Sparse Fast Fourier Transform for Exactly and Generally $K$-Sparse Signals by Downsampling and Sparse Recovery

Sung-Hsien Hsieh*,**, Chun-Shien Lu**, and Soo-Chang Pei*

*Graduate Inst. Comm. Eng., National Taiwan University, Taipei, Taiwan
**Institute of Information Science, Academia Sinica, Taipei, Taiwan

Abstract

Fast Fourier Transform (FFT) is one of the most important tools in digital signal processing. FFT costs $O(N \log N)$ for transforming a signal of length $N$. Recently, researchers at MIT have proposed Sparse Fast Fourier Transform (sFFT) [1][2] as a breakthrough with computational complexity $O(K \log N)$ and $O(K \log N \log \frac{N}{K})$ for exactly $K$-sparse signal (with only $K$ non-zero frequency grids) and generally $K$-sparse signal (with $K$ significant frequency grids), respectively, to outperform classic FFT.

In this paper, a new sparse Fast Fourier Transform by downsampling in the time domain (sFFT-DT) is proposed for both exactly and generally $K$-sparse signals, based on the assumption that the distribution of the non-zero frequency grids is uniform. The idea behind sFFT-DT is to downsample the original input signal at the beginning; then, subsequent processing operates under downsampled signals, where signal lengths are proportional to $O(K)$. Downsampling, however, possibly leads to “aliasing”. By the shift property of DFT, we recast the aliasing problem as a “moment-preserving problem (MPP),” which is solvable. We prove two theorems related to initializing the downsampling factors under different conditions to have computational complexity, $O(K \log K)$ and $O(K^{\frac{3}{2}} \log K)$. Moreover, for generally $K$-sparse signals, solutions to the MPP are inaccurate due to interference from non-significant frequency grids. We observe that significant frequency grids in aliasing are “sparse”. This property is exploited, and a new sparse signal recovery algorithm in the context of compressive sensing is presented to refine the solution to MPP. The computational complexity still costs $O(K \log K)$ but requires a larger Big-O constant compared to the case of exactly $K$-sparse signals. We conduct theoretical complexity analyses and simulations to demonstrate that our method (sFFT-DT) outperforms classic FFT and MIT’s sFFT.

Index Terms

Compressed Sensing, Downsampling, FFT, Sparse FFT, Sparsity
I. INTRODUCTION

A. Background and Related Work

Fast Fourier transform (FFT) is one of the most important approaches for fast computing Discrete Fourier transform (DFT) of a signal with time complexity $O(N \log N)$, where $N$ is the signal length. FFT has been used widely in the communities of signal processing and communications. How to outperform FFT, however, remains a challenge and persistently receives attention.

Recently, H. Hassanieh et al. from MIT proposed, as a breakthrough, a new technique, called Sparse Fast Fourier Transform (sFFT) \cite{1,2}, which was proved to outperform FFT. sFFT relies on the assumption that input signals exhibit “sparsity” in the frequency domain. Let $x \in \mathbb{C}^N$ be the input signal in the time domain, and let $X \in \mathbb{C}^N$ be the Discrete Fourier Transform (DFT) of $x$. $x$ is exactly $K$-sparse if there are $K$ non-zero entries in $X$ and $K < N$. On the other hand, $x$ is a generally $K$-sparse signal if there are more than $K$ non-zero entries in $X$, but it is desirable to keep only the first $K$-largest (significant) entries in terms of magnitude and ignore the remainder. For both types of signals, sFFT costs $O(K \log N)$ and $O(K \log N \log \frac{N}{K})$, respectively.

The idea behind sFFT is to sample fewer frequency (abbreviated as freq. hereafter) grids (proportional to $K$) instead of keeping all freq. grids since most freq. grids are zero or insignificant. FFT based on such a subsampling strategy will only cost $O(K \log K)$ calculations ideally. Nevertheless, as the locations and values of the $K$ non-zero entries are unknown, subsampled freq. grids often lead to data loss and cannot achieve perfect (100%) reconstruction. To cope with this difficulty, sFFT utilizes the strategies of filtering and permutation, which can increase the probability of capturing useful information from subsampled freq. grids. These operations cost $O(K \log N)$. According to \cite{1,2}, sFFT is faster than FFTW \cite{3} (a very fast C subroutine library for computing FFT) when $X$ is exactly $K$-sparse with $K \leq \frac{N}{2^6}$. sFFT also outperforms previous works, such as \cite{4,5}.

Nevertheless, for a generally $K$-sparse signal, subsampled freq. grids certainly face interference from neighboring insignificant freq. grids. To alleviate this interference, which is assumed to be random noise in \cite{1}, sFFT permutes $X$ randomly in order to make neighboring entries randomly distributed. In this way, sFFT produces a number of subsampled signals under different permutations, where each subsampled signal is considered a candidate and the median is identified from these candidates.

Even though sFFT \cite{1,2} is outstanding, there are some limitations, summarized as follows. 1) Filtering and permutation are operated on $x$. Since $x \in \mathbb{C}^N$, these operations are related to $N$. Thus, the complexity of sFFT still involves $N$ and cannot achieve the theoretical ideal complexity $O(K \log K)$. 2) sFFT only guarantees that it succeeds with a constant probability (e.g., $2/3$). 3) The implementation of sFFT for generally $K$-sparse signals is very complicated as it involves too many parameters.
that are difficult to set.†

Recently, Ghazi et al. in MIT proposed another sFFT version [6] that costs $O(K \log K)$ operations for exactly $K$-sparse signals. The basic idea is similar to our previous work [7]. Both methods first downsample original signals before recovering $K$ non-zero freq. grids from the downsampled signals via error correction techniques, where [6] uses Reed-Solomon code, which is equivalent to the Moment-preserving problem considered in [7]. The key difference is that Ghazi et al.’s method recovers all $K$ non-zero freq. grids once, while we propose a top-down strategy to solve $K$ non-zero freq. grids iteratively. The comparison between these two methods in terms of computational complexity and successful probability will be discussed later in Sec. II-D.

In addition to a series of results regarding sFFT from MIT, Pawar and Ramchandran [8] proposed an algorithm, called FFAST (Fast Fourier Aliasing-based Sparse Transform), which focuses only on exactly $K$-sparse signals. Their approach is based on filterless downsampling of the input signal $x$ using a small set of co-prime downsampling factors guided by the Chinese Remainder Theorem (CRT). These aliasing patterns of different downsampled signals are formulated as parity-check constraints of good erasure-correcting sparse-graph codes. They also show that computing $X$ is equivalent to decoding these sparse-graph codes. As claimed in [8], FFAST costs $O(K \log K)$ and outperforms sFFT [1][2]. Nevertheless, FFAST depends on co-prime downsampling factors. In other words, the signal length $N$ must be a product of a number (typically 3 to 9) of distinct primes; otherwise FFAST cannot work. Moreover, the smallest downsampling factor bounds FFAST’s computational cost. For example, if $N = 2^{20}3^2$ and $K = 2^{16}$, the smallest downsampling factor is $3^2$. In this case, the computational cost of calculating FFT of a downsampled signal with length $N/3^2$ is (far) higher than $O(K \log K)$. Actually, these limitations are possibly harsh.

B. Our Contributions

In our previous work [7], we proposed a new algorithm, called sFFT-DT, by downsampling in the time domain with time complexity of $O(K \log K)$ only for exactly $K$-sparse signals. The idea behind sFFT-DT is to downsample the input signal in the time domain before directly conducting all subsequent operations on the downsampled signals. By choosing an appropriate downsampling factor to make the length of a downsampled signal be $O(K)$, no operations related to $N$ are required in sFFT-DT. Downsampling, however, possibly leads to “aliasing,” where different freq. grids become indistinguishable in terms of

†In fact, according to our private communication with the authors of [1][2], they would not recommend implementing this code since it is not trivial. The authors also suggest that it is not easy to clearly illustrate which setting will work best because of the constants in the Big-O functions and because of the dependency on the implementation. The authors themselves did not implement it since they believed that the constants would be large and that it would not realize much improvement over FFTW.
their locations and values. To overcome this problem, the locations and values of these $K$ non-zero entries are considered as unknown variables and the “aliasing problem” is reformulated as “moment-preserving problem (MPP),” which can be solved via existing approaches, such as that in [7]. Furthermore, sFFT-DT is conducted in a manner of a top-down iterative strategy under different downsampling factors, which can efficiently reduce the computational cost.

In this paper, the accurate computational cost and theoretical performance lower bound of sFFT-DT are proven further for exactly $K$-sparse signals. We also derive the Big-O constants of computational complexity of sFFT-DT and show that they are smaller than those of Ghazi et al.’s sFFT [6]. In addition, all operations of sFFT-DT are solved via analytical solutions but those of Ghazi et al.’s sFFT involve a numerical root finding algorithm, which is more complicated in terms of hardware implementation.

In previous works [1][2][6][8], performance and computational complexity have been analyzed based on the assumption that sparsity $K$ is known in advance. In practice, however, $K$ is unknown and an input parameter decided by the user. If $K$ is not guessed correctly, the performance is degraded and/or the computational overhead is higher than expected because the choice of some parameters depends on $K$. In this paper, we also propose a very simple solution to address this problem and relax this impractical assumption. We show that the cost for deciding $K$ is the same as that required for sFFT-DT with known $K$.

Different from our previous study [7], we further extend sFFT-DT to generally $K$-sparse signals in this paper, in addition to conducting more advanced theoretical analyses. For generally $K$-sparse signals, since almost all freq. grids are non-zero, each freq. grid of a downsampled signal is composed of significant and insignificant entries due to aliasing from downsampling. To extract significant entries from each freq. grid, the concept of sparse signal recovery from fewer samples, originating from compressive sensing (CS) [10], is employed since significant entries are “sparse”. Under this circumstance, the insignificant freq. grids act like Gaussian random noise due to central limit theory and the signal-to-noise ratio (SNR) between the significant and insignificant freq. grids affects the successful probability of sparse signal recovery.

Finally, we conclude that our methods are easy to implement and are demonstrated to outperform the state-of-the-art in terms of theoretical analyses and simulation results.

C. Organization of This Paper

The remainder of this paper is organized as follows. In Sec. II we describe the proposed method for exactly $K$-sparse signals. Our method for generally $K$-sparse signals will be expounded in Sec. III. Conclusions will be provided in Sec. IV.
TABLE I
NOTATION TABLE

| Notation | Description |
|----------|-------------|
| $x$, $X$ | The input signal in the time domain and frequency domain, respectively |
| $x_d$, $X_d$ | The downsampled signal in the time domain and frequency domain, respectively |
| $X[k]$ | $k$’th entry or freq. grid of $X$ |
| $X_d[k]$ | $k$’th entry or downsampled freq. grid of $X_d$ |
| $d$ | The downsampling factor |
| $l$ | The shift operator |
| $x_{d,l}$, $X_{d,l}$ | The downsampled signal with shift $l$ in the time domain and frequency domain, respectively |
| $N$ | The dimension of $x$ |
| $K$ | Sparsity |
| $N^+$ | The ratio of $N/d$ to $K$ |
| $m_k$ | $k$’th order moment |

II. SFFT-DT FOR EXACTLY $K$-SPARSE SIGNALS

We describe the proposed method for exactly $K$-sparse signals and provide analyses for parameter setting, computational complexity, and recovery performance. The proposed method contains three steps. 1) Downsample the original signal in the time domain. 2) Calculate Discrete Fourier Transform (DFT) of the downsampled signal by FFT. 3) Use the DFT of the downsampled signal to locate and estimate $K$ non-zero freq. grids of $X$. Steps 1 and 2 are simple and straightforward. Thus, we focus on Step 3 here. Table I where bold font represents a matrix or vector, summarizes the notations frequently used in this paper.

A. Problem Formulation

Let $x_d$ be the signal downsampled from an original signal $x$, where $x_d[k] = x[dk], k \in [0, \frac{N}{d} - 1]$, and integer $d \geq 1$ is a downsampling factor. The length of the downsampled signal $x_d$ is $\frac{N}{d}$. Let $X_d$ be DFT of $x_d$, where

$$X_d[k] = (X[k] + X[k + \frac{N}{d}] + X[k + 2\frac{N}{d}] + ... + X[k + (d - 1)\frac{N}{d}])/d. \tag{1}$$

The objective of this paper is to locate and estimate $K$ non-zero freq. grids of $X$ from the DFT $X_d$ of the downsampled signal $x_d$.

Note that each freq. grid of $X_d$ is a sum of $d$ terms of $X$. When more than two terms of $X$ are non-zero, “aliasing” occurs, as illustrated in Fig. I. Fig. I(a) shows an original signal in the frequency domain, where only three freq. grids are non-zero (appearing at normalized frequencies = 0π, 0.5π, and π). Fig. I(b) shows the downsampled signal in the frequency
domain when \( d = 2 \), where the downsampled freq. grid at \( 0 \pi \) incurs aliasing and is the sum of the original freq. grids at \( 0 \pi \) and \( \pi \), while no aliasing occurs at \( \pi \). In Fig. 1(b), we solve all non-zero downsampled freq. grids once, no matter whether aliasing occurs or not. This procedure is called non-iterative sFFT-DT and will be discussed in detail later. Instead of solving all of the downsampled freq. grids once, Fig. 1(c) illustrates an example of iteratively solving freq. grids. At the first iteration, the downsampled freq. grid without aliasing at \( 1 \pi \) is solved. This makes the remaining downsampled freq. grids more sparse. Then, the signal is downsampled again with \( d = 4 \). At the second iteration, we solve the downsampled freq. grid with aliasing at \( 0 \pi \). This procedure, called iterative sFFT-DT, will be discussed further in Sec. II-E.

![Fig. 1. Aliasing and its iterative solver. (a) Original signal in frequency domain. (b) Downsampled signal in frequency domain with \( d = 2 \). If we want to solve all freq. grids once, it requires 4 FFTs. (c) Similar to (b), however, the freq. grid (at normalized frequency \( 1 \pi \)) at \( d = 2 \) is solved first and requires 2 FFTs. (d) Remaining freq. grid (at \( 0 \pi \)) requires 2 extra FFTs at \( d = 4 \).](image-url)

To achieve the objective mentioned above, we first have to solve the aliasing problem of each freq. grid of \( X_d \). The shift property of DFT is useful in solving the aliasing problem. Let \( x_{d,l}[k] = x[dk + l] \), where \( l \) denotes the shift factor. Each freq. grid of \( X_{d,l} \) is denoted as:

\[
X_{d,l}[k] = (X[k]e^{i2\pi kl/N} + X[k + N_d]e^{i2\pi (k + N_d)l/N} + \ldots + X[k + (d - 1)N_d]e^{i2\pi (k + (d-1)N_d)l/N})/d. \tag{2}
\]

Thus, Eq. (2) degenerates to Eq. (1) when \( l = 0 \).

In practice, all we can obtain are \( X_{d,l}[k] \)'s for different \( l \)'s. For each downsampling factor \( d \), there will be no more than \( d \) terms on the right side of Eq. (2), where each term contains two unknown variables. For example, \( X[k]e^{i2\pi kl/N} \) is composed of two variables, \( X[k] \) and \( e^{i2\pi kl/N} \). Let \( a, 1 \leq a \leq d \), denote the number of terms on the right side of Eq. (2). Therefore, we
need $2a$ equations to solve these $2a$ variables, and $l$ is within the range of $[0, 2a - 1]$. By taking the above into consideration, the problem of solving the $2a$ unknown variables on the right side of Eq. (2) can be formulated as:

$$m_0 = p_0 z_0^0 + p_1 z_1^0 + ... + p_{a-1} z_{a-1}^0,$$

$$m_1 = p_0 z_0^1 + p_1 z_1^1 + ... + p_{a-1} z_{a-1}^1,$$

$$\vdots$$

$$m_{2a-1} = p_0 z_0^{2a-1} + p_1 z_1^{2a-1} + ... + p_{a-1} z_{a-1}^{2a-1},$$

(3)

where $X_{d,l}[k]$ is known and denoted as $m_l$ while $p_j$ and $z_j^i$ represent unknown $X[s_j]$ and $e^{i2\pi s_j/dN}$, respectively, for $s_j \in \{k, k + \frac{N}{a}, ..., k + (d-1)\frac{N}{a}\}$ and $j \in [0, a - 1]$. To simplify the notation, we let $S_k = \{k, k + \frac{N}{a}, ..., k + (d-1)\frac{N}{a}\}$ and $U_k = \{e^{i2\pi kl/N}, e^{i2\pi (k+\frac{N}{a})l/N}, ..., e^{i2\pi (k+(d-1)\frac{N}{a})l/N}\}$.

It is trivial that no aliasing occurs if $a = 1$, irrespective of the downsampling factor. Under this circumstance, we have $m_0 = X_{d,0}[k]$, $m_1 = X_{d,1}[k]$, $m_0 = p_0 z_0^0 = X[s_0]/d$, and $m_1 = p_0 z_1^1 = X[s_0]e^{i2\pi s_0/N}/d$, according to Eq. (3). We obtain that $|m_0| = |X[s_0]|/d = |m_1|$ and $m_1/m_0 = e^{i2\pi s_0/N}$. After some derivations, we can solve $s_0$ and obtain $X[s_0] = dX_{d,0}[k]$ at the position $s_0$. The above solution is based on the shift property of DFT and can only work under a non-aliasing environment. Nevertheless, when aliasing appears (i.e., $a > 1$), Eq. (3) is unsolvable because, for example, under $a = 2$, we have $\frac{m_1}{m_0} = \frac{p_0 z_0^1 + p_1 z_1^1}{p_0 z_0^0 + p_1 z_1^0}$.

To solve the aliasing problem, we consider Eq. (3) from another point of view. It is observed that the $i$’th row in Eq. (3) is the $i$’th moment with $m_i = \sum_{j=0}^{a-1} p_j z_j^i$. The issue of solving $p_j$’s and $z_j$’s given different moments ($m_i$’s) is known as the “Moment-preserving problem (MPP)”. In addition, MPP is also equivalent to the error locator polynomial problem in Ghazi et al.’s sFFT, which is a commonly used step in Reed-Solomon decoding [11]. The solution to MPP [9][12] based on orthogonal polynomials is useful and is discussed in the next subsection.

### B. The Solution to Moment-Preserving Problem

Note that the moment-preserving problem (Eq. (3)) is nonlinear and cannot be solved by simple linear matrix operations. On the contrary, we have to solve $z_j$’s first, such that Eq. (3) becomes linear. Then, $p_j$’s can be solved by matrix inversion. Thus, the main difficulty is how to solve $z_j$’s given known moments. According to [9], given the unique moments with $m_0$, $m_1$, ..., $m_{2a-1}$, there must exist the corresponding orthogonal polynomial equation, $P(z)$, with roots $z_j$’s for $0 \leq j \leq a - 1$. That is, $z_j$’s can be obtained as the roots of $P(z)$. The steps for solving MPP are as follows.

Step (i): Let the orthogonal polynomial equation $P(z)$ be:

$$P(z) = z^a + c_{a-1}z^{a-1} + ... + c_1 z + c_0,$$

(4)
The relationship between $P(z)$ and the moments is as follows:

$$-m_a = c_0 m_0 + c_1 m_1 + \ldots + c_{a-1} m_{a-1},$$

$$-m_{a+1} = c_0 m_1 + c_1 m_2 + \ldots + c_{a-1} m_a,$$

$$\vdots$$

$$-m_{2a-1} = c_0 m_{a-1} + c_1 m_a + \ldots + c_{a-1} m_{2a-2}.$$  \hspace{1cm} (5)

Eq. (5) can be formulated as $m = M c$, where $M_{i,j} = m_{i+j}$, $c = \begin{bmatrix} c_0 \ c_1 \ \ldots \ c_{a-1} \end{bmatrix}^T$, and $m = \begin{bmatrix} -m_a & -m_{a+1} & \ldots & -m_{2a-1} \end{bmatrix}^T$.

Thus, Eq. (5) can be solved by matrix inversion $M^{-1}$ to obtain $c_j$'s.

Step (ii): Find the roots of $P(z)$ in Eq. (4). These roots are the solutions of $z_0, z_1, \ldots, z_{a-1}$, respectively.

Step (iii): Substitute all $z_j$'s into Eq. (3), and solve the resulting equations to obtain $p_j$'s.

Tsai \cite{12} proposed a complete analytical solution composed of the aforementioned three steps for $a \leq 4$, based on the constraint that $p_0 + p_1 + \ldots + p_{a-1} = 1$. Nevertheless, for the aliasing problem considered here, the constraint is $p_0 + p_1 + \ldots + p_{a-1} = X_{d,0}[k]$, as indicated in Eq. (2). Thus, we derive the complete analytical solution for $a = 2$ as:

$$c_d = \begin{vmatrix} m_0 & m_1 \\ m_1 & m_2 \end{vmatrix},$$

$$c_0 = \left( \frac{1}{c_d} \right) \begin{vmatrix} -m_2 & m_1 \\ -m_3 & m_2 \end{vmatrix}, \quad c_1 = \left( \frac{1}{c_d} \right) \begin{vmatrix} m_0 & -m_2 \\ m_1 & m_3 \end{vmatrix},$$

$$z_0 = \frac{1}{2} \left[ -c_1 - (c_1^2 - 4c_0)^{1/2} \right], \quad z_1 = \frac{1}{2} \left[ -c_1 + (c_1^2 - 4c_0)^{1/2} \right],$$

$$p_d = z_1 - z_0,$$

$$p_0 = \left( \frac{1}{p_d} \right) \begin{vmatrix} m_0 & 1 \\ m_1 & z_1 \end{vmatrix}, \quad p_1 = m_0 - p_0.$$  \hspace{1cm} (6)

The analytical solutions for an univariate polynomial with $a \leq 4$ cost $O(a^2)$ operations. Since there are $\frac{N}{d}$ freq. grids, the computational cost of MPP is $O(\frac{N}{d} a^2)$. For $a > 4$, Step (i) still costs $O(a^2)$, according to the Berlekamp-Massey algorithm \cite{13}, which is well-known in Reed-Solomon decoding \cite{11}. In addition, Step (iii) is designed to calculate the inverse matrix of a Vandermonde matrix and costs $O(a^2)$ \cite{14}. There is, however, no analytical solution of Step (ii) for $a > 4$. Thus, numerical methods of root finding algorithms with finite precision are required. A fast algorithm proposed by Pan \cite{15} can approximate all of the roots with $O(a (\log \log N)^{O(1)})$, where the detailed proof was shown in \cite{6}. If $(\log \log N)^{O(1)} > a$, Step (ii) will dominate the cost of solving the MPP.

It is noted that the actual number of aliasing terms for each freq. grid, $a$, is unknown in advance. In practice, we let $a_m$ be user-defined and denoted as the maximum number of aliasing terms for all downsampled freq. grids during the whole

\textsuperscript{†}The solutions for $a = 3$ and $a = 4$ are described in the Appendix.
process of solving the MPP. Therefore, $2a_m$ moments are required to solve the MPP. If $a$’s of all downsampled freq. grids are smaller than or equal to $a_m$, the MPP perfectly recovers all of the freq. grids; i.e., it resolves all non-zero values of $X$ and their corresponding locations. Otherwise, some non-zero entries of $X$ cannot be recovered due to insufficient information. Although a larger $a_m$ guarantees better recovery performance, it also means that more moments and higher computational cost are required.

In sum, the cost of the MPP consists of two parts. Since the size of a downsampled signal is $N/d$, the cost of generating the required moments via FFT is $O(2a_m N d \log N d)$, which is called the “P1 cost of MPP” hereafter. Second, as previously mentioned, solving the aforementioned Steps (i), (ii), and (iii) for MPP will cost $O(\frac{N d a_m^2}{d})$ for $a_m \leq 4$ and cost $O(\frac{N d a_m (\log \log N)^{O(1)}}{d})$ for $a_m > 4$, where either of which is defined as the “P2 cost of MPP”.

So far, our method of solving all downsampled freq. grids is based on fixing downsampling factor $d$ (and $a_m$), as an example illustrated in Fig. 1(b). In this case, we call this approach, non-iterative sFFT-DT. Its iterative counterpart, iterative sFFT-DT, which is the focus of this paper, will be described later in Secs. II-E and II-F.

C. MPP in sFFT-DT and Ghazi et al.’s sFFT

Following the discussions in the previous subsection, the P2 cost of MPP, in fact, is equivalent to solving the error locator polynomial of Reed-Solomon decoding in Ghazi et al.’s sFFT [6]. Thus, the computational cost of Ghazi et al.’s sFFT, similar to ours, still is composed of the P1 cost and P2 cost of MPP. The main difference between our non-iterative sFFT-DT and Ghazi et al.’s sFFT [6] is how to set $a_m$ and $d$. In non-iterative sFFT-DT, $a_m = 4$ is fixed and an appropriate $d$ is selected to make $a$’s of most downsampled freq. grids smaller than or equal to $a_m$. Under this situation, the total cost is $O(2a_m \frac{N}{d} \log \frac{N}{d} + \frac{N}{d} a_m^2)$.

On the other hand, Ghazi et al.’s sFFT is to set $d = \frac{N \log K}{K}$ and $a_m = C \log K$ ($C$ is a constant), and the total cost is $O(2a_m \frac{N}{d} \log \frac{N}{d} + \frac{N}{d} a_m (\log \log N)^{O(1)})$. In fact, these two kinds of parameter settings lead to different computational cost and recovery performance. We will discuss this issue further in the next subsection.

Moreover, in order to reduce computational cost in our framework further, a top-down iterative strategy for sFFT-DT will be presented in Sec. II-E.

D. Analysis

In this section, we first will study the relationship between $a_m$ and $d$ and analyze the probability of a downsampled freq. grid with an aliasing number larger than $a_m$. Second, we will discuss computational complexity and recovery performance of our non-iterative sFFT-DT. Third, we will compare our method with Ghazi et al.’s sFFT [6]. In addition, the Big-O constant of
complexity is induced in order to highlight the computational simplicity of non-iterative sFFT-DT. Finally, we will conclude by presenting an iterative sFFT-DT to reduce computational cost further.

1) Relationship between Maximum Aliasing Number and Downsampling Factor: Now, we consider the relationship between \( a_m \) and \( d \). If \( a_m \) is set to \( d \), then we always can recover any \( X \) without errors but the computational cost will be larger than that of FFT. Thus, it is preferable to set \( 1 \leq a_m < d \). In fact, small \( a_m \) is feasible when \( X \) is uniformly distributed. For each freq. grid, the number of aliasing, \( a \), will be small with high probability if \( \frac{N}{dK} \) is larger than 1. Let \( N^+ = \frac{N}{dK} \) denote the ratio of the length \( \frac{N}{d} \) of a downsampled signal to \( K \) hereafter. Our empirical observations, shown in Fig. 2 indicate the probability of aliasing at different \( N^+ \)'s. For \( a > 4 \), the probability of aliasing is very low. Therefore, if \( a_m \) is set to 4, 8 FFTs for a downsampled signal are required and most freq. grids of \( X \) can be recovered. One can see that the performance of sparse FFT is directly related to the number of downsampled freq. grids with \( a > a_m \). When \( a > a_m \), it means that the MPP cannot attain the correct values and locations in the frequency domain, leading to degraded performance. Lemma 1 shows the probability that a downsampled freq. grid with aliasing number \( a > a_m \) is related to \( \frac{N}{d} \), \( K \), and \( a_m \). Since \( \frac{N}{d} \) and \( a_m \) are user-defined, it can be very low, based on an appropriate setting, which is proven further in Theorem 1 and Theorem 2.

![Fig. 2. The distribution of all \( a \)'s at different \( N^+ \)'s, where \( a \) denotes the number of aliasing terms on the right side of Eq. (2). The results show that aliasing (\( a > 2 \)), in fact, seldom occurs.](image)

**Lemma 1.** Suppose \( K \) non-zero entries distribute uniformly (i.e., with probability \( \frac{K}{N} \)) in \( X \). Let \( Pr(d, a_m) \) denote the probability that there is at least a downsampled freq. grid with aliasing number \( a > a_m \) when the downsampling factor is \( d \). Then, \( Pr(d, a_m) \leq \frac{N}{d} \left( \frac{eK}{N(a_m+1)} \right)^{a_m+1} \), where \( e \) is Euler’s. The probability that \( X \) can be perfectly reconstructed is defined as \( \delta = 1 - Pr(d, a_m) \).
Proof: For each downsampled freq. grid, the probability of \( a > a_m \) is \( \sum_{i=a_m+1}^{d} \left( \frac{d}{a_m + 1} \right) \left( \frac{K}{N} \right)^{a_m + 1} \), which is smaller than \( \left( \frac{d}{a_m + 1} \right) \left( \frac{K}{N} \right)^{a_m + 1} \). Under this circumstance, the probability of at least a downsampled freq. grid with \( a > a_m \) is bounded by \( \frac{N}{d} \left( \frac{d}{a_m + 1} \right) \left( \frac{K}{N} \right)^{a_m + 1} \). Thus, we can derive:

\[
\Pr(d, a_m) \leq \frac{N}{d} \left( \frac{d}{a_m + 1} \right) \left( \frac{K}{N} \right)^{a_m + 1} = \frac{N}{d} \frac{d!}{(d - a_m - 1)!(a_m + 1)!} \left( \frac{K}{N} \right)^{a_m + 1} \leq \frac{N}{d} \frac{dK}{N} \left( \frac{eK}{N} \right)^{a_m + 1} \leq \frac{N}{d} \frac{dK}{N} \left( \frac{eK}{N} \right)^{a_m + 1} \tag{7}
\]

(7)

2) Computational Cost and Recovery Performance: From Lemma [1] non-iterative sFFT-DT obtains perfect recovery with probability \( \delta = 1 - \Pr(d, a_m) \) at least. Now, we first discuss our approach with fixed \( a_m = 4 \), under which most freq. grids can be solved, as an example illustrated in Fig. [2]. The computational cost and recovery performance of non-iterative sFFT-DT are depicted further in Theorem [1] and Theorem [2] respectively, based on different settings of \( d \).

**Theorem 1.** If non-zero freq. grids of \( X \) distribute uniformly, given \( a_m = 4 \) and \( \delta \), the upper bound of \( d \) is \( 2.15(1 - \delta)^{1/4} NK^{-5/4} \). sFFT-DT perfectly recovers \( X \) with probability at least \( \delta \) and the computational cost is \( O(\frac{1}{(1 - \delta)^{1/4}} K^{\frac{5}{4}} \log K) \).

**Proof:** According to Eq. (7), we can derive:

\[
\delta \leq 1 - \frac{N}{d} \left( \frac{dK}{N(a_m + 1)} \right)^{a_m + 1} \Rightarrow d^4 \left( \frac{eK}{5N} \right)^5 \leq \left( \frac{1 - \delta}{N} \right) \Rightarrow d \leq 2.15(1 - \delta)^{1/4} NK^{-5/4}, \tag{8}
\]

(8)

where \( \frac{1}{(\frac{eK}{5N})^5} \approx 2.15 \). Thus, according to the MPP cost derived in Sec. [1-B] the P1 cost of MPP is \( O(\frac{1}{(1 - \delta)^{1/4}} K^{\frac{5}{4}} \log K) \), the P2 cost of MPP is \( O(\frac{1}{(1 - \delta)^{1/4}} K^{\frac{5}{4}}) \), and the total cost is \( O(\frac{1}{(1 - \delta)^{1/4}} K^{\frac{5}{4}} \log K) \).

**Theorem 2.** If non-zero freq. grids of \( X \) distribute uniformly, given \( a_m = 4 \) and \( d = O(\frac{N}{K}) \), the output of sFFT-DT perfectly recovers \( X \) with the probability upper bounded by \( 1 - O(K)(\frac{e}{5})^5 \) with the computational cost being \( O(K \log K) \).

**Proof:** According to the leftmost inequality of Eq. (8) plus \( d = O(\frac{N}{K}) \), we can derive the upper bound of the probability of recovery as:

\[
\delta \leq 1 - \frac{N}{d} \left( \frac{dK}{N(a_m + 1)} \right)^{a_m + 1} \Rightarrow \frac{K}{N} \left( \frac{e}{5} \right)^5 \leq \left( \frac{1 - \delta}{N} \right) \Rightarrow \delta \leq 1 - O(K)(\frac{e}{5})^5. \tag{9}
\]

(9)

In addition, given \( d = O(\frac{N}{K}) \), the computational cost can be derived to be \( O(K \log K) \), as described in Sec. [1-B].

In summary, Theorem [1] guarantees perfect recovery with probability being larger than \( \delta \) and computational cost being bounded by \( O(K^{\frac{5}{4}} \log K) \), whatever \( K \) is, while Theorem [2] achieves the ideal computational cost \( O(K \log K) \) with recovery performance being degraded along with the increase of \( K \).
3) Comparison with [6]: We will compare our results in Theorem 1 and Theorem 2 with the one described in Theorem 3 below.

**Theorem 3.** *(Rewritten from Theorem 4.6 [6])* If non-zero freq. grids of $X$ distribute uniformly, given $a_m = C \log K$ and $d = O\left(\frac{N \log K}{K}\right)$, Ghazi et al.’s sFFT costs $O(K \log K + K (\log K)^{(1)})$ for perfect recovery with probability $1 - O\left(\frac{1}{K^{0.5 \log C}}\right)$.

Ghazi et al.’s sFFT aims to minimize the cost without the constraint of $a_m \leq 4$. Compared to our results in Theorem 1 and Theorem 2, Ghazi et al.’s method picks a larger downsampling factor and solves MPP under a larger $a_m$. Thus, a root finding algorithm [15] is required and its complexity is related to the signal length $N$. The seeming advantage of Ghazi et al.’s sFFT is that its computational cost is lower than that of sFFT-DT, derived in Theorem 1, and its performance becomes better with the increase of $K$. Nevertheless, if we further analyze the practical cost of additions and multiplications in detail, namely the Big-O constants of computational complexity, we can find that the cost in Theorem 3 is often larger than those derived in Theorem 1 and Theorem 2. Moreover, we shall show later that the Big-O constants in Theorem 2 are larger than those in Theorem 1.

Recall that the computational cost of our method sFFT-DT is composed of two parts: performing FFTs for obtaining moments (the P1 cost of MPP) and solving Steps (i), (ii), and (iii) of MPP (the P2 cost of MPP). Since the Big-O constants of addition and multiplication of complex-valued split-radix FFT [16] are about $3N \log N$ and $2N \log N$, respectively, the Big-O constants for FFT in Theorem 1 are about 36 with $\delta = 0.99$ for addition and 24 for multiplication. It should be noted that the P2 cost of MPP in Theorem 1 can be ignored since it only needs $\frac{N}{d} a_m^2 = O(K^{1/2})$, which is relatively smaller than the P1 cost of MPP. For Theorem 2 since $d = O\left(\frac{N}{K}\right)$ and was set to $\frac{N}{4K}$ in our simulations, the Big-O constants for FFT are 96 for addition and 64 for multiplication. Since the P2 cost of MPP in Theorem 2 ($\frac{N}{d} a_m^2 = O(K)$) is also relatively smaller than the P1 cost of MPP, it is ignored.

In contrast to Theorem 1 and Theorem 2, the Big-O constants of the P1 cost of MPP in Theorem 3 [6] are about $6C$ for addition and $4C$ for multiplication ($C$ must be larger than or equal to 2; otherwise Ghazi et al.’s sFFT cannot work). Nevertheless, the Big-O constants of one of the Steps (i) and (iii) within the P2 cost of MPP need about 96 for addition and 160 for multiplication (the detailed cost analysis is based on [14]). Even though we do not take Step (ii) into account due to the lack of detailed analysis, the Big-O constants for multiplication in Theorem 1 and Theorem 2 are far smaller than those of Ghazi et al.’s sFFT. In addition, Ghazi et al.’s sFFT should be more complex than Theorem 1 and Theorem 2 because an extra numerical procedure is required and the computational cost involves $N$. From the aspect of hardware design, an analytical

\[\text{Recall that the P1 cost of MPP is } N \cdot \log \frac{N}{d} \cdot 2a_m \frac{N}{d} \text{. Under the situation that } a_m = 4 \text{ and } \frac{N}{d} = \frac{K^{5/4}}{2.15(1 - \delta)^{1/4}} \text{, the Big-O constant is } 2a_m \frac{N}{d} \times 3 = \frac{2a_m}{2.15(1 - \delta)^{1/4}}, \text{ where } 3 \text{ comes from the constant of additions of FFT [19]. For } \delta = 0.99, \text{ the Big-O constant is about 36.}\]
solution is easier to implement than a numerical procedure. We conclude that there are two main advantages in Theorem 1 and Theorem 2 compared to Theorem 3 [6]. First, the Big-O constants in Theorem 1 and Theorem 2 are smaller than those of Ghazi et al.’s sFFT. Second, our analytical solution is hardware-friendly in terms of implementation.

On the other hand, when the signal is not so sparse with \( K \) approaching \( N \) (e.g., \( K = \frac{N}{8} \) and \( N^+ = 2^9 \)), the cost of 8 FFTs in a downsampled signal is almost equivalent to that of one FFT in the original signal. To further reduce the cost, a top-down iterative strategy is proposed in the next subsection.

It also should be noted that the above discussions (and prior works) are based on the assumption that \( K \) is known. In practice, \( K \) is unknown in advance. Unfortunately, how to automatically determine \( K \) is ignored in the literature. Instead of skipping this problem, in this paper, we present a simple but effective strategy in Sec. II-H to address this issue.

\[ E. \ Top-Down \ Iterative \ Strategy \ for \ Iterative \ sFFT-DT \]

According to Fig. 2, the probability of aliasing decreases fast with the increase of \( a \) when \( N^+ \geq 1 \). Thus, we can solve these downsampled freq. grids with \( a = 1 \) before subtracting these solved freq. grids from \( X_d \). This will make \( X_d \) more sparse. Then, under this circumstance, \( d \) subsequently can be set to be larger values to reduce computational cost without sacrificing the recovery performance. Recall that, when \( d \) is increased, \( a \) is increased as well. In this section, an iterative strategy is proposed to solve the aliasing problem with an iterative increase of the downsampling factor \( d \). Fig 1 illustrates such an example. In Fig. 1(b), if we try to solve all aliasing problems in the first iteration, 4 FFTs are required, since the maximum value of \( a \) is 2. On the other hand, if we first solve the downsampled freq. grids with \( a = 1 \) (at normalized frequency = \( \pi \)), it costs 2 FFTs, as shown in Fig. 1(c). Since 2 FFTs are insufficient for solving the aliasing problem completely under \( a = 2 \), extra 2 FFTs are required to solve a more sparse signal.

The key is how to calculate the 2 extra FFTs in the above example with lower cost. The idea motivated by sFFT is to discard the solved freq. grids in the frequency domain. Thus, if, for example, \( K' \) freq. grids are subtracted from the original \( X \), the sparsity of the remaining signal is \( K - K' \). Since a more sparse signal is generated in an iterative manner, \( d \) can be set to be larger under fixed \( N^+ \). As shown in Fig. 1(d), 2 extra FFTs can be done quickly with a larger \( d (=4) \) to solve the downsampled freq. grid with \( a = 2 \) (at normalized frequency = \( 0\pi \)). Consequently, \( d \) is doubled iteratively in our method and the total cost is dominated by that required at the first iteration.

The proposed method with the top-down iterative strategy is called iterative sFFT-DT.
In this section, our method, iterative sFFT-DT, is developed based on Theorem 2 and is depicted in Algorithm 1, which is composed of three functions, main, SubFreq, and MPP. In fact, the only difference between Theorem 1 and Theorem 2 is how to initialize \(d\). Thus, the algorithm iterative sFFT-DT is modified easily to satisfy Theorem 1. Basically, iterative sFFT-DT solves downsampled freq. grids from \(a_m = 1\) to \(4\) with an iterative increase of \(d\). Note that, its variation, non-iterative sFFT-DT, solves all downsampled freq. grids with \(a_m = 4\) and fixed \(d\).

At the initialization stage, the sets \(S\) and \(T\), recording the positions of solved and unsolved freq. grids, respectively, are set to be empty. \(a_m\) and \(d\) are initialized via Theorem 2. The algorithmic steps are explained in detail as follows.

Function main, which is executed in a top-down manner by doubling the downsampling factor iteratively, is depicted from Line 1 to Line 16. In Lines 3-4, the input signal \(x\) is represented by two shift factors \(2^l\) and \(2^l+1\). Then they are used to perform FFT to obtain \(X_{d,2l}\) and \(X_{d,2l+1}\) in Lines 5-6. In Line 7, the function SubFreq, depicted between Line 17 and Line 22, is executed to remove freq. grids from \(X_{d,2l}\) and \(X_{d,2l+1}\) that were solved in previous iterations. The goal of function SubFreq is to make the resulting signal more sparse.

Line 9 in function main is used to judge if there are still unsolved freq. grids. In particular, the condition \(X_{d,l}[k] = 0\), initially defined in Eq. (2), may imply: 1) \(X[k + j\frac{N}{d}]\)'s for all \(j \in [0, d - 1]\) are zero, meaning that there is no unsolved freq. grid and 2) \(X[k + j\frac{N}{d}]\)'s are non-zero but their sum is zero, meaning that there exist unsolved freq. grids. To distinguish both, \(|X_{d,j}[k]| > 0\) for \(j \in [0, 2l + 1]\) is a sufficient condition. More specifically, if \(a\) is less than or equal to \(2l + 2\), it is enough to distinguish both by checking whether any one of the \(2l + 2\) equations is not equal to 0. If yes, it implies that at least a frequency grid is non-zero; otherwise, all \(X[k + j\frac{N}{d}]\)'s are definitely zero. Moreover, Line 9 is equivalent to checking \(2l + 2\) equations at the \(l\)'th iteration. At \(l = 0\), two equations \((X_{d,0}[k] \text{ and } X_{d,1}[k])\) are verified to ensure that all freq. grids with \(a \leq 2\) are distinguished. At \(l = 1\), if \(k \in T\), it is confirmed that \(X[k + j\frac{N}{d}]\)'s are non-zero at the previous iteration. On the contrary, if \(k \notin T\), extra 2 equations \((X_{d,2}[k] \text{ and } X_{d,3}[k])\) are added to ensure that all freq. grids with \(a \leq 4\) are distinguished. Thus, at the \(l\)'th iteration, there are in total \(2l + 2\) equations checked.

In Line 11, the function MPP, depicted in Lines 23-35 (which was described in detail in Sec. II-B), solves freq. grids when aliasing occurs. sFFT-DT iteratively solves downsampled freq. grids from \(a = 1\) to \(a \leq a_m = 4\). Nevertheless, we do not know \(a\)'s in advance. For example, it is possible that some downsampled freq. grids with \(a = 4\) are solved in the first three iterations, and these solutions definitely fail. In this case, the solved locations do not belong to \(S_k\) (defined in Sec. II-A). On the contrary, if the downsampled freq. grid is solved correctly, the locations must belong to \(S_k\). Thus, by checking whether or not the solution satisfies the condition, \(s_j \mod d = k\) for all \(j \in [0, l]\) (Line 30), we can guarantee that all downsampled freq.
grids are solved under correct a’s. Finally, the downsampling factor is doubled, as indicated in Line 14, to solve the unsolved freq. grids in an iterative manner. This means that the downsampled signal in the next iteration will become shorter and can be dealt faster than that in the previous iterations.

Algorithm 1 Iterative sFFT-DT for exactly K-sparse signals.

| Algorithm 1 Iterative sFFT-DT for exactly K-sparse signals. |
|-------------------------------------------------------------|
| **Input:** x, K; **Output:** X;                            |
| **Initialization:** X = 0, d = O(N/K), S = {}, T = {}, a_m = 4; |
| 01. function main()                                        |
| 02. for l = 0 to a_m - 1                                    |
| 03. x_d,2l[k] = x[dk + 2l] for k ∈ [0, N/d - 1];            |
| 04. x_d,2l+1[k] = x[dk + 2l + 1] for k ∈ [0, N/d - 1];      |
| 05. X_d,2l = FFT(x_d,2l) × d;                              |
| 06. X_d,2l+1 = FFT(x_d,2l+1) × d;                          |
| 07. SubFreq(X_d,2l, X_d,2l+1, X, d, l, S);                 |
| 08. for k = 0 to N/d - 1                                    |
| 09. if (k ∈ T or |X_d,2l[k]| > 0 or |X_d,2l+1[k]| > 0)     |
| 10. m_j = X_d,j[k] for j ∈ [0, 2l + 1];                     |
| 11. MPP(m, l, d, k, X, S, T);                              |
| 12. end if                                                  |
| 13. end for                                                |
| 14. d = 2d;                                                |
| 15. All elements in T modulo N/d.                          |
| 16. end for                                                |
| 17. function SubFreq (X_d,2l, X_d,2l+1, X, d, l, S)        |
| 18. for k ∈ S                                             |
| 19. k_d = k mod N/d;                                       |
| 20. X_d,2l[k_d] = X_d,2l[k_d] - X[k]e^{i2πk_d(2l+1)/N};    |
| 21. X_d,2l+1[k_d] = X_d,2l+1[k_d] - X[k]e^{i2πk_d(2l+1)/N}; |
| 22. end for                                                |
| 23. function MPP (m, l, d, k, X, S, T)                     |
| 24. if l = 0                                              |
| 25. z_0 = (m_1/m_0); p_0 = m_0;                            |
| 26. else                                                  |
| 27. Solve the aliasing problem with a = l + 1 by            |
| the solution to MPP, described in Sec. II-B                |
| 28. end if                                                |
| 29. s_j = (ln z_j)N/2iπ for all j ∈ [0, l];                 |
| 30. if (s_j mod d) = k for all j ∈ [0, l]                  |
| 31. S = S ∪ s;                                            |
| 32. X[s_j] = p_j for all j ∈ [0, l];                       |
| 33. else                                                  |
| 34. T = T ∪ s;                                            |
| 35. end if                                                |
G. Performance and Computational Complexity of Iterative sFFT-DT

We first discuss the complexity of iterative sFFT-DT. The cost of the outer loop in function **main** (Steps 5 and 6) is bounded by two FFTs. As mentioned in Theorem 2, \( d \) is set to be \( O\left(\frac{N}{K}\right) \), the dimensions of \( x_{d,2l} \) and \( x_{d,2l+1} \) are \( O(K) \), and FFT costs \( O(K \log K) \) in the first iteration. Since \( d \) is doubled iteratively, the total cost of \( a_m \) iterations is still bounded by \( O(K \log K) \).

In addition, the function **SubFreq** costs \( O(K) \) operations due to \( |S| \leq K \).

The inner loop of the function **main** totally runs \( O(K) \) times, which is not related to the outer loop, since at most \( K \) freq. grids must be solved. The cost at each iteration is bounded by the function **MPP**. Recall that the P2 cost of MPP, as described in Sec. II-B requires \( O\left(\frac{N}{d}a^2\right) \). More specifically, since \( d \) is doubled iteratively, \( \frac{N}{d} \) can be derived to depend on \( O\left(\frac{K}{2^l}\right) \) from the initial setting \( d = O\left(\frac{N}{K}\right) \). Therefore, MPP at the \( l \)'th iteration costs \( O\left(\frac{K}{2^l}(l+1)^2\right) \), due to \( a = l+1 \), illustrated in Line 27 of Algorithm 1 and requires \( O\left(\frac{K}{2^l}1^2 + \frac{K}{2^l}2^2 + ... + \frac{K}{2^l}4^2\right) \leq O\left(6.25K\right) = O(K) \) in total. That is, the inner loop (Steps 8~13) costs \( O(K) \), given an initial downsampling factor of \( d = O\left(\frac{N}{K}\right) \) and \( a_m = 4 \).

In sum, the proposed algorithm, iterative sFFT-DT, is dominated by “FFT” and costs \( O(K \log K) \) operations. Now, we discuss Big-O constants for operations of addition and multiplication, respectively. Since \( d \) is doubled iteratively, the P1 cost of MPP gradually is reduced in the later iterations. The total cost is \( \sum_{i=1}^{a_m}O\left(2\frac{N}{2^i-1}d\log\frac{N}{2^i-1}d\right) \), where \( a_m = 4 \). Due to the fact that iterative sFFT-DT possibly recovers \( X \) with less than \( a_m = 4 \) iterations, the benefit in reducing the computational cost depends on the number of iterations. In the worst case, the cost is about \( O\left((2 + 1 + \frac{1}{2} + \frac{1}{4})\frac{N}{d}\log\frac{N}{d}\right) = O\left(3.75\frac{N}{d}\log\frac{N}{d}\right) \) under \( a_m = 4 \). Recall that the P1 cost of MPP in non-iterative sFFT-DT is \( O\left(2a_m\frac{N}{d}\log\frac{N}{d}\right) \). With \( a_m = 4 \), the Big-O constants in non-iterative sFFT-DT are two times larger than those in iterative sFFT-DT. Similarly, in the best case (i.e., \( a \)'s of all freq. grids are 1), the former is about 4 times larger than the latter. Thus, it is easy to further infer Big-O constants of iterative sFFT-DT. For instance, since the Big-O constant of addition for non-iterative sFFT-DT is 36 (in Theorem 1), the Big-O constants for iterative sFFT-DT addition range from \( \frac{36 \times 2}{8} = 9 \) (the best case) to \( \frac{36 \times 3.75}{8} \approx 18 \) (the worst case) and those for multiplication range from 6 to 12.

As for recovery performance in iterative sFFT-DT, since the downsampling factor \( d \) is doubled along with the increase of iterations, a question, which naturally arises, is if a larger downsampling factor leads to more new aliasing artifacts. If yes, these newly generated aliasing artifacts degrade the performance of iterative sFFT-DT. If no, the iterative style is good since it reduces computational cost and maintains recovery performance.

In Lemma 2 we prove that the probability of producing new aliasing artifacts after a sufficient number of iterations will approach zero.

**Lemma 2.** Suppose \( K \) non-zero entries of \( X \) distribute uniformly (i.e., with probability \( \frac{K}{N} \)). Let \( Pr_{i}^{\text{ali}} \) be the probability
that new aliasing artifacts are produced at the $l$'th iteration in iterative sFFT-DT. Let $K_l$ be the number of freq. grids with $a \geq l + 1$ at the $l$'th iteration ($0 \leq l \leq 3$, $K_0 = K$). If $K_l \leq \frac{K}{2^l}$, we have $P_{\text{rali}}^l < \frac{1}{2^{l+1}} \left( \frac{d}{N} \right) K^2$.

Proof: According to Algorithm 1 after the first iteration ($l = 0$), all downsampled freq. grids with only $a = 1$ aliasing term are solved. Thus, we focus on discussing the probability of producing new aliasing artifacts under $l \geq 1$. By the same idea of Lemma 1, we can define $\frac{N}{d} \left( \frac{d}{2} \right) \left( \frac{K}{N} \right)^2$ to be the probability that there is a downsampled freq. grid with $a \geq 2$ aliasing terms. Under the condition of $l \geq 1$ and $a \geq 2$, however, some non-zero freq. grids have been solved in previous iterations. Thus, the number of remaining non-zero freq. grids are no longer $K$ and $\frac{K}{N}$ and should be modified. In other words, the number of downsampled freq. grids with $a \geq 2$ must be less than $K_l$ for $l \geq 1$ and $\frac{N}{d} \left( \frac{d}{2} \right) \left( \frac{K_l}{N} \right)^2$ becomes the upper bound of the probability that there exists a downsampled freq. grid of producing new aliasing artifacts.

According to our iterative sFFT-DT algorithm, let $d_l = 2^l d$ for $l \geq 1$. We can derive:

$$P_{\text{rali}}^l \leq \frac{N}{d_l} \left( \frac{d_l}{2} \right) \left( \frac{K_l}{N} \right)^2 \leq \frac{N}{2^l d} \left( \frac{2^l d}{N} \right)^2 \left( \frac{K_l}{N} \right)^2 \leq 2^{l-1} \left( \frac{d}{N} \right) K_l^2.$$  \hspace{1cm} (10)

Eq. (10) converges to 0 only when $K_l \leq \frac{K}{2^l}$. As shown in Fig. 2 by initializing $d$ properly, almost $K$ freq. grids can be solved in the first few iterations. This makes $K_l \leq \frac{K}{2^l}$ easy to be satisfied. Under this circumstance, $N^+ \in [2^1, 2^3]$ would be a good choice. By replacing $K_l$ with $\frac{K}{2^l}$, we can derive $P_{\text{rali}}^l \leq \frac{1}{2^{l+1}} \left( \frac{d}{N} \right) K^2$. When $l$ increases to be large enough, the probability of $P_{\text{rali}}^l$ will be small because of $\frac{1}{2^{l+1}} \left( \frac{d}{N} \right) K^2 \rightarrow 0$.

Lemma 2 indicates the probability of producing new aliasing artifacts in an asymptotic manner. This provides us the information that the probability of producing new aliasing artifacts finally converges to zero. In our simulations, we will show that the exact probability with new aliasing is very low if $d$ is selected based on either Theorem 1 or Theorem 2. In other words, the iterative approach can reduce the computational cost and maintain the recovery performance when there is no new aliasing generated at later iterations.

H. A Simple Strategy for Estimating Unknown Sparsity $K$

As previously described, the sparsity $K$ of a signal is important in deciding the downsampling factor $d$. Nevertheless, $K$ is, in general, unknown. In this section, we provide a simple bottom-up strategy to address this issue.

First, we set a large downsampling factor $d = N$, and then run sFFT-DT. If there is any downsampled freq. grid that cannot be solved, then $d$ is halved and sFFT-DT is applied to solve $X$ again. When $d$ is halved iteratively until the condition in either Theorem 1 or Theorem 2 is satisfied, sFFT-DT guarantees one to stop with the probability indicated in either Theorem 1 or Theorem 2. This strategy needs the same computational complexity required in sFFT-DT with known $K$ because the cost
with \( d = N \) is \( O(2a_m \frac{N}{d} \log \frac{N}{d}) = O(2a_m) \) and the total cost, thus, is \( O(2a_m) + O(2a_m 2 \log 2) + \ldots + O(2a_m K \log K) < O(4a_m K \log K) \). Therefore, sFFT-DT with the strategy of automatically determining \( K \) costs double the one with known \( K \).

I. Simulation Results for Exactly K-Sparse Signals

Our method, iterative sFFT-DT, was numerically verified and compared with FFTW (http://www.fftw.org/), sFFT-v3 [1] (its code was downloaded from http://spiral.net/software/sfft.html), and Ghazi et al.’s sFFT [6] (based on Theorem 3) for exactly \( K \)-sparse signals. The simulations for sFFT-DT, FFTW, and Ghazi et al.’s sFFT were conducted with an Intel CPU Q6600 and 2.99 GB RAM under Win 7. sFFT-v3 was run in Linux because the source code was released in Linux’s platform. The signal \( x \) in time domain was produced as follows: 1) Generate a \( K \)-sparse signal \( X_{ori} \) and 2) \( x \) is obtained by inverse FFT of \( X_{ori} \).

For sFFT-DT, the initial \( d \) is set according to \( d = O(N \frac{K}{K}) \), based on Theorem 2. More specifically, \( d = \frac{N}{4K} \) was adopted here. For sFFT-v3, \( d \) was automatically assigned, according to the source code. For Ghazi et al.’s sFFT, \( d = \frac{N}{K} 2^{\lceil \log \log K \rceil} \) and \( a_m = 2 \log K \), where \( 2^{\lceil \log \log K \rceil} \) is involved to enforce \( \frac{N}{d} \) being an integer. If \( \log \log K \) is an integer, \( d = \frac{N}{K} 2^{\lceil \log \log K \rceil} = \frac{N \log K}{K} \).

Fig. 3(a) shows the results of computational time versus sparsity under \( N = 2^{24} \). For \( K \leq \frac{N}{2^4} \), our algorithm outperforms FFTW. Moreover, sFFT-v3 [1][2] is only faster than FFTW when \( K \leq \frac{N}{2^6} \) and is comparable to Ghazi et al.’s sFFT. We can also observe from Fig. 3(a) that Ghazi et al.’s sFFT is slower than iterative sFFT-DT because the P2 cost of MPP in Ghazi et al.’s sFFT dominates the computation. Compared to sFFT-v3 and Ghazi et al.’s sFFT, our method, iterative sFFT-DT, is able to deal with FFT of signals with large \( K \). Fig. 3(b) shows the results of computational time versus signal dimension under fixed \( K \). It is observed that the computational time of iterative sFFT and Ghazi et al.’s sFFT is invariant to \( N \), but our method is the fastest.

Moreover, according to Theorem 2 and Theorem 3, the performance of non-iterative sFFT-DT seems to be inferior to that of Ghazi et al.’s sFFT. Nevertheless, the successful probability described in Theorem 1 and Theorem 2 is merely a lower bound. In our simulations, we compare the recovery performance among three approaches: non-iterative sFFT-DT, iterative sFFT-DT, and Ghazi et al.’s sFFT [6]. The parameters for both proposed approaches and Ghazi et al.’s sFFT are set based on Theorem 2 and Theorem 3 respectively.

We have the following observations from Fig. 4, where signal length is \( N = 2^{20} \). First, although the theoretical result derived in Theorem 2 indicates that the performance decreases along with the increase of \( K \), it is often better than Ghazi et al.’s sFFT [6]. In fact, it is observed that the performance of Ghazi et al.’s sFFT oscillates. The oscillation is due to the fact...
that the floor operation in $2^{\lfloor \log \log K \rfloor}$ (from $d = \frac{N}{K}2^{\lfloor \log \log K \rfloor}$) acts like a discontinuous function and leads to large variations of setting $d$. The recovery performance would benefit by setting small $d$ at the expense of requiring greater computational cost. Second, iterative sFFT-DT degrades the recovery performance gradually as $K$ increases while, at the same time, the number of aliasing ($0 \leq a \leq d$) decreases as well. That is the reason the performance returns to 100% when $K = 2^{16}$ under the case that $d = \frac{N}{4K}$.

---

**Fig. 3.** Comparison of computational time for exact $K$-sparse signals. (a) Computational time vs. sparsity under $N = 2^{24}$. (b) Computational time vs. signal dimension under $K = 2^{16}$ and $a_m = 4$.

**Fig. 4.** Recovery performance comparison among non-iterative sFFT-DT, iterative sFFT-DT, and Ghazi et al.’s sFFT \[6\] for exact $K$-sparse signal. The signal length is $N = 2^{20}$.
III. SFFT-DT FOR GENERALLY K-SPARSE SIGNALS

For sparse FFT of a generally K-sparse signal \(x\), the goal is to compute an approximate transform \(X_{approx}\) satisfying:

\[
X_{approx} = \arg \min_{X'} \|X - X'\|_2,
\]

(11)

where \(X_{approx}\) is exactly K-sparse and \(X\) is generally K-sparse. Without loss of generality, we assume that all freq. grids in \(X\) are non-zero. Similar to exactly K-sparse signals, we assume that K significant freq. grids (with the first K largest magnitudes) of \(X\) distribute uniformly.

Due to generally K-sparsity of \(X\), the right-hand side of Eq. (11) will contain \(d\) terms. If we try to solve the MPP with \(a = d\), the total computational cost will exceed FFT, thereby creating nonsense. Therefore, our policy is to solve the MPP with \(a (a < d)\) significant terms in \(X_d[k]\). For example, if \(X_d[k]\) only contains two significant terms, we formulate this aliasing problem with \(a = 2\). Nevertheless, \(X_d[k]\) is, in fact, composed of \(d\) terms instead of 2 terms. In this case, from Eq. (2), the sum of \(d - 2\) insignificant terms can be considered as a normal distribution by central limit theory and act as noise, leading to inaccurate estimation in Eq. (11). Apparently, the degree of inaccuracy depends on the signal-to-noise ratio (SNR) \(SNR_{ori}\) between the significant terms and insignificant terms. More precisely, let \(X_{opt}\) be the optimal solution of Eq. (11). \(SNR_{ori}\) is defined as:

\[
SNR_{ori} = 10 \log_{10} \frac{MSE(X_{opt})}{MSE(X - X_{opt})},
\]

(12)

where \(MSE(\cdot)\) is the function of calculating the mean squared error, \(X_{opt}\) is equivalent to significant terms, and \(X - X_{opt}\) represents insignificant terms.

Due to the impact of insignificant terms, the solved roots (estimated positions), \(z\)’s, of \(P(z) = 0\) (Eq. 4), no longer belong to the set \(U_k\) of real roots, as defined in the paragraph below Eq. (3), which correspond to candidate/correct positions. Nevertheless, if \(SNR_{ori}\) is sufficiently large (> 20db), our empirical observations show that \(z\)’s can approach the real roots. Fig. 5 plots the real roots (in red) and the solved roots (in blue) in complex coordinates under different \(SNR_{ori}\) values. The real roots must locate within the unit circle with radius = 1 since they belong to \(U_k\) and each element of \(U_k\) satisfies \(\|e^{i2\pi(k+N)/N}\|_2 = 1\). Nevertheless, the solved roots could deviate from the real roots and not be located within the unit circles. In order to close the gap between the candidate positions and estimated positions, a refinement strategy is proposed in terms of sparse signal recovery in the context of compressive sensing; then, it is combined with non-iterative sFFT-DT because iterative sFFT-DT cannot work well in the aforementioned scenario. More specifically, the idea behind iterative sFFT-DT is to remove the solved freq. grids in an iterative manner to make the remaining unsolved signal more sparse. Nevertheless, we cannot guarantee an exact solution can be attained at each iteration to avoid error propagation into subsequent iterations.
The refinement strategy by sparse signal recovery is motivated by the following facts. 1) The magnitudes of significant terms must be larger than those of insignificant terms. 2) The number of significant terms is less than that of insignificant terms. Thus, estimating the locations and values of significant terms is reformulated as a sparse signal recovery problem.

Fig. 5. The bias between solved roots/estimated locations (blue circle) and real roots/candidate locations (red circle) in the polar plane under (a) $\text{SNR}_{\text{ori}} = 32$ dB and (b) $\text{SNR}_{\text{ori}} = 22$ dB.

A. Refinement via Sparse Signal Recovery

Instead of considering the locations ($z_j^l$) and values ($p_j$) as variables in Eq. (3), only values ($p_j$) are thought of as variables here. Then, we reformulate the aliasing problem in terms of a matrix form $m = Wb$ as:

$$
\begin{bmatrix}
m_{n_0} \\
m_{n_1} \\
\vdots \\
m_{n_{r-1}}
\end{bmatrix} =
\begin{bmatrix}
W_k(0, n_0) & \cdots & W_k(d-1, n_0) \\
W_k(0, n_1) & \cdots & W_k(d-1, n_1) \\
\vdots & \ddots & \vdots \\
W_k(0, n_{r-1}) & \cdots & W_k(d-1, n_{r-1})
\end{bmatrix}
\begin{bmatrix}
b_0 \\
b_1 \\
\vdots \\
b_{d-1}
\end{bmatrix},
$$

(13)

where $W_k(t, l) = e^{i2\pi(k+tN/d)l/N}$ and $t \in [0, d - 1]$, $b_i$ is the value at $(k + tN/d)$'th freq. grid, and $W \in \mathbb{C}^{r \times d}$. By comparing Eq. (3) and Eq. (13), we have $r = 2a$ and $n_j = j$ for $0 \leq j \leq r - 1$ in the exact $K$-sparse case. In the generally $K$-sparse case, however, $r$ is equal to $O(a)$, where the Big-O constant is larger than 2 and $n_j$ falls within $[0, d - 1]$. Moreover, each candidate location forms a corresponding column in $W$. For example, $[W_k(t, n_0) \ W_k(t, n_1) \ldots W_k(t, n_{r-1})]^T$ is composed of moments with different shift operators $n_j$'s. Note that Eq. (13) is an underdetermined system because $d$ is far larger than $r$ with $O(a) \ll d$. This makes Eq. (13) have infinite solutions. Nevertheless, for an exactly $K$-sparse signal $x$, $b$ is a sparse vector and $\|b\|_0 = a$ because only $a$ locations are non-zero while, for a generally $K$-sparse signal, $x$, $b$ is approximately sparse
when $SNR_{ori}$ is large enough. With the assumption of sparsity, Eq. (13) is equivalent to a sparse signal recovery problem. Conventionally, two strategies \[10\], $\ell_1$-minimization and greedy approaches, are used for sparse signal recovery. Computational complexity and recovery performance, however, are two main issues that need further attention.

First, the complexities of these standard CS approaches require polynomial time. For example, Orthogonal Matching Pursuit (OMP) \[17\] is one of the greedy algorithms and requires $O(ard)$ for solving Eq. (13). OMP runs at most $N d$ times and leads to the total cost of sFFT-DT being $O(ardN)$. This is unacceptably slower than conventional FFT. Fig. 5 illustrates the fact that the distance between the solved root and the real root is related to $SNR_{ori}$. In other words, the real roots that are far from the corresponding solved root can be ignored when $SNR_{ori}$ is sufficiently large ($>20$ db from our empirical observations). Thus, a large number of columns corresponding to the real roots that are far from the solved roots can be deleted. Moreover, the number of remaining columns is related to the number of solved roots, which is $O(a)$, and OMP costs $O(ra^2)$ now. Nevertheless, how to decide $a$ for each freq. grid is difficult. In addition, the details about how the solved roots are affected will be addressed later in Sec. III-B.

Second, how to set $r$ is important in terms of computational complexity, $O(ra^2)$, and performance. Candes and Wakin \[18\] pointed out that $r$ required to recover sparse signal must satisfy $r \geq Ca \log \frac{s}{a}$, where $C$ is a constant and $s$ is the column dimension of $W$. If $d = O(\frac{N}{K})$ is not related to $s$, then $r \geq Ca \log \frac{N}{\sqrt{K}}$. In other words, $N$ is also a parameter that impacts the size of $r$. This will make the cost of solving Eq. (13) related to $N$. Thus, by reducing the number of candidate roots to $O(a)$ (mentioned in the last paragraph), $r$ can be set according to $r = Ca \log \frac{s}{a} = Ca$. Consequently, OMP costs $O(a^3)$.

Third, from the theory of sparse signal recovery, the recovery performance also depends on mutual coherence of $W$, which is defined as:

$$\mu = \max_{i,j,i \neq j} \langle W_i, W_j \rangle,$$

(14)

where $W_i$ is the $i^\text{th}$ column of $W$ and $\mu$ is expected to be as small as possible. In this case, the phase difference between the $W_k(t, l)$ and $W_k(t+1, l)$ is $2\pi \frac{l}{d}$, as defined in Eq. (13). Recall $l \in \{n_0, n_1, ..., n_{r-1}\}$. In the exactly $K$-sparse case, the shift operator ranges from 0 to $2a - 1$. In other words, we set $n_0 = 0, n_1 = 1, ..., n_{r-1} = 2a - 1$. Even though the maximum shift operator $l = 2a - 1$ is encountered, the phase difference between the $W_k(t, l)$ and $W_k(t+1, l)$ still approaches 0 with $2\pi \frac{l}{d} = 2\pi \frac{2a - 1}{d} \to 0 (d \gg a)$. Under this circumstance, $\mu \to 1$ and perfect sparse recovery will become impossible. This is why the elements of $\{n_0, n_1, ..., n_{r-1}\}$ should be chosen randomly within the set $[0, d-1]$. In this case, $W$ will form a random partial Fourier matrix and its mutual coherence will be small, as shown in \[19\][20].

In sum, sFFT-DT for generally $K$-sparse signals must satisfy the three aforementioned requirements simultaneously and perform more FFTs of downsampled signals with different shift factors. Under this situation, $W \in \mathbb{C}^{r \times d}$ in Eq. (13) is
simplified to \( W \in \mathbb{C}^{O(a) \times O(a)} \) and the acceptable cost of OMP for recovering sparse signals is within \( O(a^3) \). In experiments, we suggest \( r = 3a \).

Consequently, \( 3a \) moments with random shift operators are required to minimize the mutual coherence of \( W \) for refining the solved roots. Furthermore, we still need the moments with shift factors ranging from 0 to \( 2a - 1 \) to obtain solved roots by MPP. In sum, the total number of moments required to solve an aliasing problem containing \( a \) significant terms is \( 5a \). As a result, for the generally \( K \)-sparse case, it requires \( 5a_m \) moments, where \( a_m \) denotes the maximum number of significant terms within all of the downsampled freq. grids.

\[ B. \ How\ to\ Decide\ the\ Number\ of\ Solved\ Roots\ (a)\ for\ Each\ Freq.\ Grid \]

The previous section for refinement via sparse signal recovery exploits the assumption that \( a \) is known for each downsampled freq. grid. In reality, though, \( a \) is unknown. Thus, deciding \( a \) for each freq. grid is an important task.

Now, we redefine the problem in Eq. \( \text{(5)} \) for generally \( K \)-sparse signals. Let \( S_s \) be an ascending sequence containing all indices of significant terms and let \( S_{ns} \) be the one defined for insignificant terms, where \( S_s \cap S_{ns} = \{ \} \), \( S_s \cup S_{ns} = \{ 0, 1, \ldots, N - 1 \} \), \( |S_s| = a \) and \( |S_{ns}| = N - a \), and \( s_{s,i} \in S_s \) and \( s_{ns,j} \in S_{ns} \) for \( 1 \leq i \leq a \) and \( 1 \leq j \leq N - a \). Thus, the moment in Eq. \( \text{(3)} \) can be rewritten as:

\[
m_i = p_0 z_{i0}^0 + p_1 z_{i1}^1 + \ldots + p_{N-1} z_{iN-1}^{N-1} = m_{s,i} + m_{ns,i},
\]

where \( m_{s,i} = p_{s_r,i} z_{s_r,1}^1 + p_{s_r,2} z_{s_r,2}^{i_2} + \ldots + p_{s_r,N-1} z_{s_r,N-1}^{i_{N-1}} \) and \( m_{ns,i} = p_{ns,i} z_{ns,1}^1 + p_{ns,2} z_{ns,2}^{i_2} + \ldots + p_{ns,N-a} z_{ns,N-a}^{i_{N-a}} \). Now, the matrix \( M \) in Eq. \( \text{(5)} \) can be rewritten as:

\[
M = \begin{bmatrix}
m_{n0} & \cdots & m_{n_{a-1}} \\
m_{n1} & \cdots & m_{n_n} \\
\vdots & \ddots & \vdots \\
m_{n_{a-1}} & \cdots & m_{n_{2a-2}}
\end{bmatrix}
= M_s + M_{ns},
\]

where \( n_j = j \) like the setting in exactly \( K \)-sparse signals. \( M_{ns} \) acts like a “noise” matrix produced by insignificant components. Thus, it is apparent that

\[
\frac{\|M_s\|_F}{\|M_{ns}\|_F},
\]

where \( \cdot \|_F \) denotes the Frobenius norm, is proportional to \( SNR_{ori} \), as defined in Eq. \( \text{(12)} \). In addition, letting \( n_j = n_0 + v_j \) for \( j \in [0, 2a_m - 2] \), where \( v \) is a constant (discussed later), \( M \) can also be expressed as:

\[
M = \sum_{i=0}^{N-1} p_i z_i z_i^T = M_s + M_{ns} = \sum_{i=1}^a p_{s,i} z_{s,i} z_{s,i}^T + \sum_{i=1}^{N-a} p_{ns,i} z_{ns,i} z_{ns,i}^T,
\]

where \( z_i = [z_{i0} \ z_i^1 \ldots z_{iN-1}^{i_{N-1}}]^T \) is a column vector.

Now, we discuss how to decide \( a \) from \( M \). It is worth noting that Eq. \( \text{(16)} \) is similar to singular value decomposition (SVD). Nevertheless, there are some differences between them: (1) \( p_{s,i} \)'s and \( p_{ns,i} \)'s are complex but not real and both \( \|z_{s,i}\|_2 \) and \( \|z_{ns,i}\|_2 \) are not normalized; (2) \( z_{s,i} \)'s and \( z_{ns,i} \)'s are not orthogonal vectors; and (3) The formulation of SVD is

\[
M = \sum_{i=0}^{N-1} p_i z_i z_i^*,
\]

where \( * \) denotes a conjugate transpose.
To alleviate the first difference, Eq. (16) is rewritten as:

$$M = \sum_{i=1}^{a} \sqrt{a_{m}} |p_{s,i}| \hat{z}_{s,i}\hat{z}_{s,i}^T + \sum_{j=1}^{N-a} \sqrt{a_{m}} |p_{n,s,j}| \hat{z}_{n,s,j}\hat{z}_{n,s,j}^T,$$

(17)

where $\hat{z}_{s,i} = \frac{1}{\sqrt{a}} e^{\sqrt{-1} \theta_{s,i}} [z_{i}^{n_{0}} z_{i}^{n_{1}} \ldots z_{i}^{n_{M-1}}]^T$ and $p_{s,i} = |p_{s,i}| e^{\sqrt{-1} \theta_{s,i}}$. Thus, we have $\|\hat{z}_{s,i}\|_2 = 1$. In addition, the third difference, mentioned above, can be solved by symmetric SVD (SSVD) instead of SVD. This is because the singular values of SSVD have been proven to be the same as those of SVD for the same matrix. Thus, Eq. (17) can be used to solve SSVD for matrix $M_s$, interfered with by $M_{ns}$. In our case, if $|p_{s,i}| > |p_{n,s,j}|$ for $i \in [1, a]$ and $j \in [1, N - a]$, the first $a$ singular values of $M$ approximate $\sqrt{a_{m}} |p_{s,i}|$'s when $M_{ns} = 0$. This claim holds when all $\hat{z}_{s,i}$'s and $\hat{z}_{n,s,j}$'s are orthogonal. To alleviate the second difference, which is due to the correlation between $\hat{z}_{s,i}$ and $\hat{z}_{n,s,j}$, the idea is similar in principle to mutual coherence minimization, described in Sec. III-A. Nevertheless, there is an extra constraint, $n_j = n_0 + v j$ for $j \in [0, 2a_m - 2]$. Combined with the conclusion that $n_j$'s should be chosen randomly from $[0, d - 1]$, as described in Sec. III-A, $n_j$'s for $j \leq 2a_m - 2$ finally are chosen as the sequence $\{0, v, 2v, \ldots, (2a_m - 2)v\}$ with $(2a_m - 2)v \leq d - 1$. In addition, $n_j$'s for $j \geq 2a_m$ still are chosen randomly from $[0, d - 1]$ and $n_i \neq n_j$ for all $i \neq j$. Based on the above setting, SSVD is an efficient method for approximating $\sqrt{a_{m}} |p_{s,i}|$'s when $SNR_{ori}$ is high enough, $a \leq a_m$, and $M \approx M_s$. In other words, we can know which freq. grids with larger magnitudes we want to solve from $|p_{s,i}|$'s. Thus, sFFT-DT computes all singular values from all $M$'s generated under different downsampling factors. We index each singular value according to which downsampled freq. grid it is from. Consequently, we are interested in finding the first $K$ largest singular values to resolve the $K$ non-zero frequency grids.

C. Non-iterative sFFT-DT: Algorithm for Generally K-Sparse Signals

For exactly $K$-sparse signals, sFFT-DT adopts the proposed top-down iterative strategy for reducing the computational cost. Nevertheless, it is not appropriate for generally $K$-sparse signals since we cannot guarantee exact solutions that can be obtained at each iteration, leading to error propagation. Thus, for generally $K$-sparse signals, sFFT-DT solves the aliasing problem once, as shown in Algorithm 2.

The function main contains four parts. Part 1: In Lines 2-11, several downsampled signals are generated for performing FFTs with different shift factors. Specifically, the downsampled signals in Lines 2-5 are used for solving the MPP and the downsampled signals in Lines 6-9 are used for deciding $a$ for each freq. grid and CS refinement. To distinguish between the two, signals from Lines 2-5 are represented by $x_d$ and those from Lines 6-10 are represented by $x_s$. Part 2: Lines 11-21 decide the number of significant terms in the downsampled freq. grids. $V_{\sin}$ is a set used to save all singular values of $M$'s (defined in Eq. (15)) corresponding to freq. grids. Part 3: Lines 22-25 calculate $a$ solved roots by $z = \arg \min_{z} |P(z)|$, where $z$ belongs to the set $U_k$. If $a > 1$, those $z$'s with the first $a$ minimum errors are assigned to the set $R$. Part 4: Lines 26-29 refine the
MPP solutions via sparse signal recovery, as mentioned in Sec. III-A

Algorithm 2 sFFT-DT for generally $K$-sparse signals.

**Input:** $x$, $K$; **Output:** $X_{approx}$

**Initialization:** $X_{approx} = 0$, $d = O\left(\frac{N}{K}\right)$, $R = \{\}$, $V_{sin} = \{\}$, $a_m = 4$

01. function main()
02.  for $l = 0$ to $a_m - 1$
03.    $x_{d,2l}[k] = x[dk + 2l]$ for $k \in [0, \frac{N}{d} - 1]$;
04.    $x_{d,2l+1}[k] = x[dk + 2l + 1]$ for $k \in [0, \frac{N}{d} - 1]$;
05.  end for
06. Generate the sequence $\{n_0, n_1, \ldots, n_{3a_m - 1}\}$ discussed in Sec. III-B;
07.  for $l = 0$ to $3a_m - 1$
08.    $x_{s,l}[k] = x[dk + n_l]$ for $k \in [0, \frac{N}{d} - 1]$;
09.  end for
10. Do FFT of all $x_d$'s, $x_s$'s to obtain $X_d$'s and $X_s$'s.
11.  for $k = 0$ to $\frac{N}{d} - 1$
12.    $m_j = X_{s,j}[k]$ for $j \in [0, 2a_m - 1]$;
13.    Use $m_j$'s to form $M$ defined in Sec. III-B;
14.    Do SVD of $M$ and put singular values into the set $V_{sin}$;
15.  end for
16. Find the first $K$ largest singular values from $V_{sin}$ and save them as $\sigma_1, \sigma_2, \ldots, \sigma_K$.
17.  for $l = 1$ to $K$
18.    if ($\sigma_l$ originates from the $k$'th freq. grid)
19.      $a$ of the $k$'th freq. grid increases by 1;
20.    end if
21.  end for
22.  for $k = 0$ to $\frac{N}{d} - 1$
23.    $m_j = X_{d,j}[k]$ for $j \in [0, 2a_m - 1]$;
24.  ROOTS($m, a_m, R$);
25.  end for
26.  for $k = 0$ to $\frac{N}{d} - 1$
27.    $m_j = X_{s,j}[k]$ for $j \in [0, 3a_m - 1]$;
28.    Refine the roots and values included in $R$;
29.  end for
30. end function
31. function ROOTS ($m, a_m, R$)
32.  Solve $c$ from Step (i) of MPP, as described in Eq. 5;
33.  Solve Step (ii) of MPP to obtain $a$ solved roots;
34. end function
D. Computational Complexity of sFFT-DT for Generally K-Sparse Signals

In this section, we analyze the computational cost of sFFT-DT for generally K-sparse signals based on Theorem 1 or Theorem 2 for the four parts of the Main function.

Part 1 is to do FFT for downsampled signals, and it costs \( O(a_m \frac{N}{d} \log \frac{N}{d}) \). Part 2 solves SVD of \( M \in C^{a_m \times a_m} \) for each downsampled freq. grid. Since SVD will totally run \( O(\frac{N}{d}) \) times, Part 2 will cost \( O(\frac{N^3}{da_m^2}) \), according to \([21]\). Part 3 is to solve the MPP without Step (iii), as described in Sec. II-B; thus, it costs \( O(\frac{N}{d} a_m^2) \). Finally, CS refinement in Part 4 needs \( O(\frac{N}{d} a_m^3) \), as described in Sec. III-A.

Consequently, the computational cost of sFFT-DT for generally K-sparse signals still is impacted by \( a_m \) and \( d \) as in the exactly-K sparse case. If significant freq. grids distribute uniformly, according to Fig. 2 both \( a_m \) and \( d \) can be set based on Theorem 1 or Theorem 2. In this case, since \( a_m = 4 \), the computational cost is bounded by Part 1, which is the same as in the exactly-K sparse case. It should be noted that the Big-O constants of the generally K-sparse case are larger than those of the exactly K-sparse case because the former needs more moments.

E. Simulation Results for Generally K-Sparse Signals

The simulation environment is similar to the one described in Sec. II-I. We only compare sFFT-DT with FFTW because sFFT \([1][2]\) does not release the code and the code of sFFT for the generally K-sparse case is difficult to implement (as mentioned in the footnote on Page 3). Therefore, no experimental results for generally K-sparse signals are shown in their papers or websites.

Here, the test signals were generated from the mixture Gaussian model as:

\[
X \sim pN(0, \sigma_{on}^2) + (1-p)N(0, \sigma_{off}^2),
\]

where \( p = \frac{K}{N} \) is the active probability that decides which Gaussian model is used and \( \sigma_{on} > \sigma_{off} \). For each test signal, its real root is defined as \( X_{opt} \), as described in Sec. III and \( X_{approx} \) is the output signal obtained from sFFT-DT. Due to the fixed noise model, when \( \frac{N}{K} \) becomes larger, \( SNR_{ori} \) decreases accordingly.

The comparison of computational time is depicted in Fig. 6. Fig. 6(a) shows the results of computational time versus signal sparsity under fixed \( N \). It is observed that sFFT-DT is remarkably faster than FFTW, except for the cases with \( K \geq 2^{15} \). Fig. 6(b) shows the results of computational time versus signal dimension under fixed \( K \). It is apparent that the computational time of sFFT-DT is not related to \( N \).

Fig. 7 shows the reconstruction accuracy versus \( \frac{N}{K} \), where \( d = \frac{N}{8K} \) and \( a_m = 3 \). Recovery accuracy was measured in terms of SNR, defined in Eq. (12). The experimental results show that sFFT-DT can approximate the first \( K \)-largest freq. grids.
accurately when $SNR_{ori}$ is large enough. For low $SNR_{ori}$ with large $\frac{N}{K}$, the performance degrades.

In sum, compared with [1][2], the proposed sFFT-DT for generally $K$-sparse signals is the first algorithm with the reasonable Big-O constants and is verified to be faster than FFTW.

**Fig. 6.** Comparison between non-iterative sFFT-DT and FFTW for generally $K$-sparse signals. (a) Computational time vs. sparsity under $N = 2^{24}$ and $a_m = 4$. (b) Computational time vs. signal dimension under $K = 2^{12}$ and $a_m = 4$.

**Fig. 7.** Reconstruction accuracy vs. $\frac{N}{K}$ under $d = \frac{N}{8K}$ and $a_m = 4$.

**IV. CONCLUSIONS AND FUTURE WORK**

We have presented new sparse Fast Fourier Transform methods based on downsampling in the time domain (sFFT-DT) for both exactly $k$-sparse and generally $K$-sparse signals in this paper. The accurate computational cost and theoretical performance lower bound of sFFT-DT are proven for exactly $K$-sparse signals. We also derive the Big-O constants of computational complexity of sFFT-DT and show that they are smaller than those of MIT’s methods [1][2][6]. In addition, sFFT-DT is more
hardware-friendly, compared with other algorithms, since all operations of sFFT-DT are linear and involved in an analytical solution. On the other hand, previous works, such as [1][2][6], are based on the assumption that sparsity $K$ is known in advance. To address this issue, we proposed a simple solution to estimate $K$ and relax this impractical assumption. We showed that the extra cost for deciding $K$ is the same as that required for sFFT-DT with known $K$. Moreover, we extended sFFT-DT to generally $K$-sparse signals in this paper. To solve the interference from insignificant freq. grids in aliasing, we presented a new sparse signal recovery algorithm to refine the solution to the MPP. Overall, theoretical complexity analyses and simulation results demonstrate that our sFFT-DT outperforms the state-of-the-art.

In future work, we want to extend sFFT-DT to a real application, such as GPS [22]. In this case, the sparsity constraint is 1 but $X$ is interfered with massive noises. The most important thing is to identify the location with the maximum magnitude. A future direction needing further attention is to combine the mechanism of location estimation in our algorithm and sFFT, as shown in Fig. 8. Fig. 8(a) shows an extremely sparse signal $X$ with only one non-zero entry ($K = 1$). The red region of Fig. 8(b) is the estimated locations using sFFT (detailed descriptions are ignored due to limited space). The red spikes in Fig. 8(c) are the possible $d$ locations derived from our method. By combining both algorithms, the possible location is reduced to the red circle, as shown in Fig. 8(a). Such a new mechanism is more efficient than the one proposed in [22].

![Fig. 8. The proposed method, sFFT-DT, combined with sFFT [1][2] can achieve efficient localization in GPS.](image)

V. ACKNOWLEDGMENT

This work was supported by National Science Council under grants NSC 100-2628-E-001-005-MY2 and NSC 102-2221-E-001-022-MY2.
VI. APPENDIX

The analytical solution of the moment preserving problem with \( a = 3 \) is

\[
c_d = \begin{vmatrix}
  m_0 & m_1 & m_2 & m_3 \\
  m_1 & m_2 & m_3 & m_4 \\
  m_2 & m_3 & m_4 & m_5 \\
  m_3 & m_4 & m_5 & m_6 \\
\end{vmatrix}
\]

\[
c_0 = \left( \frac{1}{c_d} \right)
\]

\[
c_1 = \left( \frac{1}{c_d} \right)
\]

\[
c_2 = \left( \frac{1}{c_d} \right)
\]

\[
\begin{align*}
  c_0 &= \left( \frac{1}{c_d} \right) \\
  c_1 &= \left( \frac{1}{c_d} \right) \\
  c_2 &= \left( \frac{1}{c_d} \right)
\end{align*}
\]

\[
z_0 = -\frac{c_2}{3} - A - B, \quad z_1 = -\frac{c_2}{3} - W_1 A - W_2 B, \quad z_2 = -\frac{c_2}{3} - W_2 A - W_1 B,
\]

\[
A = \left( \frac{c_0}{2} - \frac{c_1 c_2}{6} + \frac{c_2^2}{27} \right) - \left[ \left( \frac{c_0}{2} - \frac{c_1 c_2}{6} + \frac{c_2^2}{27} \right)^2 + \left( \frac{c_1}{3} - \frac{c_2}{9} \right)^2 \right]^{1/2},
\]

\[
B = -\left( \frac{c_0}{2} - \frac{c_1 c_2}{6} \right),
\]

\[
W_1 = \frac{1}{2} + \frac{\sqrt{3}}{2}, \quad W_2 = \frac{1}{2} - \frac{\sqrt{3}}{2},
\]

\[
p_d = \begin{vmatrix}
  1 & 1 & 1 & m_0 & 1 & 1 & 1 & m_0 & 1 & 1 & m_0 & 1 \\
  z_0 & z_1 & z_2 & m_1 & z_1 & z_2 & m_1 & z_1 & z_2 & m_1 & z_1 & z_2 \\
  z_0^2 & z_1^2 & z_2^2 & m_2 & z_1^2 & z_2^2 & m_2 & z_1^2 & z_2^2 & m_2 & z_1^2 & z_2^2 \\
\end{vmatrix}
\]

Similarly, the solution with \( a = 4 \) is

\[
c_d = \begin{vmatrix}
  m_0 & m_1 & m_2 & m_3 & m_4 \\
  m_1 & m_2 & m_3 & m_4 & m_5 \\
  m_2 & m_3 & m_4 & m_5 & m_6 \\
  m_3 & m_4 & m_5 & m_6 & m_7 \\
\end{vmatrix}
\]

\[
c_0 = \left( \frac{1}{c_d} \right)
\]

\[
c_1 = \left( \frac{1}{c_d} \right)
\]

\[
c_2 = \left( \frac{1}{c_d} \right)
\]

\[
c_3 = \left( \frac{1}{c_d} \right)
\]

\[
\begin{align*}
  c_0 &= \left( \frac{1}{c_d} \right) \\
  c_1 &= \left( \frac{1}{c_d} \right) \\
  c_2 &= \left( \frac{1}{c_d} \right) \\
  c_3 &= \left( \frac{1}{c_d} \right)
\end{align*}
\]

\[
n_0 = -m_4 - m_5 - m_6 - m_7, \quad n_1 = -m_5 - m_6 - m_7, \quad n_2 = -m_6 - m_7, \quad n_3 = -m_7
\]
\[ z_0 = \frac{1}{2} \left\{ -\left( \frac{c_3}{2} + A \right) - \left[ \frac{c_3}{2} + A \right]^2 - 4(Y + B) \right\}^{\frac{1}{2}}, \quad z_1 = \frac{1}{2} \left\{ -\left( \frac{c_3}{2} + A \right) + \left[ \frac{c_3}{2} + A \right]^2 - 4(Y + B) \right\}^{\frac{1}{2}}, \]
\[ z_2 = \frac{1}{2} \left\{ -\left( \frac{c_3}{2} - A \right) - \left[ \frac{c_3}{2} - A \right]^2 - 4(Y - B) \right\}^{\frac{1}{2}}, \quad z_3 = \frac{1}{2} \left\{ -\left( \frac{c_3}{2} - A \right) + \left[ \frac{c_3}{2} - A \right]^2 - 4(Y - B) \right\}^{\frac{1}{2}}, \]
\[ A = \frac{1}{2} \left( \frac{c_3 - 4c_2 + 8Y}{3} \right)^{\frac{1}{2}}, \quad B = \frac{c_3 Y - c_1}{A}, \quad Y = \frac{c_2}{6} - C - D, \]
\[ C = [G + (G^2 + H^2)]^{\frac{1}{4}}, \quad D = \frac{-H}{C}, \quad G = \frac{1}{432} \left( 72c_0c_2 + 9c_1c_2c_3 - 27c_1^2 - 27c_0c_3^2 - 2c_2^3 \right), \]
\[ H = \frac{1}{36} \left( 3c_1c_3 - 12c_0 - c_2^2 \right), \]

\[ p_d = \begin{pmatrix}
 1 & 1 & 1 & 1 \\
 z_0 & z_1 & z_2 & z_3 \\
 z_0^2 & z_1^2 & z_2^2 & z_3^2 \\
 z_0^3 & z_1^3 & z_2^3 & z_3^3
\end{pmatrix}, \quad p_0 = \left( \frac{1}{p_d} \right)
\]
\[ p_1 = \left( \frac{1}{p_d} \right)
\]
\[ p_2 = \left( \frac{1}{p_d} \right)
\]
\[ p_3 = 1 - p_0 - p_1 - p_2. \]

**References**

[1] H. Hassanieh, P. Indyk, D Katabi, and Eric Price, “Nearly optimal sparse fourier transform,” STOC, 2012.

[2] H. Hassanieh, P. Indyk, D Katabi, and Eric Price, “Simple and practical algorithm for sparse fourier transform,” SODA, 2012.

[3] M. Frigo and S. G. Johnson, “The design and implementation of fftw3,” in Proceedings of the IEEE, 2005, pp. 216–231.

[4] A. Gilbert, M. Muthukrishnan, and M. Straussn, “Improved time bounds for near-optimal space fourier representations,” in SPIE Conference, Wavelets, 2005.

[5] M. A. Iwen, A. Gilbert, and M. Straussn, “Empirical evaluation of a sub-linear time sparse dft algorithm,” in Communications in Mathematical Sciences, 2007.

[6] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and Lixin Shi, “Sample-optimal average-case sparse fourier transform in two dimensions,” Allerton, 2013.

[7] S.-H. Hsieh, C.-S. Lu, and S.-C. Pei, “Sparse fast fourier transform by downsampling,” in IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 5637–5641.

[8] S. Pawar and K. Ramchandran, “Computing a k-sparse n-length discrete fourier transform using at most 4k samples and o(klogk) complexity,” arXiv, 2013.

[9] G. Szego, Orthogonal Polynomials, Amer. Math. Soc., 1975.

[10] D.L. Donoho, “Compressed sensing,” IEEE Transactions on Information Theory, vol. 52, no. 4, pp. 1289–1306, 2006.

[11] F. J. MacWilliams and N. J. A. Sloane, The Theory of Error-Correcting Codes, North-Holland Mathematical Library, 1977.

[12] W. H. Tsai, “Moment-preserving thresholding,” Comput. Vision, Graphics, Image Processing, vol. 29, pp. 377–393, 1985.

[13] J. Massey, “Shift-register synthesis and bch decoding,” IEEE Transactions on Information Theory, vol. 15, pp. 122–127, 1963.

[14] N. Chen and Z. Yan, “Complexity analysis of reed-solomon decoding over gf(2m) without using syndromes,” EURASIP J. Wirel. Commun. Netw., vol. 2008, no. 16, pp. 1–11, 2008.

[15] V. Y. Pan, “Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding,” Journal of Symbolic Computation, vol. 33, no. 5, pp. 701–733, 2002.
[16] A. Saidi, “Decimation-in-time-frequency fft algorithm,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1994.

[17] J.A. Tropp and A.C. Gilbert, “Signal recovery from random measurements via orthogonal matching pursuit,” *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[18] E.J. Candes and M.B. Wakin, “An introduction to compressive sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[19] L. Welch, “Lower bounds on the maximum cross correlation of signals,” *IEEE Transactions on Information Theory*, vol. 20, pp. 397–399, 1974.

[20] L. Gan, C. Ling, T. T. Do, and T. D. Tran, “Analysis of the statistical restricted isometry property for deterministic sensing matrices using stein’s method,” Preprint, 2009.

[21] B.-G. Angelika and B. G. William, “Singular value decompositions of complex symmetric matrices,” *Journal of Computational and Applied Mathematics*, vol. 21, pp. 41–54, 1988.

[22] H. Hassanieh, P. Indyk, D Katabi, and Eric Price, “Faster gps via the sparse fourier transform,” in *ACM MOBICOM*, 2012.
\[ 4O\left(\frac{N}{d} \log \frac{N}{d}\right) \]

\[ 2O\left(\frac{N}{d} \log \frac{N}{d}\right) \]

Sum of two terms