b})$ *has linearly independent rows. If* $\mathcal{M}_{d_h}(\mathbf{b}) = \mathbf{LB}$ *is the GSO of* $\mathcal{M}_{d_h}(\mathbf{b})$*, then the matrix* $\mathbf{L}$ *admits the following decomposition*

$$\mathbf{L} = \prod_{i=1}^{h} \mathcal{M}_{d_{h-i}}(\mathbf{L}_i)$$

where the matrices $\mathbf{L}_i$s have a block diagonal structure with d_{i-1} unit lower triangular blocks of the form

$$\mathbf{L}_i = \begin{bmatrix} \mathbf{1} & \mathbf{0} & & & \\ L_{i,1} & \mathbf{1} & & & \\ & & \ddots & & \\ & & & \mathbf{1} & \mathbf{0} \\ & & & L_{i,d_{i-1}} & \mathbf{1} \end{bmatrix} \in \mathcal{R}_{d_{h-i}}^{d_i \times d_i}.$$

Proof. We will proceed by induction on h. When $h = 1$, this is the standard GSO, so the hypothesis holds.

Suppose that the hypothesis holds for $h-1$. Consider the matrix $\mathcal{M}_{d_h}^{d_{h-1}}(\mathbf{b})$ and its GSO

$$\mathcal{M}_{d_h}^{d_{h-1}}(\mathbf{b}) = \mathbf{L}_1 \tilde{\mathbf{B}} \quad \text{where} \quad \mathbf{L}_1 = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ L_{1,1} & \mathbf{0} \end{bmatrix} \in \mathcal{R}_{d_{h-1}}^{2\times 2}.$$

Now, say $\tilde{\mathbf{B}}$ has row representation $\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}^T$ where the rows $\mathbf{b}_1$ and $\mathbf{b}_2$ are orthogonal to each other with $\mathbf{b}_j \in \mathcal{R}_{d_{h-1}}^{2m}$. Then by induction hypothesis, we have the following GSOs for $j = 1, 2$

$$\mathcal{M}_{d_{h-1}}(\mathbf{b}_j) = \prod_{i=2}^{h} \mathcal{M}_{d_{h-i}}(\mathbf{L}_{i,j}) \cdot \mathbf{B}_j.$$

Note, this decomposition is the same as in the statement of the theorem, except that for notational convenience here we reindex i as $i+1$. Therefore, here each $\mathbf{L}_{i,j}$ is a $d_{i-1} \times d_{i-1}$ matrix with entries from $\mathcal{R}_{d_{h-i}}$, further $\mathbf{L}_{i,j}$ is block diagonal with d_{i-2} unit lower triangular blocks. With the established properties of the matrixify operation, we note that

$$\begin{aligned} \mathcal{M}_{d_h}(\mathbf{b}) &= \mathcal{M}_{d_{h-1}} \circ \mathcal{M}_{d_h}^{d_{h-1}}(\mathbf{b}) \\ &= \mathcal{M}_{d_{h-1}}(\mathbf{L}_1) \begin{bmatrix} \mathcal{M}_{d_{h-1}}(\mathbf{b}_1) \\ \mathcal{M}_{d_{h-1}}(\mathbf{b}_2) \end{bmatrix} \\ &= \mathcal{M}_{d_{h-1}}(\mathbf{L}_1) \prod_{i=2}^{h} \begin{bmatrix} \mathcal{M}_{d_{h-i}}(\mathbf{L}_{i,1}) & \\ & \mathcal{M}_{d_{h-i}}(\mathbf{L}_{i,2}) \end{bmatrix} \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix} \\ &= \prod_{i=1}^{h} \mathcal{M}_{d_{h-i}}(\mathbf{L}_i) \cdot \mathbf{B} \end{aligned}$$

with the identifications of $\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{bmatrix}$, and $\mathbf{L}_i = \begin{bmatrix} \mathbf{L}_{i,1} & \\ & \mathbf{L}_{i,2} \end{bmatrix}$ for $i = 2, \ldots, h$.

To complete the proof we need to verify the dimensions of the blocks $\mathbf{L}_i$ for $i \geq 2$, and verify that $\mathbf{B}$ is indeed an orthogonal matrix. Indeed, since each $\mathbf{L}_{i,j}$ is a $d_{i-1} \times d_{i-1}$ block diagonal matrix with d_{i-2} unit lower triangular blocks and entries from $\mathcal{R}_{d_{h-i}}$, it follows that $\mathbf{L}_i$ is $d_i \times d_i$ block diagonal matrix with d_{i-1} unit lower triangular blocks with entries from $\mathcal{R}_{d_{h-i}}$.

For the orthogonality of $\mathbf{B}$, consider two rows $\mathbf{u}_1$ and $\mathbf{u}_2$ of $\mathbf{B}$. If both of them belong to either $\mathbf{B}_1$ or $\mathbf{B}_2$, they are orthogonal by the orthogonality of $\mathbf{B}_1$ and $\mathbf{B}_2$, which is ensured by the induction hypothesis. Now, without loss of generality, assume $\mathbf{u}_1$ is a row of $\mathbf{B}_1$ and $\mathbf{u}_2$ of $\mathbf{B}_2$. Then there exists row vectors $\mathbf{c}_1$ and $\mathbf{c}_2$ such that for $j = 1, 2$

$$\mathbf{u}_j = \mathbf{c}_j \mathcal{M}_{d_{h-1}}(\mathbf{b}_j),$$

since the rows of the orthogonal matrix obtained by GSO are linear combinations of the rows of the original matrix. Let $c_j = \mathcal{V}_1^{d_{h-1}}(\mathbf{c}_j)$. Then

$$\begin{aligned}
\langle \mathbf{u}_1, \mathbf{u}_2 \rangle &= \left\langle \mathcal{V}_{d_{h-1}}(c_1)\mathcal{M}_{d_{h-1}}(\mathbf{b}_1), \mathcal{V}_{d_{h-1}}(c_2)\mathcal{M}_{d_{h-1}}(\mathbf{b}_2) \right\rangle \\
&= \left\langle \mathcal{V}_{d_{h-1}}(c_1\mathbf{b}_1), \mathcal{V}_{d_{h-1}}(c_2\mathbf{b}_2) \right\rangle \\
&= \langle c_1\mathbf{b}_1, c_2\mathbf{b}_2 \rangle \\
&= 0,
\end{aligned}$$

where the second and third inequality follows from proposition 2.1 and the last inequality is a consequence of $\mathbf{b}_1$ and $\mathbf{b}_2$ being orthogonal by definition. □

Theorem 1 is defined so far for vectors. However, it is easily extended to matrices. Continuing with the theme of stating our results for powers of 2, let us look at a matrix $\mathbf{A} \in \mathcal{R}_{d_h}^{2\times 2}$ which admits the Gram-Schmidt Orthogonalization $\mathbf{A} = \mathbf{L}_0\tilde{\mathbf{B}}$ in $\mathcal{R}_{d_h}$. Note here that $\tilde{\mathbf{B}}$ is redefined here from the previous proof, however the same notation is used as the argument is identical. Then,

$$\begin{aligned}
\mathcal{M}_{d_h}(\mathbf{A}) &= \mathcal{M}_{d_h}(\mathbf{L}_0) \cdot \mathcal{M}_{d_h}(\tilde{\mathbf{B}}) \\
&= \mathcal{M}_{d_h}(\mathbf{L}_0) \cdot \begin{bmatrix} \mathcal{M}_{d_h}(\mathbf{b}_1) \\ \mathcal{M}_{d_h}(\mathbf{b}_2) \end{bmatrix},
\end{aligned}$$

where each $\mathcal{M}_{d_h}(\mathbf{b}_1)$ can be factored using theorem 1. Collecting these factors in diagonal blocks, like we did in the proof of thorem 1, we get the following corollary.

Corollary 3.1. *Let $d_i = 2^i$ for $i = 0, \ldots, h$. Let $\mathbf{A} \in \mathcal{R}_{d_h}^{2\times 2}$ be nonsingular. If $\mathcal{M}_{d_h}(\mathbf{A}) = \mathbf{LB}$ is the GSO of $\mathcal{M}_{d_h}(\mathbf{A})$, then the matrix $\mathbf{L}$ admits the following decomposition*

$$\mathbf{L} = \prod_{i=0}^{h} \mathcal{M}_{d_{h-i}}(\mathbf{L}_i)$$

where the matrices $\mathbf{L}_i$s have a block diagonal structure with d_i unit lower triangular blocks of the form

$$\mathbf{L}_i = \begin{bmatrix} \mathbf{1} & \mathbf{0} & & & \\ L_{i,1} & \mathbf{1} & & & \\ & & \ddots & & \\ & & & \mathbf{1} & \mathbf{0} \\ & & & L_{i,d_i} & \mathbf{1} \end{bmatrix} \in \mathcal{R}_{d_{h-i}}^{d_{i+1}\times d_{i+1}}.$$

The proofs of the above results also allow us a constructive approach to compute the factored representation of the GSO of $\mathcal{M}_{d_h}(\mathbf{A})$, and store it in a compact

manner. Even though $\mathbf{L}_j$ is a $d_{j+1} \times d_{j+1}$ matrix over $\mathcal{R}_{d_{h-j}}$, its structure allows us to represent it only using d_i ring elements $\{L_{i,1}, \ldots, L_{j,d_j}\}$. Further, all these elements can be stored in the j-th level of a binary tree due to the structure of their computation.

More precisely, let us carefully reconstruct the jth step of the process. At the jth step we have

$$\mathcal{M}_{d_h}(\mathbf{A}) = \prod_{i=0}^{j} \mathcal{M}_{d_{h-j}}(\mathbf{L}_j) \cdot \mathcal{M}_{d_{h-j}} \left(\begin{bmatrix} \mathbf{B}_{j,1} \\ \vdots \\ \mathbf{B}_{j,d_j} \end{bmatrix} \right).$$

At this point, for each $k = 1, \ldots, d_j$ we write

$$\begin{aligned} \mathcal{M}_{d_{h-j}}(\mathbf{B}_{j,k}) &= \mathcal{M}_{d_{h-j-1}} \circ \mathcal{M}_{d_{h-j}}^{d_{h-j-1}}(\mathbf{B}_{j,k}) \\ &= \mathcal{M}_{d_{h-j-1}} \circ \mathcal{M}_{d_{h-j}}^{d_{h-j-1}} \left(\begin{bmatrix} \mathbf{b}_{j,k,1} \\ \mathbf{b}_{j,k,2} \end{bmatrix} \right) \\ &= \mathcal{M}_{d_{h-j-1}} \left(\begin{bmatrix} \mathbf{L}_{j+1,2k-1} & \\ & \mathbf{L}_{j+1,2k} \end{bmatrix} \right) \mathcal{M}_{d_{h-j-1}} \left(\begin{bmatrix} \mathbf{B}_{j+1,2k-1} \\ \mathbf{B}_{j+1,2k} \end{bmatrix} \right). \end{aligned}$$

What we see above is that the matrices at the $(j+1)$st step $\mathbf{L}_{j+1,2k-1}$ and $\mathbf{L}_{j+1,2k}$ are offshoots of the matrixification of the rows of $\mathbf{B}_{j,k}$. This immediately suggests a tree structure, which leads us to our next definition.

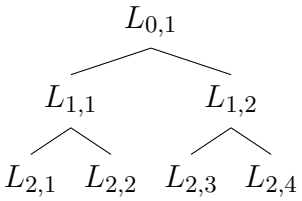
Definition 3.1. *Let $\mathbf{A} \in \mathcal{R}_{d_h}^{2\times 2}$. Then the binary $\mathcal{L}$ is called the **L-tree** of $\mathbf{A}$ and has the following properties:*

- *$\mathcal{L}$ has $h+1$ levels from $0, \ldots, h$.*
- *A node $L_{i,k}$ at the ith level has children $L_{i+1,2k-1}$ and $L_{i+1,2k}$.*
- *Every node $L_{i,k} \in \mathcal{R}_{d_{h-i}}$ is precisely the elements appearing in the compact factorization of the matrix $\mathbf{L}$ which appears in the GSO of $\mathcal{M}_{d_h}(\mathbf{A})$ as in corollary 3.1.*

Example 2. Consider a matrix $\mathbf{A} \in \mathcal{R}_4^{2\times 2}$ such that $\mathcal{M}_4(\mathbf{A}) = \mathbf{LB}$. According to corollary 3.1 suppose $\mathbf{L}$ has the following decomposition.

$$\mathbf{L} = \mathcal{M}_4 \begin{bmatrix} \mathbf{1} & \\ L_{0,1} & \mathbf{1} \end{bmatrix} \mathcal{M}_2 \begin{bmatrix} \mathbf{1} & & & \\ L_{1,1} & \mathbf{1} & & \\ & & \mathbf{1} & \\ & & L_{1,2} & \mathbf{1} \end{bmatrix} \mathcal{M}_1 \begin{bmatrix} \mathbf{1} & & & & & & & \\ L_{2,1} & \mathbf{1} & & & & & & \\ & & \mathbf{1} & & & & & \\ & & L_{2,2} & \mathbf{1} & & & & \\ & & & & \mathbf{1} & & & \\ & & & & L_{2,3} & \mathbf{1} & & \\ & & & & & & \mathbf{1} & \\ & & & & & & L_{2,4} & \mathbf{1} \end{bmatrix}.$$

Then $\mathbf{A}$ will have a L-tree $\mathcal{L}$ of the following form.



Thus $\mathbf{L}$ can be easily reconstructed from $\mathcal{L}$.

3.2 Fast Algorithm to Compute L-tree

The above formulation of an L-tree immediately furnishes an algorithm to compute it. We present it as algorithm 3 below. We call our algorithm FF-GSO as the L-tree of $\mathbf{A}$ essentially instantiates the GSO of $\mathcal{M}_d(\mathbf{A})$ where d is a power of 2, and the FF is reminiscent of the Fast Fourier flavor of matrixifying (or splitting) the ring elements.

Algorithm 3 $\mathsf{FF\text{-}GSO}_{\mathcal{R}_d}(\mathbf{A})$

Require: $\mathbf{A} \in \mathcal{R}_d^{2\times 2}$, a nonsingular matrix
Ensure: $\mathcal{L}$, the L-tree of $\mathbf{A}$ in FFT format
1: $\mathbf{L}, \mathbf{B} \leftarrow \mathsf{GSO}_{\mathcal{R}_d}(\mathbf{A})$
2: **if** $d = 1$ **then**
3: **return** $(\mathbf{L})$
4: **end if**
5: **for** $i = 1, 2$ **do**
6: $\mathcal{L}_i \leftarrow \mathsf{FF\text{-}GSO}_{\mathcal{R}_{d/2}}\left(\mathcal{M}_d^{d/2}(\mathbf{b}_i)\right)$
7: **end for**
8: **return** $(\mathbf{L}, [\mathcal{L}_1, \mathcal{L}_2])$

Running Time. We note that the GSO of a 2×2 matrix $\mathbf{A}$ on $\mathcal{R}_d$ requires a constant number of ring operations. Each ring operation can be done using in FFT format in $\mathcal{O}(d \log d)$ time. The computation of $\mathcal{M}_d^{d/2}(\mathbf{A})$ is done in $\mathcal{O}(d)$, refer to remark 2.3. Then if the algorithm runs in time $\beta(d)$, then $\beta(d)$ follows the following recurrence

$$\beta(d) = \mathcal{O}(d \log d) + 2\beta\left(d/2\right).$$

One can solve the recurrence using a telescopic sum to obtain $\beta(d) = \mathcal{O}(d \log d)$.

4 Fast Fourier Nearest Plane

We will modify algorithm 2 to curate a variant of the Nearest Plane algorithm that solves CVP on lattices whose basis is in $\mathcal{R}_d^{2\times 2}$, and compatible with the L-tree produced in algorithm 3.

The intuition here is to recursively compute $\mathbf{z}$. Instead of computing $z_j = \lfloor t_j \rceil$, here we postpone the nearest integer transformation to the child ring in a recursive manner, and lift the solution up to the current ring.

Algorithm 4 $\mathsf{FF\text{-}NP}_{\mathcal{R}_d}(\mathbf{t}, \mathcal{L})$

Require: $\mathbf{t} \in \mathcal{R}_d^2$, with $d = 2^k$ such that $\mathbf{tA}$ is the target vector, where $\mathbf{A} \in \mathcal{R}_d^{2\times 2}$, $\mathcal{L}$, which is a precomputed L-tree of $\mathbf{A}$
Ensure: $\mathbf{z} \in \mathcal{Z}_d^2$ such that $\mathcal{V}_d\left([\mathbf{z}-\mathbf{t}]\mathbf{A}\right) \in \mathcal{P}(\mathbf{B})$ where $\mathbf{B}$ is the orthogonalization of $\mathcal{M}_d(\mathbf{A})$.

if $d = 1$ **then**
  $L \leftarrow \mathcal{L}$
  return $\mathsf{NP}_{\mathbb{R}}(L, \mathbf{t})$
end if
$L, [\mathcal{L}_1, \mathcal{L}_2] \leftarrow \mathcal{L}$
for $j = 2, 1$ **do**
  $\bar{t}_j \leftarrow t_j + \sum_{i>j}(t_i - z_i)L_{ij}$
  $z_j \leftarrow \mathcal{V}_{d/2}^{d}\left[\mathsf{FF\text{-}NP}_{\mathcal{R}_{d/2}}\left(\mathcal{V}_d^{d/2}(\bar{t}_j), \mathcal{L}_2\right)\right]$
end for
return $\mathbf{z} = \begin{bmatrix} z_1 & z_2 \end{bmatrix}$

Using lemma 2.1 we prove the following theorem about the correctness of algorithm 4.

Theorem 2. *Algorithm 4 outputs a vector* $\mathbf{z}$ *such that* $\mathcal{V}_d\left([\mathbf{z}-\mathbf{t}]\mathbf{A}\right) \in \mathcal{P}(\mathbf{B})$ *where* $\mathbf{B}$ *is the orthogonalization of* $\mathcal{M}_d(\mathbf{A})$.

Proof. The proof of this theorem will closely link with the proof of theorem 1. We will proceed by induction on d. When $d = 1$, $\mathsf{FF\text{-}NP}_{\mathcal{R}_d}$ is just $\mathsf{NP}_{\mathbb{R}}$, and hence this case holds.

Suppose $\mathsf{FF\text{-}NP}_{\mathcal{R}_{d/2}}$ is correct. First, recall $\mathbf{A} = \mathbf{L}_0\tilde{\mathbf{B}}$, where $\tilde{\mathbf{B}} = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{bmatrix}^T$. Recall further that $\mathcal{L}_j$ is the L-tree for $\mathcal{M}_{d/2}(\mathbf{b}_j)$. Then by induction hypothesis we get

$$\mathcal{V}_{d/2}\left(\left[\mathcal{V}_d^{d/2}(z_j) - \mathcal{V}_d^{d/2}(\bar{t}_j)\right]\mathbf{b}_j\right) \in \mathcal{P}(\mathbf{B}_j),$$

where $\mathbf{B}_j$ is the orthogonalization of $\mathcal{M}_{d/2}(\mathbf{b}_j)$. Thus for $j = 1, 2$ we have

$$\begin{aligned} &\mathcal{V}_d\left([z_j - \bar{t}_j]\mathbf{b}_j\right) \in \mathcal{P}(\mathbf{B}_j) \\ \implies &\mathcal{V}_d\left([\mathbf{z} - \bar{\mathbf{t}}]\tilde{\mathbf{B}}\right) \in \mathcal{P}(\mathbf{B}) \\ \implies &\mathcal{V}_d\left([\mathbf{z} - \mathbf{t}]\mathbf{A}\right) \in \mathcal{P}(\mathbf{B}), \end{aligned}$$

where the last inequality follows lemma 2.1. This concludes the proof. □

Running Time. Let $\beta(d)$ be the running time of $\mathsf{FF\text{-}NP}$ on $\mathcal{R}_d$, then $\beta(d)$ shares the same recurrence relation as the running time of $\mathsf{FF\text{-}GSO}$ presented in algorithm 3. Hence, the running time of $\mathsf{FF\text{-}NP}$ is also $\mathcal{O}(d \log d)$.

Example 3. Similar to vectorization, for $d = 8$ we go down along the rings $\mathcal{R}_8, \mathcal{R}_4, \mathcal{R}_2$ in indeterminates x, y and z respectively, achieved by ring homomorphisms $x^2 \to y$ and $y^2 \to z$. Here, we see $\mathcal{M}_8(f)$ where $f = x + 2x^2 + \cdots + 7x^7$ for $d = 8$.

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 7 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 7 & 0 & 1 & 2 & 3 & 4 & 5 \\ 5 & 6 & 7 & 0 & 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 & 0 & 1 & 2 & 3 \\ 3 & 4 & 5 & 6 & 7 & 0 & 1 & 2 \\ 2 & 3 & 4 & 5 & 6 & 7 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 \end{bmatrix} \quad \text{OR} \quad \begin{matrix} x + 2x^2 + 3x^3 + 4x^4 \\ 5x^5 + 6x^6 + 7x^7 \end{matrix}$$

$$\downarrow \mathcal{M}_8^4$$

$$\begin{bmatrix} 0 & 2 & 4 & 6 & 1 & 3 & 5 & 7 \\ 6 & 0 & 2 & 4 & 7 & 1 & 3 & 5 \\ 4 & 6 & 0 & 2 & 5 & 7 & 1 & 3 \\ 2 & 4 & 6 & 0 & 3 & 5 & 7 & 1 \\ 7 & 1 & 3 & 5 & 0 & 2 & 4 & 6 \\ 5 & 7 & 1 & 3 & 6 & 0 & 2 & 4 \\ 3 & 5 & 7 & 1 & 4 & 6 & 0 & 2 \\ 1 & 3 & 5 & 7 & 2 & 4 & 6 & 0 \end{bmatrix} \quad \text{OR} \quad \begin{bmatrix} 2y + 4y^2 + 6y^3 & 1 + 3y + 5y^2 + 7y^3 \\ 7 + y + 3y^2 + 5y^3 & 2y + 4y^2 + 6y^3 \end{bmatrix}$$

$$\downarrow \mathcal{M}_4^2$$

$$\begin{bmatrix} 0 & 4 & 2 & 6 & 1 & 5 & 3 & 7 \\ 4 & 0 & 6 & 2 & 5 & 1 & 7 & 3 \\ 6 & 2 & 0 & 4 & 7 & 3 & 1 & 5 \\ 2 & 6 & 4 & 0 & 3 & 7 & 5 & 1 \\ 7 & 3 & 1 & 5 & 0 & 4 & 2 & 6 \\ 3 & 7 & 5 & 1 & 4 & 0 & 6 & 2 \\ 5 & 1 & 7 & 3 & 6 & 2 & 0 & 4 \\ 1 & 5 & 3 & 7 & 2 & 6 & 4 & 0 \end{bmatrix} \quad \text{OR} \quad \begin{bmatrix} 4z & 2 + 6z & 1 + 5z & 3 + 7z \\ 6 + 2z & 4z & 7 + 3z & 1 + 5z \\ 7 + 3z & 1 + 5z & 4z & 2 + 6z \\ 5 + z & 7 + 3z & 6 + 2z & 4z \end{bmatrix}$$

Note that the last step $\mathcal{M}_2^1$ keeps the order of the matrix elements unchanged.

References

[Bab86] László Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

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

[DP16] Léo Ducas and Thomas Prest. Fast fourier orthogonalization. In *Proceedings of the 2016 ACM International Symposium on Symbolic and Algebraic Computation*, page 191198. ACM, 2016.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory: Third International Symposium, ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288, Berlin, Heidelberg, 1998. Springer.

[PFH+19] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over NTRU. Technical report, National Institute of Standards and Technology, 2019. Submission to the NIST Post-Quantum Cryptography Standardization Project.

[Reg05] Oded Regev. On lattices, learning with errors, and post-quantum cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.