A Superfast Direct Solver for 2D Type-II Inverse Nonuniform Discrete Fourier Transform Based on Hierarchically Semiseparable Matrix

Yingzhou Li*, Jingyu Liu†

July 2, 2026

Abstract

This paper proposes a direct inversion method for the 2D type-II nonuniform discrete Fourier transform (NUDFT). The NUDFT matrix $\boldsymbol{A}$ is factored as $\boldsymbol{A} = \boldsymbol{G}\boldsymbol{F}$, where $\boldsymbol{G}$ can be expressed as a kernel matrix and $\boldsymbol{F}$ is the 2D DFT matrix. We show that $\boldsymbol{G}$ can be approximated by a hierarchically semiseparable (HSS) matrix and give an estimate of the HSS rank. Then, using the least-squares solver for HSS matrix and the two-dimensional inverse fast Fourier transform, the inverse NUDFT problem can be solved efficiently. Our algorithm has an offline complexity of $\mathcal{O}\big(M + N^{3/2}\log^3 N\big)$ where M and N are the size of rows and columns of the NUDFT matrix, respectively. Once the direct solver is built, it can be applied to a vector with an online complexity of $\mathcal{O}\big(M + N\log^3 N\big)$. The proposed method can be used as a preconditioner for iterative methods, especially when the sample points are distributed on a grid such that $\boldsymbol{A}$ is ill-conditioned. Numerical results are provided to show the scaling performance of the inversion method and demonstrate the efficiency and robustness of it as a preconditioner.

Keywords nonuniform discrete Fourier transform, hierarchically semi-separable matrix

1 Introduction

This paper considers the two-dimensional type-II *nonuniform discrete Fourier transform* (NUDFT) of the following form:

$$f_j = \sum_{k^{[x]}=0}^{n^{[x]}-1}\sum_{k^{[y]}=0}^{n^{[y]}-1} c_{k^{[x]},k^{[y]}}\,\mathrm{e}^{-2\pi\mathrm{i}(k^{[x]}x_j + k^{[y]}y_j)}, \quad 0 \le j \le M-1, \tag{1.1}$$

where the *sample points* $\{(x_j, y_j)\}$ are distributed arbitrarily in $[0,1)^2$ and the *frequencies* $\{(k^{[x]}, k^{[y]})\}$ are distributed on a Cartesian grid of contiguous integers. Throughout the paper, the number of samples points M is assumed to be larger than or equal to the number of frequencies, i.e., $N = n^{[x]}n^{[y]}$. When the *coefficients* $\{c_{k^{[x]},k^{[y]}}\}$ are given and one aims to compute the *target values* $\{f_j\}$, the problem is called the *forward*

*School of Mathematical Sciences, Fudan University; Shanghai Key Laboratory for Contemporary Applied Mathematics, Fudan University, yingzhouli@fudan.edu.cn

†School of Mathematical Sciences, Fudan University, jyliu22@m.fudan.edu.cn

NUDFT. When the target values $\{f_j\}$ are given and one aims to determine the coefficients $\{c_{k^{[x]},k^{[y]}}\}$, the problem is called the *inverse* NUDFT. Both the forward and inverse NUDFT have wide applications in scientific computing, such as signal processing [2], image reconstruction [7], and fast convolution [19] etc. Let the NUDFT matrix $\boldsymbol{A}$ be defined as[1]

$$\boldsymbol{A}\big(j,(k^{[x]},k^{[y]})\big)=\mathrm{e}^{-2\pi\mathrm{i}(k^{[x]}x_j+k^{[y]}y_j)},\quad 0\le j\le M-1,\ 0\le k^{[x]}\le n^{[x]}-1,\ 0\le k^{[y]}\le n^{[y]}-1, \tag{1.2}$$

where the column index $(k^{[x]},k^{[y]})$ is ordered in a lexicographical manner. That is, we have an underlying 1-to-1 map $(k^{[x]},k^{[y]})\leftrightarrow k^{[y]}+k^{[x]}n^{[y]}$. The forward NUDFT can be expressed as a matrix-vector multiplication $\boldsymbol{f}=\boldsymbol{A}\boldsymbol{c}$ and the inverse NUDFT can be formulated as a linear least-squares problem $\min_{\boldsymbol{c}}\|\boldsymbol{A}\boldsymbol{c}-\boldsymbol{f}\|_2$, where $\boldsymbol{c}\in\mathbb{C}^N$ is the vectorized form of the coefficients $\{c_{k^{[x]},k^{[y]}}\}$ and $\boldsymbol{f}\in\mathbb{C}^M$ is the vector corresponding to the target values $\{f_j\}$. Typically, the forward and inverse NUDFT have a complexity of $\mathscr{O}(MN)$ and $\mathscr{O}(MN^2)$ respectively.

1.1 Related Work

If the sample points are distributed on a uniformly Cartesian grid, i.e., $(x_{j^{[x]}},y_{j^{[y]}})=(j^{[x]}/n^{[x]},j^{[y]}/n^{[y]})$ for $0\le j^{[x]}\le n^{[x]}-1$ and $0\le j^{[y]}\le n^{[y]}-1$, the NUDFT reduces to the *discrete Fourier transform* (DFT). Consequently, the NUDFT matrix $\boldsymbol{A}$ has a Kronecker product structure $\boldsymbol{A}=\boldsymbol{A}^{[x]}\otimes\boldsymbol{A}^{[y]}$, where $\boldsymbol{A}^{[x]}(j^{[x]},k^{[x]})=\mathrm{e}^{-2\pi k^{[x]}j^{[x]}/n^{[x]}}$ and $\boldsymbol{A}^{[y]}(j^{[y]},k^{[y]})=\mathrm{e}^{-2\pi\mathrm{i}k^{[y]}j^{[y]}/n^{[y]}}$ are the DFT matrices in the x and y directions, respectively. In this case, the forward NUDFT can be efficiently computed by the *fast Fourier transform* (FFT) algorithm [11] in $\mathscr{O}\big(N\log N\big)$ time. And the inverse NUDFT can also be efficiently solved by the *inverse fast Fourier transform* (iFFT) algorithm in $\mathscr{O}\big(N\log N\big)$ time. More generally, if $(x_{j^{[x]}},y_{j^{[y]}})=(j^{[x]}/m^{[x]},j^{[y]}/m^{[y]})$ where $m^{[x]}\ge n^{[x]}$ and $m^{[y]}\ge n^{[y]}$, the matrix $\boldsymbol{A}$ is a submatrix of the DFT matrix, and both the forward and inverse NUDFT can still be efficiently computed by the FFT algorithm in $\mathscr{O}\big(M\log M\big)$ time.

Unfortunately, when the sample points are not distributed on a uniformly Cartesian grid, FFT cannot be directly applied. Many algorithms are developed to compute the forward NUDFT efficiently, which are usually referred to as the *nonuniform fast Fourier transform* (NUFFT) algorithms [1, 3, 13, 14, 16, 22, 28, 29]. Commonly, the NUFFT algorithms have a complexity of $\mathscr{O}\big(M+N\log N\big)$.

In the nonuniform case, the pseudoinverse of the NUDFT matrix is no longer a scaled version of its adjoint. Consequently, specialized algorithms are required to solve the associated inverse problem efficiently. A widely adopted approach is the iterative method. Thanks to the efficient NUFFT algorithms for the forward computation, the inverse NUDFT is often solved by the *conjugate gradient* (CG) method. For example, using the MATLAB command "lsqr" with the NUFFT algorithm as the matrix-vector multiplication subroutine [27]. When the sample points are well-distributed, iterative methods usually converge in a small number of iterations. However, when the sample points are distributed in a highly nonuniform manner, which is often the case in practical applications, CG may require a large number of iterations to converge or even fail to converge. Preconditioners are needed to accelerate the convergence of iterative methods in this case.

Direct inversion methods are also developed. For example, Kircheis and Potts [20] proposed a direct inversion method for the type-II NUDFT in 1D case and they extended it to the 2D case [21]. Recently, in [30], the authors proposed a direct method to solve the type-II NUDFT in 1D based on the *hierarchically semiseparable* (HSS) matrix, which has a complexity of $\mathscr{O}\big((M+N)\log^2 N\big)$. Further, the authors in [23] extended this method to the case of type-III NUDFT in 1D, by factoring the type-III NUDFT matrix into a product of a type-II NUDFT matrix and an HSS matrix. For the 2D problem, if the sample points are distributed on a tensor grid, the NUDFT matrix has a Kronecker product structure, and one can apply the 1D solver in each direction to solve the problem. However, the extension of the method [30] to the general 2D case is nontrivial.

[1] In this paper, the starting index of a matrix can be 0 or 1, depending on the context.

1.2 Contributions

In this paper, we propose a direct inversion method for the 2D type-II NUDFT. Let $\boldsymbol{F} \in \mathbb{C}^{N\times N}$ be the 2D DFT matrix. We first derive an explicit formula for the entries of the matrix $\boldsymbol{G} = \boldsymbol{A}\boldsymbol{F}^{-1}$, showing that it can be expressed as a kernel matrix. Based on this expression, we analyze the low-rank structure of $\boldsymbol{G}$ and show that it can be efficiently compressed into an HSS matrix.

The matrix $\boldsymbol{G}$ can also be expressed as a face-splitting product of two matrices $\boldsymbol{G}^{[x]}$ and $\boldsymbol{G}^{[y]}$, where $\boldsymbol{G}^{[x]}$ and $\boldsymbol{G}^{[y]}$ are the transformed NUDFT matrices in the x- and y- directions, respectively. Since $\boldsymbol{G}^{[x]}$ and $\boldsymbol{G}^{[y]}$ can be approximated by HSS matrices, this representation suggests that $\boldsymbol{G}$ can also be approximated by an HSS matrix. To make this argument precise, we define the face-splitting product of two HSS trees and prove that this product is again an HSS tree. We further prove that the face-splitting product of two HSS matrices remains an HSS matrix, and provide an estimate for the resulting HSS rank. Finally, we establish an approximation error bound for the HSS approximation of $\boldsymbol{G}$.

Using the kernel matrix expression of $\boldsymbol{G}$, the HSS construction method based on the interpolative decomposition [10] and the proxy surface technique [25] can be applied to approximate $\boldsymbol{G}$ by an HSS matrix $\widetilde{\boldsymbol{G}}$ efficiently. After the construction, the URV factorization [31] is used as a direct solver for the least-squares problem associated with $\widetilde{\boldsymbol{G}}$. The construction and factorization of the HSS matrix constitute the *offline* stage, whose complexity is $\mathcal{O}\big(M + N^{3/2}\log^3 N\big)$. Once the solver is built, the inverse NUDFT problem can be solved by applying the HSS solver followed by the iFFT. This is referred to as the *online* stage, and its complexity is $\mathcal{O}\big(M + N\log^3 N\big)$.

The proposed method can also be used as a preconditioner for iterative methods, which is particularly useful when the sample points are distributed in a highly nonuniform manner. Numerical experiments agree with our theoretical analysis and demonstrate the efficiency of the proposed method.

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we introduce some notations and preliminaries that will be used in the later paper. We derive the low-rank structure of the NUDFT matrix in Section 3 and propose our direct solver in Section 4. In Section 5, we present numerical results to demonstrate the efficiency of the proposed method. Finally, we conclude the paper in Section 6 and discuss possible future directions.

2 Preliminaries

2.1 Notations and Preliminaries

We use $\mathbb{S} \coloneqq \{z \in \mathbb{C} : |z| = 1\}$ to denote the unit circle. For a set $\mathcal{J}$, we use $|\mathcal{J}|$ to denote the cardinality of $\mathcal{J}$. We use MATLAB notation to denote the submatrix of $\boldsymbol{A}$ with row and column indices in $\mathcal{R}$ and $\mathcal{C}$, i.e., $\boldsymbol{A}(\mathcal{R},\mathcal{C})$. A colon in the index means all the indices in that dimension, e.g., $\boldsymbol{A}(:,\mathcal{C})$ means the submatrix of $\boldsymbol{A}$ with all rows and column indices in $\mathcal{C}$.

Let $\|\cdot\|$ be a unitarily invariant matrix norm. For a given tolerance $0 < \varepsilon < 1$, the numerical rank of a matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$, denoted by $\operatorname{rank}_\varepsilon(\boldsymbol{A})$, is defined as the smallest integer r such that there exists a matrix $\widetilde{\boldsymbol{A}} \in \mathbb{C}^{m\times n}$ with $\|\boldsymbol{A} - \widetilde{\boldsymbol{A}}\| < \varepsilon\|\boldsymbol{A}\|$ and $\operatorname{rank}(\widetilde{\boldsymbol{A}}) = r$.

For two matrices $\boldsymbol{A}_1 \in \mathbb{C}^{m\times n_1}$ and $\boldsymbol{A}_2 \in \mathbb{C}^{m\times n_2}$ of the same size of rows, the *face-splitting product* of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ is defined as the Kronecker product of the corresponding rows of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, i.e.,

$$\boldsymbol{A}_1 \bullet \boldsymbol{A}_2 \coloneqq \begin{bmatrix} \boldsymbol{A}_1(1,:) \otimes \boldsymbol{A}_2(1,:) \\ \boldsymbol{A}_1(2,:) \otimes \boldsymbol{A}_2(2,:) \\ \vdots \\ \boldsymbol{A}_1(m,:) \otimes \boldsymbol{A}_2(m,:) \end{bmatrix} \in \mathbb{C}^{m\times(n_1 n_2)}.$$

The Frobenius norm of the face-splitting product can be estimated by the following lemma, which can be proved through direct calculation.

Lemma 2.1. *Let* $\boldsymbol{A}_1 \in \mathbb{C}^{m\times n_1}$ *and* $\boldsymbol{A}_2 \in \mathbb{C}^{m\times n_2}$*, then*

$$\|\boldsymbol{A}_1 \bullet \boldsymbol{A}_2\|_{\mathrm{F}} \leq \min\Big\{ \max_{1\leq j\leq m} \big\{\|\boldsymbol{A}_1(j,:)\|_2\big\}\|\boldsymbol{A}_2\|_{\mathrm{F}}, \max_{1\leq j\leq m} \big\{\|\boldsymbol{A}_2(j,:)\|_2\big\}\|\boldsymbol{A}_1\|_{\mathrm{F}}\Big\}.$$

2.2 HSS Matrices

HSS matrices are a class of hierarchical matrices [6, 18, 32]. Since the storage and algebraic operations for HSS matrices typically have linear complexity, they have been widely used in scientific computing [12, 26]. We first introduce the definition of HSS tree, which is the underlying data structure for HSS matrices.

Definition 2.2 (HSS tree, [31])**.** *A tree* T *is called an* HSS tree *with row and column indices* $\mathcal{J}$ *and* $\mathcal{K}$ *if each node* τ *of* T *is associated with two index sets* $\mathcal{J}_\tau$ *and* $\mathcal{K}_\tau$*, satisfying the following conditions*

(1) $\mathcal{J}_\rho = \mathcal{J}$, $\mathcal{K}_\rho = \mathcal{K}$ *for the root node* ρ*.*

(2) For a nonleaf node τ*, we have* $\mathcal{J}_\tau = \sqcup_{\alpha\in\mathrm{ch}(\tau)}\mathcal{J}_\alpha$ *and* $\mathcal{K}_\tau = \sqcup_{\alpha\in\mathrm{ch}(\tau)}\mathcal{K}_\alpha$*, where the notation* $\sqcup$ *means the disjoint union.*

For a node τ*, we use* $\mathrm{ch}(\tau)$ *and* $\mathrm{sib}(\tau)$ *to denote the set of children of* τ *and the set of siblings of* τ*, respectively. For an HSS tree* T*, we define the level of each node as follows: For the root node* ρ*,* $\mathrm{level}(\rho) = 0$*. If* τ *is a nonleaf node and* $\alpha \in \mathrm{ch}(\tau)$*, then* $\mathrm{level}(\alpha) = \mathrm{level}(\tau) + 1$*. The maximum level of the tree is denoted by* L*.*

Definition 2.3 (HSS matrix, [31])**.** *Let* T *be an HSS tree of row and column indices* $\mathcal{J}$ *and* $\mathcal{K}$*. A matrix* $\boldsymbol{H} \in \mathbb{C}^{|\mathcal{J}|\times|\mathcal{K}|}$ *is called an* HSS matrix *about* T *if there are matrices* $\boldsymbol{D}_\tau$*,* $\boldsymbol{U}_\tau$*,* $\boldsymbol{V}_\tau$*,* $\boldsymbol{R}_\tau$*,* $\boldsymbol{W}_\tau$ *and* $\boldsymbol{B}_{\tau,\sigma}$ *(called HSS generators) associated with each node* τ*, which satisfy the following conditions:*

(1) For a leaf node τ*,* $\boldsymbol{D}_\tau = \boldsymbol{H}(\mathcal{J}_\tau, \mathcal{K}_\tau)$ *is a dense matrix.*

(2) For a nonleaf node τ *with children* $\{\alpha_p\}_{p=1}^b$*,* $\boldsymbol{D}_\tau = \boldsymbol{H}(\mathcal{J}_\tau, \mathcal{K}_\tau)$ *is a* $b\times b$ *block matrix, with the* (p,q)*-th block* $\boldsymbol{D}_{\tau;p,q}$ *being* $\boldsymbol{D}_{\alpha_p}$ *for* $p = q$ *and* $\boldsymbol{U}_{\alpha_p}\boldsymbol{B}_{\alpha_p,\alpha_q}\boldsymbol{V}_{\alpha_q}^*$ *for* $p \neq q$*. The matrices* $\boldsymbol{U}_\tau$ *and* $\boldsymbol{V}_\tau$ *is a* $b\times 1$ *block matrix, with the* p*-th block being* $\boldsymbol{U}_{\alpha_p}\boldsymbol{R}_{\alpha_p}$ *and* $\boldsymbol{V}_{\alpha_p}\boldsymbol{W}_{\alpha_p}$*, respectively.*

We call $\boldsymbol{U}_\tau$ *and* $\boldsymbol{V}_\tau^*$ *the* basis matrices*,* $\boldsymbol{R}_\tau$ *and* $\boldsymbol{W}_\tau$ *the* transfer matrices*, and* $\boldsymbol{B}_{\tau,\sigma}$ *the* interaction matrices*.*

One important property of HSS matrices, called the *shared basis property*, is that the basis matrices $\boldsymbol{U}_\tau$ and $\boldsymbol{V}_\tau^*$ are bases for the column and row spaces of the corresponding off-diagonal blocks (called the *HSS blocks*) $\boldsymbol{H}(\mathcal{J}_\tau, \mathcal{K}_\tau^{\mathrm{c}})$ and $\boldsymbol{H}(\mathcal{J}_\tau^{\mathrm{c}}, \mathcal{K}_\tau)$ of $\boldsymbol{H}$, respectively. Here $\mathcal{J}_\tau^{\mathrm{c}} = \mathcal{J}\setminus\mathcal{J}_\tau$ and $\mathcal{K}_\tau^{\mathrm{c}} = \mathcal{K}\setminus\mathcal{K}_\tau$. the *HSS rank* r of $\boldsymbol{H}$ is defined as the maximum rank among all the HSS blocks.

Throughout this paper, we assume that the HSS matrix is associated with a given HSS tree T with maximum level L. For simplicity, we also assume that T is a full tree, i.e., all the leaf nodes are on the same level L, and that the HSS row and column blocks of a node τ have a same rank r_τ.

2.3 The URV Factorization for HSS Matrices

Consider the least-squares problem $\min_{\boldsymbol{c}} \|\boldsymbol{H}\boldsymbol{c} - \boldsymbol{f}\|_2$, where $\boldsymbol{H}$ is an HSS matrix and $\boldsymbol{f}$ is a given vector. In this section, we review the URV factorization for HSS matrices [30, 31], which serves as a direct solver for the least-squares problem.

The URV factorization for HSS matrices is performed in a bottom-up manner on the HSS tree. Suppose τ is a leaf node and $\boldsymbol{D}_\tau \in \mathrm{c}^{m_\tau\times n_\tau}$. If $m_\tau \gg n_\tau$, a *size reduction* step is performed by QR factorization:

$$\begin{bmatrix}\boldsymbol{U}_\tau & \boldsymbol{D}_\tau\end{bmatrix} = \boldsymbol{\Omega}_\tau \left[\begin{array}{cc} \boldsymbol{R}_{1,1} & \boldsymbol{R}_{1,2} \\ \boldsymbol{0} & \boldsymbol{R}_{2,2} \\ \hline \boldsymbol{0} & \boldsymbol{0}\end{array}\right] = \boldsymbol{\Omega}_\tau \begin{bmatrix}\breve{\boldsymbol{U}}_\tau & \breve{\boldsymbol{D}}_\tau \\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}, \tag{2.1}$$

where $\boldsymbol{\Omega}_\tau \in \mathbb{C}^{m_\tau \times m_\tau}$ is unitary. The reduced row size, i.e., the number of rows of $\breve{\boldsymbol{U}}_\tau$ and $\breve{\boldsymbol{D}}_\tau$, is $\breve{m}_\tau = n_\tau + r_\tau$. If no size reduction is adapted, we set $\boldsymbol{\Omega}_\tau$ to be the empty and let $\breve{\boldsymbol{U}}_\tau = \boldsymbol{U}_\tau$, $\breve{\boldsymbol{D}}_\tau = \boldsymbol{D}_\tau$ and $\breve{m}_\tau = n_\tau + r_\tau$.

The next is a *basis elimination* step. For every leaf node τ, zeros are introduced into $\boldsymbol{V}_\tau$ by

$$\boldsymbol{V}_\tau = \boldsymbol{Q}_\tau \begin{bmatrix} \boldsymbol{0} \\ \widehat{\boldsymbol{V}}_{\tau;2} \end{bmatrix}, \tag{2.2}$$

where $\boldsymbol{Q}_\tau \in \mathbb{C}^{n_\tau \times n_\tau}$ is unitary and $\widehat{\boldsymbol{V}}_{\tau;2} \in \mathbb{C}^{r_\tau \times r_\tau}$. This can be done by a reverse QR decomposition. The diagonal block is then modifed by $\widetilde{\boldsymbol{D}}_\tau = \widehat{\boldsymbol{D}}_\tau \boldsymbol{Q}_\tau$

Then a *partial decomposition* on the matrix $\widetilde{\boldsymbol{D}}_\tau$ is performed to decouple the unknowns corresponding to the zero and dense parts of (2.2). First, a QR decomposition is computed on the first block column of $\widetilde{\boldsymbol{D}}_\tau$ as

$$\begin{bmatrix} \widetilde{\boldsymbol{D}}_{\tau;1,1} \\ \widetilde{\boldsymbol{D}}_{\tau;2,1} \end{bmatrix} = \boldsymbol{P}_\tau \begin{bmatrix} \widehat{\boldsymbol{D}}_{\tau;1,1} \\ \boldsymbol{0} \end{bmatrix}. \tag{2.3}$$

Then it follows that

$$\widetilde{\boldsymbol{D}}_\tau = \boldsymbol{P}_\tau \begin{bmatrix} \widehat{\boldsymbol{D}}_{\tau;1,1} & \widehat{\boldsymbol{D}}_{\tau;1,2} \\ \boldsymbol{0} & \widehat{\boldsymbol{D}}_{\tau;2,2} \end{bmatrix}, \tag{2.4}$$

where

$$\begin{bmatrix} \widehat{\boldsymbol{D}}_{\tau;1,2} \\ \widehat{\boldsymbol{D}}_{\tau;2,2} \end{bmatrix} = \boldsymbol{P}_\tau^* \begin{bmatrix} \widetilde{\boldsymbol{D}}_{\tau;1,2} \\ \widetilde{\boldsymbol{D}}_{\tau;2,2} \end{bmatrix}. \tag{2.5}$$

The basis matrix $\boldsymbol{U}_\tau$ is then modified by

$$\boldsymbol{P}_\tau^* \boldsymbol{U}_\tau = \begin{bmatrix} \widehat{\boldsymbol{U}}_{\tau;1} \\ \widehat{\boldsymbol{U}}_{\tau;2} \end{bmatrix}. \tag{2.6}$$

After that, the variables corresponding to the first block column of $\widetilde{\boldsymbol{D}}_\tau$ can be decoupled from the rest variables, and the least-squares problem can be solved by a recursive process, which we describe in the following. For a nonleaf node τ with children $\{\alpha_p\}$, appropriate blocks of the children are merged to form the new generators. Let $\boldsymbol{D}_\tau^+$, $\boldsymbol{U}_\tau^+$ and $\boldsymbol{V}_\tau^+$ be the block matrices given by

$$\begin{aligned} \boldsymbol{D}_{\tau;p,q}^+ &= \begin{cases} \widehat{\boldsymbol{D}}_{\alpha_p;2,2}, & p = q, \\ \widehat{\boldsymbol{U}}_{\alpha_p;2} \boldsymbol{B}_{\alpha_p,\alpha_q} \widehat{\boldsymbol{V}}_{\alpha_q;2}^*, & p \neq q \end{cases} \\ \boldsymbol{U}_{\tau;p}^+ &= \widehat{\boldsymbol{U}}_{\alpha_p;2} \boldsymbol{R}_{\alpha_p}, \quad \boldsymbol{V}_{\tau;p}^+ = \widehat{\boldsymbol{V}}_{\alpha_p;2} \boldsymbol{W}_{\alpha_p}. \end{aligned} \tag{2.7}$$

Then the children of τ can be "removed" and the node τ can be treated as a leaf node. The generators $\{\boldsymbol{D}_\tau^+\}$, $\{\boldsymbol{U}_\tau^+\}$, $\{\boldsymbol{V}_\tau^+\}$ $\{\boldsymbol{R}_\tau\}$, $\{\boldsymbol{W}_\tau\}$ and $\{\boldsymbol{B}_{\tau,\sigma}\}$ define a new HSS matrix of maximum level $L-1$. This HSS can be further factorized by the same discussion above (with $\boldsymbol{D}_\tau$, $\boldsymbol{U}_\tau$ and $\boldsymbol{V}_\tau$ replaced by $\boldsymbol{D}_\tau^+$, $\boldsymbol{U}_\tau^+$ and $\boldsymbol{V}_\tau^+$) until the root node is reached. At the root node ρ, the QR factorization is performed, i.e., $\boldsymbol{D}_\rho^+ = \boldsymbol{P}_\rho \widehat{\boldsymbol{D}}_\rho^+$.

Once the URV factors are computed, the solution is obtained by a bottom-up pass and a top-down pass. The bottom-up pass is to apply the unitary transformations $\boldsymbol{\Omega}_\tau$ and $\boldsymbol{P}_\tau$ to the right-hand side $\boldsymbol{f}$. Starting from every leaf node τ, if size reduction is performed, the vector $\boldsymbol{f}_\tau$ is updated by

$$\boldsymbol{\Omega}_\tau^* \boldsymbol{f}_\tau = \begin{bmatrix} \breve{\boldsymbol{f}}_\tau \\ \times \end{bmatrix}, \quad \widehat{\boldsymbol{f}}_\tau = \boldsymbol{P}_\tau^* \breve{\boldsymbol{f}}_\tau = \begin{bmatrix} \widehat{\boldsymbol{f}}_{\tau;1} \\ \widehat{\boldsymbol{f}}_{\tau;2} \end{bmatrix},$$

where the $\times$ means the corresponding entries are not used in the following steps and can be ignored. If no size reduction is performed, the first equation is omitted and $\breve{\boldsymbol{f}}_\tau$ is replaced by $\boldsymbol{f}_\tau$ in the second equation. Then for a nonleaf node τ with children $\{\alpha_p\}$, $\boldsymbol{f}_\tau^+$ is obtained by merging its children blocks, i.e.,

$\boldsymbol{f}^+_{\tau;p} = \widehat{\boldsymbol{f}}_{\alpha_p;2}$. This process is repeated (with $\boldsymbol{f}_\tau$ replaced by $\boldsymbol{f}^+_\tau$ in the above equations) until the root node is reached.

The top-down pass is to perform back substitution to obtain the solution $\boldsymbol{c}$. At the root node ρ, after solving $\widehat{\boldsymbol{D}}^+_\rho \boldsymbol{c}^+_\rho = \widehat{\boldsymbol{f}}_\rho$, for each child $\alpha \in \mathrm{ch}(\rho)$, the vectors $\widehat{\boldsymbol{c}}_{\alpha;2}$ is assigned as the corresponding block in $\widehat{\boldsymbol{c}}^+_\rho$. Further, the vectors $\widehat{\boldsymbol{h}}_\alpha$, $\widehat{\boldsymbol{g}}_\alpha$ and $\widehat{\boldsymbol{b}}_\alpha$ are intialized by $\boldsymbol{0}$, $\sum_{\beta\in\mathrm{sib}(\alpha)} \boldsymbol{B}_{\alpha,\beta}\widehat{\boldsymbol{V}}_{\beta;2}\widehat{\boldsymbol{c}}_{\beta;2}$ and $\widehat{\boldsymbol{h}}_\alpha + \widehat{\boldsymbol{g}}_\alpha$, respectively. For a nonroot node τ, the vector $\boldsymbol{c}^+_\tau$ is updated by $\boldsymbol{c}^+_\tau = \boldsymbol{Q}_\tau \begin{bmatrix} \widehat{\boldsymbol{c}}_{\tau;1} \\ \widehat{\boldsymbol{c}}_{\tau;2} \end{bmatrix}$, where $\widehat{\boldsymbol{c}}_{\tau;1} = \widehat{\boldsymbol{D}}^{-1}_{\tau;1,1}(\widehat{\boldsymbol{f}}_{\tau;1} - \widehat{\boldsymbol{D}}_{\tau;1,2}\widehat{\boldsymbol{c}}_{\tau;2} - \widehat{\boldsymbol{U}}_{\tau;1}\widehat{\boldsymbol{b}}_\tau)$. If τ is nonleaf, then for every child $\alpha \in \mathrm{ch}(\tau)$, $\widehat{\boldsymbol{c}}_{\alpha;2}$ is assigned as the corresponding block in $\boldsymbol{c}^+_\tau$ and $\widehat{\boldsymbol{h}}_\alpha = \boldsymbol{R}_\alpha \widehat{\boldsymbol{h}}_\tau$. The vectors $\widehat{\boldsymbol{g}}_\alpha$ and $\widehat{\boldsymbol{b}}_\alpha$ are given by $\widehat{\boldsymbol{g}}_\alpha = \sum_{\beta\in\mathrm{sib}(\alpha)} \boldsymbol{B}_{\alpha,\beta}\widehat{\boldsymbol{V}}_{\beta;2}\widehat{\boldsymbol{c}}_\beta$ and $\widehat{\boldsymbol{b}}_\alpha = \widehat{\boldsymbol{h}}_\tau + \widehat{\boldsymbol{g}}_\tau$. This process is repeated until the leaf nodes are reached, where $\boldsymbol{c}_\tau = \boldsymbol{c}^+_\tau$ for each leaf node τ.

The complexity of these two algorithms can be analyzed similarly as that in [31]. Assume that the HSS tree T is a full quad tree with maximum level L, and that the number of rows and columns of the HSS matrix $\boldsymbol{H}$ are $|\mathcal{J}| = M = 4^L M_L$ and $|\mathcal{K}| = N = 4^L N_L$ according to the HSS tree, where M_L and N_L are the row and column sizes of the leaf nodes, respectively. We also assume that $N = n^2$ where $n = 2^L n_L$. For each level ℓ, there are 4^ℓ nodes, and assume that the rank of the HSS blocks of a node on level ℓ is bounded by $r_\ell = \mathscr{O}(n_\ell \log n_\ell)$ where $n_\ell = 2^{L-\ell} n_L$. Then the complexity of the URV factorization and solution algorithms are $\mathscr{O}\big(M + N^{3/2}\log^3 N\big)$ and $\mathscr{O}\big(M + N\log^3 N\big)$, respectively. We point out here that the URV factorization can also be used compute $\boldsymbol{H}^{*,\dagger}\boldsymbol{f}$ for a given vector $\boldsymbol{f}$. The solution process is similar and we omit the details here for brevity.

2.4 A Review of the Direct Inverse Method in 1D Case

Consider the 1D type-II NUDFT matrix given by $\boldsymbol{A}(j,k) = \mathrm{e}^{-2\pi \mathrm{i} k x_j}$ where $x_j \in [0,1)$ for $j = 0, 1, \ldots, M-1$ and $k = 0, 1, \ldots, N-1$. If we define $\gamma_j = \mathrm{e}^{-2\pi \mathrm{i} x_j} \in \mathbb{S}$, then $\boldsymbol{A}(j,k) = \gamma_j^k$ and it can be directly verified that $\boldsymbol{A}$ satisfies the following Sylvester equation

$$\boldsymbol{\Gamma}\boldsymbol{A} - \boldsymbol{A}\boldsymbol{C} = \boldsymbol{u}\boldsymbol{e}^\top_{N-1}, \tag{2.8}$$

where $\boldsymbol{\Gamma} = \mathrm{diag}(\gamma_0, \gamma_1, \ldots, \gamma_{M-1})$ is a diagonal matrix, and

$$\boldsymbol{C} = \begin{bmatrix} & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \\ 1 & & & & \end{bmatrix}$$

is a circulant shift matrix and $u_j = \gamma_j^N - 1$ for $j = 0, 1, \ldots, M-1$. The matrix $\boldsymbol{D}$ can be diagonalized by the DFT matrix, i.e., $\boldsymbol{D} = \boldsymbol{F}^{-1}\boldsymbol{D}\boldsymbol{F}$ where $\boldsymbol{F}$ is the DFT matrix given by $\boldsymbol{F}(k,\ell) = \mathrm{e}^{-2\pi \mathrm{i} k\ell/N}$ and $\boldsymbol{D} = \mathrm{diag}(\xi_0, \xi_1, \ldots, \xi_{N-1})$ is a diagonal matrix. Here $\xi_k = \zeta_N^k \in \mathbb{S}$ and $\zeta_N = \mathrm{e}^{-2\pi \mathrm{i}/N}$ is the Nth root of unity. Substituting the diagonalization of $\boldsymbol{C}$ into (2.8) and multiplying both sides by $\boldsymbol{F}^{-1}$ on the right, it follows that

$$\boldsymbol{\Gamma}\boldsymbol{G} - \boldsymbol{G}\boldsymbol{D} = \boldsymbol{u}\boldsymbol{v}^\top, \tag{2.9}$$

where $\boldsymbol{G} = \boldsymbol{A}\boldsymbol{F}^{-1}$ and $\boldsymbol{v} = \boldsymbol{F}^{-1}\boldsymbol{e}_{N-1}$. The entries of $\boldsymbol{v}$ are given by $v_k = \xi_k/N$ for $k = 0, 1, \ldots, N-1$. Therefore, $\boldsymbol{G}$ can be expressed as a Cauchy-like matrix with the entries given by

$$\boldsymbol{G}(j,k) = \frac{u_j v_k}{\gamma_j - \xi_k} = \frac{1}{N}\frac{(\gamma_j/\xi_k)^N - 1}{(\gamma_j/\xi_k) - 1} =: \Psi(\gamma_j, \xi_k). \tag{2.10}$$

In [30], the authors utilize the *displacement structure* [4] of $\boldsymbol{G}$ in (2.9) to derive the *low-rank property* of $\boldsymbol{G}$. They showed that $\boldsymbol{G}$ can be approximated by an HSS matrix $\widetilde{\boldsymbol{G}}$ with numerical HSS rank $r =$

$\mathscr{O}\big(\log(1/\varepsilon)\log N\big)$, where ε is the accuracy parameter for the approximation. Specifically, we have the following Theorem.

Theorem 2.4 (Numerical rank of HSS blocks of $\boldsymbol{G}$, Theorem 3.2 in [30])**.** *Let $\boldsymbol{G}$ be the matrix defined by (2.10) and $0 < \varepsilon < 1$ and $\mathcal{J} = \{0, 1, \ldots, M-1\}$ and $\mathcal{K} = \{0, 1, \ldots, N-1\}$ be the row and column index sets of $\boldsymbol{G}$ respectively. Suppose that $[a, b) \subset [0, 1)$ is an interval and $\mathcal{R} = \{j : x_j \in [a, b)\} \subset \mathcal{J}$ and $\mathcal{C} = \{k : k/N \in [a, b)\} \subset \mathcal{K}$ are the row and column index sets corresponding to this interval. Then for the 2-norm or the Frobenius norm, the HSS row or column block of $\boldsymbol{G}$ with respect to $\mathcal{R}$ or $\mathcal{C}$ has a numerical rank at most $r = \mathscr{O}\big(\log(1/\varepsilon)\log n_{\mathrm{c}}\big)$ where $n_{\mathrm{c}} = |\mathcal{C}|$.*

There are two main differences between Theorem 2.4 and the original result in [30]. First, the original result in [30] is for the 2-norm while we add the Frobenius norm in Theorem 2.4. This is because the singular values of the corresponding submatrix decay exponentially. The second is that the rank bound here is more refined by carefully tracking the inequality in the proof of in [30, Theorem 3.2]. From the Sylvester equation (2.9), the HSS approximation $\widetilde{\boldsymbol{G}}$ can be constructed by combining the *factored alternating direction implicit* (fADI) method [5] and interpolative decomposition [10], at a complexity of $\mathscr{O}\big((M+N)\log^2 N\big)$.

Once the HSS approximation $\widetilde{\boldsymbol{G}}$ is constructed, the URV factorization discussion in Section 2.3 is performed, these are the offline steps of the direct inverse method. In the online step, given a vector $\boldsymbol{f} \in \mathbb{C}^M$, the solution of the least-squares problem associated with $\boldsymbol{A}$ can be obtained by $\boldsymbol{c} = \boldsymbol{F}^{-1}\widetilde{\boldsymbol{G}}^{\dagger}\boldsymbol{f}$, which involves one URV solution of $\widetilde{\boldsymbol{G}}$ andone iFFT. Therefore, the online complexity is $\mathscr{O}\big((M+N)\log N\big)$.

3 The Low-Rank Property of the NUDFT Matrix

This section discusses the low-rank property of the 2D type-II NUDFT matrix. Section 3.1 gives a kernel matrix expression of the transformed 2D type-II NUDFT matrix and derives the low-rank property of the matrix. In Section 3.2, we define the face-splitting product of two HSS matrices and show that the face-splitting product of two HSS matrices is still an HSS matrix.

3.1 Kernel Matrix Perspective

Consider the 2D type-II NUDFT matrix $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ given by (1.2). Let $\boldsymbol{A}^{[x]} \in \mathbb{C}^{M\times n^{[x]}}$ and $\boldsymbol{A}^{[y]} \in \mathbb{C}^{M\times n^{[y]}}$ be the type-II NUDFT in x and y direction respectively, i.e., $\boldsymbol{A}^{[x]}(j, k^{[x]}) = \mathrm{e}^{-2\pi \mathrm{i} k^{[x]} x_j}$ and $\boldsymbol{A}^{[y]}(j, k^{[y]}) = \mathrm{e}^{-2\pi \mathrm{i} k^{[y]} y_j}$. Then $\boldsymbol{A}$ can be written as the face-splitting product $\boldsymbol{A} = \boldsymbol{A}^{[x]} \bullet \boldsymbol{A}^{[y]}$.

Let $\boldsymbol{F}^{[x]} \in \mathbb{C}^{n^{[x]}\times n^{[x]}}$ and $\boldsymbol{F}^{[y]} \in \mathbb{C}^{n^{[y]}\times n^{[y]}}$ be the DFT matrices in x and y direction respectively and $\boldsymbol{F} = \boldsymbol{F}^{[x]} \otimes \boldsymbol{F}^{[y]} \in \mathbb{C}^{N\times N}$ be the DFT matrix in 2D. Following the discussions in Section 2.4, if we define $\boldsymbol{G}^{[x]} = \boldsymbol{A}^{[x]}\boldsymbol{F}^{[x],-1}$, $\boldsymbol{G}^{[y]} = \boldsymbol{A}^{[y]}\boldsymbol{F}^{[y],-1}$ and $\boldsymbol{G} = \boldsymbol{A}\boldsymbol{F}^{-1}$, then it can also be verified that $\boldsymbol{G} = \boldsymbol{G}^{[x]} \bullet \boldsymbol{G}^{[y]}$. Using (2.10), the entries of $\boldsymbol{G}$ can be written as

$$\begin{aligned}
\boldsymbol{G}\big(j, (k^{[x]}, k^{[y]})\big) &= \frac{1}{n^{[x]}} \frac{\big(\gamma_j^{[x]}/\xi_{k^{[x]}}^{[x]}\big)^{n^{[x]}} - 1}{\big(\gamma_j^{[x]}/\xi_{k^{[x]}}^{[x]}\big) - 1} \frac{1}{n^{[y]}} \frac{\big(\gamma_j^{[y]}/\xi_{k^{[y]}}^{[y]}\big)^{n^{[y]}} - 1}{\big(\gamma_j^{[y]}/\xi_{k^{[y]}}^{[y]}\big) - 1} \\
&= \Psi^{[x]}\big(\gamma_j^{[x]}, \xi_{k^{[x]}}^{[y]}\big)\Psi^{[y]}\big(\gamma_j^{[y]}, \xi_{k^{[y]}}^{[y]}\big) \\
&=: \Psi\big(\boldsymbol{\gamma}_j, \boldsymbol{\xi}_{(k^{[x]}, k^{[y]})}\big),
\end{aligned} \tag{3.1}$$

where $\gamma_j^{[x]} = \mathrm{e}^{-2\pi \mathrm{i} x_j}$, $\gamma_j^{[y]} = \mathrm{e}^{-2\pi \mathrm{i} y_j}$, $\xi_{k^{[x]}}^{[x]} = \mathrm{e}^{-2\pi \mathrm{i} k^{[x]}/n^{[x]}}$ and $\xi_{k^{[y]}}^{[y]} = \mathrm{e}^{-2\pi \mathrm{i} k^{[y]}/n^{[y]}}$ belong to the unit circle $\mathbb{S}$, $\Psi^{[x]}$ and $\Psi^{[y]}$ are corresponding kernel functions in x and y direction respectively, and $\boldsymbol{\gamma}_j = (\gamma_j^{[x]}, \gamma_j^{[y]})$ and $\boldsymbol{\xi}_{(k^{[x]}, k^{[y]})} = (\xi_{k^{[x]}}^{[x]}, \xi_{k^{[y]}}^{[y]})$ are points on $\mathbb{S}^2$. Consequently, $\boldsymbol{G}$ can be viewed as a kernel matrix with the kernel function

$$\Psi(\boldsymbol{z}, \boldsymbol{w}) = \Psi^{[x]}(z_1, w_1)\Psi^{[y]}(z_2, w_2), \quad \boldsymbol{z} = (z_1, z_2) \in \mathbb{S}^2, \quad \boldsymbol{w} = (w_1, w_2) \in \mathbb{S}^2. \tag{3.2}$$

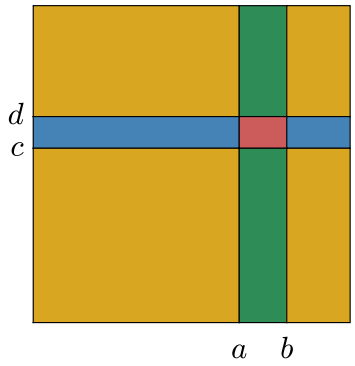

Figure 3.1: The partition of $[0,1)^2$ according to a rectangular domain. The red box represents the rectangular domain $[a,b)\times[c,d)$. The blue boxes represent the points whose x-coordinate not in $[a,b)$ but y-coordinate in $[c,d)$. The green boxes represent the points whose x-coordinate in $[a,b)$ but y-coordinate not in $[c,d)$. The yellow boxes represent the points whose x-coordinate not in $[a,b)$ and y-coordinate not in $[c,d)$.

The equation (3.1) gives a kernel matrix representation of $\boldsymbol{G}$, implying that it can be approximated by an HSS matrix. In the remaining part of this section, we give some intuition and discussion on the HSS rank. We first give a preliminary lemma, which shows that the face-splitting product of two low-rank matrices is still low-rank.

Lemma 3.1 (Rank of the matrix face-splitting product)**.** *Let $\boldsymbol{A}_1\in\mathbb{C}^{m\times n_1}$ and $\boldsymbol{A}_2\in\mathbb{C}^{m\times n_2}$ be two matrices and $\boldsymbol{A}=\boldsymbol{A}_1\bullet\boldsymbol{A}_2$, then* $\operatorname{rank}(\boldsymbol{A})\le\operatorname{rank}(\boldsymbol{A}_1)\operatorname{rank}(\boldsymbol{A}_2)$.

Proof. Suppose $\operatorname{rank}(\boldsymbol{A}_1)=r_1$ and $\operatorname{rank}(\boldsymbol{A}_2)=r_2$. We factorize them as $\boldsymbol{A}_t=\boldsymbol{U}_t\boldsymbol{S}_t\boldsymbol{V}_t^*$ for $t=1,2$, where $\boldsymbol{U}_t\in\mathbb{C}^{m\times r_t}$, $\boldsymbol{S}_t\in\mathbb{C}^{r_t\times r_t}$ and $\boldsymbol{V}_t\in\mathbb{C}^{n_t\times r_t}$. Then for $1\le j\le m$, $1\le k_1\le n_1$ and $1\le k_2\le n_2$, the $\big(j,(k_1,k_2)\big)$ entry of $\boldsymbol{A}$ is given by

$$\begin{aligned}\boldsymbol{A}\big(j,(k_1,k_2)\big)&=\boldsymbol{A}_1\big(j,k_1\big)\boldsymbol{A}_2\big(j,k_2\big)\\&=\Bigg(\sum_{\alpha_1,\alpha_2=1}^{r_1}\boldsymbol{U}_1(j,\alpha_1)\boldsymbol{S}_1(\alpha_1,\alpha_2)\overline{\boldsymbol{V}}_1(k_1,\alpha_2)\Bigg)\Bigg(\sum_{\beta_1,\beta_2=1}^{r_2}\boldsymbol{U}_2(j,\beta_1)\boldsymbol{S}_2(\beta_1,\beta_2)\overline{\boldsymbol{V}}_2(k_2,\beta_2)\Bigg)\\&=\sum_{\alpha_1=1}^{r_1}\sum_{\beta_1=1}^{r_2}\Big(\boldsymbol{U}_1(j,\alpha_1)\boldsymbol{U}_2(j,\beta_1)\Big)\sum_{\alpha_2=1}^{r_1}\sum_{\beta_2=1}^{r_2}\Big(\boldsymbol{S}_1(\alpha_1,\alpha_2)\boldsymbol{S}_2(\beta_1,\beta_2)\Big)\Big(\overline{\boldsymbol{V}}_1(k_1,\alpha_2)\overline{\boldsymbol{V}}_2(k_2,\beta_2)\Big).\end{aligned}$$

Therefore, $\boldsymbol{A}=(\boldsymbol{U}_1\bullet\boldsymbol{U}_2)(\boldsymbol{S}_1\otimes\boldsymbol{S}_2)(\boldsymbol{V}_1\otimes\boldsymbol{V}_2)^*$ and $\operatorname{rank}(\boldsymbol{A})$ is at most r_1r_2. □

Since each $z\in\mathbb{S}$ can be represented by $z=\mathrm{e}^{-2\pi\mathrm{i}t}$ where $t\in[0,1)$, every point in $\mathbb{S}^2$ can be identified with a point in a periodic domain $[0,1)^2$. Accordingly, the kernel function in (3.2) can be viewed as a function defined on $[0,1)^2$. Consider a rectangular domain $[a,b)\times[c,d)$ in $[0,1)^2$, shown as the red rectangle in Figure 3.1. We partition $[0,1)^2$ into nine rectangles by cutting along the boundaries of $[a,b)\times[c,d)$. These rectangles can be further classified into four classes. The first class is the rectangle $[a,b)\times[c,d)$ itself. The second class consists of points whose x-coordinate lies outside $[a,b)$ while whose y-coordinate lies in $[c,d)$. The third class consists of points whose x-coordinate lies in $[a,b)$ while whose y-coordinate lies outside $[c,d)$. These two classes are colored as shown in blue and green, respectively, in Figure 3.1. The fourth class consists of points whose x-coordinate and y-coordinate both lie outside $[a,b)$ and $[c,d)$, respectively, and is colored as yellow in Figure 3.1.

Let $\mathcal{R}$ and $\mathcal{C} = \mathcal{C}^{[x]} \times \mathcal{C}^{[y]}$ be row and column index sets of $\boldsymbol{G}$ respectively, such that the corresponding row and column points belong to the first class. Define $\mathcal{R}^{\mathrm{c}} = \mathcal{J} \setminus \mathcal{R}$ and $\mathcal{C}^{\mathrm{c}} = \mathcal{K} \setminus \mathcal{C}$. Suppose $|\mathcal{C}^{[x]}| = n_{\mathrm{c}}^{[x]}$ and $|\mathcal{C}^{[y]}| = n_{\mathrm{c}}^{[y]}$ and $n_{\mathrm{c}} = \max\{n_{\mathrm{c}}^{[x]}, n_{\mathrm{c}}^{[y]}\}$.

Consider the HSS row block $\boldsymbol{G}(\mathcal{R}, \mathcal{C}^{\mathrm{c}})$. We can partition it as

$$\boldsymbol{G}(\mathcal{R}, \mathcal{C}^{\mathrm{c}}) = \begin{bmatrix} \boldsymbol{G}(\mathcal{R}, \mathcal{C}_{\mathrm{b}}) & \boldsymbol{G}(\mathcal{R}, \mathcal{C}_{\mathrm{g}}) & \boldsymbol{G}(\mathcal{R}, \mathcal{C}_{\mathrm{y}}) \end{bmatrix}, \tag{3.3}$$

where $\mathcal{C}_{\mathrm{b}}$, $\mathcal{C}_{\mathrm{g}}$ and $\mathcal{C}_{\mathrm{y}}$ denote the column index sets corresponding to the second, third and fourth classes respectively. Note that these submatrices can be further factorized as

$$\begin{aligned} \boldsymbol{G}\big(\mathcal{R}, \mathcal{C}_{\mathrm{b}}\big) &= \boldsymbol{G}^{[x]}\big(\mathcal{R}, \mathcal{C}^{[x],\mathrm{c}}\big) \bullet \boldsymbol{G}^{[y]}\big(\mathcal{R}, \mathcal{C}^{[y]}\big), \\ \boldsymbol{G}\big(\mathcal{R}, \mathcal{C}_{\mathrm{g}}\big) &= \boldsymbol{G}^{[x]}\big(\mathcal{R}, \mathcal{C}^{[x]}\big) \bullet \boldsymbol{G}^{[y]}\big(\mathcal{R}, \mathcal{C}^{[y],\mathrm{c}}\big), \\ \boldsymbol{G}\big(\mathcal{R}, \mathcal{C}_{\mathrm{y}}\big) &= \boldsymbol{G}^{[x]}\big(\mathcal{R}, \mathcal{C}^{[x],\mathrm{c}}\big) \bullet \boldsymbol{G}^{[y]}\big(\mathcal{R}, \mathcal{C}^{[y],\mathrm{c}}\big). \end{aligned} \tag{3.4}$$

From Theorem 2.4, for a given accuracy parameter $0 < \varepsilon < 1$, the numerical ranks of $\boldsymbol{G}^{[x]}(\mathcal{R}, \mathcal{C}^{[x],\mathrm{c}})$ and $\boldsymbol{G}^{[y]}(\mathcal{R}, \mathcal{C}^{[y],\mathrm{c}})$ are bounded by $r = \mathscr{O}\big(\log(1/\varepsilon) \log n_{\mathrm{c}}\big)$. Suppose for simplicity that the ranks of these two matrices are exactly at most r. Then from Lemma 3.1, the ranks of $\boldsymbol{G}(\mathcal{R}, \mathcal{C}_{\mathrm{b}})$, $\boldsymbol{G}(\mathcal{R}, \mathcal{C}_{\mathrm{g}})$ and $\boldsymbol{G}(\mathcal{R}, \mathcal{C}_{\mathrm{y}})$ are at most rn_{c}, rn_{c} and r^2 respectively. Consequently, the rank of $\boldsymbol{G}(\mathcal{R}, \mathcal{C}^{\mathrm{c}})$ is at most $2rn_{\mathrm{c}} + r^2 = \mathscr{O}\big(\log(1/\varepsilon) n_{\mathrm{c}} \log n_{\mathrm{c}}\big)$. The same procedure also applies to the HSS column block $\boldsymbol{G}(\mathcal{R}^{\mathrm{c}}, \mathcal{C})$. Note that in the HSS tree, the largest size of n_{c} is $\mathscr{O}(\sqrt{N})$. Therefore, intuitively one may argue that the HSS rank of the HSS approximation of $\boldsymbol{G}$ is at most $\mathscr{O}\big(\log(1/\varepsilon)\sqrt{N} \log N\big)$.

3.2 Face-Splitting Product of HSS Trees and HSS Matrices

Let $\mathsf{T}^{[x]}$ be an HSS tree with row and column index sets $\mathcal{J}$ and $\mathcal{K}^{[x]}$ and $\mathsf{T}^{[y]}$ be an HSS tree with row and column index sets $\mathcal{J}$ and $\mathcal{K}^{[y]}$. Suppose $\mathcal{K} = \mathcal{K}^{[x]} \times \mathcal{K}^{[y]}$. We construct a new HSS tree T with row and column index sets $\mathcal{J}$ and $\mathcal{K}$ as follows.

Definition 3.2 (Face-splitting product of HSS trees)**.** *Let $\mathsf{T}^{[x]}$ and $\mathsf{T}^{[y]}$ be two HSS trees of row and column index sets $\mathcal{J}$ and $\mathcal{K}^{[x]}$ and $\mathcal{J}$ and $\mathcal{K}^{[y]}$ respectively, then the face-splitting product of $\mathsf{T}^{[x]}$ and $\mathsf{T}^{[y]}$, denoted by $\mathsf{T} = \mathsf{T}^{[x]} \bullet \mathsf{T}^{[y]}$, is defined as follows.*

(1) The root of T is $\rho = (\rho^{[x]}, \rho^{[y]})$, where $\rho^{[x]}$ and $\rho^{[y]}$ are the roots of $\mathsf{T}^{[x]}$ and $\mathsf{T}^{[y]}$ respectively.

(2) If $\tau^{[x]}$ is a leaf and $\tau^{[y]}$ is a nonleaf, then $\tau = (\tau^{[x]}, \tau^{[y]})$ is a nonleaf node of T and its children are given by $\mathrm{ch}(\tau) = \{(\tau^{[x]}, \alpha^{[y]}) : \alpha^{[y]} \in \mathrm{ch}(\tau^{[y]})\}$.

(3) If $\tau^{[x]}$ is a nonleaf and $\tau^{[y]}$ is a leaf, then $\tau = (\tau^{[x]}, \tau^{[y]})$ is a nonleaf node of T and its children are given by $\mathrm{ch}(\tau) = \{(\alpha^{[x]}, \tau^{[y]}) : \alpha^{[x]} \in \mathrm{ch}(\tau^{[x]})\}$.

(4) If $\tau^{[x]}$ and $\tau^{[y]}$ are nonleaf, then $\tau = (\tau^{[x]}, \tau^{[y]})$ is a nonleaf node of T and its children are given by $\mathrm{ch}(\tau) = \{(\alpha^{[x]}, \alpha^{[y]}) : \alpha^{[x]} \in \mathrm{ch}(\tau^{[x]}), \alpha^{[y]} \in \mathrm{ch}(\tau^{[y]})\}$.

(5) If $\tau^{[x]}$ and $\tau^{[y]}$ are both leafs, then $\tau = (\tau^{[x]}, \tau^{[y]})$ is a leaf node of T.

(6) Each node $\tau = (\tau^{[x]}, \tau^{[y]})$ is associated with the row index set $\mathcal{J}_\tau = \mathcal{J}_{\tau^{[x]}} \cap \mathcal{J}_{\tau^{[y]}}$ and the column index set $\mathcal{K}_\tau = \mathcal{K}^{[x]}_{\tau^{[x]}} \times \mathcal{K}^{[y]}_{\tau^{[y]}}$.

Typically, from the discussions in Section 2.4, the HSS tree $\mathsf{T}^{[x]}$ and $\mathsf{T}^{[y]}$ are constructed by the bisection of the periodic interval $[0, 1)$. Therefore, the HSS tree T can be viewed as a tree constructed by the quadrisection of $[0, 1)^2$. Note that each node of T is a "tensor product of" a node of $\mathsf{T}^{[x]}$ and a node of $\mathsf{T}^{[y]}$, corresponding to a rectangular domain in $[0, 1)^2$. We first prove the face-splitting product of two HSS trees is still an HSS tree and then prove the face-splitting product of two HSS matrices is still an HSS matrix.

Theorem 3.3 (Face-splitting product of HSS trees is an HSS tree)**.** *Let T be a face-splitting product of two HSS trees $\mathsf{T}^{[x]}$ and $\mathsf{T}^{[y]}$ of row and column index sets $\mathcal{J}$ and $\mathcal{K}^{[x]}$ and $\mathcal{J}$ and $\mathcal{K}^{[y]}$ respectively, then T is an HSS tree of row and column index sets $\mathcal{J}$ and $\mathcal{K}$.*

Proof. It suffices to verify that the conditions in Definition 2.2 for T.

(1) For the root node $\rho = (\rho^{[x]}, \rho^{[y]})$, we have $\mathcal{J}_\rho = \mathcal{J}_{\rho^{[x]}} \cap \mathcal{J}_{\rho^{[y]}} = \mathcal{J} \cap \mathcal{J} = \mathcal{J}$ and $\mathcal{K}_\rho = \mathcal{K}^{[x]}_{\rho^{[x]}} \times \mathcal{K}^{[y]}_{\rho^{[y]}} = \mathcal{K}^{[x]} \times \mathcal{K}^{[y]} = \mathcal{K}$.

(2) If $\tau = (\tau^{[x]}, \tau^{[y]})$ is a nonleaf node, then there are three cases. We only give the proof for the first case and the proof for the other two cases can be done by similar arguments. If $\tau^{[x]}$ is a leaf and $\tau^{[y]}$ is a nonleaf, then the children of τ are given by $\mathrm{ch}(\tau) = \{(\tau^{[x]}, \alpha^{[y]}) : \alpha^{[y]} \in \mathrm{ch}(\tau^{[y]})\}$. Therefore,

$$\mathcal{J}_\tau = \mathcal{J}_{\tau^{[x]}} \bigcap \mathcal{J}_{\tau^{[y]}} = \mathcal{J}_{\tau^{[x]}} \bigcap \Big(\bigsqcup_{\alpha^{[y]} \in \mathrm{ch}(\tau^{[y]})} \mathcal{J}_{\alpha^{[y]}} \Big) = \bigsqcup_{\alpha^{[y]} \in \mathrm{ch}(\tau^{[y]})} \Big(\mathcal{J}_{\tau^{[x]}} \bigcap \mathcal{J}_{\alpha^{[y]}} \Big) = \bigsqcup_{\alpha \in \mathrm{ch}(\tau)} \mathcal{J}_\alpha,$$

$$\mathcal{K}_\tau = \mathcal{K}^{[x]}_{\tau^{[x]}} \times \mathcal{K}^{[y]}_{\tau^{[y]}} = \mathcal{K}^{[x]}_{\tau^{[x]}} \times \Big(\bigsqcup_{\alpha^{[y]} \in \mathrm{ch}(\tau^{[y]})} \mathcal{K}^{[y]}_{\alpha^{[y]}} \Big) = \bigsqcup_{\alpha^{[y]} \in \mathrm{ch}(\tau^{[y]})} \Big(\mathcal{K}^{[x]}_{\tau^{[x]}} \times \mathcal{K}^{[y]}_{\alpha^{[y]}} \Big) = \bigsqcup_{\alpha \in \mathrm{ch}(\tau)} \mathcal{K}_\alpha.$$

This completes the proof. □

Theorem 3.4 (Face-splitting product of HSS matrices is an HSS matrix)**.** *Let T be a face-splitting product of two HSS trees $\mathsf{T}^{[x]}$ and $\mathsf{T}^{[y]}$ of row and column index sets $\mathcal{J}$ and $\mathcal{K}^{[x]}$ and $\mathcal{J}$ and $\mathcal{K}^{[y]}$ respectively. If $\boldsymbol{H}^{[x]}$ and $\boldsymbol{H}^{[y]}$ are two HSS matrices corresponding to $\mathsf{T}^{[x]}$ and $\mathsf{T}^{[y]}$, then $\boldsymbol{H} = \boldsymbol{H}^{[x]} \bullet \boldsymbol{H}^{[y]}$ is an HSS matrix for the HSS tree T.*

Furthermore, suppose that the rank of HSS blocks in $\mathsf{T}^{[x]}$ and $\mathsf{T}^{[y]}$ only depend on the level, that is, for any node $\tau^{[x]}$ in $\mathsf{T}^{[x]}$ with $\mathrm{level}(\tau^{[x]}) = \ell$ and any node $\tau^{[y]}$ in $\mathsf{T}^{[y]}$ with $\mathrm{level}(\tau^{[y]}) = \ell$, the rank of the HSS blocks corresponding to $\tau^{[x]}$ and $\tau^{[y]}$ is at most $r_\ell^{[x]}$ and $r_\ell^{[y]}$ respectively, then for any node $\tau = (\tau^{[x]}, \tau^{[y]})$ in T with $\mathrm{level}(\tau) = \ell$, the rank of the HSS blocks corresponding to τ is at most $r_\ell^{[x]} d_\ell^{[y]} + r_\ell^{[y]} d_\ell^{[x]} + r_\ell^{[x]} r_\ell^{[y]}$ where $d_\ell^{[x]}$ and $d_\ell^{[y]}$ are the maximum size of all diagonal blocks of $\boldsymbol{H}^{[x]}$ and $\boldsymbol{H}^{[y]}$ at level ℓ respectively.

Proof. Without loss of generality, assume that $\mathsf{T}^{[x]}$ and $\mathsf{T}^{[y]}$ are full binary HSS trees of maximum level L. This means that the cases (2) and (3) in Definition 3.2 will not happen. The maximum level of T is L, and every nonleaf node of T has exactly four children. General cases can be proved by similar arguments.

Let $\{\boldsymbol{D}^{[x]}, \boldsymbol{U}^{[x]}, \boldsymbol{V}^{[x]}, \boldsymbol{R}^{[x]}, \boldsymbol{W}^{[x]}, \boldsymbol{B}^{[x]}\}$ and $\{\boldsymbol{D}^{[y]}, \boldsymbol{U}^{[y]}, \boldsymbol{V}^{[y]}, \boldsymbol{R}^{[y]}, \boldsymbol{W}^{[y]}, \boldsymbol{B}^{[y]}\}$ the generator of $\boldsymbol{H}^{[x]}$ and $\boldsymbol{H}^{[y]}$ respectively. For a node $\lambda = (\tau, \sigma) \in \mathsf{T}$, we define

$$\boldsymbol{D}_\lambda \coloneqq \boldsymbol{H}(\mathcal{J}_\lambda, \mathcal{K}_\lambda) = \boldsymbol{H}\big(\mathcal{J}_\lambda, \mathcal{K}^{[x]}_\tau \times \mathcal{K}^{[y]}_\sigma\big) = \boldsymbol{H}^{[x]}(\mathcal{J}_\lambda, \mathcal{K}^{[x]}_\tau) \bullet \boldsymbol{H}^{[y]}(\mathcal{J}_\lambda, \mathcal{K}^{[y]}_\sigma).$$

By $\boldsymbol{D}^{[x]}_\tau = \boldsymbol{H}^{[x]}(\mathcal{J}_\mathsf{T}, \mathcal{K}^{[x]}_\tau)$ and $\mathcal{J}_\lambda = \mathcal{J}_\tau \cap \mathcal{J}_\sigma$, $\boldsymbol{H}^{[x]}(\mathcal{J}_\lambda, \mathcal{K}^{[x]}_\tau)$ is a submatrix corresponding to some rows of $\boldsymbol{D}^{[x]}_\tau$. We denote $\boldsymbol{H}^{[x]}(\mathcal{J}_\lambda, \mathcal{K}^{[x]}_\tau) = \boldsymbol{D}^{[x]}_\lambda$. Similarly, we denote $\boldsymbol{H}^{[y]}(\mathcal{J}_\lambda, \mathcal{K}^{[y]}_\sigma) = \boldsymbol{D}^{[y]}_\lambda$. Therefore,

$$\boldsymbol{D}_\lambda = \boldsymbol{D}^{[x]}_\lambda \bullet \boldsymbol{D}^{[y]}_\lambda.$$

If $\lambda = (\tau, \sigma)$ is a leaf, then both τ and σ are leaf, meaning that both $\boldsymbol{D}^{[x]}_\tau$ and $\boldsymbol{D}^{[y]}_\sigma$ are dense matrices, thus $\boldsymbol{D}_\lambda$ defined above is a dense matrix as well.

If $\lambda = (\tau, \sigma)$ is a nonleaf, then by our assumption both τ and σ are nonleaf and $\boldsymbol{D}^{[x]}_\tau$ and $\boldsymbol{D}^{[y]}_\sigma$ are 2×2 block matrices. Let $\mathrm{ch}(\tau) = \{\alpha_1, \alpha_2\}$ and $\mathrm{ch}(\sigma) = \{\beta_1, \beta_2\}$, then

$$\mathrm{ch}(\tau) = \{(\alpha_1, \beta_1), (\alpha_1, \beta_2), (\alpha_2, \beta_1), (\alpha_2, \beta_2)\}.$$

The columns of $\boldsymbol{D}_\lambda$ can be partitioned into 4 column blocks corresponding to $\mathrm{ch}(\lambda)$, with the (r, s)-th column block of is

$$\boldsymbol{D}_{\lambda;:,(r,s)} = \boldsymbol{D}^{[x]}_{\lambda;:,r} \bullet \boldsymbol{D}^{[y]}_{\lambda;:,s} = \boldsymbol{H}^{[x]}(\mathcal{J}_\lambda, \mathcal{K}^{[x]}_{\alpha_r}) \bullet \boldsymbol{H}^{[y]}(\mathcal{J}_\lambda, \mathcal{K}^{[y]}_{\beta_s}).$$

We partition the rows of $\boldsymbol{D}_\lambda$ into 4 blocks according to its children, where the (p,q)-th row block of $\boldsymbol{D}_\lambda$ is

$$\boldsymbol{D}_{\lambda;(p,q),:} = \boldsymbol{D}^{[x]}_{\lambda;p,:} \bullet \boldsymbol{D}^{[y]}_{\lambda;q,:} = \boldsymbol{H}^{[x]}(\mathcal{J}_{(\alpha_p,\beta_q)}, \mathcal{K}^{[x]}_{\tau}) \bullet \boldsymbol{H}^{[y]}(\mathcal{J}_{(\alpha_p,\beta_q)}, \mathcal{K}^{[y]}_{\sigma}).$$

Consequently, the $((p,q),(r,s))$-th block of $\boldsymbol{D}_\lambda$ is

$$\boldsymbol{D}_{\lambda;(p,q),(r,s)} = \boldsymbol{D}^{[x]}_{\lambda;p,r} \bullet \boldsymbol{D}^{[y]}_{\lambda;q,s} = \boldsymbol{H}^{[x]}(\mathcal{J}_{(\alpha_p,\beta_q)}, \mathcal{K}^{[x]}_{\alpha_r}) \bullet \ \boldsymbol{H}^{[y]}(\mathcal{J}_{(\alpha_p,\beta_q)}, \mathcal{K}^{[y]}_{\beta_s}).$$

In the following, we use the one-to-one mapping of the indices

$$(1,1) \leftrightarrow (\alpha_1,\beta_1) \leftrightarrow 1, \quad (1,2) \leftrightarrow (\alpha_1,\beta_2) \leftrightarrow 2, \quad (2,1) \leftrightarrow (\alpha_2,\beta_1) \leftrightarrow 3, \quad (2,2) \leftrightarrow (\alpha_2,\beta_2) \leftrightarrow 4.$$

We fix the row child node to be $\xi_1 = (\alpha_1,\beta_1)$ and discuss the block structure of $\boldsymbol{D}_\lambda$. Note that for the off-diagonal blocks, we have

$$\boldsymbol{D}^{[x]}_{\tau;1,2} = \boldsymbol{U}^{[x]}_{\alpha_1} \boldsymbol{B}^{[x]}_{\alpha_1,\alpha_2} \boldsymbol{V}^{[x],*}_{\alpha_2}, \quad \boldsymbol{D}^{[y]}_{\sigma;1,2} = \boldsymbol{U}^{[y]}_{\beta_1} \boldsymbol{B}^{[y]}_{\beta_1,\beta_2} \boldsymbol{V}^{[y],*}_{\beta_2}.$$

- Case: The column child node is $\xi_1 = (\alpha_1,\beta_1)$, corresponding to block $(1,1) = ((1,1),(1,1))$. In this case,
$$\boldsymbol{D}_{\lambda;1,1} = \boldsymbol{H}^{[x]}(\mathcal{J}_{\xi_1}, \mathcal{K}^{[x]}_{\alpha_1}) \bullet \boldsymbol{H}^{[y]}(\mathcal{J}_{\xi_1}, \mathcal{K}^{[y]}_{\beta_1}) = \boldsymbol{D}_{\xi_1}.$$
- Case: The column child node is $\xi_2 = (\alpha_1,\beta_2)$, corresponding to block $(1,2) = ((1,1),(1,2))$. Using the proof of Lemma 3.1, we have
$$\begin{aligned} \boldsymbol{D}_{\lambda;1,2} &= \boldsymbol{D}^{[x]}_{\lambda;1,1} \bullet \boldsymbol{D}^{[y]}_{\lambda;1,2} = \boldsymbol{D}^{[x]}_{\alpha_1} \bullet \big(\boldsymbol{U}^{[y]}_{\xi_1} \boldsymbol{B}^{[y]}_{\beta_1,\beta_2} \boldsymbol{V}^{[y],*}_{\beta_2}\big) \\ &= \big(\boldsymbol{D}^{[x]}_{\xi_1} \bullet \boldsymbol{U}^{[y]}_{\xi_1}\big)\big(\boldsymbol{I} \otimes \boldsymbol{B}^{[y]}_{\beta_1,\beta_2}\big)\big(\boldsymbol{I} \otimes \boldsymbol{V}^{[y]}_{\beta_2}\big)^*, \end{aligned} \tag{3.5}$$
where $\boldsymbol{U}^{[y]}_{\xi_1}$ is the submatrix of $\boldsymbol{U}^{[y]}_{\beta_1}$ corresponding to the row index set $\mathcal{J}_{\xi_1}$.
- Case: The column child node is $\xi_3 = (\alpha_2,\beta_1)$, corresponding to block $(1,3) = ((1,1),(2,1))$. In this case, by a similar argument as in (3.5), we have
$$\boldsymbol{D}_{\lambda;1,3} = \big(\boldsymbol{U}^{[x]}_{\xi_1} \boldsymbol{B}^{[x]}_{\alpha_1,\alpha_2} \boldsymbol{V}^{[x],*}_{\alpha_2}\big) \bullet \boldsymbol{D}^{[y]}_{\xi_1} = \big(\boldsymbol{U}^{[x]}_{\xi_1} \bullet \boldsymbol{D}^{[y]}_{\xi_1}\big)\big(\boldsymbol{B}^{[x]}_{\alpha_1,\alpha_2} \otimes \boldsymbol{I}\big)\big(\boldsymbol{V}^{[x]}_{\alpha_2} \otimes \boldsymbol{I}\big)^*. \tag{3.6}$$
- Case: The column child node is $\xi_4 = (\alpha_2,\beta_2)$, corresponding to block $(1,4) = ((1,1),(2,2))$. In this case, we have
$$\begin{aligned} \boldsymbol{D}_{\lambda;1,4} &= \big(\boldsymbol{U}^{[x]}_{\xi_1} \boldsymbol{B}^{[x]}_{\alpha_1,\alpha_2} \boldsymbol{V}^{[x],*}_{\alpha_2}\big) \bullet \big(\boldsymbol{U}^{[y]}_{\xi_1} \boldsymbol{B}^{[y]}_{\beta_1,\beta_2} \boldsymbol{V}^{[y],*}_{\beta_2}\big) \\ &= \big(\boldsymbol{U}^{[x]}_{\xi_1} \bullet \boldsymbol{U}^{[y]}_{\xi_1}\big)\big(\boldsymbol{B}^{[x]}_{\alpha_1,\alpha_2} \otimes \boldsymbol{B}^{[y]}_{\beta_1,\beta_2}\big)\big(\boldsymbol{V}^{[x]}_{\alpha_2} \otimes \boldsymbol{V}^{[y],*}_{\beta_2}\big). \end{aligned} \tag{3.7}$$

Therefore, for a nonroot node $\xi = (\alpha,\beta)$, if we define

$$\boldsymbol{U}_\xi = \begin{bmatrix} \boldsymbol{D}^{[x]}_{\xi} \bullet \boldsymbol{U}^{[y]}_{\xi} & \boldsymbol{U}^{[x]}_{\xi} \bullet \boldsymbol{D}^{[y]}_{\xi} & \boldsymbol{U}^{[x]}_{\xi} \bullet \boldsymbol{U}^{[y]}_{\xi} \end{bmatrix}, \quad \boldsymbol{V}_\xi = \begin{bmatrix} \boldsymbol{I} \otimes \boldsymbol{V}^{[y]}_{\beta} & \boldsymbol{V}^{[x]}_{\alpha} \otimes \boldsymbol{I} & \boldsymbol{V}^{[x]}_{\alpha} \otimes \boldsymbol{V}^{[y]}_{\beta} \end{bmatrix}, \tag{3.8}$$

then the off-diagonal blocks (3.5), (3.6) and (3.7) can be written as

$$\begin{aligned} \boldsymbol{D}_{\lambda;1,2} &= \boldsymbol{U}_{\xi_1;} \boldsymbol{B}_{\xi_1,\xi_2} \boldsymbol{V}^*_{\xi_2;}, \quad \boldsymbol{B}_{\xi_1,\xi_2} = \operatorname{diag}\big(\boldsymbol{I} \otimes \boldsymbol{B}^{[y]}_{\beta_1,\beta_2}, \boldsymbol{0}, \boldsymbol{0}\big) \\ \boldsymbol{D}_{\lambda;1,3} &= \boldsymbol{U}_{\xi_1;} \boldsymbol{B}_{\xi_1,\xi_3} \boldsymbol{V}^*_{\xi_3;}, \quad \boldsymbol{B}_{\xi_1,\xi_3} = \operatorname{diag}\big(\boldsymbol{0}, \boldsymbol{I} \otimes \boldsymbol{B}^{[x]}_{\alpha_1,\alpha_2}, \boldsymbol{0}\big) \\ \boldsymbol{D}_{\lambda;1,4} &= \boldsymbol{U}_{\xi_1;} \boldsymbol{B}_{\xi_1,\xi_4} \boldsymbol{V}^*_{\xi_4;}, \quad \boldsymbol{B}_{\xi_1,\xi_4} = \operatorname{diag}\big(\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{B}^{[x]}_{\alpha_1,\alpha_2} \otimes \boldsymbol{B}^{[y]}_{\beta_1,\beta_2}\big). \end{aligned}$$

The remaining is to verify the shared basis property. Suppose $\lambda = (\tau, \sigma)$ is a nonroot and nonleaf node. We only check it for the first row block corresponding to $\xi_1 = (\alpha_1, \beta_1)$. Consider $\boldsymbol{V}_\lambda$ in (3.8) first, following a similar argument as above, we have

$$\begin{aligned}
\boldsymbol{V}_{\lambda;1} &= \begin{bmatrix} \boldsymbol{I} \otimes \boldsymbol{V}_{\sigma;1}^{[y]} & \boldsymbol{V}_{\tau;1}^{[x]} \otimes \boldsymbol{I} & \boldsymbol{V}_{\tau;1}^{[x]} \otimes \boldsymbol{V}_{\sigma;1}^{[y]} \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{I} \otimes (\boldsymbol{V}_{\beta_1}^{[y]} \boldsymbol{W}_{\beta_1}^{[y]}) & (\boldsymbol{V}_{\alpha_1}^{[x]} \boldsymbol{W}_{\alpha_1}^{[x]}) \otimes \boldsymbol{I} & (\boldsymbol{V}_{\alpha_1}^{[x]} \boldsymbol{W}_{\alpha_1}^{[x]}) \otimes (\boldsymbol{V}_{\beta_1}^{[y]} \boldsymbol{W}_{\beta_1}^{[y]}) \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{I} \otimes \boldsymbol{V}_{\beta_1}^{[y]} & \boldsymbol{V}_{\alpha_1}^{[x]} \otimes \boldsymbol{I} & \boldsymbol{V}_{\alpha_1}^{[x]} \otimes \boldsymbol{V}_{\beta_1}^{[y]} \end{bmatrix} \operatorname{diag}\big(\boldsymbol{I} \otimes \boldsymbol{W}_{\beta_1}^{[y]}, \boldsymbol{W}_{\alpha_1}^{[x]} \otimes \boldsymbol{I}, \boldsymbol{W}_{\alpha_1}^{[x]} \otimes \boldsymbol{W}_{\beta_1}^{[y]}\big) \\
&= \boldsymbol{V}_{\xi_1;} \boldsymbol{W}_{\xi_1},
\end{aligned}$$

where $\boldsymbol{W}_{\xi_1}$ is defined as the block diagonal matrix in the above equation. Similarly, for $\boldsymbol{U}_\lambda$ in (3.8), we have

$$\boldsymbol{U}_{\lambda;1} = \begin{bmatrix} \boldsymbol{D}_{\lambda;1,:}^{[x]} \bullet \boldsymbol{U}_{\lambda;1}^{[y]} & \boldsymbol{U}_{\lambda;1}^{[x]} \bullet \boldsymbol{D}_{\lambda;1,:}^{[y]} & \boldsymbol{U}_{\lambda;1}^{[x]} \bullet \boldsymbol{U}_{\lambda;1}^{[y]} \end{bmatrix}.$$

The first column block can be further factorized as

$$\begin{aligned}
\boldsymbol{U}_{\lambda;1}^{[1]} &= \begin{bmatrix} \boldsymbol{D}_{\xi_1}^{[x]} & \boldsymbol{U}_{\xi_1}^{[x]} \boldsymbol{B}_{\alpha_1,\alpha_2}^{[x]} \boldsymbol{V}_{\alpha_2}^{[x],*} \end{bmatrix} \bullet (\boldsymbol{U}_{\xi_1}^{[y]} \boldsymbol{R}_{\beta_1}^{[y]}) \\
&= \begin{bmatrix} \boldsymbol{D}_{\xi_1}^{[x]} \bullet (\boldsymbol{U}_{\xi_1}^{[y]} \boldsymbol{R}_{\beta_1}^{[y]}) & (\boldsymbol{U}_{\xi_1}^{[x]} \boldsymbol{B}_{\alpha_1,\alpha_2}^{[x]} \boldsymbol{V}_{\alpha_2}^{[x],*}) \bullet (\boldsymbol{U}_{\xi_1}^{[y]} \boldsymbol{R}_{\beta_1}^{[y]}) \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{D}_{\xi_1}^{[x]} \bullet \boldsymbol{U}_{\xi_1}^{[y]} & \boldsymbol{U}_{\xi_1}^{[x]} \bullet \boldsymbol{D}_{\xi_1}^{[y]} & \boldsymbol{U}_{\xi_1}^{[x]} \bullet \boldsymbol{U}_{\xi_1}^{[y]} \end{bmatrix} \operatorname{diag}\Big(\boldsymbol{I} \otimes \boldsymbol{R}_{\beta_1}^{[y]}, \boldsymbol{0}, (\boldsymbol{B}_{\alpha_1,\alpha_2}^{[x]} \boldsymbol{V}_{\alpha_2}^{[x],*}) \otimes \boldsymbol{R}_{\beta_1}^{[y]}\Big) \\
&= \boldsymbol{U}_{\xi_1} \boldsymbol{R}_{\xi_1}^{[1]},
\end{aligned}$$

where $\boldsymbol{R}_{\xi_1}^{[1]}$ is defined as the block diagonal matrix in the above equation. Similarly, the second and third column blocks of $\boldsymbol{U}_{\lambda;1}$ can be factorized as $\boldsymbol{U}_{\lambda;1}^{[2]} = \boldsymbol{U}_{\xi_1} \boldsymbol{R}_{\xi_1}^{[2]}$ and $\boldsymbol{U}_{\lambda;1}^{[3]} = \boldsymbol{U}_{\xi_1} \boldsymbol{R}_{\xi_1}^{[3]}$. Therefore,

$$\boldsymbol{U}_{\lambda;1}^{[1]} = \begin{bmatrix} \boldsymbol{U}_{\lambda;1}^{[1]} & \boldsymbol{U}_{\lambda;1}^{[2]} & \boldsymbol{U}_{\lambda;1}^{[3]} \end{bmatrix} = \boldsymbol{U}_{\xi_1} \begin{bmatrix} \boldsymbol{R}_{\xi_1}^{[1]} & \boldsymbol{R}_{\xi_1}^{[2]} & \boldsymbol{R}_{\xi_1}^{[3]} \end{bmatrix} = \boldsymbol{U}_{\xi_1} \boldsymbol{R}_{\xi_1}.$$

Combining all the above results, we conclude that $\boldsymbol{H}$ is an HSS matrix about T. The results on the rank of HHS blocks can be easily verified by the construction process. □

For our NUDFT matrix, assume that $n^{[x]} = n^{[y]} = n$ and ε is a given accuracy parameter. Then for both $\boldsymbol{H}^{[x]}$ and $\boldsymbol{H}^{[y]}$, the rank of HSS blocks at level ℓ is $\mathscr{O}\big(\log(1/\varepsilon) \log(n/2^\ell)\big)$ and the maximum size of the diagonal blocks at level ℓ is $\mathscr{O}(n/2^\ell)$. Therefore the rank of $\boldsymbol{H}$ at level ℓ is $\mathscr{O}\big(\log(1/\varepsilon)(n/2^\ell) \log(n/2^\ell)\big)$, implying that the HSS rank of $\boldsymbol{H}$ is $\mathscr{O}\big(\log(1/\varepsilon)\sqrt{N} \log N\big)$.

Next we give a result on the approximation error of the face-splitting product of HSS approximations of $\boldsymbol{G}^{[x]}$ and $\boldsymbol{G}^{[y]}$ to $\boldsymbol{G}$. Before that, we point out that since every row of $\boldsymbol{A}^{[x]}$ has the same norm and $\boldsymbol{F}^{[x]}$ is unitary up to a scaling, every row of $\boldsymbol{G}^{[x]} = \boldsymbol{A}^{[x]} \boldsymbol{F}^{[x],-1}$ has the same norm. The similar result also holds for $\boldsymbol{G}^{[y]}$ and consequently for $\boldsymbol{G}$. These facts serve as a key tool in our proofs and we write it as a lemma here, whose proof is straightforward and omitted.

Lemma 3.5 (Equal row norm property)**.** *Consider the 2D type-II NUDFT matrix $\boldsymbol{A} = \boldsymbol{A}^{[x]} \bullet \boldsymbol{A}^{[y]}$ defined by* (1.2)*, where $\boldsymbol{A}^{[x]}$ and $\boldsymbol{A}^{[y]}$ be the type-II NUDFT matrices in x and y direction respectively. Let $\boldsymbol{F} = \boldsymbol{F}^{[x]} \otimes \boldsymbol{F}^{[y]}$ be the DFT matrix in 2D and $\boldsymbol{G} = \boldsymbol{A}\boldsymbol{F}^{-1} = \boldsymbol{G}^{[x]} \bullet \boldsymbol{G}^{[y]}$ where $\boldsymbol{G}^{[x]} = \boldsymbol{A}^{[x]} \boldsymbol{F}^{[x],-1}$ and $\boldsymbol{G}^{[y]} = \boldsymbol{A}^{[y]} \boldsymbol{F}^{[y],-1}$. Then every row of $\boldsymbol{G}^{[x]} = \boldsymbol{A}^{[x]} \boldsymbol{F}^{[x],-1}$ has the same norm, denoted by $Q^{[x]}$ and every row of $\boldsymbol{G}^{[y]} = \boldsymbol{A}^{[y]} \boldsymbol{F}^{[y],-1}$ has the same norm, denoted by $Q^{[y]}$. Consequently, every row of $\boldsymbol{G} = \boldsymbol{A}\boldsymbol{F}^{-1} = \boldsymbol{G}^{[x]} \bullet \boldsymbol{G}^{[y]}$ has the same norm $Q^{[x]}Q^{[y]}$. In particular, $\|\boldsymbol{G}^{[x]}\|_{\mathrm{F}} = \sqrt{M} Q^{[x]}$, $\|\boldsymbol{G}^{[y]}\|_{\mathrm{F}} = \sqrt{M} Q^{[y]}$ and $\|\boldsymbol{G}\|_{\mathrm{F}} = \sqrt{M} Q^{[x]} Q^{[y]}$.*

Theorem 3.6. *Using the same notations as in Lemma 3.5, for a given $0 < \varepsilon < 1$, suppose $\widetilde{\boldsymbol{G}}^{[x]}$ and $\widetilde{\boldsymbol{G}}^{[y]}$ are two HSS approximations of $\boldsymbol{G}^{[x]}$ and $\boldsymbol{G}^{[y]}$ such that $\|\boldsymbol{G}^{[x]} - \widetilde{\boldsymbol{G}}^{[x]}\|_{\mathrm{F}} \leq \varepsilon\|\boldsymbol{G}^{[x]}\|_{\mathrm{F}}$ and $\|\boldsymbol{G}^{[y]} - \widetilde{\boldsymbol{G}}^{[y]}\|_{\mathrm{F}} \leq \varepsilon\|\boldsymbol{G}^{[y]}\|_{\mathrm{F}}$. Assume that exists a constant C such that $\max_j\{\|\widetilde{\boldsymbol{G}}^{[x]}(j,:)\|_2\} \leq C\max_j\{\|\boldsymbol{G}^{[x]}(j,:)\|_2\}$. Then $\widetilde{\boldsymbol{G}} = \widetilde{\boldsymbol{G}}^{[x]} \bullet \widetilde{\boldsymbol{G}}^{[y]}$ is an HSS approximation of $\boldsymbol{G} = \boldsymbol{G}^{[x]} \bullet \boldsymbol{G}^{[y]}$ and $\|\widetilde{\boldsymbol{G}} - \boldsymbol{G}\|_{\mathrm{F}} \leq (1 + C)\varepsilon\|\boldsymbol{G}\|_{\mathrm{F}}$.*

Proof. From Lemma 2.1 and Lemma 3.5, we have

$$\begin{aligned}
\|\boldsymbol{G} - \widetilde{\boldsymbol{G}}\|_{\mathrm{F}} &= \|(\boldsymbol{G}^{[x]} - \widetilde{\boldsymbol{G}}^{[x]}) \bullet \boldsymbol{G}^{[y]} + \widetilde{\boldsymbol{G}}^{[x]} \bullet (\boldsymbol{G}^{[y]} - \widetilde{\boldsymbol{G}}^{[y]})\|_{\mathrm{F}} \\
&\leq \|(\boldsymbol{G}^{[x]} - \widetilde{\boldsymbol{G}}^{[x]}) \bullet \boldsymbol{G}^{[y]}\|_{\mathrm{F}} + \|\widetilde{\boldsymbol{G}}^{[x]} \bullet (\boldsymbol{G}^{[y]} - \widetilde{\boldsymbol{G}}^{[y]})\|_{\mathrm{F}} \\
&\leq \varepsilon\|\boldsymbol{G}^{[x]}\|_{\mathrm{F}}Q^{[y]} + CQ^{[x]}\varepsilon\|\boldsymbol{G}^{[y]}\|_{\mathrm{F}} \\
&= \varepsilon\sqrt{M}Q^{[x]}Q^{[y]} + C\varepsilon\sqrt{M}Q^{[x]}Q^{[y]} = (1 + C)\varepsilon\|\boldsymbol{G}\|_{\mathrm{F}}.
\end{aligned}$$

This completes the proof. □

From $\|\boldsymbol{G}^{[x]} - \widetilde{\boldsymbol{G}}^{[x]}\|_{\mathrm{F}} \leq \varepsilon\|\boldsymbol{G}^{[x]}\|_{\mathrm{F}}$, it is not hard to prove that C can be chosen as $1 + \sqrt{M}\varepsilon$. Therefore, one can expect the approximation error in Theorem 3.6 is not large when $\sqrt{M}\varepsilon$ is small. However, this estimate is quite pessimistic. If every row of $\boldsymbol{G}^{[x]} - \widetilde{\boldsymbol{G}}^{[x]}$ has approximately the same norm (and this could be the case in practice), then one could expect that C is a small constant independent of M and N, meaning that the approximation error in Theorem 3.6 is small.

4 A Direct Solver for the Inverse NUDFT

In this section, we present a superfast solver for the inverse NUDFT problem (1.1) based on the HSS approximation of $\boldsymbol{G}$. Section 4.1 describes the construction of the HSS approximation of $\boldsymbol{G}$ and Section 4.2 presents the algorithm for the approximate direct inversion.

4.1 The Construction of the HSS Approximation

Theorem 3.4 shows that $\boldsymbol{G} = \boldsymbol{G}^{[x]} \bullet \boldsymbol{G}^{[y]}$ could be approximated by an HSS matrix $\widetilde{\boldsymbol{G}}$, and the proof of it also provides a way to construct $\widetilde{\boldsymbol{G}}$ directly from the HSS approximations of $\boldsymbol{G}^{[x]}$ and $\boldsymbol{G}^{[y]}$. But, instead, our construction of $\widetilde{\boldsymbol{G}}$ works on the kernel matrix expression (3.1) of $\boldsymbol{G}$ directly, which is a common way in many literatures such as [9, 25, 26]. The main tool in the construction is the *interpolative decomposition* (ID) [10], which is defined as follows.

Lemma 4.1 (Interpolative decomposition [10])**.** *Let $\boldsymbol{A}$ be an $m \times n$ matrix with rank $r \leq \min\{m, n\}$. Then there exist a subset $\widehat{\mathcal{K}} \subset \{1, 2, \ldots, n\}$ of size r and a matrix $\boldsymbol{Y} \in \mathbb{C}^{n\times r}$ such that $\boldsymbol{A} = \boldsymbol{A}(:, \widehat{\mathcal{K}})\boldsymbol{Y}^*$ and $\boldsymbol{Y}(\widehat{\mathcal{K}}, :) = \boldsymbol{I}$. This is called the column ID of $\boldsymbol{A}$. Similarly, there exist a subset $\widehat{\mathcal{J}} \subset \{1, 2, \ldots, m\}$ of size r and a matrix $\boldsymbol{L} \in \mathbb{C}^{m\times r}$ such that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{A}(\widehat{\mathcal{J}}, :)$ and $\boldsymbol{L}(\widehat{\mathcal{J}}, :) = \boldsymbol{I}$. This is called the row ID of $\boldsymbol{A}$. The matrices $\boldsymbol{Y}$ and $\boldsymbol{L}$ are called the interpolation matrices.*

Numerically, ID can be computed by rank-revealing QR factorization (RRQR) [17] or be approximated using the QR factorization with column pivoting (QRCP). Since $\boldsymbol{G}$ has a kernel matrix expression, the *proxy surface* technique [25, 26] is used to accelerate the construction process. By (3.1), for any row and column index sets $\mathcal{R}$ and $\mathcal{C}$, the submatrix can be expressed as $\boldsymbol{G}(\mathcal{R}, \mathcal{C}) = \Psi(\mathscr{Z}_{\mathcal{R}}, \mathscr{W}_{\mathcal{C}})$ where Ψ is the kernel function, $\mathscr{Z}_{\mathcal{R}} = \{\boldsymbol{\gamma}_j\}_{j\in\mathcal{R}}$ and $\mathscr{W}_{\mathcal{C}} = \{\boldsymbol{\xi}_k\}_{k\in\mathcal{C}}$.

We give a brief introduction of the construction process. Starting from the leaf nodes, we set $\boldsymbol{D}_\tau = \boldsymbol{G}(\mathcal{J}_\tau, \mathcal{K}_\tau)$ directly. For each node τ, let $\mathscr{P}_\tau$ be a set of proxy points such that

$$\boldsymbol{G}(\mathcal{J}_\tau, \mathcal{K}_\tau^{\mathrm{c}}) = \Psi(\mathscr{Z}_{\mathcal{J}_\tau}, \mathscr{W}_{\mathcal{K}_\tau^{\mathrm{c}}}) \approx \Psi(\mathscr{Z}_{\mathcal{J}_\tau}, \mathscr{P}_\tau)\boldsymbol{Q}_\tau^*, \quad \boldsymbol{G}(\mathcal{J}_\tau^{\mathrm{c}}, \mathcal{K}_\tau) = \Psi(\mathscr{Z}_{\mathcal{J}_\tau^{\mathrm{c}}}, \mathscr{W}_{\mathcal{K}_\tau}) \approx \boldsymbol{P}_\tau\Psi(\mathscr{P}_\tau, \mathscr{W}_{\mathcal{K}_\tau})$$

for some matrix $\boldsymbol{P}_\tau$ and $\boldsymbol{Q}_\tau$. If we have the row ID $\Psi(\mathscr{Z}_{\mathcal{J}_\tau}, \mathscr{P}_\tau) \approx \boldsymbol{L}_\tau \Psi(\mathscr{Z}_{\mathcal{J}_\tau}, \mathscr{P}_\tau)$, then

$$\boldsymbol{G}(\mathcal{J}_\tau, \mathcal{K}_\tau^{\mathrm{c}}) \approx \Psi(\mathscr{Z}_{\mathcal{J}_\tau}, \mathscr{W}_{\mathcal{K}_\tau^{\mathrm{c}}}) \approx \boldsymbol{L}_\tau \Psi(\mathscr{Z}_{\mathcal{J}_\tau}, \mathscr{P}_\tau) \boldsymbol{Q}_\tau^*,$$

which gives the row ID of $\boldsymbol{G}(\mathcal{J}_\tau, \mathcal{K}_\tau^{\mathrm{c}})$. Similarly, the column ID of $\boldsymbol{G}(\mathcal{J}_\tau^{\mathrm{c}}, \mathcal{K}_\tau)$ can be obtained from the column ID of $\Psi(\mathscr{P}_\tau, \mathscr{W}_{\mathcal{K}_\tau})$. Therefore, after computing the row and column IDs of $\Psi(\mathscr{Z}_{\mathcal{K}_\tau}, \mathscr{P}_\tau)$ and $\Psi(\mathscr{P}_\tau, \mathscr{W}_{\mathcal{K}_\tau})$, the basis matrices can be obtained as $\boldsymbol{U}_\tau = \boldsymbol{L}_\tau$ and $\boldsymbol{V}_\tau = \boldsymbol{Y}_\tau$.

After computing the generators for all leaf nodes on level L, we move to level $L-1$. For a node τ on level $L-1$, we define the row index sets $\mathcal{J}_\tau^+ = \cup_{\alpha \in \mathrm{ch}(\tau)} \widehat{\mathcal{J}}_\alpha$, and set $\mathcal{J}^+ = \cup_{\mathrm{level}(\tau)=L-1} \mathcal{J}_\tau^+$ and $\mathcal{J}_\tau^{\mathrm{c},+} = \mathcal{J}^+ \setminus \mathcal{J}_\tau^+$. The column index sets $\mathcal{K}_\tau^+$, $\mathcal{K}^+$ and $\mathcal{K}_\tau^{\mathrm{c},+}$ are defined similarly. For a node τ on level $L-1$, the interaction matrices of its children are given by $\boldsymbol{B}_{\alpha_p,\alpha_q} = \boldsymbol{G}(\widehat{\mathcal{J}}_{\alpha_p}, \widehat{\mathcal{K}}_{\alpha_q})$ for $p \neq q$. We then compute the interpolation matrices $\boldsymbol{L}_\tau$ and $\boldsymbol{Y}_\tau$ by the row and column IDs of $\Psi(\mathscr{Z}_{\mathcal{J}_\tau^+}, \mathscr{P}_\tau)$ and $\Psi(\mathscr{P}_\tau, \mathscr{W}_{\mathcal{K}_\tau^+})$ respectively. Partitioning $\boldsymbol{L}_\tau$ and $\boldsymbol{Y}_\tau$ according to the children of τ, the transfer matrices are given as $\boldsymbol{R}_{\alpha_p} = \boldsymbol{L}_{\tau;p}$ and $\boldsymbol{W}_{\alpha_p} = \boldsymbol{Y}_{\tau;p}$. This process is repeated until we reach the root node ρ.

Typically, the size of the proxy point set is $\mathscr{O}(r_\tau)$ where r_τ is the rank of the HSS blocks corresponding to τ. In our case, $\mathscr{P}_\tau$ is chosen as the random points surrounding the bounding box of $\mathscr{Z}_{\mathcal{J}_\tau}$ and $\mathscr{W}_{\mathcal{K}_\tau}$, which is shown in Figure 4.1. More speficically, suppose the box contains $n_{\mathrm{c}} \times n_{\mathrm{c}}$ column points. Let the 1D rank estimate in Theorem 2.4 be given as s_τ (hence $r_\tau \approx s_\tau n_{\mathrm{c}}$). Then we sample $s_\tau n_{\mathrm{c}}$ points uniformly on the horizontal and vertical sides of the box, and sample s_τ^2 points uniformly on the diagonal boundary of the box. Following the discussions in [25], the complexity of the construction of the HSS approximation of $\boldsymbol{G}$ is $\mathscr{O}\big(M + N^{3/2} \log^3 N\big)$.

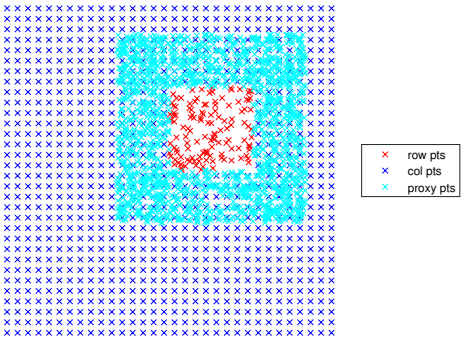

Figure 4.1: Proxy surfaces for the row ID in the construction of the HSS approximation of $\boldsymbol{G}$. The red points are the row points $\mathscr{Z}_{\mathcal{J}_\tau}$, the blue points are the the column points $\mathscr{W}_{\mathcal{K}_\tau}$ and the cyan points are the proxy points $\mathscr{P}_\tau$.

4.2 An Algorithm for the Direct Inversion of Type-II 2D NUDFT

We design an algorithm for solving the 2D type-II NUDFT based on the HSS matrix approximation of $\boldsymbol{G}$, based on the discussions from the previous sections. The algorithm consists of two stages. The *offline* stage first constructs the HSS approximation of $\boldsymbol{G}$ using the construction method described in Section 4.1, then computes the URV factorization of the HSS matrix as described in Section 2.3. This is is independent of the target values $\boldsymbol{f}$ and can be reused. The *online* stage computes the solution of the inverse NUDFT problem by the URV solution of HSS matrix and two-dimensional iFFT. The pseudocode of the algorithm is summarized in Algorithm 1. Algebraically, we have a factorization

$$\boldsymbol{A} = \boldsymbol{G}\boldsymbol{F} \approx \boldsymbol{A}_{\mathrm{fast}} \coloneqq \widetilde{\boldsymbol{G}}\boldsymbol{F},$$

for the type-II 2D NUDFT matrix $\boldsymbol{A}$, where $\widetilde{\boldsymbol{G}}$ is the HSS approximation of $\boldsymbol{G}$ and $\boldsymbol{F}$ is the 2D DFT matrix. The pseudo-inverse of $\boldsymbol{A}$ can be approximated by

$$\boldsymbol{A}^{\dagger} \approx \boldsymbol{A}_{\text{fast}}^{\dagger} = \boldsymbol{F}^{-1}\widetilde{\boldsymbol{G}}^{\dagger},$$

where $\widetilde{\boldsymbol{G}}^{\dagger}$ is represented by its URV factorization and $\boldsymbol{F}^{-1}$ can be computed by two-dimensional iFFT. We also remark that our fast direct solver for the 2D type-II NUDFT can be used as a preconditioner for iterative solvers such as lsqr, as long as we can efficiently apply $\boldsymbol{A}_{\text{fast}}^{\dagger,*} = \widetilde{\boldsymbol{G}}^{\dagger *}\boldsymbol{F}/N$, which involves the URV solution of HSS matrix mentioned in the end of Section 2.3 and two-dimensional FFT.

Algorithm 1 A direct solver for type-II 2D NUDFT problem

Input: Sample points $\{(x_j, y_j)\}_{j=0}^{M-1}$ and number of frequencies $n^{[x]}$ and $n^{[y]}$, target values $\{f_j\}_{j=0}^{M-1}$.

Output: Coefficients $\boldsymbol{c}$.

1: Construct the HSS approximation $\widetilde{\boldsymbol{G}}$ of $\boldsymbol{G} = \boldsymbol{A}\boldsymbol{F}^{-1}$ using the kernel matrix expression (3.1).
2: Compute the URV factorization of $\widetilde{\boldsymbol{G}}$.
3: Compute $\boldsymbol{c} = \boldsymbol{F}^{-1}\widetilde{\boldsymbol{G}}^{\dagger}\boldsymbol{f}$ by the URV solution and two-dimensional iFFT.

We analyze the complexity of Algorithm 1 under a simplified setting. Suppose $\mathsf{T}^{[x]}$ and $\mathsf{T}^{[y]}$ are both full binary HSS trees with maximum level L constructed by bisecting the column index sets, and the maximum level of both trees is L, and all leaf nodes have exact column size n_L, where n_L is a small constant independent of $n^{[x]}$ and $n^{[y]}$. That is, $n^{[x]} = n^{[y]} = n = 2^L n_L$ and $N = 4^L N_L$ where $N_L = n_L^2$. Let $\mathsf{T} = \mathsf{T}^{[x]} \bullet \mathsf{T}^{[y]}$ and assume further that $M = 4^L M_L$ correspondingly where M_L is the number of sample points in the leaf nodes. For level ℓ, there are 4^ℓ nodes and the HSS blocks on level ℓ be denoted by r_ℓ. By the discussions in Section 3.1 and Section 3.2, we assume that $r_\ell = \mathscr{O}(n_\ell \log n_\ell)$. Follwing the results in Section 4.1 and Section 2.3, the complexity of the offline stage is $\mathscr{O}\big(M + N^{3/2}\log^3 N\big)$, while the complexity of the URV solution is $\mathscr{O}\big(M + N\log^3 N\big)$.

5 Numerical Results

In this section, we present numerical results of our direct solver for the 2D type-II NUDFT. Two types of nonuniform grids are considered: the random grid and the (modified) polar grid [15]. The random grid is generated by sampling M points distributed uniformly randomly in $[0,1)^2$. The polar grid is defined as the zero point $(0,0)$ together with points of the form

$$x_{(p,q)} = \frac{1}{2} + r_p\cos(2\pi t_q), \quad y_{(p,q)} = \frac{1}{2} + r_p\sin(2\pi t_q),$$
$$r_p = \frac{\sqrt{2}}{2}\frac{p}{n_r}, \quad t_q = \frac{q}{n_t}, \quad p = 1,\ldots,n_r-1, \quad q = 0,1,\ldots,n_t-1,$$

where we exclude the points outside $[0,1)^2$. Figure 5.1 shows the two grid examples.

In our experiments, we first let n ranges in $\{32, 64, 128, 256, 512\}$ and set $n^{[x]} = n^{[y]} = n$ and $N = n^{[x]}n^{[y]} = n^2$. For random grid, the number of sample points is set as $M = \alpha N$, where α is selected to be 1.5. For polar grid, we set $n_r = n$ and $n_t \approx \beta n\log_2 n$ for $\beta = 0.6$, which gives $M \approx 0.8\beta N\log_4 N$, where the factor 0.8 comes from the sample area that is inside $[0,1)^2$. The well-known Shepp-Logan phantom, which is widely used in the field of magnetic resonance imaging (MRI) [7], is used as the exact solution. That is, the coefficients $\boldsymbol{c}$ are sampled on a $n \times n$ grid of the Shepp-Logan phantom, and the target values $\boldsymbol{f}$ are generated by applying the type-II 2D NUFFT to $\boldsymbol{c}$. The tolerance parameter ε in the construction of the HSS approximation is set as 10^{-2} or 10^{-4} for all experiments. We use the MATLAB command "lsqr" [27]

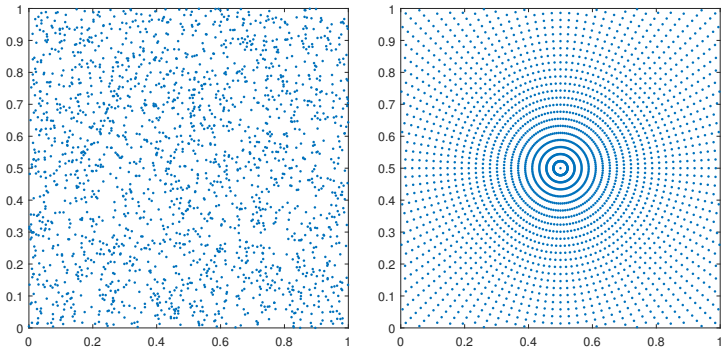

Figure 5.1: Nonuniform grids. $N = 32^2$. Left: Random grid with $M = 2N$ points. Right: Polar grid with $M \approx 0.54N \log_4(N)$ points.

as the iterative solver, with the maximum number of iterations set as 500 and the tolerance parameter set as 10^{-12}. All algorithms are implemented in MATLAB R2023b and the FINUFFT library [3] are utilized for forward NUFFT, and are carried out on a server with two Intel Gold 6226R CPUs at 2.90 GHz and 1000.6 GB of RAM.

The following quantities are used to evaluate the performance of the algorithms. We use r_{h} to denote the HSS rank of the HSS approximation of $\boldsymbol{G}$, t_{c} to denote the walltime of constructing the HSS approximation, t_{f} to denote the walltime of computing the URV factorization of the HSS approximation, t_{s} to denote the walltime of computing the solution by the URV solution of HSS matrix and iFFT, r_{s} to denote the relative residual of the solution, i.e., $\|\boldsymbol{A}\boldsymbol{c} - \boldsymbol{f}\|_2/\|\boldsymbol{f}\|_2$, and e_{s} to denote the relative error of the solution, i.e., $\|\boldsymbol{c} - \boldsymbol{c}_{\mathrm{ex}}\|_2/\|\boldsymbol{c}_{\mathrm{ex}}\|_2$ where $\boldsymbol{c}_{\mathrm{ex}}$ is the exact solution. For iterative solvers, we use t_{pre} to denote the walltime of constructing the preconditioner, i.e., the offline computation, which equals to $t_{\mathrm{c}} + t_{\mathrm{f}}$, t_{iter} to denote the walltime of all iterations when solving the problem by MATLAB's "lsqr" command with or without a preconditioner, and n_{iter} to denote the number of iterations.

5.1 Random Grid

Figure 5.2 shows the scaling of the HSS rank, construction time, factorization time, and solution time of Algorithm 1 on random grids with $M = 1.5N$. The corresponding numerical results are summarized in Table 5.1. It can be observed that the HSS rank grows as $\mathcal{O}\left(\sqrt{N}\log N\right)$, which is consistent with our analysis. Moreover, both the construction and factorization times follow the predicted $\mathcal{O}\left(M + N^{3/2}\log^3 N\right)$ complexity. The solving time is significantly smaller than the construction and factorization times, indicating that the dominant computational cost lies in the offline stage, while the online stage is negligible in comparison. For example, when $N = 512^2$, $M = 1.5N$ and $\varepsilon = 10^{-2}$, the construction takes approximately 1.4 hours, whereas the factorization and solution times are only about 28 minutes and 50 seconds, respectively. Furthermore, the tolerance parameter ε has a significant impact on the computational cost. For the same example, when ε is decreased from 10^{-2} to 10^{-4}, the construction, factorization, and solution times increase by factors of approximately 4.7, 3.2, and 1.3, respectively. In most cases, the relative residuals and relative errors of the direct solver are on the order of 10^{-2} for $\varepsilon = 10^{-2}$ and 10^{-4} for $\varepsilon = 10^{-4}$, suggesting that the algorithm remains stable as the problem size increases.

Table 5.2 demonstrates the behavior of the iterative solver using our direct solver as a preconditioner, where lsqr denotes the standard lsqr method and plsqr denotes the preconditioned lsqr method. First, the results show that, for random grids with $M = 1.5N$, standard lsqr fails to converge within 500 iterations for all tested problem sizes. In contrast, plsqr converges in 8 to 40 iterations when $\varepsilon = 10^{-2}$ and 4 to 14 iterations when $\varepsilon = 10^{-4}$, and the relative residual and relative error achieve at a level of 10^{-12} in both

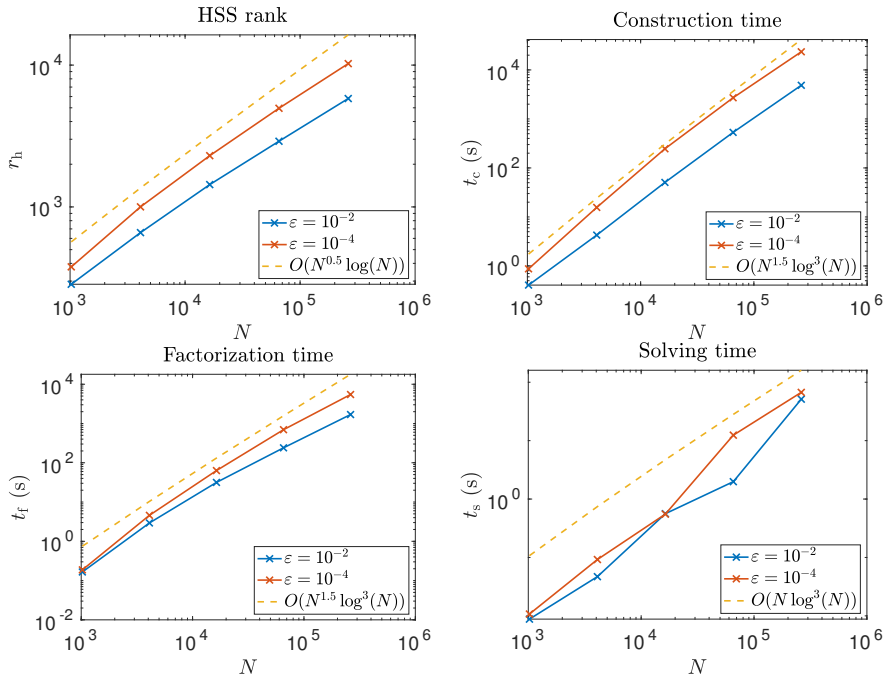

Figure 5.2: HSS rank, construction time, factorization time and solution time of the direct solver on a random grid. $M = 1.5N$.

N	M	ε	t_{c}	t_{f}	t_{s}	r_{s}	e_{s}
32^2	1536	10^{-2}	4.1e-01	1.7e-01	8.8e-03	2.1e-03	2.3e-03
		10^{-4}	8.7e-01	1.9e-01	1.1e-02	1.0e-05	1.2e-05
64^2	6144	10^{-2}	4.3e+00	2.9e+00	4.7e-02	3.9e-03	5.9e-03
		10^{-4}	1.6e+01	4.6e+00	9.2e-02	3.0e-05	4.3e-05
128^2	24576	10^{-2}	5.1e+01	3.2e+01	5.6e-01	5.0e-03	7.1e-03
		10^{-4}	2.5e+02	6.3e+01	5.5e-01	1.4e-04	2.1e-04
256^2	98304	10^{-2}	5.3e+02	2.4e+02	2.0e+00	7.6e-03	1.1e-02
		10^{-4}	2.7e+03	6.9e+02	1.2e+01	4.2e-04	6.0e-04
512^2	393216	10^{-2}	4.9e+03	1.7e+03	5.1e+01	1.1e-02	1.7e-02
		10^{-4}	2.3e+04	5.5e+03	6.7e+01	7.3e-04	1.1e-03

Table 5.1: Construction time, factorization time, solve time, relative residual and relative error of the direct solver on a random grid. $M = 1.5N$.

N	M	method	t_{pre}	t_{iter}	n_{iter}	r_{s}	e_{s}
32^2	1536	lsqr	-	1.2e+00	500	5.2e-10	5.7e-10
		plsqr ($\varepsilon = 10^{-2}$)	5.7e-01	1.4e-01	8	2.6e-12	3.0e-12
		plsqr ($\varepsilon = 10^{-4}$)	1.1e+00	1.1e-01	4	4.4e-15	5.2e-15
64^2	6144	lsqr	-	2.7e+00	500	2.8e-08	5.4e-08
		plsqr ($\varepsilon = 10^{-2}$)	7.2e+00	1.0e+00	12	5.6e-13	9.6e-13
		plsqr ($\varepsilon = 10^{-4}$)	2.0e+01	8.9e-01	4	1.3e-13	2.0e-13
128^2	24576	lsqr	-	4.8e+00	500	6.5e-05	9.4e-05
		plsqr ($\varepsilon = 10^{-2}$)	8.2e+01	8.5e+00	14	4.3e-12	7.0e-12
		plsqr ($\varepsilon = 10^{-4}$)	3.1e+02	7.6e+00	6	2.5e-14	3.5e-14
256^2	98304	lsqr	-	1.2e+01	500	1.2e-04	2.1e-04
		plsqr ($\varepsilon = 10^{-2}$)	7.7e+02	6.6e+01	23	2.0e-12	3.1e-12
		plsqr ($\varepsilon = 10^{-4}$)	3.4e+03	5.6e+01	8	8.8e-13	1.3e-12
512^2	393216	lsqr	-	3.0e+01	500	1.9e-04	3.2e-04
		plsqr ($\varepsilon = 10^{-2}$)	6.5e+03	5.1e+02	39	3.6e-12	5.5e-12
		plsqr ($\varepsilon = 10^{-4}$)	2.9e+04	4.6e+02	14	2.7e-12	4.0e-12

Table 5.2: Time cost, iteration number, relative residual and relative error of lsqr and plsqr on a random grid. $M = 1.5N$.

cases. It means that the preconditioner is effective and efficient.

5.2 Polar Grid

Following the setting introduced at the beginning of Section 5, for the polar grid we set $n_r = n$ and $n_t \approx \beta n \log_2 n$ with $\beta = 0.6$. This gives $M \approx 0.47N \log_4 N$. Figure 5.3 reports the scaling of the HSS rank, construction time, factorization time, and solution time of Algorithm 1 on the polar grid with $\beta = 0.6$. The corresponding numerical results are summarized in Table 5.3. Similar to the random-grid case, the HSS rank grows approximately as $\mathcal{O}\big(\sqrt{N}\log N\big)$, in agreement with the estimation in Section 4.2. The construction and factorization times also follow the predicted $\mathcal{O}\big(M + N^{3/2}\log^3 N\big)$ scaling. In particular, since $M \approx 0.47N \log_4 N$ for the polar grid, the dominant cost in both stages comes from the $\mathcal{O}\big(N^{3/2}\log^3 N\big)$ term rather than the linear $\mathcal{O}(M)$ term. The solution time is much smaller than the construction and factorization times. For instance, when $N = 512^2$ and $\varepsilon = 10^{-2}$, the construction takes about 1.7 hours, whereas the factorization and solution times are approximately 33 minutes and 34 seconds, respectively. When the tolerance is tightened to $\varepsilon = 10^{-4}$, the construction, factorization, and solution times increase to 7.2 hours, 1.9 hours, and 92 seconds, respectively. Moreover, the relative residuals and relative errors of the direct solver are generally on the order of 10^{-2} for $\varepsilon = 10^{-2}$ and 10^{-4} for $\varepsilon = 10^{-4}$, indicating that the proposed direct solver remains stable for the polar grid as the problem size increases.

Table 5.4 shows the performance of lsqr and plsqr on the polar grid. For $\beta = 0.6$, lsqr fails to converge within 500 iterations for all but the smallest problem size. In contrast, plsqr converges in 9 to 38 iterations when $\varepsilon = 10^{-2}$, and in 4 to 10 iterations when $\varepsilon = 10^{-4}$. In both cases, the relative residuals and relative errors reach the level of 10^{-11} to 10^{-12}, demonstrating the effectiveness of the proposed preconditioner. Overall, these results confirm that the proposed preconditioner significantly improves the convergence of lsqr on polar grids, especially for larger and more ill-conditioned problems.

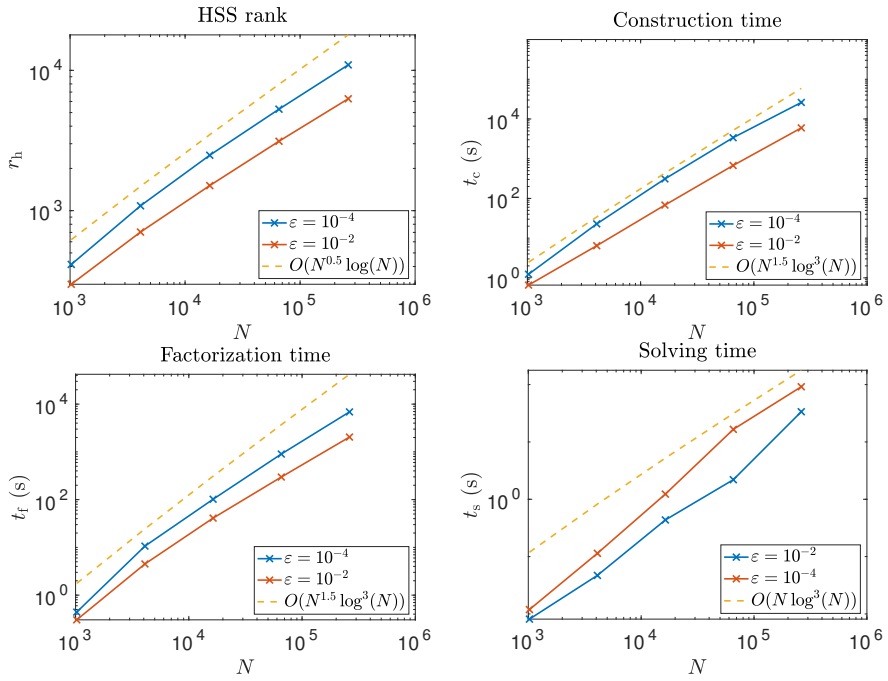

Figure 5.3: HSS rank, construction time, factorization time and solution time of the direct solver on a polar grid. $\beta = 0.6$. $M \approx 0.47N\log_4(N)$.

N	M	ε	t_{c}	t_{f}	t_{s}	r_{s}	e_{s}
32^2	2397	10^{-2}	6.6e-01	3.0e-01	8.1e-03	6.6e-04	7.0e-03
		10^{-4}	1.2e+00	4.4e-01	1.2e-02	3.3e-06	1.3e-05
64^2	11620	10^{-2}	6.4e+00	4.5e+00	4.7e-02	2.9e-02	3.1e-02
		10^{-4}	2.3e+01	1.1e+01	1.1e-01	2.1e-04	2.2e-04
128^2	54373	10^{-2}	6.8e+01	4.1e+01	4.4e-01	2.3e-03	3.2e-02
		10^{-4}	3.1e+02	1.0e+02	1.2e+00	8.0e-05	2.0e-03
256^2	249074	10^{-2}	6.8e+02	3.0e+02	2.2e+00	5.5e-02	5.4e-02
		10^{-4}	3.4e+03	9.0e+02	1.7e+01	2.9e-03	2.8e-03
512^2	1122019	10^{-2}	6.0e+03	2.0e+03	3.4e+01	6.1e-03	5.7e-03
		10^{-4}	2.6e+04	6.9e+03	9.2e+01	4.6e-04	4.2e-04

Table 5.3: Construction time, factorization time, solve time, relative residual and relative error of the direct solver on a polar grid. $M \approx 0.47N\log_4(N)$.

N	M	method	t_{pre}	t_{iter}	n_{iter}	r_{s}	e_{s}
32^2	2397	lsqr	-	1.3e+00	470	8.9e-13	6.8e-12
		plsqr ($\varepsilon = 10^{-2}$)	9.6e-01	1.9e-01	9	2.9e-13	9.9e-12
		plsqr ($\varepsilon = 10^{-4}$)	1.7e+00	1.5e-01	4	7.7e-16	6.5e-15
64^2	11620	lsqr	-	2.9e+00	500	3.8e-03	4.0e-03
		plsqr ($\varepsilon = 10^{-2}$)	1.1e+01	2.2e+00	21	1.5e-11	1.6e-11
		plsqr ($\varepsilon = 10^{-4}$)	3.3e+01	1.4e+00	5	8.9e-11	9.3e-11
128^2	54373	lsqr	-	5.6e+00	500	2.2e-05	1.6e-03
		plsqr ($\varepsilon = 10^{-2}$)	1.1e+02	2.8e+01	38	2.6e-13	1.4e-11
		plsqr ($\varepsilon = 10^{-4}$)	4.1e+02	1.5e+01	10	2.2e-13	6.7e-11
256^2	249074	lsqr	-	1.7e+01	500	1.9e-03	1.8e-03
		plsqr ($\varepsilon = 10^{-2}$)	9.7e+02	1.3e+02	37	1.3e-11	1.2e-11
		plsqr ($\varepsilon = 10^{-4}$)	4.3e+03	1.0e+02	12	5.7e-11	5.5e-11
512^2	1122019	lsqr	-	6.5e+01	500	4.8e-04	4.6e-04
		plsqr ($\varepsilon = 10^{-2}$)	8.0e+03	4.2e+02	24	2.2e-12	2.1e-12
		plsqr ($\varepsilon = 10^{-4}$)	3.3e+04	4.2e+02	10	2.5e-12	2.4e-12

Table 5.4: Time cost, iteration number, relative residual and relative error of lsqr and plsqr on a polar grid. $M \approx 0.47N \log_4(N)$.

6 Conclusions and Future Directions

This paper presents a direct solver for the 2D type-II NUDFT based on the HSS matrix approximation $\widetilde{\boldsymbol{G}}$ of the transformed NUDFT matrix $\boldsymbol{G} = \boldsymbol{A}\boldsymbol{F}^{-1}$. The construction of the HSS approximation is based on the kernel matrix expression of $\boldsymbol{G}$ and the proxy surface technique, resulting in the complexity of $\mathcal{O}\big(M + N^{3/2}\log^3 N\big)$. The URV factorization is then applied to $\widetilde{\boldsymbol{G}}$, which also takes $\mathcal{O}\big(M + N^{3/2}\log^3 N\big)$ time. Finally, the solution can be obtained using the URV factorization of $\widetilde{\boldsymbol{G}}$ and two-dimensional FFT, which takes $\mathcal{O}\big(M + N\log^3 N\big)$ time. The proposed method can also be used as a preconditioner for iterative solvers. Numerical results show the efficiency and accuracy of our direct solver for the 2D type-II NUDFT with both random and polar grids. It also demonstrates the effectiveness of the proposed solver as a preconditioner for iterative solvers, especially when the grid is ill-conditioned.

There are several future directions. The first is to extend the proposed method to the 2D type-III NUDFT. This could be done by using the same idea proposed in [23]. The second is to give a rigorous discussion to the estimation of the numerical rank of the HSS blocks of $\boldsymbol{G}$. The discussion in Section 3.1 provides an intuition of the numerical rank of the HSS blocks. To achieve this we need a numerical rank version of Lemma 3.1, which is not straightforward. The last is about the extension to higher dimensions. Our method factorizes $\boldsymbol{A}$ approximately as a product of an HSS matrix $\widetilde{\boldsymbol{G}}$ and a fast transform $\boldsymbol{F}$. As we have shown intuitively and numerically, the HSS rank is $r = \mathcal{O}\big(\sqrt{N}\log N\big)$, which is quite large compared to the 1D case, where the HSS rank in 1D is $r = \mathcal{O}(\log N)$. Moreover, by the same discussion one can also prove that in 3D case the HSS rank will be $r = \mathcal{O}\big(N^{2/3}\log N\big)$, which is even larger. Consequently, the complexity of our algorithm in 3D case will be $\mathcal{O}\big(N^2\log^\alpha N\big)$ for some $\alpha > 0$, which is less efficient, especially when N is large. Therefore, it is interesting to investigate whether there are other data-sparse approximations of $\boldsymbol{G}$ with smaller rank, which can further reduce the complexity of the algorithm. For example, the *butterfly factorization* [8, 24] for Fourier integral operators may be a good candidate, which achieves a quasi-linear complexity $\mathcal{O}(N\log N)$ for the representation of the matrix when $M = N$.

References

[1] C. Anderson and M. D. Dahleh, *Rapid computation of the discrete Fourier transform*, SIAM Journal on Scientific Computing, 17 (1996), pp. 913–919.

[2] S. Bagchi and S. K. Mitra, *The nonuniform discrete Fourier transform and its applications in signal processing*, Springer US, 2012.

[3] A. H. Barnett, J. Magland, and L. af Klinteberg, *A parallel nonuniform fast fourier transform library based on an "exponential of semicircle" kernel*, SIAM Journal on Scientific Computing, 41 (2019), pp. C479–C504.

[4] B. Beckermann and A. Townsend, *On the singular values of matrices with displacement structure*, SIAM Journal on Matrix Analysis and Applications, 38 (2017), pp. 1227–1248.

[5] P. Benner, R.-C. Li, and N. Truhar, *On the ADI method for Sylvester equations*, J. Comput. Appl. Math., 233 (2009), pp. 1035–1045.

[6] S. Börm, L. Grasedyck, and W. Hackbusch, *Introduction to hierarchical matrices with applications*, Engineering Analysis with Boundary Elements, 27 (2003), pp. 405–422.

[7] M. Bourgeois, F. T. A. W. Wajer, D. van Ormondt, and D. Graveron-Demilly, *Reconstruction of MRI images from non-uniform sampling and its application to intrascan motion correction in functional MRI*, Birkhäuser Boston, Boston, MA, 2001, pp. 343–363.

[8] E. Candès, L. Demanet, and L. Ying, *A fast butterfly algorithm for the computation of Fourier integral operators*, Multiscale Model. Simul., 7 (2009), pp. 1727–1750.

[9] S. Chandrasekaran, M. Gu, and T. Pals, *A fast ULV decomposition solver for hierarchically semiseparable representations*, SIAM Journal on Matrix Analysis and Applications, 28 (2006), pp. 603–622.

[10] H. Cheng, Z. Gimbutas, P.-G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM Journal on Scientific Computing, 26 (2005), pp. 1389–1404.

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

[12] E. Corona, P.-G. Martinsson, and D. Zorin, *An $O(N)$ direct solver for integral equations on the plane*, Applied and Computational Harmonic Analysis, 38 (2015), pp. 284–317.

[13] A. Dutt and V. Rokhlin, *Fast Fourier transforms for nonequispaced data*, SIAM Journal on Scientific Computing, 14 (1993), pp. 1368–1393.

[14] ———, *Fast Fourier transforms for nonequispaced data, II*, Applied and Computational Harmonic Analysis, 2 (1995), pp. 85–100.

[15] M. Fenn, S. Kunis, and D. Potts, *On the computation of the polar FFT*, Appl. Comput. Harmon. Anal., 22 (2007), pp. 257–263.

[16] L. Greengard and J.-Y. Lee, *Accelerating the nonuniform fast Fourier transform*, SIAM Review, 46 (2004), pp. 443–454.

[17] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM Journal on Scientific Computing, 17 (1996), pp. 848–869.

[18] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108.

[19] S. Jiang and L. Greengard, *A dual-space multilevel kernel-splitting framework for discrete and continuous convolution*, Commun. Pure Appl. Math., 78 (2025), pp. 1086–1143.

[20] M. Kircheis and D. Potts, *Direct inversion of the nonequispaced fast Fourier transform*, Linear Algebra and its Applications, 575 (2019), pp. 106–140.

[21] ———, *Fast and direct inversion methods for the multivariate nonequispaced fast Fourier transform*, Front. Appl. Math. Stat., Volume 9 (2023).

[22] J.-Y. Lee and L. Greengard, *The type 3 nonuniform FFT and its applications*, Journal of Computational Physics, 206 (2005), pp. 1–5.

[23] Y. Li and J. Liu, *A superfast direct solver for type-III inverse nonuniform discrete Fourier transform*, arxiv, (2025).

[24] Y. Li, H. Yang, E. R. Martin, K. L. Ho, and L. Ying, *Butterfly factorization*, Multiscale Model. Simul., 13 (2015), pp. 714–732.

[25] P.-G. Martinsson, *Fast direct solvers for elliptic PDEs*, Fast direct solvers for elliptic partial differential equations, Society for Industrial Applied Mathematics, 2019.

[26] P.-G. Martinsson and V. Rokhlin, *A fast direct solver for boundary integral equations in two dimensions*, Journal of Computational Physics, 205 (2005), pp. 1–23.

[27] C. C. Paige and M. A. Saunders, *Lsqr: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Softw., 8 (1982), pp. 43–71.

[28] D. Potts, G. Steidl, and M. Tasche, *Fast Fourier transforms for nonequispaced data: A tutorial*, Birkhäuser Boston, Boston, MA, 2001, pp. 247–270.

[29] D. Ruiz-Antolín and A. Townsend, *A nonuniform fast Fourier transform based on low rank approximation*, SIAM Journal on Scientific Computing, 40 (2018), pp. A529–A547.

[30] H. Wilber, E. N. Epperly, and A. H. Barnett, *Superfast direct inversion of the nonuniform discrete fourier transform via hierarchically semiseparable least squares*, SIAM Journal on Scientific Computing, (2025), pp. A1702–A1732.

[31] Y. Xi, J. Xia, S. Cauley, and V. Balakrishnan, *Superfast and stable structured solvers for Toeplitz least squares via randomized sampling*, SIAM Journal on Matrix Analysis and Applications, 35 (2014), pp. 44–72.

[32] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numerical Linear Algebra with Applications, 17 (2010), pp. 953–976.