Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

SIAM J. SCI. COMPUT.
Vol. 48, No. 6, pp. A3615–A3637
© 2026 Society for Industrial and Applied Mathematics

A SUPERFAST DIRECT SOLVER FOR TYPE-III INVERSE NONUNIFORM DISCRETE FOURIER TRANSFORM*

YINGZHOU LI† AND JINGYU LIU‡

Abstract. Various fields in scientific computing require the use of the nonuniform discrete Fourier transform (NUDFT) and its inversion. In [H. Wilber, E. N. Epperly, and A. H. Barnett, *SIAM J. Sci. Comput.,* 47 (2025), pp. A1702–A1732], the authors proposed a direct inversion method for the type-II NUDFT, which achieves quasi-linear complexity by approximating the type-II NUDFT matrix as a product of a hierarchical semiseparable (HSS) matrix and a DFT matrix. In this article, we propose a novel superfast direct inversion method for type-III NUDFT. The proposed method approximates the type-III NUDFT matrix as a product of a type-II NUDFT matrix and an HSS matrix. This decomposition enables both the forward application and the backward inversion to be accomplished with quasi-linear complexity. The fast inversion can serve as a high-accuracy direct solver or as an efficient preconditioner. Additionally, we provide an error bound for the approximation when the sample points are independent and identically distributed uniform random variables and the frequencies are perturbed continuous integers. Numerical results are presented to verify the relevant theoretical properties and demonstrate the efficiency of the proposed methods.

Key words. nonuniform discrete Fourier transform, hierarchical semiseparable matrices

MSC codes. 65T50, 65F20, 65F55

DOI. 10.1137/25M1829000

1. Introduction.

In this paper, a superfast direct solver is proposed for the *nonuniform discrete Fourier transform* (NUDFT). In one dimension, the NUDFT *forward* problem has the following form:

$$f_j = \sum_{k=0}^{N-1} c_k \mathrm{e}^{-2\pi \mathrm{i} \omega_k x_j}, \quad 0 \le j \le M-1, \tag{1.1}$$

where $\{x_j\} \subset [0,1)$ are *sample points,* $\{\omega_k\} \subset [-1/2, N-1/2)$ are *frequencies,* $\{c_k\} \subset \mathbb{C}$ are *coefficients*, and $\{f_j\} \subset \mathbb{C}$ are *target values.* We assume throughout the paper that $M \ge N$. Since the forward problem is a matrix-vector multiplication, direct computation requires $\mathcal{O}(MN)$ operations. If $M = N$, the sample points are equispaced and the frequencies are continuous integers, i.e., $x_j = j/M$ and $\omega_k = k$, the NUDFT becomes uniform, and (1.1) can be computed using the *fast Fourier transform* (FFT) algorithm in $\mathcal{O}(N \log N)$ operations [10]. Unfortunately, the FFT cannot be directly applied when either the sample points or the frequencies are nonuniform. In these cases, the NUDFT problem can be categorized into three types according to the nonuniformity [12, 31]:

*Submitted to the journal's Numerical Algorithms for Scientific Computing section December 8, 2025; accepted for publication (in revised form) July 9, 2026; published electronically November 25, 2026.

https://doi.org/10.1137/25M1829000

Funding: This work was supported in part by the National Natural Science Foundation of China under grants 12271109 and 12526211; by the Shanghai Pilot Program for Basic Research-Fudan University under grant 21TQ1400100 (22TQ017); by the Scientific Research Innovation Capability Support Project for Young Faculty under grant SRICSPYF-ZY2025159; and by the Xuemin Institute of Advanced Studies, Fudan University.

†School of Mathematical Sciences, Fudan University, Shanghai 200433, China; Shanghai Key Laboratory for Contemporary Applied Mathematics, Fudan University, Shanghai, 200433, China (yingzhouli@fudan.edu.cn).

‡School of Mathematical Sciences, Fudan University, Shanghai 200433, China (jyliu22@m.fudan.edu.cn).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

- Type-I: Uniform sample points and nonuniform frequencies, i.e., $x_j = j/M$;
- Type-II: Nonuniform sample points and uniform frequencies, i.e., $\omega_k = k$;
- Type-III: Nonuniform sample points and nonuniform frequencies.

Various algorithms have been developed to address the NUDFT forward problem with a complexity of $\mathcal{O}(M + N + N \log N)$ while maintaining the desired accuracy. These algorithms are commonly referred to as the *nonuniform fast Fourier transform* (NUFFT). Some methods approximate the NUDFT matrix by the Hadamard product of the discrete Fourier transform (DFT) matrix and a low-rank matrix [1, 31], while others utilize convolution with a smooth, nearly locally supported function combined with the FFT on an oversampled grid [3, 11, 12, 14, 22, 29, 30]. It is worthwhile to point out that typically in the NUFFT algorithms, the linear term $\mathcal{O}(M + N)$ comes from the number of sample points and frequencies, while the quasi-linear term $\mathcal{O}(N \log N)$ is from the fact that the frequencies are distributed in an interval of length $\mathcal{O}(N)$. For more general problems such as the Fourier integral operator (FIO), *butterfly algorithms* have been designed for efficient forward computation [8, 24] at a complexity of $\mathcal{O}(N \log N)$ when $M = N$.

In this article, we are mainly interested in the *inverse* NUDFT problem.

PROBLEM 1.1 (Inverse NUDFT (INUDFT)). *Given sample points* $\{x_j\}$ *and frequencies* $\{\omega_k\}$*, determine the coefficients* $\{c_k\}$ *from target values* $\{f_j\}$*.*

Algebraically, this problem can be modeled as a linear least-squares problem associated with the NUDFT matrix $\boldsymbol{A}$, whose entries are given by $\boldsymbol{A}(j,k) = \mathrm{e}^{-2\pi \mathrm{i} x_j \omega_k}$. Unlike the uniform case, the pseudoinverse of the NUDFT matrix is not simply equal to its adjoint (up to a scaling factor), meaning that the fast inverse algorithm differs from that used for the forward problem.

Another alternative perspective on the inverse problem is to view it as a function approximation task. Specifically, given values $\{f(x_j)\}$ of a function $f\colon [0,1] \to \mathbb{C}$ at sample points $\{x_j\}$, we aim to determine the coefficients $\{c_k\}$ such that the expansion $\sum_{k=0}^{N-1} c_k \mathrm{e}^{-2\pi \mathrm{i} \omega_k x}$ provides a good approximation of f. According to the Kadec-1/4 theorem [36], when frequencies are perturbed integers satisfying $\sup_{k\in\mathbb{Z}} |\omega_k - k| \le \alpha$ with $0 \le \alpha < 1/4$, the nonharmonic Fourier modes $\{\mathrm{e}^{-2\pi \mathrm{i} \omega_k x}\}_{k\in\mathbb{Z}}$ form a Riesz basis of $L^2[0,1]$. Hence the INUDFT problem can be interpreted as finding the coefficients of a given function on some nonharmonic Fourier basis. For more applications of the type-III NUDFT such as the heat flow and the magnetic resonance imaging (MRI), we refer the reader to [22].

Typically, the INUDFT problem is solved using iterative methods [11, 13, 31] due to the efficiency of its forward computation. Direct methods have also been developed, as noted in [20, 21]. The authors design a direct solver for the type-II problem by finding an approximation of the form $\boldsymbol{A}\boldsymbol{D}^*\boldsymbol{F}^*\boldsymbol{S} \approx N\boldsymbol{I}_M$ where the sparse matrix $\boldsymbol{S}$ is the only matrix to be determined. However, these direct methods are for the type-I or type-II problems and cannot be easily extended to the type-III problem. Recently, in [32], the authors proposed a direct solver for the type-II INUDFT problem. A key observation they use is that type-II NUDFT matrix $\boldsymbol{A}$ is a Vandermonde matrix and thus satisfies a Sylvester matrix equation with a low-rank right-hand side. Consequently, the transformed matrix $\tilde{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{F}^{-1}$ can be compressed into a *hierarchically semiseparable* (HSS) matrix [35], where $\boldsymbol{F} \in \mathbb{C}^{N\times N}$ is the DFT matrix defined by $\boldsymbol{F}(j,k) = \mathrm{e}^{-2\pi \mathrm{i} jk/N}$. Their direct inversion method is achieved by applying the HSS least-squares solver [34] and the *inverse fast Fourier transform* (iFFT). The proposed method could be naturally extended to the type-I INUDFT problem but not to the type-III INUDFT problem, since the type-III NUDFT matrix

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

is no longer a Vandermonde matrix and no such Sylvester matrix equation can be found.

1.1. Contributions. The main purpose of this paper is to develop a direct solver for the type-III INUDFT problem. We only consider the case where the frequencies are the perturbed continuous integers. By projecting the frequencies onto the integers, the type-III NUDFT matrix is approximated as $\boldsymbol{A} \approx \boldsymbol{B}\boldsymbol{H}$, where $\boldsymbol{B} \in \mathbb{C}^{M\times N}$ is a type-II NUDFT matrix and $\boldsymbol{H} \in \mathbb{C}^{N\times N}$ is a dense matrix that can be compressed into an HSS matrix. We provide a theoretical error bound for this approximation under the assumption that the sample points are independent and identically distributed (i.i.d.) uniform random variables.

By applying relevant fast inversion algorithms to $\boldsymbol{B}$ and $\boldsymbol{H}$, the least-squares problem associated with $\boldsymbol{A}$ can be solved rapidly. To be more specific, the usage of this superfast direct solver can be divided into four stages: preliminary, construction, factorization, and solution. In the preliminary stage, we build a fast structure $\boldsymbol{B}_{\text{fast}}$ for the type-II NUDFT matrix $\boldsymbol{B}$. In the construction stage, the matrix $\boldsymbol{B}_{\text{fast}}^{\dagger}\boldsymbol{A}$ is compressed into an HSS matrix, denoted as $\boldsymbol{H}_{\text{HSS}}$. This compression is performed efficiently using an algorithm based on random sampling. In the factorization stage, the HSS matrix $\boldsymbol{H}_{\text{HSS}}$ undergoes the URV factorization [34]. Once the factorization stage finishes, we use $\boldsymbol{A}_{\text{fast}}^{\dagger} = \boldsymbol{H}_{\text{HSS}}^{-1}\boldsymbol{B}_{\text{fast}}^{\dagger}$ as a direct solver for the least-squares problem. Additionally, when considering the normal equation of (1.1), our solver can also serve as an efficient preconditioner in preconditioned conjugate gradient (PCG) methods. Numerical experiments are conducted to investigate the properties of the approximation and to demonstrate the efficiency of the proposed method.

1.2. Organization. The rest of this paper is organized as follows. In section 2, we review the related definitions and algorithms related to HSS matrices, as well as the type-II INUDFT solver [32], which serves as a preliminary for the subsequent sections. Section 3 introduces our novel method on the type-III NUDFT problem. We discuss first its derivation and related theoretical results, followed by the practical implementation of the algorithm. Numerical results are presented in section 4 to illustrate the efficiency of our methods. Finally, section 5 concludes the paper with a discussion on future directions.

2. Preliminaries.

2.1. HSS matrices. HSS matrices are a special class of $\mathcal{H}^2$-matrices [7, 15, 16] which have linear complexity for storage and numerical linear algebra operations, such as matrix-vector products and solving linear systems [35]. The definition of an HSS matrix relies on a tree structure (HSS tree), which is also included in the following definition.

DEFINITION 2.1 (HSS tree and HSS matrix [34]). *An $M \times N$ matrix $\boldsymbol{A}$ is called an HSS matrix with the associated HSS tree T if the following conditions hold:*

- *T is a full binary tree with root node ρ.*
- *There are two index sets $\mathcal{J}_\tau$ and $\mathcal{K}_\tau$ associated with each node τ of T, satisfying the following conditions:*
  1. *$\mathcal{J}_\rho = \{0, \ldots, M-1\}$, $\mathcal{K}_\rho = \{0, \ldots, N-1\}$.*
  2. *For a nonleaf node τ with children α_1 and α_2, we have $\mathcal{J}_\tau = \mathcal{J}_{\alpha_1} \sqcup \mathcal{J}_{\alpha_2}$ and $\mathcal{K}_\tau = \mathcal{K}_{\alpha_1} \sqcup \mathcal{K}_{\alpha_2}$, where the notation $\sqcup$ means the disjoint union.*
- *There are matrices $\boldsymbol{D}_\tau$ and $\boldsymbol{U}_\tau$, $\boldsymbol{V}_\tau$, $\boldsymbol{B}_{\tau,\sigma}$ called HSS generators associated with each node τ and its sibling σ, satisfying the following recursion:*
  1. *If τ is a leaf node, $\boldsymbol{D}_\tau = \boldsymbol{A}(\mathcal{J}_\tau, \mathcal{K}_\tau)$ is a dense matrix. If τ is a nonleaf node with children α_1 and α_2,*

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

$$\boldsymbol{D}_\tau = \boldsymbol{A}(\mathcal{J}_\tau, \mathcal{K}_\tau) = \begin{bmatrix} \boldsymbol{D}_{\alpha_1} & \boldsymbol{U}_{\alpha_1}^{\mathrm{big}} \boldsymbol{B}_{\alpha_1,\alpha_2} \boldsymbol{V}_{\alpha_2}^{\mathrm{big},*} \\ \boldsymbol{U}_{\alpha_2}^{\mathrm{big}} \boldsymbol{B}_{\alpha_2,\alpha_1} \boldsymbol{V}_{\alpha_1}^{\mathrm{big},*} & \boldsymbol{D}_{\alpha_2} \end{bmatrix},$$

where $\boldsymbol{V}_{\alpha_j}^{\mathrm{big},*}$ *means* $(\boldsymbol{V}_{\alpha_j}^{\mathrm{big}})^*$. *This notation will be used throughout the paper.*

(2) *If* τ *is a leaf node, we define* $\boldsymbol{U}_\tau^{\mathrm{big}} = \boldsymbol{U}_\tau$ *and* $\boldsymbol{V}_\tau^{\mathrm{big}} = \boldsymbol{V}_\tau$. *When* τ *is a nonleaf and nonroot node, the matrices* $\boldsymbol{U}_\tau$ *and* $\boldsymbol{V}_\tau$ *satisfy the following recursions:*

$$\boldsymbol{U}_\tau^{\mathrm{big}} = \begin{bmatrix} \boldsymbol{U}_{\alpha_1}^{\mathrm{big}} & \\ & \boldsymbol{U}_{\alpha_2}^{\mathrm{big}} \end{bmatrix} \boldsymbol{U}_\tau \quad \textit{and} \quad \boldsymbol{V}_\tau^{\mathrm{big}} = \begin{bmatrix} \boldsymbol{V}_{\alpha_1}^{\mathrm{big}} & \\ & \boldsymbol{V}_{\alpha_2}^{\mathrm{big}} \end{bmatrix} \boldsymbol{V}_\tau.$$

Given an HSS tree, we define the level of each node as follows: For the root node ρ, *we set* $\mathrm{level}(\rho) = 0$. *If* τ *is a nonleaf node and* α *is a child of* τ, *we set* $\mathrm{level}(\alpha) = \mathrm{level}(\tau) + 1$. *The maximum level of the tree is denoted by* L.

A key observation regarding HSS matrices is that for any nonroot node τ, the matrices $\boldsymbol{U}_\tau^{\mathrm{big}}$ and $\boldsymbol{V}_\tau^{\mathrm{big},*}$ are bases for the column or row spaces of the corresponding blocks (called the *HSS blocks*) $\boldsymbol{A}(\mathcal{J}_\tau, \mathcal{K}_\tau^{\mathrm{c}})$ and $\boldsymbol{A}(\mathcal{J}_\tau^{\mathrm{c}}, \mathcal{K}_\tau)$, respectively. Here $\mathcal{J}_\tau^{\mathrm{c}} = \mathcal{J}_\rho \backslash \mathcal{J}_\tau$ and $\mathcal{K}_\tau^{\mathrm{c}} = \mathcal{K}_\rho \backslash \mathcal{K}_\tau$. This is often called the *shared basis property*. Matrices $\boldsymbol{U}_\tau^{\mathrm{big}}$ and $\boldsymbol{V}_\tau^{\mathrm{big}}$ are called *basis matrices*. The *HSS rank* of $\boldsymbol{A}$ is defined as the maximum numerical rank among all the HSS blocks of $\boldsymbol{A}$. It is well known that the complexity for both storage and the matrix-vector product of an HSS matrix with HSS rank k is $\mathcal{O}\big(k(M+N)\big)$ [9]. When the context is clear, we may omit the index sets $\mathcal{J}_\tau$ and $\mathcal{K}_\tau$ in the subscript, using τ directly to represent the corresponding block. For instance, $\boldsymbol{A}_{\tau,\sigma}$ and $\boldsymbol{A}_{\tau,\sigma^{\mathrm{c}}}$ represent $\boldsymbol{A}(\mathcal{J}_\tau, \mathcal{K}_\sigma)$ and $\boldsymbol{A}(\mathcal{J}_\tau, \mathcal{K}_\sigma^{\mathrm{c}})$, respectively.

The hierarchical structure of HSS matrices is often represented in a different form known as the *telescoping factorization* [26, 27]. Specifically, for each level $1 \le \ell \le L$, we define the following block diagonal matrices:

$$\boldsymbol{U}^{(\ell)} = \mathrm{diag}\big(\boldsymbol{U}_\tau \colon \mathrm{level}(\tau) = \ell\big), \quad \boldsymbol{V}^{(\ell)} = \mathrm{diag}\big(\boldsymbol{V}_\tau \colon \mathrm{level}(\tau) = \ell\big). \tag{2.1}$$

For each level $0 \le \ell \le L-1$, we define

$$\boldsymbol{B}^{(\ell)} = \mathrm{diag}\big(\boldsymbol{B}_\tau \colon \mathrm{level}(\tau) = \ell\big) \quad \text{with} \quad \boldsymbol{B}_\tau = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{B}_{\alpha_1,\alpha_2} \\ \boldsymbol{B}_{\alpha_2,\alpha_1} & \boldsymbol{0} \end{bmatrix}. \tag{2.2}$$

Then the HSS matrix $\boldsymbol{A}$ can be recursively factorized as

$$\begin{aligned} \boldsymbol{A}^{(L)} &= \boldsymbol{U}^{(L)} \boldsymbol{A}^{(L-1)} \boldsymbol{V}^{(L),*} + \boldsymbol{D}^{(L)}, \\ \boldsymbol{A}^{(\ell)} &= \boldsymbol{U}^{(\ell)} \boldsymbol{A}^{(\ell-1)} \boldsymbol{V}^{(\ell),*} + \boldsymbol{B}^{(\ell)}, \quad 1 \le \ell \le L-1, \\ \boldsymbol{A}^{(0)} &= \boldsymbol{B}^{(0)}, \end{aligned} \tag{2.3}$$

where $\boldsymbol{A}^{(L)} = \boldsymbol{A}$ and $\boldsymbol{D}^{(L)} = \mathrm{diag}\big(\boldsymbol{D}_\tau \colon \mathrm{level}(\tau) = L\big)$.

2.2. A black-box construction algorithm for HSS matrices. In this section we review the randomized algorithm [23] for computing the HSS compression of a given matrix $\boldsymbol{A}$, under the limitation that interactions with the matrix can only be accessed through applying it and its conjugate transpose on vectors. This scenario is often referred to as the *black-box* setting [25]. The HSS compression of a matrix $\boldsymbol{A}$ is constructed from the information contained in $\{\boldsymbol{\Omega}, \boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}, \boldsymbol{\Psi}, \boldsymbol{Z} = \boldsymbol{A}^*\boldsymbol{\Psi}\}$, where $\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}$ are two random *test matrices* and $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are corresponding *sample matrices*. We emphasize that compared to the algorithm in [23], our algorithm needs a postprocessing step to enforce the structure of $\boldsymbol{B}_\tau$ as in (2.2), which makes it slightly different from the original one.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Suppose $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ is an HSS matrix with HSS rank k and leaf node size m. Let r be a number slightly larger than k, e.g., $r = k+5$, and let $\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}$ be $N \times s$ Gaussian test matrices, where the number of samples $s \geq \max(r+m, 3r)$. The construction can be divided into two steps: First, we conduct a bottom-up process to compute all basis matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. Some auxiliary matrices are also computed in this step. Next, a top-down process is employed to complete the construction by assigning matrices $\boldsymbol{B}$ and $\boldsymbol{D}$.

We begin by describing how to compute matrices $\boldsymbol{U}_\tau$ and $\boldsymbol{V}_\tau$ for a leaf node τ in level $\ell = L$. By considering the row block corresponding to τ in the equation $\boldsymbol{Y} = \boldsymbol{A\Omega}$, we have

$$\boldsymbol{Y}_\tau = \boldsymbol{D}_\tau \boldsymbol{\Omega}_\tau + \boldsymbol{A}_{\tau,\tau^c} \boldsymbol{\Omega}_{\tau^c}. \tag{2.4}$$

The key observation is that if there exists an orthonormal matrix $\boldsymbol{P}_\tau \in \mathbb{C}^{s\times r}$ that belongs to the null space of $\boldsymbol{\Omega}_\tau$, such that $\boldsymbol{\Omega}_\tau \boldsymbol{P}_\tau = \boldsymbol{0}$, multiplying both sides of (2.4) by $\boldsymbol{P}_\tau$ on the right yields

$$\boldsymbol{Y}_\tau \boldsymbol{P}_\tau = \boldsymbol{A}_{\tau,\tau^c} \boldsymbol{\Omega}_{\tau^c} \boldsymbol{P}_\tau. \tag{2.5}$$

Such a matrix $\boldsymbol{P}_\tau$ can be obtained by first performing a QR factorization with column pivoting (QRCP) on $\boldsymbol{\Omega}_\tau^*$, then taking the last r columns from the Q-factor. Note that in (2.5) the contribution from $\boldsymbol{A}_{\tau,\tau}$ is removed. Then $\boldsymbol{Y}_\tau \boldsymbol{P}_\tau$ can be viewed as a sampling of $\boldsymbol{A}_{\tau,\tau^c}$ with test matrix $\boldsymbol{\Omega}_{\tau^c} \boldsymbol{P}_\tau$. Consequently, the basis matrix $\boldsymbol{U}_\tau$ can be constructed by taking the first r columns from the Q-factor in the QRCP of $\boldsymbol{Y}_\tau \boldsymbol{P}_\tau$. A similar procedure using $\boldsymbol{\Psi}$ and $\boldsymbol{Z}$ yields the matrix $\boldsymbol{V}_\tau$.

With matrices $\boldsymbol{U}_\tau$ and $\boldsymbol{V}_\tau$ of τ in hand, the diagonal block can be decomposed into two parts:

$$\boldsymbol{D}_\tau = \boldsymbol{U}_\tau (\boldsymbol{U}_\tau^* \boldsymbol{D}_\tau \boldsymbol{V}_\tau) \boldsymbol{V}_\tau^* + (\boldsymbol{D}_\tau - \boldsymbol{U}_\tau \boldsymbol{U}_\tau^* \boldsymbol{D}_\tau \boldsymbol{V}_\tau \boldsymbol{V}_\tau^*) =: \boldsymbol{U}_\tau \hat{\boldsymbol{D}}_\tau \boldsymbol{V}_\tau^* + \check{\boldsymbol{D}}_\tau. \tag{2.6}$$

The first term represents the projection of $\boldsymbol{D}_\tau$ onto basis matrices, while the second term is the remaining part. We will address the computation of $\hat{\boldsymbol{D}}_\tau$ later in the top-down process and focus on construction of $\check{\boldsymbol{D}}_\tau$ first. By rewriting (2.6) as

$$\check{\boldsymbol{D}}_\tau = \boldsymbol{D}_\tau - \boldsymbol{U}_\tau \boldsymbol{U}_\tau^* \boldsymbol{D}_\tau \boldsymbol{V}_\tau \boldsymbol{V}_\tau^* = (\boldsymbol{I} - \boldsymbol{U}_\tau \boldsymbol{U}_\tau^*) \boldsymbol{D}_\tau + \boldsymbol{U}_\tau \boldsymbol{U}_\tau^* \boldsymbol{D}_\tau (\boldsymbol{I} - \boldsymbol{V}_\tau \boldsymbol{V}_\tau^*),$$

it suffices to find a way to obtain $(\boldsymbol{I} - \boldsymbol{U}_\tau \boldsymbol{U}_\tau^*) \boldsymbol{D}_\tau$ and $\boldsymbol{D}_\tau (\boldsymbol{I} - \boldsymbol{V}_\tau \boldsymbol{V}_\tau^*)$. Multiplying $(\boldsymbol{I} - \boldsymbol{U}_\tau \boldsymbol{U}_\tau^*)$ on the left of (2.4) and using the fact that $\boldsymbol{U}_\tau$ forms the column basis of $\boldsymbol{A}_{\tau,\tau^c}$, we obtain

$$(\boldsymbol{I} - \boldsymbol{U}_\tau \boldsymbol{U}_\tau^*) \boldsymbol{Y}_\tau = (\boldsymbol{I} - \boldsymbol{U}_\tau \boldsymbol{U}_\tau^*) \boldsymbol{D}_\tau \boldsymbol{\Omega}_\tau.$$

Therefore, $(\boldsymbol{I} - \boldsymbol{U}_\tau \boldsymbol{U}_\tau^*) \boldsymbol{D}_\tau = (\boldsymbol{I} - \boldsymbol{U}_\tau \boldsymbol{U}_\tau^*) \boldsymbol{Y}_\tau \boldsymbol{\Omega}_\tau^\dagger$ can be recovered through a least-squares solving. A similar process yields $\boldsymbol{D}_\tau (\boldsymbol{I} - \boldsymbol{V}_\tau \boldsymbol{V}_\tau^*) = ((\boldsymbol{I} - \boldsymbol{V}_\tau \boldsymbol{V}_\tau^*) \boldsymbol{Z}_\tau \boldsymbol{\Psi}_\tau^\dagger)^*$. Combining these results, $\check{\boldsymbol{D}}_\tau$ is obtained through

$$\check{\boldsymbol{D}}_\tau = (\boldsymbol{I} - \boldsymbol{U}_\tau \boldsymbol{U}_\tau^*) \boldsymbol{Y}_\tau \boldsymbol{\Omega}_\tau^\dagger + \boldsymbol{U}_\tau \boldsymbol{U}_\tau^* \Big((\boldsymbol{I} - \boldsymbol{V}_\tau \boldsymbol{V}_\tau^*) \boldsymbol{Z}_\tau \boldsymbol{\Psi}_\tau^\dagger \Big)^*.$$

Suppose $0 \leq \ell \leq L-1$ and we have computed the basis matrices for all nodes in level $\ell+1$. By extracting these basis matrices, we obtain the following recursive decomposition of $\boldsymbol{A}^{(\ell+1)}$ starting from $\boldsymbol{A}^{(L)} = \boldsymbol{A}$:

$$\boldsymbol{A}^{(\ell+1)} = \boldsymbol{U}^{(\ell+1)} \boldsymbol{A}^{(\ell)} \boldsymbol{V}^{(\ell+1),*} + \check{\boldsymbol{D}}^{(\ell+1)}, \tag{2.7}$$

where

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

$$\boldsymbol{U}^{(\ell+1)} = \operatorname{diag}\big(\boldsymbol{U}_\tau : \ \operatorname{level}(\tau) = \ell+1\big),$$
$$\boldsymbol{V}^{(\ell+1)} = \operatorname{diag}\big(\boldsymbol{V}_\tau : \ \operatorname{level}(\tau) = \ell+1\big),$$
$$\check{\boldsymbol{D}}^{(\ell+1)} = \operatorname{diag}\big(\check{\boldsymbol{D}}_\tau : \ \operatorname{level}(\tau) = \ell+1\big).$$

Therefore, $\boldsymbol{A}^{(\ell)} = \boldsymbol{U}^{(\ell+1),*}\boldsymbol{A}^{(\ell+1)}\boldsymbol{V}^{(\ell+1)}$. If we "eliminate" all nodes in level $\ell+1$, it can be verified that $\boldsymbol{A}^{(\ell)}$ remains an HSS matrix with maximum level ℓ. The only requirement is to update the generators at level ℓ. Specifically, for every node τ in level ℓ with children α_1 and α_2, τ becomes a leaf node of the new HSS tree, with corresponding generators updated by

$$\boldsymbol{D}_\tau \leftarrow \begin{bmatrix} \hat{\boldsymbol{D}}_{\alpha_1} & \boldsymbol{B}_{\alpha_1,\alpha_2} \\ \boldsymbol{B}_{\alpha_2,\alpha_1} & \hat{\boldsymbol{D}}_{\alpha_2} \end{bmatrix}. \tag{2.8}$$

Let $\boldsymbol{\Omega}^{(L)} = \boldsymbol{\Omega}$ and $\boldsymbol{Y}^{(L)} = \boldsymbol{Y}$. By multiplying (2.7) with $\boldsymbol{\Omega}^{(\ell+1)}$ and rearranging the equation, we obtain

$$\boldsymbol{U}^{(\ell+1),*}\big(\boldsymbol{Y}^{(\ell+1)} - \check{\boldsymbol{D}}^{(\ell+1)}\boldsymbol{\Omega}^{(\ell+1)}\big) = \boldsymbol{A}^{(\ell)}\big(\boldsymbol{V}^{(\ell+1),*}\boldsymbol{\Omega}^{(\ell+1)}\big).$$

This can be interpreted as a sampling of $\boldsymbol{A}^{(\ell)}$. More specifically, let τ be a node and α_1 and α_2 be its children. The matrices $\boldsymbol{\Omega}_\tau^{(\ell)}$ and $\boldsymbol{Y}_\tau^{(\ell)}$ are defined as

$$\boldsymbol{\Omega}_\tau^{(\ell)} = \begin{bmatrix} \boldsymbol{V}_{\alpha_1}^* \boldsymbol{\Omega}_{\alpha_1}^{(\ell+1)} \\ \boldsymbol{V}_{\alpha_2}^* \boldsymbol{\Omega}_{\alpha_2}^{(\ell+1)} \end{bmatrix} \quad \text{and} \quad \boldsymbol{Y}_\tau^{(\ell)} = \begin{bmatrix} \boldsymbol{U}_{\alpha_1}^* \big(\boldsymbol{Y}_{\alpha_1}^{(\ell+1)} - \check{\boldsymbol{D}}_{\alpha_1}\boldsymbol{\Omega}_{\alpha_1}^{(\ell+1)}\big) \\ \boldsymbol{U}_{\alpha_2}^* \big(\boldsymbol{Y}_{\alpha_2}^{(\ell+1)} - \check{\boldsymbol{D}}_{\alpha_2}\boldsymbol{\Omega}_{\alpha_2}^{(\ell+1)}\big) \end{bmatrix}.$$

Matrices $\boldsymbol{\Psi}_\tau^{(\ell)}$ and $\boldsymbol{Z}_\tau^{(\ell)}$ are defined analogously. By the previous discussions, we can compute $\check{\boldsymbol{D}}_\tau$, $\boldsymbol{U}_\tau$, and $\boldsymbol{V}_\tau$. This procedure is iteratively repeated until reaching the root node. If τ is the root node, we compute $\boldsymbol{D}_\tau = \boldsymbol{Y}_\tau \boldsymbol{\Omega}_\tau^\dagger$ directly.

Next, a top-down procedure is performed to construct matrices $\boldsymbol{B}$ and $\boldsymbol{D}$. By comparing the factorization (2.7) with (2.3), the only difference is that the matrix $\boldsymbol{B}_\tau$ in (2.2) has a structure in which the diagonal blocks corresponding to its children are zero. Therefore, the key idea here is to "push" the diagonal part into its children blocks until we reach the leaf nodes.

For a level $0 \le \ell \le L-1$, suppose τ is a nonleaf node on level ℓ with children α_1 and α_2. We define the matrices $\boldsymbol{D}_\tau^{(\ell)}$, $\boldsymbol{U}_\tau^{(\ell+1)}$, and $\boldsymbol{V}_\tau^{(\ell+1)}$ to be the submatrices of $\boldsymbol{D}^{(\ell)}$, $\boldsymbol{U}^{(\ell+1)}$, and $\boldsymbol{V}^{(\ell+1)}$ corresponding to the node τ, respectively, and let

$$\boldsymbol{D}_\tau^{(\ell)} = \begin{bmatrix} \boldsymbol{D}_{\tau;\alpha_1,\alpha_1} & \boldsymbol{D}_{\tau;\alpha_1,\alpha_2} \\ \boldsymbol{D}_{\tau;\alpha_2,\alpha_1} & \boldsymbol{D}_{\tau;\alpha_2,\alpha_2} \end{bmatrix}.$$

The corresponding block with respect to τ in the factorization (2.7) can be expressed as

$$\begin{aligned} \boldsymbol{A}_\tau^{(\ell+1)} &= \boldsymbol{U}_\tau^{(\ell+1)} \boldsymbol{A}_\tau^{(\ell)} \boldsymbol{V}_\tau^{(\ell+1),*} + \check{\boldsymbol{D}}_\tau^{(\ell+1)} \\ &= \boldsymbol{U}_\tau^{(\ell+1)} \left(\boldsymbol{U}_\tau^{(\ell)} \boldsymbol{A}_\tau^{(\ell-1)} \boldsymbol{V}_\tau^{(\ell),*} + \boldsymbol{D}_\tau^{(\ell)} \right) \boldsymbol{V}_\tau^{(\ell+1)*} + \check{\boldsymbol{D}}_\tau^{(\ell+1)} \\ &= \boldsymbol{U}_\tau^{(\ell+1)} \boldsymbol{U}_\tau^{(\ell)} \boldsymbol{A}_\tau^{(\ell-1)} \boldsymbol{V}_\tau^{(\ell),*} \boldsymbol{V}_\tau^{(\ell+1)*} + \boldsymbol{U}_\tau^{(\ell+1)} \boldsymbol{D}_\tau^{(\ell)} \boldsymbol{V}_\tau^{(\ell+1)*} + \check{\boldsymbol{D}}_\tau^{(\ell+1)}, \end{aligned}$$

where $\boldsymbol{A}_\tau^{(\ell-1)}$ has been updated. The second and third parts in the last equation can be merged together as

$$\begin{aligned} &\begin{bmatrix} \boldsymbol{U}_{\alpha_1} & \\ & \boldsymbol{U}_{\alpha_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{D}_{\tau;\alpha_1,\alpha_1} & \boldsymbol{D}_{\tau;\alpha_1,\alpha_2} \\ \boldsymbol{D}_{\tau;\alpha_2,\alpha_1} & \boldsymbol{D}_{\tau;\alpha_2,\alpha_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_{\alpha_1} & \\ & \boldsymbol{V}_{\alpha_2} \end{bmatrix}^* + \begin{bmatrix} \check{\boldsymbol{D}}_{\alpha_1} & \\ & \check{\boldsymbol{D}}_{\alpha_2} \end{bmatrix} \\ &\quad = \begin{bmatrix} \boldsymbol{U}_{\alpha_1} & \\ & \boldsymbol{U}_{\alpha_2} \end{bmatrix} \begin{bmatrix} & \boldsymbol{B}_{\alpha_1,\alpha_2} \\ \boldsymbol{B}_{\alpha_2,\alpha_1} & \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_{\alpha_1} & \\ & \boldsymbol{V}_{\alpha_2} \end{bmatrix}^* + \begin{bmatrix} \boldsymbol{D}_{\alpha_1} & \\ & \boldsymbol{D}_{\alpha_2} \end{bmatrix}, \end{aligned}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

where $\boldsymbol{B}_{\alpha_i,\alpha_j} = \boldsymbol{D}_{\tau;\alpha_i,\alpha_j}$ and $\boldsymbol{D}_\alpha = \check{\boldsymbol{D}}_\alpha + \boldsymbol{U}_\alpha \boldsymbol{D}_{\tau;\alpha,\alpha} \boldsymbol{V}_\alpha^*$. Note that the matrix $\boldsymbol{D}_{\tau;\alpha,\alpha}$ is actually the matrix $\hat{\boldsymbol{D}}_\alpha$ defined in (2.6). The process is repeated until reaching the leaf nodes. When τ is a leaf node, the wanted matrix $\boldsymbol{D}_\tau$ is obtained after the update, and the process is finished.

The algorithm requires $\mathscr{O}(k)$ sample points and has a complexity of $\mathscr{O}\big(k^2(M+N) + kT_{\text{mult}}\big)$, where T_{mult} is the complexity of applying $\boldsymbol{A}$ and $\boldsymbol{A}^*$ to a vector. Therefore, it is particularly suitable for problems where fast computations of matrix-vector products are available.

2.3. A superfast least-squares solver for HSS matrices. In this section, we review the HSS least-squares solver proposed in [32, 34]. The problem we consider is defined as $\min_{\boldsymbol{c}\in\mathbb{C}^N} \|\boldsymbol{A}\boldsymbol{c} - \boldsymbol{f}\|$ where $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ is an HSS matrix associated with an HSS tree whose maximum level is L. The key algorithm is the URV factorization, which can be viewed as a generalized QR factorization or the transpose version of the ULV factorization [9, 35]. Generally speaking, it factorizes an HSS matrix into a product of block diagonal unitary matrices and block upper-triangular matrices, where each component in the product is easy to invert. The computation of the factorization usually costs $\mathscr{O}\big(k^2(M+N)\big)$ operations, where k is the HSS rank, and once it is finished, $\boldsymbol{A}^\dagger$ can be efficiently applied on a vector with $\mathscr{O}\big(k(M+N)\big)$ operations.

We give a brief description of the algorithm. For convenience we assume that all HSS blocks of $\boldsymbol{A}$ have the exact rank k. The factorization begins from each leaf node. Suppose the numbers of rows and columns of a leaf node τ are m_τ and n_τ, respectively. If $m_\tau \gg n_\tau$, a *size reduction* step is taken to decrease the row size of the problem. This introduces zeros into $\boldsymbol{D}_\tau$ and $\boldsymbol{U}_\tau$ through a QR decomposition

$$\begin{bmatrix}\boldsymbol{U}_\tau & \boldsymbol{D}_\tau\end{bmatrix} = \boldsymbol{\Omega}_\tau \begin{bmatrix}\boldsymbol{R}_{1,1} & \boldsymbol{R}_{1,2}\\ \boldsymbol{0} & \boldsymbol{R}_{2,2}\\ \hline \boldsymbol{0} & \boldsymbol{0}\end{bmatrix} = \boldsymbol{\Omega}_\tau \begin{bmatrix}\check{\boldsymbol{U}}_\tau & \check{\boldsymbol{D}}_\tau\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix},$$

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 $\check{\boldsymbol{U}}_\tau$ and $\check{\boldsymbol{D}}_\tau$, is denoted by $\check{m}_\tau = n_\tau + k$. If no size reduction is adopted, we set $\boldsymbol{\Omega}_\tau$ to be empty and let $\check{\boldsymbol{U}}_\tau = \boldsymbol{U}_\tau$, $\check{\boldsymbol{D}}_\tau = \boldsymbol{D}_\tau$ and $\check{m}_\tau = m_\tau$.

The next is a basis elimination step. We introduce zeros into $\boldsymbol{V}_\tau \in \mathbb{C}^{n_\tau\times k}$ by a reverse QR factorization

$$\boldsymbol{V}_\tau = \boldsymbol{Q}_\tau \begin{bmatrix}\boldsymbol{0}\\ \hat{\boldsymbol{V}}_{\tau;2}\end{bmatrix}, \tag{2.9}$$

where $\boldsymbol{Q}_\tau \in \mathbb{C}^{n_\tau\times n_\tau}$ is unitary and $\hat{\boldsymbol{V}}_{\tau;2} \in \mathbb{C}^{k\times k}$. By the shared basis property of the HSS matrix, multiplying each leaf block τ by $\boldsymbol{Q}_\tau$ on the right zeroes out corresponding columns of the node τ. See Figure 1(a) for an illustration. The diagonal block is then modified by $\tilde{\boldsymbol{D}}_\tau = \check{\boldsymbol{D}}_\tau \boldsymbol{Q}_\tau$.

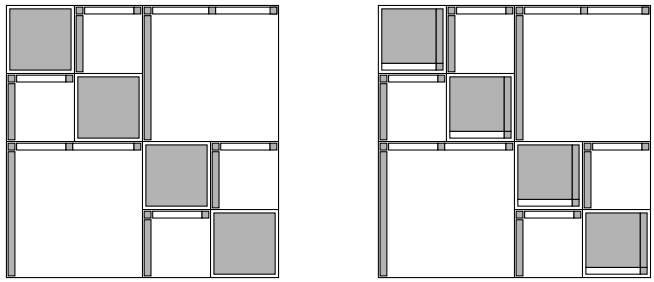
By partitioning the diagonal block as $\tilde{\boldsymbol{D}}_\tau = \left[\begin{smallmatrix}\tilde{\boldsymbol{D}}_{\tau;1,1} & \tilde{\boldsymbol{D}}_{\tau;1,2}\\ \tilde{\boldsymbol{D}}_{\tau;2,1} & \tilde{\boldsymbol{D}}_{\tau;2,2}\end{smallmatrix}\right]$, our goal is to eliminate the block $\tilde{\boldsymbol{D}}_{\tau;2,1}$ by a unitary transformation. This can be done by a QR factorization of the first column block:

$$\begin{bmatrix}\tilde{\boldsymbol{D}}_{\tau;1,1}\\ \tilde{\boldsymbol{D}}_{\tau;2,1}\end{bmatrix} = \boldsymbol{P}_\tau \begin{bmatrix}\hat{\boldsymbol{D}}_{\tau;1,1}\\ \boldsymbol{0}\end{bmatrix},$$

where $\boldsymbol{P}_\tau \in \mathbb{C}^{\check{m}_\tau\times\check{m}_\tau}$ is unitary. After multiplying $\boldsymbol{P}_\tau^*$ on the left, the diagonal block transforms into

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy



(a) (b)

FIG. 1. *An illustration of the URV factorization. Left: Introducing zeros into off-diagonal columns. Right: QR factorization on diagonal blocks.*

$$\hat{\boldsymbol{D}}_\tau = \boldsymbol{P}_\tau^* \tilde{\boldsymbol{D}}_\tau = \begin{bmatrix} \hat{\boldsymbol{D}}_{\tau;1,1} & \hat{\boldsymbol{D}}_{\tau;1,2} \\ \boldsymbol{0} & \hat{\boldsymbol{D}}_{\tau;2,2} \end{bmatrix}$$

and the basis matrix $\check{\boldsymbol{U}}_\tau$ is modified to

$$\boldsymbol{P}_\tau^* \check{\boldsymbol{U}}_\tau = \begin{bmatrix} \hat{\boldsymbol{U}}_{\tau;1} \\ \hat{\boldsymbol{U}}_{\tau;2} \end{bmatrix}.$$

Figure 1(b) illustrates the results after the elimination on diagonal blocks.

Now, it is remarkable that if we partition $\tau = \tau_1 \sqcup \tau_2$ conformably for all leaf nodes τ, the subproblem with respect to τ_2 is entirely decoupled from τ_1. Moreover, for a nonleaf node τ with children α_1 and α_2, by merging appropriate blocks as

$$\boldsymbol{D}_\tau^+ = \begin{bmatrix} \hat{\boldsymbol{D}}_{\alpha_1;2,2} & \hat{\boldsymbol{U}}_{\alpha_1;2} \boldsymbol{B}_{\alpha_1,\alpha_2} \hat{\boldsymbol{V}}_{\alpha_2;2}^* \\ \hat{\boldsymbol{U}}_{\alpha_2;2} \boldsymbol{B}_{\alpha_2,\alpha_1} \hat{\boldsymbol{V}}_{\alpha_1;2}^* & \hat{\boldsymbol{D}}_{\alpha_2;2,2} \end{bmatrix}, \tag{2.10}$$

$$\boldsymbol{U}_\tau^+ = \begin{bmatrix} \hat{\boldsymbol{U}}_{\alpha_1;2} & \\ & \hat{\boldsymbol{U}}_{\alpha_2;2} \end{bmatrix} \boldsymbol{U}_\tau, \quad \boldsymbol{V}_\tau^+ = \begin{bmatrix} \hat{\boldsymbol{V}}_{\alpha_1;2} & \\ & \hat{\boldsymbol{V}}_{\alpha_2;2} \end{bmatrix} \boldsymbol{V}_\tau,$$

then the children α_1 and α_2 can be "eliminated" and τ becomes a new leaf with the associated generators defined in (2.10). After that, the remaining matrix still has the HSS structure but with a maximum level $L-1$. This enables us to recursively factor the merged HSS matrix (with $\boldsymbol{D}_\tau$, $\boldsymbol{U}_\tau$, $\boldsymbol{V}_\tau$ replaced by $\boldsymbol{D}_\tau^+$, $\boldsymbol{U}_\tau^+$, $\boldsymbol{V}_\tau^+$) until the root node is the only remaining node. At that point, we only need to perform a QR decomposition $\boldsymbol{D}_\rho^+ = \boldsymbol{P}_\rho \hat{\boldsymbol{D}}_\rho$.

Once the URV factorization is computed, the solution process can be divided into two steps similar to the application of QR factorization. The first step applies unitary transforms to the right-hand side, and the second step uses backward substitution to solve a sequence of hierarchical triangular systems.

The first step involves a bottom-up traversal of the HSS tree. Starting from each leaf node τ, let $\boldsymbol{f}_\tau$ denote the corresponding block of the right-hand side. If size reduction is applied to τ, we define $\check{\boldsymbol{f}}_\tau = \boldsymbol{\Omega}_\tau^* \boldsymbol{f}_\tau = \left[\begin{smallmatrix} \check{\boldsymbol{f}}_{\tau;1} \\ \check{\boldsymbol{f}}_{\tau;2} \end{smallmatrix}\right]$ and $\hat{\boldsymbol{f}}_\tau = \boldsymbol{P}_\tau^* \check{\boldsymbol{f}}_{\tau;1} = \left[\begin{smallmatrix} \hat{\boldsymbol{f}}_{\tau;1} \\ \hat{\boldsymbol{f}}_{\tau;2} \end{smallmatrix}\right]$. If there is no size reduction step, the first equation is skipped and $\check{\boldsymbol{f}}_{\tau;1}$ is replaced by $\boldsymbol{f}_\tau$ in the second equation. Next, we merge the vectors on each nonleaf node τ with children α_1 and α_2 by $\boldsymbol{f}_\tau^+ = \left[\begin{smallmatrix} \hat{\boldsymbol{f}}_{\alpha_1;2} \\ \hat{\boldsymbol{f}}_{\alpha_2;2} \end{smallmatrix}\right]$. We continue applying the same procedure (with $\boldsymbol{f}_\tau$ replaced by $\boldsymbol{f}_\tau^+$) until τ is the root node.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

The second step is a top-down procedure. Beginning at the root ρ, we solve the triangular system $\hat{\boldsymbol{D}}_\rho \boldsymbol{c}_\rho = \hat{\boldsymbol{f}}_\rho$. For its children α_1 and α_2, we partition the solution as $\boldsymbol{c}_\rho = [\begin{smallmatrix}\hat{\boldsymbol{c}}_{\alpha_1;2}\\ \hat{\boldsymbol{c}}_{\alpha_2;2}\end{smallmatrix}]$. We then compute $\hat{\boldsymbol{g}}_{\alpha_1} = \boldsymbol{B}_{\alpha_1,\alpha_2}\hat{\boldsymbol{V}}^*_{\alpha_2;2}\hat{\boldsymbol{c}}_{\alpha_2;2}$ and $\hat{\boldsymbol{g}}_{\alpha_2} = \boldsymbol{B}_{\alpha_2,\alpha_1}\hat{\boldsymbol{V}}^*_{\alpha_1;2}\hat{\boldsymbol{c}}_{\alpha_1;2}$. For a nonroot node τ, we obtain $\hat{\boldsymbol{c}}_{\tau;1}$ by solving the triangular system $\hat{\boldsymbol{D}}_{\tau;1,1}\hat{\boldsymbol{c}}_{\tau;1} = \hat{\boldsymbol{f}}_{\tau;1} - \hat{\boldsymbol{D}}_{\tau;1,2}\hat{\boldsymbol{c}}_{\tau;2} - \hat{\boldsymbol{U}}_{\tau;1}\hat{\boldsymbol{g}}_\tau$ and update $\boldsymbol{c}_\tau = \boldsymbol{Q}_\tau[\begin{smallmatrix}\hat{\boldsymbol{c}}_{\tau;1}\\ \hat{\boldsymbol{c}}_{\tau;2}\end{smallmatrix}]$. If τ is a nonleaf node with children α_1 and α_2, we partition $\boldsymbol{c}_\tau$ conformably as $\boldsymbol{c}_\tau = [\begin{smallmatrix}\hat{\boldsymbol{c}}_{\alpha_1;2}\\ \hat{\boldsymbol{c}}_{\alpha_2;2}\end{smallmatrix}]$ and compute $\hat{\boldsymbol{g}}_{\alpha_1} = \boldsymbol{B}_{\alpha_1,\alpha_2}\hat{\boldsymbol{V}}^*_{\alpha_2;2}\hat{\boldsymbol{c}}_{\alpha_2;2} + \boldsymbol{R}_{\alpha_1}\hat{\boldsymbol{g}}_\tau$ and $\hat{\boldsymbol{g}}_{\alpha_2} = \boldsymbol{B}_{\alpha_2,\alpha_1}\hat{\boldsymbol{V}}^*_{\alpha_1;2}\hat{\boldsymbol{c}}_{\alpha_1;2} + \boldsymbol{R}_{\alpha_2}\hat{\boldsymbol{g}}_\tau$, where $\boldsymbol{U}_\tau = [\begin{smallmatrix}\boldsymbol{R}_{\alpha_1}\\ \boldsymbol{R}_{\alpha_2}\end{smallmatrix}]$. Repeating this process until we reach the leaf nodes, the solution $\boldsymbol{c}$ is obtained by collecting vectors $\boldsymbol{c}_\tau$ for all leaf nodes τ.

2.4. A superfast direct solver of type-II INUDFT problem. The type-II NUDFT matrix $\boldsymbol{A}$, defined by $\boldsymbol{A}(j,k) = \mathrm{e}^{-2\pi \mathrm{i} x_j k} = \gamma_j^k$ with $\gamma_j = \mathrm{e}^{-2\pi \mathrm{i} x_j}$ on the unit circle $\mathbb{S}$, is a Vandermonde matrix and satisfies the Sylvester equation

$$\boldsymbol{\Gamma}\boldsymbol{A} - \boldsymbol{A}\boldsymbol{C} = \boldsymbol{u}\boldsymbol{e}_{N-1}^\top, \tag{2.11}$$

where $\boldsymbol{\Gamma} = \mathrm{diag}(\gamma_0, \gamma_1, \ldots, \gamma_{M-1}) \in \mathbb{C}^{M\times M}$ is diagonal, $\boldsymbol{C} = [\begin{smallmatrix}\boldsymbol{0} & 1\\ \boldsymbol{I}_{N-1} & \boldsymbol{0}\end{smallmatrix}] \in \mathbb{C}^{N\times N}$ is the shift-down circulant matrix, and $u_j = \gamma_j^N - 1$. Here $\boldsymbol{e}_{N-1}$ is the canonical unit vector in $\mathbb{C}^N$. The circulant matrix $\boldsymbol{C}$ can be diagonalized using the DFT matrix $\boldsymbol{F}$, expressed as $\boldsymbol{C} = \boldsymbol{F}^{-1}\boldsymbol{D}\boldsymbol{F}$, where $\boldsymbol{D} = \mathrm{diag}(\xi_0, \xi_1, \ldots, \xi_{N-1})$ is a diagonal matrix with $\xi_k = \zeta_N^k$ for $k = 0, 1, \ldots, N-1$. Here $\zeta_N = \mathrm{e}^{-2\pi \mathrm{i}/N}$ is the Nth root of unity. By multiplying $\boldsymbol{F}^{-1}$ to the right of (2.11) and denoting $\tilde{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{F}^{-1}$, the equation becomes

$$\boldsymbol{\Gamma}\tilde{\boldsymbol{A}} - \tilde{\boldsymbol{A}}\boldsymbol{D} = \boldsymbol{u}\boldsymbol{v}^\top, \tag{2.12}$$

where the entries of $\boldsymbol{v}$ are given by $v_k = \zeta_N^k/N = \xi_k/N$ for $k = 0, 1, \ldots, N-1$.

In [32], the authors utilize the displacement structure (2.12) to establish the low-rank property of $\tilde{\boldsymbol{A}}$ and provide an estimate of the numerical rank [5]. Specifically, let the ε-rank of a matrix $\boldsymbol{M}$ be defined as $\mathrm{rank}_\varepsilon(\boldsymbol{M}) := \min\{k : \|\boldsymbol{X} - \boldsymbol{M}\|_2 \le \varepsilon\|\boldsymbol{M}\|_2, \mathrm{rank}(\boldsymbol{X}) = k\}$. It is shown that the ε-rank of the HSS blocks of $\tilde{\boldsymbol{A}}$ admits an upper bound of $k = \mathscr{O}(\log(1/\varepsilon)\log N)$ (Theorem 3.2 in [32]). See [4] for a more general result for the Toeplitz-like matrices. Additionally, by exploiting the displacement structure, the HSS matrix approximation can be constructed using the *factorized alternating direction implicit* (fADI) method [6, 33], resulting in an overall complexity of $\mathscr{O}\big(k^2(M+N)\big)$.

Here we present an alternative perspective on the low-rank property. Direct calculations show that the entries of $\tilde{\boldsymbol{A}}$ are explicitly given by

$$\tilde{\boldsymbol{A}}(j,k) = \frac{u_j v_k}{\gamma_j - \xi_k} = \frac{1}{N}\frac{(\gamma_j^N - 1)\xi_k}{\gamma_j - \xi_k}. \tag{2.13}$$

Consequently, $\tilde{\boldsymbol{A}}$ is a kernel matrix corresponding to kernel function

$$K(z,w) = \frac{1}{N}\frac{(z^N - 1)w}{z - w}$$

evaluated at the points $\{\gamma_j\}$ and $\{\xi_k\}$ for rows and columns. Using Taylor expansion or pole expansion, one can show that for well-separated z and w, the ε-rank of the kernel matrix is bounded by $\mathscr{O}(\log(1/\varepsilon))$, independent of size [17]. Then using the technique in [18], by recursively applying the well-separated block rank bound to the HSS blocks, we can show that the ε-rank of the HSS blocks is bounded by $\mathscr{O}(\log(1/\varepsilon)\log N)$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Algorithm 2.1. A direct solver for type-II INUDFT problem.

Require: Sample points $\{x_j\}_{j=0}^{M-1}$, number of frequencies N, target values $\{f_j\}_{j=0}^{M-1}$, accuracy parameter $\varepsilon \in (0,1)$.
Ensure: Coefficients $\{c_k\}_{k=0}^{N-1}$.
1: Approximate $\tilde{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{F}^{-1}$ by an HSS matrix $\tilde{\boldsymbol{A}}_{\text{HSS}}$ using the fADI-based construction with accuracy parameter ε.
2: Compute the URV factorization of $\tilde{\boldsymbol{A}}_{\text{HSS}}$.
3: Solve $\boldsymbol{v} = \arg\min_{\boldsymbol{w}} \|\tilde{\boldsymbol{A}}_{\text{HSS}}\boldsymbol{w} - \boldsymbol{f}\|_2$ by the HSS least-squares solver.
4: Compute $\boldsymbol{c} = \boldsymbol{F}^{-1}\boldsymbol{v}$ using the iFFT.

See the discussion in section 4 in [18] for more details. Consequently, $\tilde{\boldsymbol{A}}$ can be approximated by an HSS matrix whose HSS rank is bounded by $k = \mathscr{O}(\log(1/\varepsilon)\log N)$. This is consistent with the discussion in [32].

To conclude, the type-II NUDFT matrix $\boldsymbol{A}$ is approximated by $\boldsymbol{A}_{\text{fast}} := \tilde{\boldsymbol{A}}_{\text{HSS}}\boldsymbol{F}$, where $\tilde{\boldsymbol{A}}_{\text{HSS}}$ is the HSS approximation of $\tilde{\boldsymbol{A}}$. The type-II direct solver is then obtained by applying the least-squares solver in section 2.3 and the iFFT. We summarize the whole procedure in Algorithm 2.1. The algorithm consists of three stages: construction, factorization, and solution. In the construction and factorization stages, the HSS compression of $\tilde{\boldsymbol{A}}$ is constructed by the fADI-based method and factorized by the URV factorization, respectively. Once the factorization is finished, the solution $\boldsymbol{c}$ can be obtained using the solution process of the URV factorization and the iFFT. The complexity of the construction and factorization is $\mathscr{O}\big(k^2(M+N)\big)$, while the solution stage has a complexity of $\mathscr{O}\big(kM + N\log N\big)$, where k is the HSS rank of $\tilde{\boldsymbol{A}}_{\text{HSS}}$. Using the bound $k = \mathscr{O}(\log N)$ and $M \geq N$, the type-II solver has the complexity of $\mathscr{O}\big((M+N)\log^2 N\big) = \mathscr{O}\big(M\log^2 N\big)$ for construction and factorization and $\mathscr{O}\big((M+N)\log N\big) = \mathscr{O}(M\log N)$ for solution.

3. A superfast direct solver of the type-III INUDFT problem.

In this section, we propose a superfast direct solver for the type-III INUDFT problem. We first give some intuitions and motivations in section 3.1, and then present the detailed algorithm and its implementation in section 3.2.

3.1. Motivations.

Our motivation comes from the expansion of the nonharmonic Fourier basis in terms of the Fourier basis. The square integrable functions on $[0,1]$ form a Hilbert space $L^2[0,1]$ with the inner product defined by $\langle f, g\rangle := \int_0^1 f(x)\overline{g(x)}\mathrm{d}x$. A typically used orthonormal basis of $L^2[0,1]$ is given by $\{\mathrm{e}^{-2\pi\mathrm{i}\ell x}\}_{\ell\in\mathbb{Z}}$. We begin by expanding the function $\mathrm{e}^{-2\pi\mathrm{i}\omega_k x}$ in terms of this basis:

$$\mathrm{e}^{-2\pi\mathrm{i}\omega_k x} = \sum_{\ell\in\mathbb{Z}} \big\langle \mathrm{e}^{-2\pi\mathrm{i}\omega_k x}, \mathrm{e}^{-2\pi\mathrm{i}\ell x}\big\rangle\, \mathrm{e}^{-2\pi\mathrm{i}\ell x} = \sum_{\ell\in\mathbb{Z}} Q_{\ell,k}\mathrm{e}^{-2\pi\mathrm{i}\ell x}, \tag{3.1}$$

where the coefficients are given by

$$Q_{\ell,k} = \int_0^1 \mathrm{e}^{2\pi\mathrm{i}(\ell-\omega_k)x}\mathrm{d}x = \mathrm{e}^{\pi\mathrm{i}(\ell-\omega_k)}\frac{\sin\big(\pi(\ell-\omega_k)\big)}{\pi(\ell-\omega_k)} = G(\ell-\omega_k). \tag{3.2}$$

The assembled matrix $\boldsymbol{Q}$ can be viewed as a kernel matrix with the kernel function defined by

$$G(\omega) := \mathrm{e}^{\pi\mathrm{i}\omega}\frac{\sin(\pi\omega)}{\pi\omega}. \tag{3.3}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Since the kernel G is smooth and not highly oscillatory, the matrix $\boldsymbol{Q}$ is expected to have a low-rank property for its off-diagonal blocks. Suppose N is even and $\sigma \geq 1$ is some integer, and consider the index set $\mathcal{I}_{N;\sigma} = \{\ell \in \mathbb{Z} : -(\sigma N - N)/2 \leq \ell \leq (\sigma N + N)/2 - 1\}$, which is an expansion of $\{0, 1, \ldots, N-1\}$ about $N/2$ by a factor of σ. Truncating the series in (3.1) leads to

$$\mathrm{e}^{-2\pi \mathrm{i}\omega_k x} = \sum_{\ell \in \mathbb{Z}} Q_{\ell,k} \mathrm{e}^{-2\pi \mathrm{i}\ell x} = \sum_{\ell \in \mathcal{I}_{N;\sigma}} Q_{\ell,k} \mathrm{e}^{-2\pi \mathrm{i}\ell x} + \sum_{\ell \notin \mathcal{I}_{N;\sigma}} Q_{\ell,k} \mathrm{e}^{-2\pi \mathrm{i}\ell x},$$

where the first and second terms in the right-hand side serve as the approximation and the error, respectively. The L^2 norm of the error, denoted as $\mathrm{err}_k(x)$, is given by Parseval's identity:

$$\|\mathrm{err}_k(\cdot)\|_{L^2}^2 = \sum_{\ell \notin \mathcal{I}_{N;\sigma}} |Q_{\ell,k}|^2 = \sum_{\ell \notin \mathcal{I}_{N;\sigma}} \frac{\sin^2\big(\pi(\ell - \omega_k)\big)}{\pi^2(\ell - \omega_k)^2}.$$

Assume $\{\omega_k = k + \delta_k\}$ is a perturbation of the continuous integers, where $|\delta_k| \leq \alpha$ for some constant $\alpha \in [0, 1/2)$. Then the error can be estimated as

$$\begin{aligned}
\|\mathrm{err}_k(\cdot)\|_{L^2}^2 &= \sum_{\ell=-\infty}^{-(\sigma N-N)/2-1} \frac{\sin^2\big(\pi(\ell - \omega_k)\big)}{\pi^2(\ell - \omega_k)^2} + \sum_{\ell=(\sigma N+N)/2}^{\infty} \frac{\sin^2\big(\pi(\ell - \omega_k)\big)}{\pi^2(\ell - \omega_k)^2} \\
&= \sum_{\ell=-\infty}^{-(\sigma N-N)/2-1} \frac{\sin^2(\pi\delta_k)}{\pi^2(\ell - \omega_k)^2} + \sum_{\ell=(\sigma N+N)/2}^{\infty} \frac{\sin^2(\pi\delta_k)}{\pi^2(\ell - \omega_k)^2} \\
&\leq 2 \sum_{s=(\sigma-1)N/2}^{\infty} \frac{\sin^2(\pi\alpha)}{\pi^2(s + 1 - \alpha)^2}.
\end{aligned}$$

By the integral estimate of the tail and noting that the integrand is a convex function, we have

$$\begin{aligned}
\sum_{s=S}^{\infty} \frac{1}{(s+1-\alpha)^2} &\leq \sum_{s=S}^{\infty} \int_{s-1/2}^{s+1/2} \frac{1}{(x+1-\alpha)^2} \mathrm{d}x \\
&\leq \int_{S-1/2}^{\infty} \frac{1}{(x+1-\alpha)^2} \mathrm{d}x = \frac{1}{S + 1/2 - \alpha}.
\end{aligned}$$

Therefore,

$$\|\mathrm{err}_k(\cdot)\|_{L^2}^2 \leq \frac{2\sin^2(\pi\alpha)}{\pi^2((\sigma-1)N/2 + 1/2 - \alpha)}.$$

For our type-III NUDFT matrix, each entry can be approximated by

$$\boldsymbol{A}(j,k) = \mathrm{e}^{-2\pi \mathrm{i} x_j \omega_k} = \sum_{\ell \in \mathcal{I}_{N;\sigma}} \mathrm{e}^{-2\pi \mathrm{i} x_j \ell} Q_{\ell,k} + \sum_{\ell \notin \mathcal{I}_{N;\sigma}} \mathrm{e}^{-2\pi \mathrm{i} x_j \ell} Q_{\ell,k}.$$

The approximation can be reformulated in matrix form as

$$\boldsymbol{A} = \boldsymbol{B}\boldsymbol{Q} + \boldsymbol{E}, \tag{3.4}$$

where $\boldsymbol{B} = (\mathrm{e}^{-2\pi \mathrm{i} x_j \ell})_{j,\ell}$, $\boldsymbol{Q} = (Q_{\ell,k})_{\ell,k}$, and $\boldsymbol{E} = (\sum_{\ell \notin \mathcal{I}_{N;\sigma}} Q_{\ell,k} \mathrm{e}^{-2\pi \mathrm{i}\ell x_j})_{j,k}$ are the corresponding type-II NUDFT matrix, coefficient matrix, and error matrix, respectively. To estimate the approximation error, we assume that the sample points $\{x_j\}$ are i.i.d. uniform random variables in $[0, 1)$. The statistical properties of the random matrix $\boldsymbol{E}$ can be obtained through direct calculations. Precisely, we have $\mathbb{E}\boldsymbol{E}(j,k) = \sum_{\ell \notin \mathcal{I}_{N;\sigma}} Q_{\ell,k} \int_0^1 \mathrm{e}^{-2\pi \mathrm{i}\ell x} \mathrm{d}x = 0$ and

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Algorithm 3.1. A direct solver for type-III INUDFT problem.

Require: Sample points $\{x_j\}_{j=0}^{M-1}$, frequencies $\{\omega_k\}_{k=0}^{N-1}$, target values $\{f_j\}_{j=0}^{M-1}$, accuracy parameter $\varepsilon \in (0,1)$, and rank parameter k.

Ensure: Coefficients $\{c_k\}_{k=0}^{N-1}$.

1: Construct a type-II direct solver $\boldsymbol{B} \approx \boldsymbol{B}_{\text{fast}} = \tilde{\boldsymbol{B}}_{\text{HSS}}\boldsymbol{F}$ by Algorithm 2.1 for the type-II NUDFT matrix $\boldsymbol{B}(j,k) = \mathrm{e}^{-2\pi \mathrm{i} x_j k}$ with accuracy parameter ε.

2: Set $r = k + p$ ($p = 5$ or $p = 10$) and $s \geq \max(r + m, 3r)$, where m is the max leaf size in the HSS tree corresponding to $\boldsymbol{H}_{\text{HSS}}$.

3: Generate $N \times s$ Gaussian random matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}$.

4: Compute $\boldsymbol{Y} = \boldsymbol{B}_{\text{fast}}^{\dagger}\boldsymbol{A}\boldsymbol{\Omega}$ and $\boldsymbol{Z} = \boldsymbol{A}^{*}\boldsymbol{B}_{\text{fast}}^{\dagger,*}\boldsymbol{\Psi}$ using the type-II direct solver and NUFFT.

5: Compress $\boldsymbol{B}_{\text{fast}}^{\dagger}\boldsymbol{A}$ into an HSS matrix $\boldsymbol{H}_{\text{HSS}}$ using $\{\boldsymbol{\Omega}, \boldsymbol{Y}, \boldsymbol{\Psi}, \boldsymbol{Z}\}$.

6: Compute the URV factorization of $\boldsymbol{H}_{\text{HSS}}$.

7: Calculate $\boldsymbol{c} = \boldsymbol{H}_{\text{HSS}}^{-1}\boldsymbol{B}_{\text{fast}}^{\dagger}\boldsymbol{f}$.

$$\begin{aligned}
\mathbb{E}\|\boldsymbol{E}\|_{\text{F}}^2 &= \mathbb{E}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}\left|\sum_{\ell\notin\mathcal{I}_{N;\sigma}} Q_{\ell,k}\mathrm{e}^{-2\pi\mathrm{i}\ell x_j}\right|^2 = M\int_0^1 \sum_{k=0}^{N-1}\left|\sum_{\ell\notin\mathcal{I}_{N,\sigma}} Q_{\ell,k}\mathrm{e}^{2\pi\mathrm{i}\ell x}\right|^2 \mathrm{d}x \\
&= M\sum_{k=0}^{N-1}\sum_{\ell\notin\mathcal{I}_{N;\sigma}}\left|Q_{\ell,k}\right|^2 \leq M\sum_{k=0}^{N-1}\|\mathrm{err}_k(\cdot)\|_{L^2}^2 \\
&\leq MN\frac{2\sin^2(\pi\alpha)}{\pi^2((\sigma-1)N/2+1/2-\alpha)} = \frac{2\sin^2(\pi\alpha)}{\pi^2((\sigma-1)N/2+1/2-\alpha)}\|\boldsymbol{A}\|_{\text{F}}^2.
\end{aligned}$$

We summarize the results in the following proposition by noting that $\|\boldsymbol{A}\|_{\text{F}}^2 = MN$.

PROPOSITION 3.1. *Suppose $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ is the type-III NUDFT matrix defined as $\boldsymbol{A}(j,k) = \mathrm{e}^{-2\pi\mathrm{i}x_j\omega_k}$, where x_j are i.i.d. uniform random variables in $[0,1)$ and $\omega_k = k + \delta_k$ are perturbations of the continuous integers with $|\delta_k| \leq \alpha$ for some constant $\alpha \in [0, 1/2)$. Let $\sigma \geq 1$, and let $\boldsymbol{B}$ and $\boldsymbol{Q}$ be the matrices defined in* (3.4). *Then the error matrix $\boldsymbol{E}$ satisfies $\mathbb{E}\boldsymbol{E} = \boldsymbol{0}$ and $\mathbb{E}\|\boldsymbol{E}\|_{\text{F}}^2 \leq \frac{2\sin^2(\pi\alpha)}{\pi^2((\sigma-1)N/2+1/2-\alpha)}\|\boldsymbol{A}\|_{\text{F}}^2$.*

3.2. Algorithm. While the approximation derived from (3.4) is straightforward, it is not the form used in our practical algorithm. One main reason is that the error of this approximation can be quite large. Our strategy is to replace the matrix $\boldsymbol{Q}$ by the best candidate in the approximation $\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{H}\|_{\text{F}}$, i.e., $\boldsymbol{H} = \boldsymbol{B}^{\dagger}\boldsymbol{A}$. We will later demonstrate that this leads to a reprojection of the error matrix $\boldsymbol{E}$ against the type-II NUDFT matrix $\boldsymbol{B}$. In this section, we describe our practical superfast direct solver for the type-III INUDFT problem. The overall algorithm consists of four stages: preliminary, construction, factorization, and solution. We will discuss each stage in detail in the rest of this section, and the overall pseudocode can be found in Algorithm 3.1.

Suppose the type-III NUDFT matrix $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ is defined by $\boldsymbol{A}(j,k) = \mathrm{e}^{-2\pi\mathrm{i}x_j\omega_k}$ for $x_j \in [0,1)$ and $\omega_k \in [-1/2, N-1/2)$. Let $\boldsymbol{B}$ be the type-II NUDFT matrix defined by $\boldsymbol{B}(j,\ell) = \mathrm{e}^{-2\pi\mathrm{i}x_j\ell}$ where $\ell \in \mathcal{I}_{N;\sigma}$ is the index set defined in section 3.1. Since $\boldsymbol{B}$ is a Vandermonde matrix with distinct nodes, it has full column rank and $\boldsymbol{B}^{\dagger}\boldsymbol{B} = \boldsymbol{I}$. Multiplying $\boldsymbol{B}^{\dagger}$ on the left of (3.4) and denoting $\boldsymbol{H} = \boldsymbol{B}^{\dagger}\boldsymbol{A}$ gives

$$\boldsymbol{H} = \boldsymbol{Q} + \boldsymbol{B}^{\dagger}\boldsymbol{E}. \tag{3.5}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Moreover, by multiplying $\boldsymbol{B}$ on the left and rearranging the equation, we have

$$\boldsymbol{A} = \boldsymbol{B}\boldsymbol{H} + \left(\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^{\dagger}\right)\boldsymbol{E}. \tag{3.6}$$

The two equations (3.5) and (3.6) serve as the foundation of our algorithm. From the discussion in section 3.1, $\boldsymbol{Q}$ can be well approximated by an HSS matrix. Consequently, the matrix $\boldsymbol{H}$ in (3.5), viewed as a perturbation of $\boldsymbol{Q}$, can also be empirically compressed into an HSS matrix. On the other hand, from (3.6), $\boldsymbol{B}\boldsymbol{H}$ can be regarded as an approximation of $\boldsymbol{A}$. Noting that $\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^{\dagger}$ is a projection matrix, the following results are direct consequences based on the discussion in section 3.1:

$$\mathbb{E}\|\left(\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^{\dagger}\right)\boldsymbol{E}\|_{\mathrm{F}}^2 \leq \mathbb{E}\|\boldsymbol{E}\|_{\mathrm{F}}^2 \leq \frac{2\sin^2(\pi\alpha)}{\pi^2((\sigma-1)N/2 + 1/2 - \alpha)}\|\boldsymbol{A}\|_{\mathrm{F}}^2. \tag{3.7}$$

Therefore, we approximate the NUDFT matrix $\boldsymbol{A}$ by the product of $\boldsymbol{B}$ and $\boldsymbol{H}$, where $\boldsymbol{B}$ is a type-II NUDFT matrix and $\boldsymbol{H}$ can be approximated by an HSS matrix. In the preliminary stage, we construct a direct solver $\boldsymbol{B}_{\mathrm{fast}}$ for the type-II NUDFT matrix $\boldsymbol{B}$ as in Algorithm 2.1. Thus, the main task here is to find an efficient way to compress $\boldsymbol{H}$ into its HSS form. This can be accomplished based on the following two observations:

- The matrix-vector multiplications $\boldsymbol{A}\boldsymbol{c}$ and $\boldsymbol{A}^*\boldsymbol{f}$ correspond to the forward NUDFT computation, which can be performed very efficiently.
- The pseudoinverse (as well as the conjugate transpose of the pseudoinverse) of $\boldsymbol{B}_{\mathrm{fast}}$ can be applied rapidly by the fast solver introduced in section 2.4.

Therefore, by applying $\boldsymbol{B}_{\mathrm{fast}}^{\dagger}\boldsymbol{A}$ on random test matrices, the black-box algorithm in section 2.2 is used to obtain an HSS approximation of $\boldsymbol{H}$. After construction, the fast structure in the type-III NUDFT problem can be expressed as

$$\boldsymbol{A}_{\mathrm{fast}} = \boldsymbol{B}_{\mathrm{fast}}\boldsymbol{H}_{\mathrm{HSS}}. \tag{3.8}$$

Once the HSS matrix $\boldsymbol{H}_{\mathrm{HSS}}$ has been prepared, we use the URV factorization to give an inversion of it. Consequently, the solution of the least-squares problem is quite straightforward. We directly compute $\boldsymbol{c} = \boldsymbol{H}_{\mathrm{HSS}}^{-1}\boldsymbol{B}_{\mathrm{fast}}^{\dagger}\boldsymbol{f}$, which involves the solution processes for the type-II NUDFT matrix $\boldsymbol{B}$ and the HSS matrix $\boldsymbol{H}$.

This method can also serve as a preconditioner for iterative methods when the matrix $\boldsymbol{A}$ is too ill-conditioned. When $\boldsymbol{A}$ is not too ill-conditioned, one can obtain a solution by applying PCG to the following normal equation:

$$\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{c} = \boldsymbol{A}^*\boldsymbol{f}. \tag{3.9}$$

Because of the fast matrix-vector multiplication, each iteration of CG can be computed efficiently. In this circumstance, one can use $\boldsymbol{A}_{\mathrm{fast}}^{\dagger}\boldsymbol{A}_{\mathrm{fast}}^{\dagger *}$ as a preconditioner of system (3.9).

3.3. Complexities. We analyze the complexity of Algorithm 3.1 under the assumption that the HSS rank of both $\boldsymbol{H}_{\mathrm{HSS}}$ and $\tilde{\boldsymbol{B}}_{\mathrm{HSS}}$ is $\mathscr{O}(\log N)$. The construction stage includes three steps: As shown in section 2.4, the preliminary stage of constructing the type-II direct solver has a complexity of $\mathscr{O}(M\log^2 N)$. The construction stage applies $\boldsymbol{A}$ and $\boldsymbol{B}_{\mathrm{fast}}^{\dagger}$ on random test matrices with $\mathscr{O}(\log N)$ columns and recovers $\boldsymbol{H}_{\mathrm{HSS}}$ using the test matrices and sample matrices, which costs $\mathscr{O}(M\log^2 N + N\log^2 N)$ and $\mathscr{O}(N\log^2 N)$. Therefore, the total complexity of the construction stage is $\mathscr{O}(M\log^2 N)$. The factorization stage involves computing the URV factorization of $\boldsymbol{H}_{\mathrm{HSS}}$, which has a cost of $\mathscr{O}(N\log^2 N)$. The solution stage includes the applications of $\boldsymbol{B}_{\mathrm{fast}}^{\dagger}$ and $\boldsymbol{H}_{\mathrm{HSS}}^{\dagger}$, with complexities of $\mathscr{O}(M\log N)$ and $\mathscr{O}(N\log N)$, respectively. Hence, the total complexity for solution is $\mathscr{O}(M\log N)$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

4. Numerical results. In this section, we conduct several tests on our proposed algorithm. Throughout all experiments, two cases of type-III NUDFT problem are considered:

- Perturbation-perturbation: Sample points and frequencies are perturbed from uniform points and continuous integers, respectively, i.e., $x_j = (j + \beta\phi_j)/M$ and $\omega_k = k + \alpha\psi_k$ where each ϕ_j and ψ_k is drawn from a uniform distribution on $[-1, 1]$ and $0 \le \alpha, \beta < 1/2$ control the nonuniformity.
- Random-perturbation: Sample points are i.i.d. uniform random variables on $[0, 1)$, and the frequencies are perturbed integers given by $\omega_k = k + \alpha\psi_k$.

We briefly explain our experimental settings. The number of frequency points N is chosen to be 2^p for $p = 16 : 20$, and the number of sample points M is set to be $4N$. The leaf size of each HSS matrix is fixed at 256, the accuracy parameter in the fADI construction of $\tilde{\boldsymbol{B}}_{\mathrm{HSS}}$ is set to be 10^{-6}, and the HSS rank of $\boldsymbol{H}_{\mathrm{HSS}}$ in (3.8) is set to be approximately $10\log_2(N)$ for its black-box construction. The exact coefficient vector $\boldsymbol{c}_{\mathrm{ex}}$ is generated by a random vector in $\mathbb{C}^N$, and the target values are obtained by applying the type-III NUFFT to $\boldsymbol{c}_{\mathrm{ex}}$. For the perturbation-perturbation case, β is fixed to be 0.4 and α is chosen to be 0.1 or 0.4. For the random-perturbation case, α is set to be 0.1 and 0.4. For iterative methods, we use the MATLAB `lsqr` command [28], with tolerance set to be 10^{-12} and the maximum number of iterations set to be 500. We denote the iterative method without and with preconditioner as "lsqr" and "plsqr," respectively. All algorithms are implemented in MATLAB R2023b where the MATLAB FFT library and the FINUFFT library [3] are utilized for all corresponding fast matrix-vector multiplications. All experiments are carried out on a server with Intel Gold 6226R CPU at 2.90 GHz and 1000.6 GB of RAM.

The following values are taken into consideration when evaluating the algorithm: We use t_{cB}, t_{fB}, t_{cH}, and t_{fH} to denote the time cost for constructing $\tilde{\boldsymbol{B}}_{\mathrm{HSS}}$, factorizing $\tilde{\boldsymbol{B}}_{\mathrm{HSS}}$, constructing $\boldsymbol{H}_{\mathrm{HSS}}$, and factorizing $\boldsymbol{H}_{\mathrm{HSS}}$, respectively. For each solution, its relative residual and relative error are defined as $r_{\mathrm{s}} = \|\boldsymbol{A}\boldsymbol{c} - \boldsymbol{f}\|_2/\|\boldsymbol{f}\|_2$ and $e_{\mathrm{s}} = \|\boldsymbol{c} - \boldsymbol{c}_{\mathrm{ex}}\|_2/\|\boldsymbol{c}_{\mathrm{ex}}\|_2$, respectively. For direct methods, we use t_{d} to denote the time cost for solving the system, while for iterative methods, we use t_{pre} and t_{iter} to denote the time cost in the preparation and iteration stage, respectively. More specifically, $t_{\mathrm{pre}} = t_{\mathrm{cB}} + t_{\mathrm{fB}} + t_{\mathrm{cH}} + t_{\mathrm{fH}}$, and t_{iter} only includes the iteration time of lsqr or plsqr. Additionally, n_{iter} is the number of iterations to solve (3.9) using lsqr or plsqr to a tolerance of 10^{-12}, with a maximum number of iterations set to be 500.

4.1. Exploring properties of the type-III NUDFT. We verify the approximation error in Proposition 3.1 and (3.7) by approximating a 4096×1024 type-III NUDFT matrix $\boldsymbol{A}$ under the random-perturbation case. The approximation error is defined as $\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{Q}\|_{\mathrm{F}}^2/\|\boldsymbol{A}\|_{\mathrm{F}}^2$ and $\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{H}\|_{\mathrm{F}}^2/\|\boldsymbol{A}\|_{\mathrm{F}}^2$, respectively. Both errors are estimated by an average over 5 simulations. Figure 2 shows the error for different values of σ and α. It can be observed that the approximation error decreases as σ increases and as α decreases, which is consistent with our intuitions and the theoretical results.

4.2. Perturbation-perturbation case. In [2], the authors discuss the convergence of interpolation and quadrature for periodic functions when using perturbed points. In [37], the authors demonstrate that the condition number of a tall-and-skinny type-II NUDFT matrix with perturbed sample points can be bounded by a constant that depends only on the perturbation size β and the ratio M/N. Although there are no directly related generalizations for the type-III case, this finding provides useful insights. When the perturbation α of frequencies is small, we expect similar behaviors of the NUDFT matrix.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

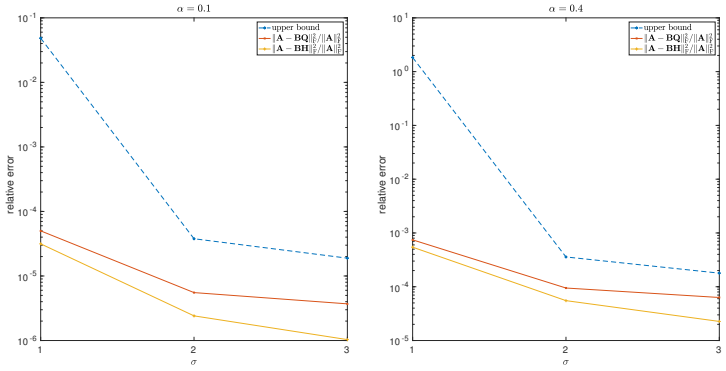


FIG. 2. *Approximation errors for various* σ *and* α. *Left:* $\alpha = 0.1$, *Right:* $\alpha = 0.4$. *The upper bound is given by Proposition* 3.1 *and* (3.7).

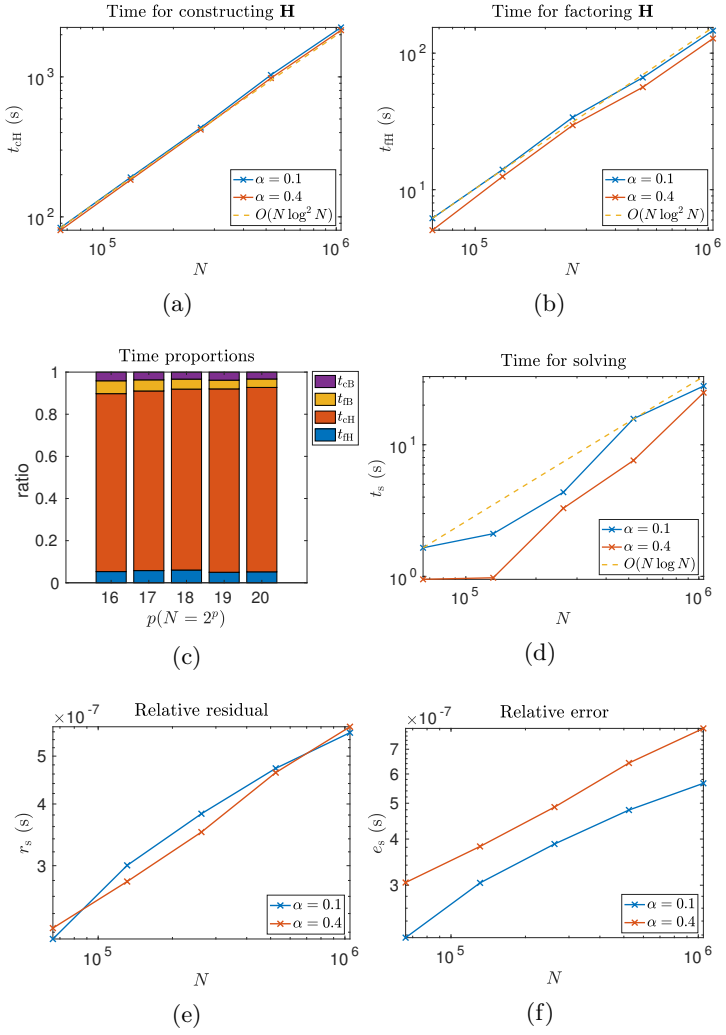


FIG. 3. *Results of the direct solver under the perturbation-perturbation case for* $\alpha = 0.1$ *and* 0.4 *and* $\beta = 0.4$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Figure 3 and Tables 1 and 2 show the results of the direct solver under the perturbation-perturbation case for $\alpha = 0.1$ and 0.4. It can be observed that the time costs of different components in the algorithm scale as predicted in Section 3.3, except for the solution stage. This is because the time cost of the solution stage is quite small compared to others. As we expect, the runtimes for $\alpha = 0.4$ and $\alpha = 0.1$ are quite similar, since our algorithm is not sensitive to distribution of sample points and frequencies. In all examples, the most time-consuming part is the black-box construction of $\boldsymbol{H}_{\mathrm{HSS}}$, accounting for nearly 90% of the total time cost; this can also be inferred from the proportion of different components shown in Figure 3(c). To be more specific, when $\alpha = 0.4$ and $N = 2^{20}$, the runtimes of t_{cB}, t_{fB}, t_{cH}, t_{fH} are 80s, 98s, 2200s, and 130s, respectively, while the solution only takes 25s. On the other hand, the relative residual and relative error are at the order of 10^{-7} for all examples, meaning that our algorithm is quite accurate and stable.

The results of the iterative solver are shown in Tables 3 and 4. For both values of α, the NUDFT matrix is quite well-conditioned, since the number of iterations for lsqr is quite small. When $\alpha = 0.1$, the number of iterations is 17 for all examples, while when $\alpha = 0.4$, the number of iterations is around 100. On the other hand, our algorithm serves as a very efficient preconditioner for the iterative method, since the number of iterations for plsqr is only 5 and 9 for $\alpha = 0.1$ and 0.4, respectively. However, the time cost for plsqr is higher than that of lsqr, since each iteration of plsqr is more expensive and the problem is easy for applying lsqr directly.

4.3. Random-perturbation case. For the random-perturbation case, we did not find any literature discussing the conditioning of the type-III NUDFT matrix. For the type-II case when sample points are i.i.d. uniform random variables, the paper [19] contains a study of the conditioning of the NUDFT matrix, showing that it is well-conditioned with high probability when $M \gtrsim N \log N$. This implies that the random-perturbation case is more challenging than the perturbation-perturbation case.

TABLE 1
Time and error results of the direct solver under the perturbation-perturbation case, $\alpha = 0.1$ and $\beta = 0.4$.

N	t_{cB}	t_{fB}	t_{cH}	t_{fH}	t_{d}	r_{s}	e_{s}
2^{16}	4.1e + 00	6.0e + 00	8.4e + 01	6.2e + 00	1.6e + 00	2.1e-07	2.2e-07
2^{17}	8.1e + 00	1.1e + 01	1.9e + 02	1.4e + 01	2.1e + 00	3.0e-07	3.0e-07
2^{18}	1.6e + 01	2.3e + 01	4.3e + 02	3.4e + 01	4.4e + 00	3.8e-07	3.9e-07
2^{19}	4.0e + 01	4.6e + 01	1.0e + 03	6.6e + 01	1.6e + 01	4.7e-07	4.8e-07
2^{20}	9.5e + 01	9.7e + 01	2.3e + 03	1.5e + 02	2.8e + 01	5.6e-07	5.7e-07

TABLE 2
Time and error results of the direct solver under the perturbation-perturbation case, $\alpha = 0.4$ and $\beta = 0.4$.

N	t_{cB}	t_{fB}	t_{cH}	t_{fH}	t_{d}	r_{s}	e_{s}
2^{16}	3.9e + 00	5.8e + 00	8.0e + 01	5.1e + 00	9.5e-01	2.2e-07	3.1e-07
2^{17}	7.9e + 00	1.1e + 01	1.8e + 02	1.2e + 01	9.8e-01	2.8e-07	3.8e-07
2^{18}	1.6e + 01	2.3e + 01	4.2e + 02	3.0e + 01	3.3e + 00	3.5e-07	4.9e-07
2^{19}	4.3e + 01	4.7e + 01	9.8e + 02	5.6e + 01	7.6e + 00	4.6e-07	6.4e-07
2^{20}	8.0e + 01	9.8e + 01	2.2e + 03	1.3e + 02	2.5e + 01	5.7e-07	8.0e-07

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

TABLE 3
Time and error results of the iterative solver under the perturbation-perturbation case, $\alpha = 0.1$ and $\beta = 0.4$.

N	Method	t_{pre}	t_{iter}	n_{iter}	r_{s}	e_{s}
2^{16}	lsqr	-	7.8e-01	17	7.4e-13	8.0e-13
	plsqr	1.0e + 02	6.9e + 00	5	4.1e-13	4.3e-13
2^{17}	lsqr	-	1.5e + 00	17	6.7e-13	7.1e-13
	plsqr	2.2e + 02	1.2e + 01	5	4.7e-14	4.9e-14
2^{18}	lsqr	-	3.1e + 00	17	6.4e-13	6.9e-13
	plsqr	5.1e + 02	2.6e + 01	5	2.8e-13	3.0e-13
2^{19}	lsqr	-	7.4e + 00	17	9.3e-13	1.0e-12
	plsqr	1.2e + 03	6.1e + 01	5	1.3e-13	1.4e-13
2^{20}	lsqr	-	1.9e + 01	17	8.0e-13	8.7e-13
	plsqr	2.6e + 03	1.3e + 02	5	5.6e-14	5.7e-14

TABLE 4
Time and error results of the iterative solver under the perturbation-perturbation case, $\alpha = 0.4$ and $\beta = 0.4$.

N	Method	t_{pre}	t_{iter}	n_{iter}	r_{s}	e_{s}
2^{16}	lsqr	-	4.1e + 00	95	8.8e-13	2.9e-12
	plsqr	9.5e + 01	8.5e + 00	8	4.0e-13	7.3e-13
2^{17}	lsqr	-	8.6e + 00	99	1.0e-12	4.0e-12
	plsqr	2.2e + 02	2.0e + 01	9	3.3e-15	6.1e-15
2^{18}	lsqr	-	1.7e + 01	101	7.8e-13	2.9e-12
	plsqr	4.9e + 02	4.7e + 01	9	6.6e-14	1.2e-13
2^{19}	lsqr	-	3.4e + 01	104	8.9e-13	3.5e-12
	plsqr	1.1e + 03	9.7e + 01	9	1.6e-14	3.2e-14
2^{20}	lsqr	-	1.2e + 02	105	9.9e-13	4.0e-12
	plsqr	2.5e + 03	2.1e + 02	9	3.6e-14	6.5e-14

Figure 4 and Tables 5 and 6 show the results of the direct solver under the random-perturbation case for $\alpha = 0.1$ and 0.4. First, it can be found that the relative residual and relative error are at the order of 10^{-6} for all examples, which is slightly larger than the perturbation-perturbation case. This is because the random-perturbation case is more ill-conditioned than the perturbation-perturbation case, which can also be inferred from the results of the iterative solver. Furthermore, the runtime of the direct solver is quite similar for both the random-perturbation and perturbation-perturbation cases, which is consistent with our expectation since our algorithm is not sensitive to the distribution of sample points.

Tables 7 and 8 summarize the results of the iterative solver. It is clear that lsqr fails to converge within 500 iterations for all examples. On the other hand, plsqr converges within 5 steps for $\alpha = 0.1$ and 9 steps for $\alpha = 0.4$, which is significantly smaller than the number of iterations without preconditioning. This demonstrates the efficiency of our method as a preconditioner for iterative methods and also implies that increasing α leads to a more ill-conditioned problem, which is consistent with our intuition. Considering the time cost, once the preconditioner is constructed (which is the most time-consuming part), the runtime of plsqr is quite small, which is about 170s for $\alpha = 0.1$ and 240s for $\alpha = 0.4$ when $N = 2^{20}$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

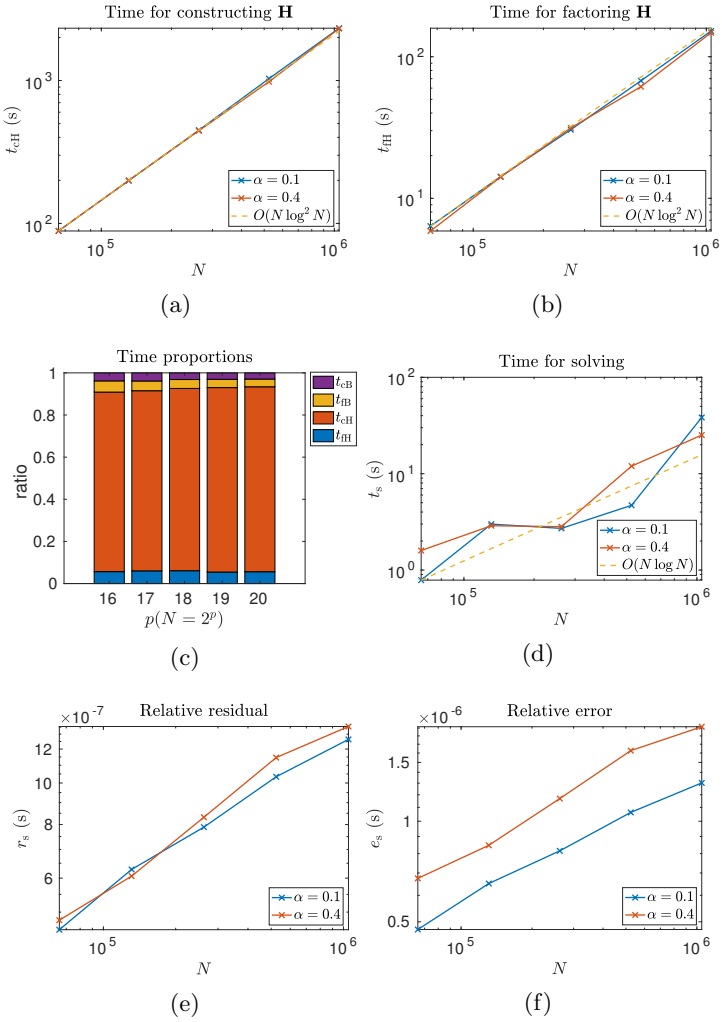


FIG. 4. *Results of the direct solver under the random-perturbation case for* $\alpha = 0.1$ *and* 0.4.

TABLE 5
Time and error results of the direct solver under the random-perturbation case, $\alpha = 0.1$.

N	t_{cB}	t_{fB}	t_{cH}	t_{fH}	t_d	r_s	e_s
2^{16}	3.9e + 00	5.4e + 00	8.9e + 01	6.4e + 00	7.9e-01	4.5e-07	4.7e-07
2^{17}	8.1e + 00	1.1e + 01	2.0e + 02	1.4e + 01	3.0e + 00	6.3e-07	6.5e-07
2^{18}	1.7e + 01	2.2e + 01	4.5e + 02	3.1e + 01	2.7e + 00	7.9e-07	8.2e-07
2^{19}	3.3e + 01	4.4e + 01	1.0e + 03	6.8e + 01	4.7e + 00	1.0e-06	1.1e-06
2^{20}	7.0e + 01	9.1e + 01	2.3e + 03	1.5e + 02	3.8e + 01	1.3e-06	1.3e-06

5. Conclusions and future work. In this paper, we introduce a superfast direct solver for the least-squares problem of the type-III INUDFT problem. Our method combines the efficient type-II INUDFT direct solver [32] and fast algorithms of HSS matrices [23, 34]. By approximating the type-III NUDFT matrix by a product of a NUDFT-II matrix and an HSS matrix, we develop a fast inversion algorithm,

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

TABLE 6
Time and error results of the direct solver under the random-perturbation case, $\alpha = 0.4$.

N	t_{cB}	t_{fB}	t_{cH}	t_{fH}	t_{d}	r_{s}	e_{s}
2^{16}	3.9e + 00	5.5e + 00	8.9e + 01	5.9e + 00	1.6e + 00	4.8e-07	6.7e-07
2^{17}	9.0e + 00	1.1e + 01	2.0e + 02	1.4e + 01	2.9e + 00	6.1e-07	8.5e-07
2^{18}	1.6e + 01	2.2e + 01	4.5e + 02	3.1e + 01	2.8e + 00	8.3e-07	1.2e-06
2^{19}	3.4e + 01	4.4e + 01	9.8e + 02	6.1e + 01	1.2e + 01	1.1e-06	1.6e-06
2^{20}	7.7e + 01	9.6e + 01	2.3e + 03	1.5e + 02	2.5e + 01	1.4e-06	1.9e-06

TABLE 7
Time and error results of the iterative solver under the random-perturbation case, $\alpha = 0.1$.

N	Method	t_{pre}	t_{iter}	n_{iter}	r_{s}	e_{s}
2^{16}	lsqr	-	2.1e + 01	500	2.3e-12	3.0e-11
	plsqr	1.0e + 02	9.2e + 00	5	5.3e-14	2.0e-13
2^{17}	lsqr	-	4.3e + 01	500	5.9e-06	1.7e-03
	plsqr	2.3e + 02	1.4e + 01	5	2.8e-13	1.6e-11
2^{18}	lsqr	-	8.4e + 01	500	2.7e-07	1.1e-05
	plsqr	5.2e + 02	2.9e + 01	5	7.8e-14	4.9e-13
2^{19}	lsqr	-	1.7e + 02	500	9.6e-09	3.0e-07
	plsqr	1.2e + 03	6.2e + 01	5	3.2e-13	1.1e-12
2^{20}	lsqr	-	5.2e + 02	500	9.7e-07	6.2e-05
	plsqr	2.6e + 03	1.7e + 02	5	1.2e-13	7.6e-13

which can serve as a direct solver or an efficient preconditioner for iterative methods. We give some theoretical results under suitable assumptions of the sample points and frequencies. Numerical experiments are carried out to explore the properties of our fast structure and to demonstrate the efficiency of the proposed method.

Some important directions for future research include the following:

- Theoretical analysis of the type-III NUDFT problem such as the properties of the NUDFT matrix. This would enable a much deeper understanding of the approximation. Current works mostly focus on the type-II case and lack a discussion of the type-III case.
- Investigate a method on the regularized least-squares problem. In many circumstances (such as when sample points and frequencies are both uniform random variables), the NUDFT matrix is ill-conditioned or even rank-deficient. Regularization techniques need to be added into the least-squares problem, and in this case, our method cannot be applied directly. The question of whether the regularized problem can be efficiently solved remains an interesting topic.
- Extension to high-dimensional cases. The challenges of the type-III INUDFT problem in high dimensions are similar to those of the type-II INUDFT. For two dimensions and higher dimensions, the displacement structure (2.12) of $\tilde{\boldsymbol{A}}$ does not hold anymore. In this case one can still derive some low-rank properties by the kernel function expression (2.13). However, it cannot be compressed into an HSS matrix since many blocks are not numerically low-rank. Designing efficient methods to solve high-dimensional INUDFT problems remains open.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

TABLE 8
Time and error results of the iterative solver under the random-perturbation case, $\alpha = 0.4$.

N	Method	t_{pre}	t_{iter}	n_{iter}	r_{s}	e_{s}
2^{16}	lsqr	-	2.2e + 01	500	3.0e-07	7.1e-06
	plsqr	1.0e + 02	1.5e + 01	8	5.4e-13	2.7e-12
2^{17}	lsqr	-	4.3e + 01	500	5.1e-10	9.1e-09
	plsqr	2.3e + 02	2.6e + 01	9	8.2e-15	3.2e-14
2^{18}	lsqr	-	8.5e + 01	500	1.2e-07	6.3e-06
	plsqr	5.1e + 02	5.2e + 01	9	9.0e-14	5.6e-13
2^{19}	lsqr	-	2.2e + 02	500	2.1e-06	6.4e-04
	plsqr	1.1e + 03	9.4e + 01	8	2.4e-13	2.6e-11
2^{20}	lsqr	-	4.8e + 02	500	4.3e-06	5.5e-04
	plsqr	2.6e + 03	2.4e + 02	9	7.1e-14	1.9e-12

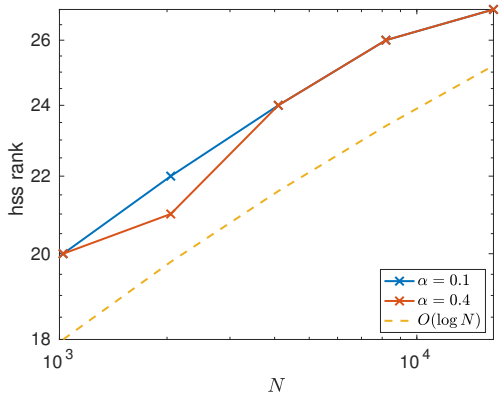


FIG. 5. *The ε-rank of the top-right submatrix in $\boldsymbol{B}_{\text{fast}}^{\dagger}\boldsymbol{A}$ for $\varepsilon = 10^{-6}$. The size of the submatrix is $N/2$, where $N = 2^p$ for $p = 10:14$.*

Appendix A. The HSS rank in the type-III NUDFT approximation. We discuss the HSS rank of $\boldsymbol{B}_{\text{fast}}^{\dagger}\boldsymbol{A}$. To do this, we extract the $N/2 \times N/2$ top-right submatrix of it under the random-perturbation case and compute its ε-rank r_ε. The results in terms of $\varepsilon = 10^{-6}$ are plotted in Figure 5. It is obvious that the numerical rank is significantly small compared to the size of the submatrix, implying the low-rank structure of $\boldsymbol{B}_{\text{fast}}^{\dagger}\boldsymbol{A}$. Furthermore, given a fixed accuracy ε, the rank of the submatrix behaves approximately as $\mathcal{O}(\log N)$ relative to the matrix size, consistent with our intuition and discussion in section 3.2.

Appendix B. Discussion of the type-III NUDFT approximation. In this section, we give a detailed discussion of the approximation in the fast structure (3.8), which can be viewed as an extension of section 4.1. Specifically, we consider the following three errors.

- The relative error of the approximation:

$$\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{Q}\|_{\text{F}}/\|\boldsymbol{A}\|_{\text{F}} \quad \text{and} \quad \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{H}\|_{\text{F}}/\|\boldsymbol{A}\|_{\text{F}}.$$

- The relative error of the inversion:

$$\|\boldsymbol{A}^{\dagger} - \boldsymbol{Q}^{\dagger}\boldsymbol{B}^{\dagger}\|_{\text{F}}/\|\boldsymbol{A}^{\dagger}\|_{\text{F}} \quad \text{and} \quad \|\boldsymbol{A}^{\dagger} - \boldsymbol{H}^{\dagger}\boldsymbol{B}^{\dagger}\|_{\text{F}}/\|\boldsymbol{A}^{\dagger}\|_{\text{F}}.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

TABLE 9
*Errors of the approximations in the type-*III *NUDFT,* $\alpha = 0.1$.

σ	1	2	3
$\|\boldsymbol{A} - \boldsymbol{BQ}\|_{\mathrm{F}}/\|\boldsymbol{A}\|_{\mathrm{F}}$	7.0e-03	2.1e-03	1.7e-03
$\|\boldsymbol{A} - \boldsymbol{BH}\|_{\mathrm{F}}/\|\boldsymbol{A}\|_{\mathrm{F}}$	5.5e-03	9.9e-04	4.9e-04
$\|\boldsymbol{A}^\dagger - \boldsymbol{Q}^\dagger\boldsymbol{B}^\dagger\|_{\mathrm{F}}/\|\boldsymbol{A}^\dagger\|_{\mathrm{F}}$	1.4e-02	7.7e+01	5.4e+08
$\|\boldsymbol{A}^\dagger - \boldsymbol{H}^\dagger\boldsymbol{B}^\dagger\|_{\mathrm{F}}/\|\boldsymbol{A}^\dagger\|_{\mathrm{F}}$	1.1e-02	7.3e+01	1.6e+07
$\|\boldsymbol{I} - \boldsymbol{Q}^\dagger\boldsymbol{B}^\dagger\boldsymbol{A}\|_{\mathrm{F}}/\|\boldsymbol{I}\|_{\mathrm{F}}$	8.8e-03	3.8e-02	1.1e+05
$\|\boldsymbol{I} - \boldsymbol{H}^\dagger\boldsymbol{B}^\dagger\boldsymbol{A}\|_{\mathrm{F}}/\|\boldsymbol{I}\|_{\mathrm{F}}$	5.9e-15	2.2e-13	8.4e-02

TABLE 10
*Errors of the approximations in the type-*III *NUDFT,* $\alpha = 0.4$.

σ	1	2	3
$\|\boldsymbol{A} - \boldsymbol{BQ}\|_{\mathrm{F}}/\|\boldsymbol{A}\|_{\mathrm{F}}$	2.7e-02	8.6e-03	6.6e-03
$\|\boldsymbol{A} - \boldsymbol{BH}\|_{\mathrm{F}}/\|\boldsymbol{A}\|_{\mathrm{F}}$	2.3e-02	5.3e-03	2.4e-03
$\|\boldsymbol{A}^\dagger - \boldsymbol{Q}^\dagger\boldsymbol{B}^\dagger\|_{\mathrm{F}}/\|\boldsymbol{A}^\dagger\|_{\mathrm{F}}$	5.9e-02	3.2e+01	2.9e+06
$\|\boldsymbol{A}^\dagger - \boldsymbol{H}^\dagger\boldsymbol{B}^\dagger\|_{\mathrm{F}}/\|\boldsymbol{A}^\dagger\|_{\mathrm{F}}$	5.4e-02	3.4e+01	4.4e+05
$\|\boldsymbol{I} - \boldsymbol{Q}^\dagger\boldsymbol{B}^\dagger\boldsymbol{A}\|_{\mathrm{F}}/\|\boldsymbol{I}\|_{\mathrm{F}}$	3.7e-02	1.4e-01	9.8e+03
$\|\boldsymbol{I} - \boldsymbol{H}^\dagger\boldsymbol{B}^\dagger\boldsymbol{A}\|_{\mathrm{F}}/\|\boldsymbol{I}\|_{\mathrm{F}}$	7.2e-15	5.6e-13	3.5e-04

- The relative error of the inversion on range($\boldsymbol{A}$):

$$\|\boldsymbol{I} - \boldsymbol{Q}^\dagger\boldsymbol{B}^\dagger\boldsymbol{A}\|_{\mathrm{F}}/\|\boldsymbol{I}\|_{\mathrm{F}} \quad \text{and} \quad \|\boldsymbol{I} - \boldsymbol{H}^\dagger\boldsymbol{B}^\dagger\boldsymbol{A}\|_{\mathrm{F}}/\|\boldsymbol{I}\|_{\mathrm{F}}.$$

Tables 9 and 10 summarize the results of the errors for different σ and α. It can be found that for $\|\boldsymbol{A} - \boldsymbol{BQ}\|_{\mathrm{F}}/\|\boldsymbol{A}\|_{\mathrm{F}}$ and $\|\boldsymbol{A} - \boldsymbol{BH}\|_{\mathrm{F}}/\|\boldsymbol{A}\|_{\mathrm{F}}$, the errors decrease as σ increases, which is consistent with our theoretical analysis. However, for $\|\boldsymbol{A}^\dagger - \boldsymbol{Q}^\dagger\boldsymbol{B}^\dagger\|_{\mathrm{F}}/\|\boldsymbol{A}^\dagger\|_{\mathrm{F}}$ and $\|\boldsymbol{A}^\dagger - \boldsymbol{H}^\dagger\boldsymbol{B}^\dagger\|_{\mathrm{F}}/\|\boldsymbol{A}^\dagger\|_{\mathrm{F}}$, the errors are less than 1 only when $\sigma = 1$, meaning that we cannot use $\sigma > 1$ for these approximations. A increasing trend with σ is also observed for $\|\boldsymbol{I} - \boldsymbol{Q}^\dagger\boldsymbol{B}^\dagger\boldsymbol{A}\|_{\mathrm{F}}/\|\boldsymbol{I}\|_{\mathrm{F}}$ and $\|\boldsymbol{I} - \boldsymbol{H}^\dagger\boldsymbol{B}^\dagger\boldsymbol{A}\|_{\mathrm{F}}/\|\boldsymbol{I}\|_{\mathrm{F}}$. Therefore we focus on the case of $\sigma = 1$. In this case, although $\|\boldsymbol{A}^\dagger - \boldsymbol{Q}^\dagger\boldsymbol{B}^\dagger\|_{\mathrm{F}}/\|\boldsymbol{A}^\dagger\|_{\mathrm{F}}$, $\|\boldsymbol{A}^\dagger - \boldsymbol{H}^\dagger\boldsymbol{B}^\dagger\|_{\mathrm{F}}/\|\boldsymbol{A}^\dagger\|_{\mathrm{F}}$, and $\|\boldsymbol{I} - \boldsymbol{Q}^\dagger\boldsymbol{B}^\dagger\boldsymbol{A}\|_{\mathrm{F}}/\|\boldsymbol{I}\|_{\mathrm{F}}$ are not very small (at the order of 10^{-2}), $\|\boldsymbol{I} - \boldsymbol{H}^\dagger\boldsymbol{B}^\dagger\boldsymbol{A}\|_{\mathrm{F}}/\|\boldsymbol{I}\|_{\mathrm{F}}$ is quite small (at the order of 10^{-15}). It implies that when the target values are in or nearly in the range of $\boldsymbol{A}$, we can expect a very accurate solution by using $\boldsymbol{H}^\dagger\boldsymbol{B}^\dagger$.

Acknowledgment. The authors would like to thank the reviewers for their valuable comments and suggestions, which have greatly improved the quality of this paper.

REFERENCES

[1] C. ANDERSON AND M. D. DAHLEH, *Rapid computation of the discrete Fourier transform*, SIAM J. Sci. Comput., 17 (1996), pp. 913–919, https://doi.org/10.1137/0917059.

[2] A. P. AUSTIN AND L. N. TREFETHEN, *Trigonometric interpolation and quadrature in perturbed points*, SIAM J. Numer. Anal., 55 (2017), pp. 2113–2122, https://doi.org/10.1137/16M1107760.

[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 J. Sci. Comput., 41 (2019), pp. C479–C504, https://doi.org/10.1137/18M120885X.

[4] B. BECKERMANN, D. KRESSNER, AND H. WILBER, *Compression properties for large Toeplitz-like matrices*, Numer. Algorithms, 100 (2025), pp. 2041–2067, https://doi.org/10.1007/s11075-025-02185-8.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

[5] B. Beckermann and A. Townsend, *On the singular values of matrices with displacement structure*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1227–1248, https://doi.org/10.1137/16M1096426.

[6] P. Benner, R.-C. Li, and N. Truhar, *On the ADI method for Sylvester equations*, J. Comput. Appl. Math., 233 (2009), pp. 1035–1045, https://doi.org/10.1016/j.cam.2009.08.108.

[7] S. Börm, L. Grasedyck, and W. Hackbusch, *Introduction to hierarchical matrices with applications*, Eng. Anal. Boundary Elem., 27 (2003), pp. 405–422, https://doi.org/10.1016/S0955-7997(02)00152-2.

[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, https://doi.org/10.1137/080734339.

[9] S. Chandrasekaran, M. Gu, and T. Pals, *A fast ULV decomposition solver for hierarchically semiseparable representations*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 603–622, https://doi.org/10.1137/S0895479803436652.

[10] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301, https://doi.org/10.1090/S0025-5718-1965-0178586-1.

[11] A. Dutt and V. Rokhlin, *Fast Fourier transforms for nonequispaced data*, SIAM J. Sci. Comput., 14 (1993), pp. 1368–1393, https://doi.org/10.1137/0914081.

[12] A. Dutt and V. Rokhlin, *Fast Fourier transforms for nonequispaced data,* II, Appl. Comput. Harmon. Anal., 2 (1995), pp. 85–100, https://doi.org/10.1006/acha.1995.1007.

[13] H. G. Feichtinger, K. Gröchenig, and T. Strohmer, *Efficient numerical methods in non-uniform sampling theory*, Numer. Math., 69 (1995), pp. 423–440, https://doi.org/10.1007/s002110050101.

[14] L. Greengard and J.-Y. Lee, *Accelerating the nonuniform fast Fourier transform*, SIAM Rev., 46 (2004), pp. 443–454, https://doi.org/10.1137/S003614450343200X.

[15] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part* I*: Introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108, https://doi.org/10.1007/s006070050015.

[16] W. Hackbusch, B. N. Khoromskij, and S. A. Sauter, *On $\mathcal{H}^2$-matrices*, in Lectures on Applied Mathematics (Munich, 1999), Springer-Verlag, Berlin, 2000, pp. 9–29.

[17] W. Hackbusch and B. N. Khoromskij, *A sparse $\mathcal{H}$-matrix arithmetic. Part* II*: Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47, https://doi.org/10.1007/PL00021408.

[18] K. L. Ho and L. Greengard, *A fast direct solver for structured linear systems by recursive skeletonization*, SIAM J. Sci. Comput., 34 (2012), pp. A2507–A2532, https://doi.org/10.1137/120866683.

[19] L. Kämmerer, T. Ullrich, and T. Volkmer, *Worst-case recovery guarantees for least squares approximation using random samples*, Constr. Approx., 54 (2021), pp. 295–352, https://doi.org/10.1007/s00365-021-09555-0.

[20] M. Kircheis and D. Potts, *Direct inversion of the nonequispaced fast Fourier transform*, Linear Algebra Appl., 575 (2019), pp. 106–140, https://doi.org/10.1016/j.laa.2019.03.028.

[21] M. Kircheis and D. Potts, *Fast and direct inversion methods for the multivariate nonequispaced fast Fourier transform*, Front. Appl. Math. Stat., 9 (2023), 1155484, https://doi.org/10.3389/fams.2023.1155484.

[22] J.-Y. Lee and L. Greengard, *The type* 3 *nonuniform FFT and its applications*, J. Comput. Phys., 206 (2005), pp. 1–5, https://doi.org/10.1016/j.jcp.2004.12.004.

[23] J. Levitt and P.-G. Martinsson, *Linear-complexity black-box randomized compression of rank-structured matrices*, SIAM J. Sci. Comput., 46 (2024), pp. A1747–A1763, https://doi.org/10.1137/22M1528574.

[24] Y. Li, H. Yang, E. R. Martin, K. L. Ho, and L. Ying, *Butterfly factorization*, Multiscale Model. Simul., 13 (2015), pp. 714–732, https://doi.org/10.1137/15M1007173.

[25] L. Lin, J. Lu, and L. Ying, *Fast construction of hierarchical matrix representation from matrix-vector multiplication*, J. Comput. Phys., 230 (2011), pp. 4071–4087, https://doi.org/10.1016/j.jcp.2011.02.033.

[26] P. G. Martinsson, *A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1251–1274, https://doi.org/10.1137/100786617.

[27] P.-G. Martinsson, *Fast Direct Solvers for Elliptic PDEs,* CBMS-NSF Regional Conf. Ser. in Appl. Math. 96, SIAM, Philadelphia, 2019, https://doi.org/10.1137/1.9781611976045.

[28] 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, https://doi.org/10.1145/355984.355989.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

[29] G. Plonka, D. Potts, G. Steidl, and M. Tasche, *Fast Fourier transforms for nonequispaced data*, in Numerical Fourier Analysis, Springer International, Cham, 2023, pp. 413–463.
[30] D. Potts, G. Steidl, and M. Tasche, *Fast Fourier Transforms for Nonequispaced Data: A Tutorial*, Birkhäuser Boston, Boston, MA, 2001, pp. 247–270.
[31] D. Ruiz-Antolín and A. Townsend, *A nonuniform fast Fourier transform based on low rank approximation*, SIAM J. Sci. Comput., 40 (2018), pp. A529–A547, https://doi.org/10.1137/17M1134822.
[32] H. Wilber, E. N. Epperly, and A. H. Barnett, *Superfast direct inversion of the nonuniform discrete Fourier transform via hierarchically semiseparable least squares*, SIAM J. Sci. Comput., 47 (2025), pp. A1702–A1732, https://doi.org/10.1137/24M1656694.
[33] H. D. Wilber, *Computing Numerically with Rational Functions*, Ph.D. thesis, Cornell University, Ithaca, NY, 2021.
[34] Y. Xi, J. Xia, S. Cauley, and V. Balakrishnan, *Superfast and stable structured solvers for Toeplitz least squares via randomized sampling*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 44–72, https://doi.org/10.1137/120895755.
[35] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976, https://doi.org/10.1002/nla.691.
[36] R. M. Young, *An Introduction to Nonharmonic Fourier Series*, Pure Appl. Math. 93, Academic Press, 1981.
[37] A. Yu and A. Townsend, *On the stability of unevenly spaced samples for interpolation and quadrature*, BIT Numer. Math., 63 (2023), 23, https://doi.org/10.1007/s10543-023-00965-z.

Downloaded 11/25/26 to 130.88.226.2 by Jingyu Liu (jyliu22@m.fudan.edu.cn). Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.