On Performance of Sparse Fast Fourier Transform and Enhancement Algorithm

Gui-Lin Chen, Shang-Ho Tsai, *Senior Member, IEEE*, and Kai-Jiun Yang

***Abstract*—Sparse fast Fourier transform (FFT) is a promising technique that can significantly reduce computational complexity. However, only a handful of research has been conducted on precisely analyzing the performance of this new scheme. Accurate theoretical results are important for new techniques to avoid numerous simulations when applying them in various applications. In this study, we analyze several performance metrics and derive the corresponding closed-form expressions for the sparse FFT including 1) inter sparse interference due to nonideal windowing effects, 2) the probability of sparse elements overlapping, and 3) the recovering rate performance. From the analytical results, we gain insights and propose a novel mode-mean estimation algorithm for improving the performance. Simulation results are provided to show the accuracy of the derived results as well as the performance enhancement. We also show how to determine parameters to achieve the lowest computational complexity using these theoretical results.**

***Index Terms*—Sparse fast Fourier transform, sparse signals, recovering rate, mode-mean estimator.**

I. Introduction

FAST Fourier Transform (FFT) is one of the most important techniques in signal processing. The demands for high-speed large-size FFT are strong and urgent. To name a few, VDSL2 [1] and DVB-T2 [2] standards with 8k- and 32k-FFT respectively. The dimension of signal processing for high-resolution multimedia, such as 4k images or high-definition audio, also needs to be scaled up for better quality. Meanwhile, lately, deep learning has replaced high dimensional convolution by FFT for speed-up in the neural network for extracting features [3] and [4]. When the FFT size is large, the computational complexity grows rapidly, and the computation units may not be able to complete the calculations in time.

Sparsity is common and inherent in signals such as communications, audio, image and video data. Recently there has been extensive research done in compressive sensing to handle sparse signals [5]–[8]. Several signals are sparse in the frequency domain, e.g., see [9]–[12]. To transform such sparse signals to the other domain, sparse FFT (SFFT) can be applied to greatly reduce computations [13]–[20].

The concept of SFFT was mentioned by the researchers in [13]–[16]. Instead of computing all the elements, the SFFT exposes new techniques to identify and calculate the sparse elements. Generally, the SFFT proceeds in three steps [17]: 1) identifying the (frequency) locations of the principal elements with large magnitude; 2) estimating the coefficients of these elements in the first step; 3) removing the attribution of the Fourier result computed by the first two steps from the original signal. These three steps are repeated until the entire sparse elements are found. H. Hassanieh *et al.* modified the third step by keeping only K largest elements while setting the others to zeros [15]. Instead of subtracting the original time domain signal, the complexity is further reduced by subtracting the reconstructed partial sparsity from the sub-sampled signals, such that the complexity of the inverse FFT is also minimized [16]. A different scheme called SFFT-DT, which handles sparse FFT problems via downsampling the source signal in the time domain without aliasing the outputs, was proposed in [18] and [19]. Lately, the SFFT has been applied in image processing to reduce the complexity and the performance loss: by optimally permuting the signal, the spectrum collision is effectively reduced [21]. The property in lattice theory is applied in the hash-to-bin process so that the permutation is performed in multi-dimension. In such a case, the peak signal-to-noise ratio (PSNR) of the 2-D image processing can be improved by several decibels compared to those with random permutation in single-dimension. Another application is the noise-robust fast Fourier aliasing-based sparse transform (R-FFAST) in [22] that speeds up the acquisition of magnetic resonance (MR) images.

The motivations of this work are as follows: The SFFT is a novel and promising technology. However to date not much research has been conducted to derive accurate performance in closed-form expressions for this new technique. Accurate performance analyses and closed-form expressions for new technologies are important because these closed-form expressions not only provide insights into the designs but also avoid cumbersome numerical simulations due to the needs of adjusting various parameters. Although some performance bounds were derived, e.g., in [14], however, those bounds may not be sufficiently tight, and thus simulations are still needed in most applications. For example, one may ask the question *what is the relationship between the computational complexity and the recovering rate under a specific parameter setting,* or the question *what is the parameter setting that achieves the lowest*

Manuscript received December 23, 2016; revised April 20, 2017 and August 2, 2017; accepted August 4, 2017. Date of publication August 15, 2017; date of current version September 5, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Dennis Wei. This work was supported by the Ministry of Science and Technology, Taiwan under Grant MOST 105-2221-E-009-033 and Grant MOST 106-2221-E-009-043. *(Corresponding author: Shang-Ho Tsai.)*

The authors are with the Department of Electrical Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: glchen.eed03g@g2.nctu.edu.tw; shanghot@alumni.usc.edu; kaijiuny.ece98g@g2.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2017.2740198

computational complexity for a 100% *recovering rate?* To answer these questions, accurate analyses are needed. Moreover, we notice that with the aid of signal updating iterations, e.g., similar concept in [16], the subsampling FFT size for the algorithm proposed in [15] can be reduced.[1] In other words, the residual sparse elements due to the shorten subsampling FFT size after the outer loop iterations can be further recovered with the assistance of the signal updating iterations. As a result, the overall complexity can be reduced. The question is *how to efficiently determine suitable subsampling FFT size and the numbers of outer loop and signal updating iterations for various parameter settings?* Again, accurate performance analyses and closed-form expressions are competent for solving this question. Furthermore, most of the existing SFFT schemes use the "median" to estimate the coefficients of the sparse elements, e.g., see [14], [15], and [16]. The reason should be to combat the outliers, because the commonly used "mean estimator" is susceptible to the influence of outliers. However, we notice that in some cases, especially when the number of outliers increases, the median estimator can lead to a serious performance degradation in terms of recovering rate. Hence some other estimator capable to conquer this issue is desired.

To address these issues, we have done the following contributions: First, we analyze the performance of the SFFT scheme and derive closed-form expressions. Consequently, one can determine suitable design parameters without conducting numerous simulation results for various cases. For example one can use these results to determine the subsampling FFT size, windowing specifications, the numbers of outer loop and signal updating iterations for a given number of sparse elements and a target recovering rate. Simulation results are provided to verify the accuracy of the proposed analysis as well as show how to use these analytical results to determine design parameters in various situations. It is worth pointing out that the analysis is derived for arbitrary sparse signals, although in the Examples and Experiments, the signal that we used are i.i.d. Gaussian since such signals are more generally used in the applications with FFT, e.g., image processing, machine learning, etc.

Moreover, we propose a mode-mean estimator to overcome the performance loss using the conventional median estimator. This proposed estimator is motivated by the observation that when a specific sparse element does not overlap with other sparse elements after subsampling, its reconstructed results from individual iterations have a high probability to be nearly identical. The proposed estimator finds the mode of the reconstructed results from individual iterations. Those reconstructed results that equal to the mode are further averaged to obtain the FFT output. Simulation results show that the proposed mode-mean estimator outperforms the conventional median estimator in terms of estimation error and overall recovering rate.

Furthermore, using the above findings, we suggest the following scheme: The SFFT applies the outer loop iterations in [15] while the median estimator can be replaced by the proposed mode-mean estimator. Signal updating iterations, like those in [16], can be used to reduce the subsampling FFT size and thus it decreases the overall complexity. The design parameters such as the subsampling FFT size, the numbers of outer loop and signal updating iterations can be efficiently determined by the proposed analytical results.

[1]Note that in [16], the algorithms in the outer loop iterations are not the same as those in [15]. Also, there is no signal updating iteration used in [15].

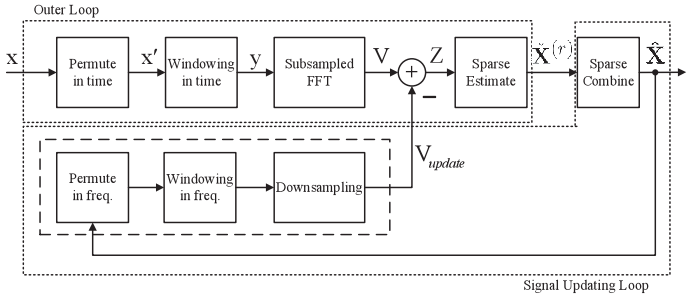


Fig. 1. An SFFT system block diagram with outer loop and signal updating iterations.

The rest of this paper is organized as follow: In Section II, the system model of the SFFT is introduced. In addition, the proposed mode-mean estimation is introduced and the induced inter sparse interference is analyzed here. In Section III, this new proposed SFFT is analyzed, and closed-form approximations are derived for the recovering rate performance. Simulation results are provided in Section IV. Conclusion remarks are given in Section V.

II. System Model and Problem Formulation

The block diagram of the suggested SFFT system is shown in Fig. 1. The input signal $\mathbf{x} \in \mathbb{C}^{\mathbf{N}\times\mathbf{1}}$ is K-sparse in the frequency domain, where N is a power-of-2 integer. Now let us introduce the main steps of the SFFT, including outer loop iterations and signal updating iterations. The outer loop iterations contain signal permutation, windowing, subsampled FFT and inner loop iterations.

A. Permutation

The permutation function is defined as $f_T(n) = (\beta_T n)_{mod\ N}$ in the time domain and $f_F(k) = (\beta_F k)_{mod\ N}$ in the frequency domain, where β_T has a uniform distribution in $\{1, 3, 5, \ldots, N-1\}$. The index β_F after the permutation in the frequency domain can be obtained by $(\beta_F \beta_T)_{mod\ N} = 1$. Because both β_T and β_F are one-to-one and onto, β_F also has uniform distribution in $\{1, 3, 5, \ldots, N-1\}$.

Let $\mathbf{x}$ be an $N \times 1$ complex signal in the time domain, and its DFT $\mathbf{X}$ is an $N \times 1$ $K-$ sparse complex signal. Then it can be shown that permutating $\mathbf{x}$ by $x'[n] = x\,[f_T(n)]$, the DFT of the permuted signal can be obtained by $X'[k] = X\,[f_F(k)]$.

B. Windowing

A Gaussian window function was used in [16]. Although the Gaussian window function is good in stopband because its magnitude decreases rapidly than the other window functions, its window length is longer than Chebyshev window function in the time domain. Also, we would like the passband region to

be as flat as possible and the transition band be as sharp as it can be. Intuitively, this can be achieved by using a rectangular function. Hence, we adopt the Chebyshev window function as the standard window and convolve it with a rectangular function to attain the properties of flat passband and short window length. The window function is redefined as: $\mathbf{g} \in \mathbb{R}^{L\times 1}$ is said to be a $(\epsilon_p, \epsilon_s, \delta_s, L)$ flat window function if its DFT $\mathbf{G} \in \mathbb{C}^{\mathbf{N}\times \mathbf{1}}$ satisfies

$$\begin{cases} 1-\delta_s \le \Big|G[k]\Big| \le 1+\delta_s,\ k \in [-\epsilon_p N, \epsilon_p N] \\ \Big|G[k]\Big| < \delta_s,\ k \notin [-\epsilon_s N, \epsilon_s N]. \end{cases}$$

C. Subsampled FFT

Let $\mathbf{y} \in \mathbb{C}^{N\times 1}$ be a complex signal in the time domain and $\mathbf{Y} \in \mathbb{C}^{N\times 1}$ be its DFT in the frequency domain. Instead of using N-point FFT, in the sparse FFT, only B-point FFT is conducted, where $\frac{N}{B}$ is called the subsampling factor. It is mentioned in [24] that downsampling in the frequency domain leads to time aliasing in the time domain. More specifically, let $\mathbf{V}$ be a downsampled vector of $\mathbf{Y}$ via $V[k] = Y[\frac{N}{B}k]$. Then, the resulting time aliasing effect can be expressed as $v[n] = \sum_{j=0}^{\frac{N}{B}-1} y[n+Bj]$. This is similar to downsampling a signal in the time domain results in aliasing in the frequency domain.

D. Inner Loop Iteration

There are two kinds of iterations in the inner loop. One is location loop iteration to determine index locations of sparse signals, and the other is estimation loop iteration to reconstruct amplitude. Each iteration in both loops contains permutation, windowing, and subsampled FFT. The location and estimation loops are introduced separately as follows:

1) Location Loop: The location loop finds dK candidate elements that have the largest magnitude from $\mathbf{V}$, where $d \ge 1$ and its purpose is to keep more candidates for finding the true K-sparse locations, and this can be determined off-line.

Let us describe how to find the locations. Let $\mathbf{H} \in \mathbb{R}^{B\times 1}$ be a vector obtained by setting the dK largest elements of $\mathbf{V}$ to 1 and the other elements to 0. Let $\mathbf{U}'$ be a signal obtained by upsampling $\mathbf{H}$ by a factor of $\frac{N}{B}$ using the following equations:

$$U'[k] = H\left[\left(round\left(\frac{N}{B}k\right)\right)_{mod\ N}\right],\ k = 0, 1, \ldots, N-1.$$

That is, in the upsampling operation, we assign the same value to the neighbor elements for reflecting the window effect. Then the reconstructed signal can be obtained by depermuting $\mathbf{U}'$ as follows:

$$U[k] = U'[(\beta_T k)_{mod\ N}],\ k = 0, 1, \ldots, N-1.$$

A $\mathbf{0}^{N\times 1}$ score table $\mathbf{T}$ is prepared in advance. At each iteration, the score table plus one at location k, *i.e.,* $T[k] = T[k] + 1$, where k corresponds to the $dK\frac{N}{B}$ non-zero elements of $U[k]$. After several iterations, one picks up the locations with the highest scores and regards them as the candidate indices of the sparse signal. We define $\mathcal{J}$ as the index set of the candidate indices of the sparse signal.

2) Estimation Loop: After determining the candidate locations of a sparse signal, namely $\mathcal{J}$, estimation loop iterations are conducted to reconstruct the amplitudes for these locations. In the subsampled FFT, because windowing in the time domain is equivalent to convolving in the frequency domain, *i.e.,* $Y[k] = G[k] * X'[k]$, the subsampled FFT signal can be expressed as

$$\begin{aligned} V[k] &= Y\left[\frac{N}{B}k\right] \\ &= \sum_{m'=0}^{N-1} G\left[\left(\frac{N}{B}k - m'\right)_{mod\ N}\right] X'[m'] \\ &= \sum_{m=0}^{N-1} G[\left(\frac{N}{B}k - f_F(m)\right)_{mod\ N}]X[m], \end{aligned} \tag{1}$$

where $k = 0,\ 1, \ldots, B-1$. Since the sparse signal consists of only K dominant elements, (1) becomes

$$V[k] = \sum_{j=0}^{K-1} G\left[\left(\frac{N}{B}k - f_F(I_i)\right)_{mod\ N}\right] X[I_i], \tag{2}$$

where I_i is used to emphasize that this is a candidate index of the sparse signal. Because we have candidate locations $I_i \in \mathcal{J}$ from the location loop, by letting $k = h(I_i) = round\left(f_F(I_i)\frac{B}{N}\right)$, one can distinguish between the desired signal and the interference. More specifically, rewriting (2) leads to

$$V[h(I_i)] = G[o(I_i)]X[I_i] + \sum_{j=0, j\neq i}^{K-1} G[d(I_i, I_j)]X[I_j], \tag{3}$$

where

$$o(I_i) = \left(h(I_i)\frac{N}{B} - f_F(I_i)\right)_{mod\ N},$$

and

$$d(I_i, I_j) = \left(h(I_i)\frac{N}{B} - f_F(I_j)\right)_{mod\ N}. \tag{4}$$

Since the window function $\mathbf{G}$ is known in advance, the value of $G[o(I_i)]$ is available. Dividing both sides by $G[o(I_i)]$, one can equalize the signal by

$$\begin{aligned} \tilde{X}[I_i] &= \frac{G[o(I_i)]X[I_i] + \sum_{j=0, j\neq i}^{K-1} G[d(I_i, I_j)]X[I_j]}{G[o(I_i)]} \\ &= \underbrace{X[I_i]}_{\text{desired signal}} + \underbrace{\frac{\sum_{j=0, j\neq i}^{K-1} G[d(I_i, I_j)]X[I_j]}{G[o(I_i)]}}_{\text{inter sparse interference}}. \end{aligned} \tag{5}$$

To combat the inter sparse interference in the equalized signal $\tilde{X}[I_i]$, several iterations shall be conducted for estimating accurate results. For this, the conventional estimation is introduced first [15]. Then we introduce the proposed new estimation to improve the performance. For representation purpose, we let $\tilde{X}^{(j)}[I_i]$ be the equalized value with index I_i in the jth outer loop iteration.

Conventional Estimator: In practice the number of outer loop iterations is limited, denoted by R_{outer}. Thus, one has the following observations:

$$\tilde{\mathbf{X}}[I_i] = \left[\tilde{X}^{(1)}[I_i], \tilde{X}^{(2)}[I_i], \cdots, \tilde{X}^{(R_{outer})}[I_i]\right]^T, \ I_i \in \mathcal{J}.$$

Due to the outliers, which are observations far from other observations, using the mean estimator is particularly susceptible to the influence of outliers. This is more pronounced when there are extreme values. In the algorithm proposed in [15], the median estimator was used, which performs better than the mean estimator when there are outliers in the observations. However, the median estimator can degrade performance in some cases. For instance, let $R_{outer} = 10$, and there are 10 observations for each candidate index. An index has the following observations: $\tilde{\mathbf{X}}[I_i] = (3, 3, 3, 3, 4, 5, 6, 7, 8, 8)^T$. In this example, the mean value is 5 and the median is 4.5. To overcome this, we notice the fact that most of the equalized values in individual iterations are very close. In the above example, the mode is 3. To utilize this finding, we propose to use a mode-mean estimator described as follows:

Proposed Mode-Mean Estimator: We notice that when the sparse signal indeed has a non-zero element in an index, the probability that this index has large equalized values after individual inner loop iterations is high; more importantly, the equalized values for individual iterations are almost the same, except for the outliers. Hence, in the proposed algorithm, we use a mode-mean method to estimate the results. More precisely, first we take mode operation, which identifies the candidate indices with close equalized values. For instance, we define a threshold M and an error deviation σ. If the number of outer loop iterations is R_{outer}, we identify an index as a sparse location if greater or equal to M of these R_{outer} equalized values have differences smaller than $10^{\lceil \log 4\sigma \rceil}$. Then these close values are averaged to obtain the reconstructed result $\check{X}[I_i]$, which is called mean operation. The proposed algorithm is summarized in Algorithm 1.

Example 1. Proposed mode-mean estimator in Algorithm 1: Let $R_{outer} = 10$, and hence we have 10 equalized values for each candidate index. An example is shown in Table I. Let $\sigma = 4.097 \times 10^{-8}$, $\gamma = -6$, $M = 3$. The corresponding rounded values are shown in the second column of this table. One can see that there are seven identical rounded values (boldfaced), which are greater than $M = 3$ and hence I_i is regarded as a sparse index. The mode value is $0.551155 + 0.382201j$. The reconstructed value for this index is $0.551155421 + 0.382201418j$, which is very close to the correct result $0.551155422 + 0.382201418j$.

Reasonable values for σ, M and R_{outer} and their meaning will be explained and become clear later in next section. The value of σ is related to inter sparse interference, and it is introduced in the following proposition:

Proposition 1: Assume that the sparse signal has i.i.d. sparse elements with zero mean and variance σ_x^2. Then the variance of inter sparse interference defined by

$$\sigma^2 = \mathrm{Var}\left\{\frac{\sum_{j=0, j\neq i}^{K-1} G[d(I_i, I_j)]X[I_j]}{G[o(I_i)]}\right\}, \tag{6}$$

Algorithm 1: Proposed mode-mean estimator.

Input: candidate index set $\mathcal{J}$; equalized values in R_{outer} individual iterations $\tilde{X}^{(\ell)}[I_i]$; predetermined M and σ;
Output: reconstructed signal $\check{X}[I_i]$;
1: **for** $I_i \in \mathcal{J}$ **do**
2: Round each equalized value $\tilde{X}^{(\ell)}[I_i]$ to decimal place $\gamma = \lceil \log 4\sigma \rceil$;
3: Obtain the mode value λ that most frequently occurs as a measure;
4: Keep and count the equalized values $\tilde{X}^{(\ell)}[I_i]$ that satisfies the condition $\left|\lambda - \tilde{X}^{(\ell)}[I_i]\right| \leq 10^{\gamma}, \ \ell = 1, \ldots, R_{outer}$;
5: If the count number is greater or equal to M, I_i is regarded as a sparse index; reconstructed signal $\check{X}[I_i]$ is obtained by averaging the kept values in Step 4.
6: **end for**

TABLE I
RECONSTRUCTING SPARSE SIGNAL USING ALGORITHM 1

Equalized values $\tilde{\mathbf{X}}[I_i]$	Rounded values	Mode λ
0.551155418 + 0.382201422j	**0.551155 + 0.382201j**	0.551155 + 0.382201j
-0.774625492 + 0.934655789j	-0.774625 + 0.934656j	
0.506512807 + 0.410869837j	0.506513 + 0.410870j	
0.551155419 + 0.382201414j	**0.551155 + 0.382201j**	
0.551155416 + 0.382201418j	**0.551155 + 0.382201j**	
0.551155422 + 0.382201422j	**0.551155 + 0.382201j**	
0.568574739 + 0.394654806j	0.568575 + 0.394655j	
0.551155421 + 0.382201416j	**0.551155 + 0.382201j**	
0.551155427 + 0.382201415j	**0.551155 + 0.382201j**	
0.551155425 + 0.382201417j	**0.551155 + 0.382201j**	

can be shown to be

$$\sigma^2 = (K-1)\sigma_x^2 \mathbb{E}\left\{\left(G[d(I_i, I_j)]\right)^2\right\}\mathbb{E}\left\{\left(\frac{1}{G[o(I_i)]}\right)^2\right\}. \tag{7}$$

Proof: Please see Appendix A. ■

From Proposition 1, since the window function is designed in advance, both $G[o(I_i)]$ and $|G[d(I_i, I_j)]|$ are known. More specifically, we know that $|G[d(I_i, I_j)]| < \delta_s$, for $d(I_i, I_j) \notin [-\epsilon_s N, \epsilon_s N]$. Note that δ_s is the maximum value of the stop band ripple, and the actual value of the stop band ripple shall be lower than δ_s. Hence, we can obtain an upper bound value for σ^2 by letting $\mathbb{E}\{(G[d(I_i, I_j)])^2\} = \delta$ as

$$\sigma^2 \leq \delta(K-1)\sigma_x^2 \mathbb{E}\left\{\left(\frac{1}{G[o(I_i)]}\right)^2\right\}. \tag{8}$$

Example 2. Inter sparse interference: Theoretical and simulation results: In this example, we show the accuracy of the theoretical result in (8). Let $\delta_s = 10^{-8}$. The sparse elements are assumed to be i.i.d. Gaussian with zero mean and unit variance. Fig. 2 shows the deviation σ as a function of K for various (original) FFT size N. Observe that the derived upper bound matches the simulation result quite well. If high accuracy is needed, one can use the theoretical result in (7).

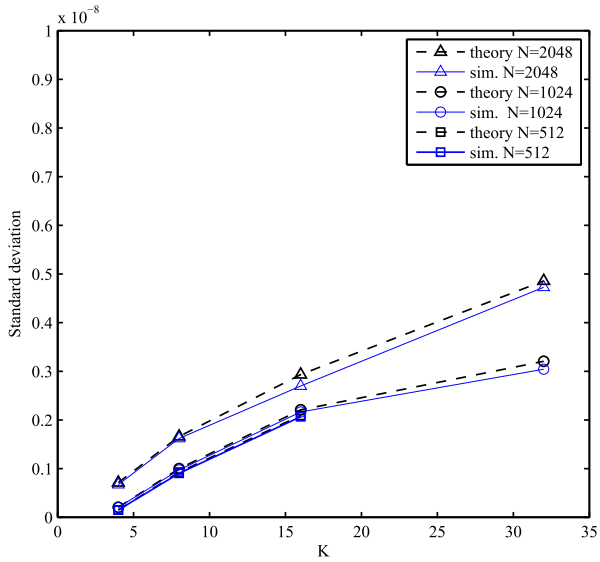


Fig. 2. The standard deviation σ of inter sparse interference for different values of N.

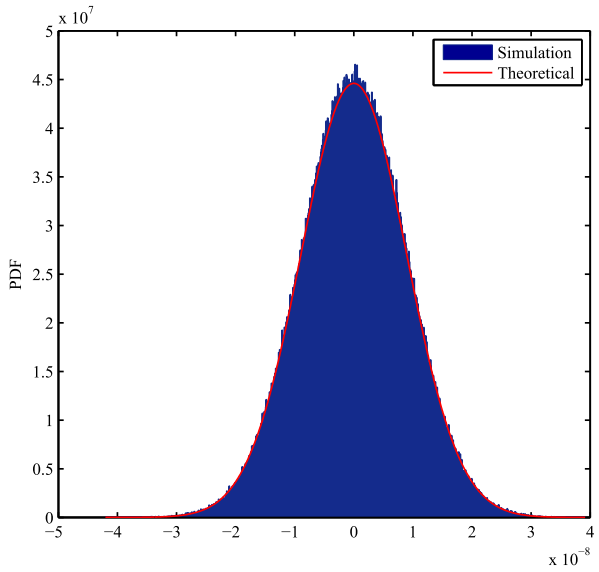


Fig. 3. Distribution of inter sparse interference.

When the value of K is large, the inter sparse interference is asymptotically Gaussian distribution. The following example demonstrates this approximation.

Example 3. Gaussian approximation of inter sparse interference: Let $N = 2^{15}$, $K = 128$, $B = 4K$ and the window function is $(8.85 \times 10^{-4}, 0.0039, 2 \times 10^{-9}, 7723)$. Fig. 3 shows the inter sparse interference and a Gaussian distribution with zero mean and standard deviation $\sigma = 1.265 \times 10^{-8}$. The two results are close thanks to the Law of Large Number.

The Gaussian approximation of the inter sparse interference can be used to determine the rounding decimal place in Algorithm 1. More specifically, since the inter sparse interference is asymptotically Gaussian with zero mean, and we have already derived its standard deviation σ in Proposition 1, the complete statistic is known. One can use this statistic to infer that the dynamic range of the equalized values from individual iterations. As mentioned before in Section II-D2, we set the dynamic range to be, e.g., 4σ, to obtain the mode of the proposed mode-mean estimator. In this case, the equalized values of a specific sparse element without overlapping with other sparse elements in individual iterations shall be close. The probability to have a difference greater than, e.g., say 4σ, shall be smaller than 10^{-3}.

E. Signal Updating Iterations

When the subsampled FFT size B approaches the value of K, for instance, $B = K$ or $B = 2K$, the recovering (reconstructing) performance degrades. Signal updating iterations can be conducted to overcome this. There is a tradeoff between complexity and performance when different numbers of outer loop and signal updating iterations are applied. That is, the main complexity lies on the FFT. When a small value of B is used, the FFT complexity is low while its recovering performance may degrade. However, if updating the signal are added to improve the performance, *i.e.,* each updating the signal subtracts part of the reconstructed sparse elements obtained from the previous iteration, the overall complexity can be lower than that obtained by using a large value of B but without updating the signal. Hence, if one fixes the recovering rate to be (nearly) 100%, there seems to have an optimal setting for the sparse FFT parameters that leads to the smallest complexity. The key to solve this problem relies on precise theoretical analysis of the recovering rate, which is a function of several parameters including SNR, the value of K, subsampled FFT size B, and original FFT size N, etc. To address this question, we need to derive closed-form solutions for the recovering rate, and this will be introduced in Section III.

Now let us describe how signal updating iterations work. The signal updating iterations subtract the identified sparse elements from previous iterations. Let $r = 1, 2, \cdots, R_{update}$, where R_{update} is the number of signal updating iterations. Referring to Fig. 1, the input signal after the rth updating iterations can be expressed as [16].

$$Z[k] = V[k] - V_{update}[k], \quad k = 0, 1, \ldots, B-1, \tag{9}$$

where $V_{update}[k]$, from (3), can be calculated by

$$V_{update}[h(I_i)] = \sum_{l=1}^{r} \sum_{j=0}^{K_r - 1} G[d(I_i, I_j)] \hat{X}[I_j], \tag{10}$$

and $K = \sum_{r=1}^{R_{update}} K_r$ with K_r is the number of sparse elements that can be identified at the rth updating iteration. After all updating iterations are done, the reconstructed signal $\hat{\mathbf{X}}$ is obtained by combining the reconstructed sparse elements from all iterations.

F. Performance Criteria

To make the sparse FFT more general, assuming that the input signal is i.i.d. complex Gaussian. In this case, the recovering rate for complex Gaussian signals shall be defined first. In this study, we define the sparse error and recovering rate as

Algorithm 2: Suggested SFFT with inner loop, outer loop, and signal updating iterations.

Input: a sparse signal $\mathbf{x}$; number of frequencies K; number largest magnitude elements in location loops, d; number iterations in updating the signal, R_{update}; number iterations in location loops, $R_{\ell oc}$; number iterations in estimated loops, R_{outer}; flat window function $\mathbf{g}$ and $\mathbf{G}$; subsampled FFT size, B;

Output: Estimated result $\hat{\mathbf{X}}$.

Initialization: $\hat{\mathbf{X}} = \mathbf{0}$.
for $r = 1 : R_{update}$ **do** % Signal updating iteration
  for $\ell = 1 : R_{outer}$ **do** % Outer loop
    Generate a uniform random number, β_T from $\{1, 3, 5, \ldots, N-1\}$ and find β_F.
    Permute signals with β_T using Section II-A.
    Window signals using Section II-B.
    Subsampled FFT and obtain $\mathbf{Z}$ using Section II-C.
    Store β_T, β_F, $\mathbf{Z}$ from all inner loop iterations.
    if $\ell \le R_{\ell oc}$ **then** % Location loop
      Make a score table.
    end if
  end for
  $\mathcal{J}$ = find candidate sparse index that is equal or more than score threshold in score table.
  for $\ell = 1 : R_{outer}$ **do** % Estimation loop
    reconstrtuct $\tilde{X}^{(\ell)}[I_i],\ I_i \in \mathcal{J}$.
  end for
  Obtain the reconstructed value $\check{\mathbf{X}}^{(r)}$ in the rth iteration using Algorithm 1;
  $\hat{\mathbf{X}} = \hat{\mathbf{X}} + \check{\mathbf{X}}^{(r)}$. % Combine sparse elements
  Take largest K magnitudes of $\hat{\mathbf{X}}$ and set the remainders to zeros.
end for

$$\text{Sparse Error} = \begin{cases} 1, & \left|\hat{X}[i] - X[i]\right| > V_{th}, \\ 0, & \left|\hat{X}[i] - X[i]\right| \le V_{th}, \end{cases}$$

$$\text{Recovering Rate} = \frac{1}{K}\sum_{i=0}^{K-1} \text{Sparse Error}, \tag{11}$$

where V_{th} can be determined according to the performance requirements. It is worth mentioning that the derived results are valid for arbitrary sparse signal including lattice points such as QAM. Taking QAM for instance, one can regard (11) as symbol error rate by setting V_{th} be a half of the minimum distance between two lattice points.

Another performance metric is the average L_1 error of the K sparse, which was defined in [15]:

$$\text{Average } L_1 \text{ Error} = \frac{1}{K}\sum_{i=0}^{N-1}\left|\hat{X}[i] - X[i]\right|. \tag{12}$$

The suggested sparse FFT using the proposed mode-mean estimator is summarized in Algorithm 2. Note that in this suggested sparse FFT, the method for the inner loop iterations is the same as that in [15] except that the proposed mode-mean estimator in Algorithm 1 is applied; while the concept to use signal updating iterations is inspired by [16]. Note that there is no signal updating iteration in [15], and the outer loop iterations used in [16] are different from those in [15].

Moreover, in this suggested scheme, location loop and estimation loop iterations have common function blocks, *i.e.,* permutation, windowing and subsampled FFT. Thus the results from these two iterations can be shared by each other to reduce iterations. That is, we can set $R_{outer} = R_{estimation} \ge R_{\ell oc}$.

III. Performance Analysis

In this section, the recovering rate performance of the proposed scheme is analyzed. The following two lemmas introduce how the sparse indices behave after subsampling in the frequency domain and help in deriving the recovering rate.

Lemma 1: If the index I_i has a uniform distribution in $[0, 1, 2, \ldots, N-1]$ and β_F defined in Section II-A has a uniform distribution in $[1, 3, 5, \ldots, N-1]$, where N is with power of 2, then the permuted index $I_{i'}$ obtained by $I_{i'} = f_F(I_i) = (\beta_F I_i)_{mod\ N}$ also has a uniform distribution in $[1, 2, 3, \ldots, N-1]$.

Proof: See [23] ■

Lemma 2: Let $h_F(I_i) = round\left(f_F(I_i)\frac{B}{N}\right)$ be a function that reflects signal permutation and subsampling of an index I_i, where B is the number of resulting samples after subsampling. If I_i has a uniform distribution in $[0, 1, 2, \ldots, N-1]$, then $h_F(I_i)$ has a uniform distribution in $[0, 1, 2, \ldots, B-1]$ with probability $\frac{1}{B}$.

Proof: See Appendix B. ■

We discuss the recovering rates for ideal and practical window functions respectively in the following two subsections:

A. Recovering Rate: Ideal Rectangular Window

Let us discuss ideal window function first, *i.e.,* ideal rectangular shape in the frequency domain. Let I_i be a sparse index, b be an index in $\{0, 1, \ldots, B-1\}$ in the subsampled FFT samples, and $h_F(I_i)$ be the function defined in Lemma 2. A sparse element cannot be recovered (or reconstructed) when it overlaps with other sparse elements after windowing and subsampling. That is, any two sparse elements overlap when the following conditions hold:

$$d(I_i, I_j) < \frac{N}{2B} \quad or \quad d(I_i, I_j) > N - \frac{N}{2B}, \tag{13}$$

where $d(I_i, I_j)$ is defined in (4). Because the ideal window has width $\frac{N}{B}$, sparse elements do not overlap before subsampling if their distances are larger than $\frac{N}{B}$. That is to say, two sparse elements overlap after subsampling when

$$h_F(I_i) = h_F(I_j). \tag{14}$$

The following two lemmas describe the probability of overlapping among sparse elements.

Lemma 3: In an ideal rectangular window function with width $\frac{N}{B}$ in the frequency domain, the probability that a specific sparse element does not overlap with other sparse elements

in one iteration can be expressed as

$$p = \left(1 - \frac{1}{B}\right)^{K-1}. \tag{15}$$

In other words, the probability that a specific sparse element overlaps with other sparse elements in one iteration is $1 - p$.

Proof: See Appendix C. ■

Now we discuss how to derive the recovering rate performance. As mentioned in the previous section, we use the mode-mean estimator to recover the FFT result. Recall that this method conducts R_{outer} iterations in the estimation loop, and averages the equalized samples whose rounded values are equal to the mode value as the reconstructed FFT result. If the number of equalized samples used in averaging is greater or equal to M, we identify it a sparse element.

That is, referring to Table I, we assume that: if a sparse element indeed appears in a frequency index I_i, the probability that several of the rounded values from the R_{outer} iterations are the same is very high. On the other hand, if there is no sparse element in a frequency index, it is very unlikely that the rounded values have the same result.

We further assume that if a specific sparse element does not overlap with other sparse elements, this sparse element should be able to be correctly reconstructed. If the above assumptions hold, the non-overlapping probability p in Lemma 3 can be regarded as the recovering probability in one iteration.

The assumptions are generally true. Although there are two situations that may violate the assumptions, the violations can be avoided as explained below: The first violation occurs when the window function is not ideal. In this case, the stop band ripple δ_s can affect the reconstructed values. As we have derived the variance of the inter sparse interference in Propositions 1, we know how to choose the rounding decimal place to avoid this violation. For instance, referring to Example 3, the famous 4σ theorem tells us that the rounded values have a probability less than 10^{-3} (this can be regarded as violation probability) to have differences larger than 4σ. This gives us an important reference to determine which decimal place to round. One can avoid this violation by adjusting the rounding decimal place such that this violation probability is in a negligible level. As we see in Example 2, the standard deviation σ is smaller than 10^{-8} for all parameter settings in this example. In this case, setting 8σ or 16σ to have negligible violation probability is not a problem.

This violation can also be avoided by setting M. Theoretically speaking, one may identify an element is indeed a sparse element if greater than or equal to $M = 2$ of its rounded values from R_{outer} iterations are the same. To avoid the violation, one can simply set $M = 3$ instead of 2. The simulation results show that although $M = 2$ may not be able to avoid this violation in some cases, setting $M = 3$ can nearly avoid this violation in most cases.

The second violation occurs when the window function is ideal, where some overlapped sparse elements from different iterations somehow have the same result, though this probability is very small. This happens more often when the number K of sparse elements is small. Taking two sparse elements for instance, if these two sparse overlap in two iterations, the rounded values for any one of these sparse elements in the two iterations are identical, because there is no inter sparse interference. In this case, the proposed algorithm may mistakenly regard the samples corresponding to the overlapped region (before subsampling) as the candidates of sparse elements, but actually these identical results are caused by overlapped sparse elements. The same situation applies to multiple sparse elements. Fortunately, when K increases, especially in the interested range of sparse FFT applications, the violation probability becomes very small. Lemma 4 describes this second violation probability.

TABLE II
SPECIFY FOR PROP. 2

Overlapping events	Probability	# events
no overlap i'	p_1	$x_{1,1}$
j' overlaps with i'	p_2	$x_{2,1}$
k' overlaps with i'		$x_{2,2}$
ℓ' overlaps with i'		$x_{2,3}$
j', k' overlap with i'	p_3	$x_{3,1}$
j', ℓ' overlap with i'		$x_{3,2}$
k', ℓ' overlap with i'		$x_{3,3}$
All elements overlap	p_4	$x_{4,1}$

Lemma 4: In an ideal window function, there may be some overlapped sparse elements from different iterations somehow have the same result. This phenomenon is more pronounced when the number K of sparse elements is small. Let p_i denotes the probability that exact i sparse elements overlap; *i.e.,* for $i = 1$ there is no overlap, for $i = 2$ two sparse elements overlap, and so on. Then p_i can be shown to be

$$p_i = \left(\frac{1}{B}\right)^{i-1}\left(1 - \frac{1}{B}\right)^{K-i}, \tag{16}$$

and the number of events with this probability is $\binom{K-1}{i-1}$.

Proof: See Appendix D. ■

From the Binomial theorem, the number of total events is 2^{K-1}.

Example 4. Overlapping events: Let us give an example to explain the overlapping events and probabilities in Lemma 4. Let $K = 4$, and the permuted sparse four indices be i', j', k', and ℓ'. From Lemma 4, there are totally 2^{4-1} events shown in Table II. Let $R_{outer} = 4$, each event may occur or not. For instance, one realization can be $x_{1,1} = x_{2,1} = x_{2,3} = x_{4,1} = 1$ and $x_{2,2} = x_{3,1} = x_{3,2} = x_{3,3} = 0$, where $\sum_i \sum_j x_{i,j} = R_{outer} = 4$.

Using these results, we have the following proposition for recovering rate performance.

Proposition 2: The recovering rate defined in (11) can be approximated by

$$S = \sum_{\alpha=M}^{R_{outer}} S_\alpha, \tag{17}$$

where S_α is given by

$$S_\alpha = \Pr\left[(x_{1,1} = \alpha) \cap (x_{2,1} < \alpha) \cap \ldots \cap (x_{K,1} < \alpha)\right], \tag{18}$$

and (18) can be calculated via

$$\Pr\left[\cap_{i=1}^{K}\cap_{j=1}^{\binom{K-1}{i-1}}(x_{i,j}=t_{i,j})\right] = \frac{R_{outer}!}{\prod_{i=1}^{K}\prod_{j=1}^{\binom{K-1}{i-1}}t_{i,j}!}\prod_{i=1}^{K}\prod_{j=1}^{\binom{K-1}{i-1}}p_i^{t_{i,j}}. \quad (19)$$

Proof: See Appendix E. ■

When K increases, the computational complexity for (18) increases rapidly. We notice that when K is sufficiently large, $t_{i,j}$ is small and may be ignored for $i > 2$. Thus we can further approximate the recovering rate in the following proposition.

Proposition 3: When the value of K is sufficiently large, the recovering rate defined in (11) can be approximated by (17), but now S_α can be further approximated by

$$S_\alpha = \binom{R_{outer}}{M}p^M(1-p)^{R_{outer}-M}, \quad (20)$$

and p is defined in (15).

Proof: See Appendix F. ■

B. Recovering Rate: Practical Window

For practical window function, the effects of transition band and stop band ripples shall be considered. Practical flat window function is defined as $(\epsilon_p, \epsilon_s, \delta_s, L)$ in Section II-B. Sparse elements after windowing in practical window overlap more often than in ideal window due to the transition band. Thus the overlapping condition for the sparse elements is modified as

$$d(I_i, I_j) \le \epsilon_s N \text{ or } d(I_i, I_j) \ge N(1-\epsilon_s), \quad (21)$$

where $d(I_i, I_j)$ is defined in (4).

In a practical window function, the overlapping condition in (14) no longer holds. Hence, we need the following two lemmas for obtaining the recovering rate.

Lemma 5: Let I_i and I_j be two different sparse indices that have a uniform distribution in $[0, 1, 2, \cdots, N-1]$. Then the distance function $d(I_i, I_j)$ defined in (21) also has the uniform distribution in $[0, 1, 2, \cdots, N-1]$.

Proof: See Appendix G. ■

Lemma 6: In a $(\omega_p, \omega_s, \delta_s, L)$ practical window function, the probability that a specific sparse element does not overlap with other sparse elements in one iteration can be expressed as

$$p = (1-2\epsilon_s)^{K-1}. \quad (22)$$

Proof: See Appendix H. ■

Using Lemmas 5–6, and previous results in Proposition 3, we have the recovering rate for practical window functions summarized in the following proposition.

Proposition 4: Given a $(\epsilon_p, \epsilon_s, \delta_s, L)$ practical window function. Let all the parameters remains the same as those in Proposition 3. Then the probability that M of the R_{outer} iterations have the same value can be expressed as in (20), where now p shall be from (22). The recovering rate defined in (11) can again be approximated by (17).

C. Signal Updating Iterations and Recovering Rate

Having the recovering rate for the outer loop iterations, the recovering rate for signal updating iterations can then be introduced in the following proposition.

Proposition 5: Given a $(\epsilon_p, \epsilon_s, \delta_s, L)$ practical window function. Let the number of signal updating iterations be R_{update}, and let all the parameters remains the same as those in Proposition 3. The recovering rate defined in (12) with the outer loop iterations can be approximated by

$$S_{R_{update}} = 1 - \prod_{r=1}^{R_{update}}(1-S_r), \quad (23)$$

where

$$S_r = \sum_{\alpha=M}^{R_{outer}}\binom{R_{outer}}{\alpha}p_r^\alpha(1-p_r)^{R_{outer}-\alpha},$$

$$p_r = (1-2\epsilon_s)^{(1-S_{r-1})K-1}.$$

For the first updating iteration, set $p_1 = (1-2\epsilon_s)^{K-1}$.

Note that when $(1-S_{r-1})K - 1 < 0$, and thus $p_r > 1$, it implies that the sparse element can be perfectly reconstructed at this updating iteration. In this case, we set $p_r = 1$ so that this position can be functional well.

IV. Simulation Results

Simulation results are provided to 1) demonstrate the accuracy of the derived analytical results, and 2) compare the performance of the proposed algorithm and the scheme in [15]. The input signal is complex Gaussian with zero mean and unit variance in the experiments. Practical window is applied unless specifically mentioned. The window function is designed by setting Bcst_loc = Bcst_est and $\delta = 10^{-8}$, see [15].

Experiment 1. Performance comparisons of conventional median and proposed mode-mean estimators: In this experiment, the proposed mode-mean estimator in Algorithm 1 is compared with the conventional median estimator in [15]. Letting $N = 2^{14}$ and $R_{outer} = 20$, the mean square errors as functions of different values of B/K for these two estimators are shown in Fig. 4. From the figure, the proposed estimator outperforms the median estimator in the interested values of B/K for $M = 2$ and $M = 3$. This result is not surprising because the proposed estimator finds the mode and thus can handle the outliers more robustly than the median estimator, whose performance is usually dominated by the accuracy of the location loop iterations.

Experiment 2. Performance comparison of conventional and proposed schemes: In this example, we compare the performance of the sFFT 1.0 [15] and the proposed schemes in terms of the recovering rate and the average $\mathbf{L_1}$ error defined in (11) and (12), respectively. The parameter setting is as follows: Let $K = 16$. The number of iterations for location loop is $R_{loc} = 7$ and that for estimation loops is $R_{outer} = 14$.

Fig. 5 shows the recovering rate as a function of N. Fig. 5 (b) is the zoom-in version of Fig. 5 (a). Solid and dashed curves are respectively for the proposed and the sFFT 1.0 schemes. We

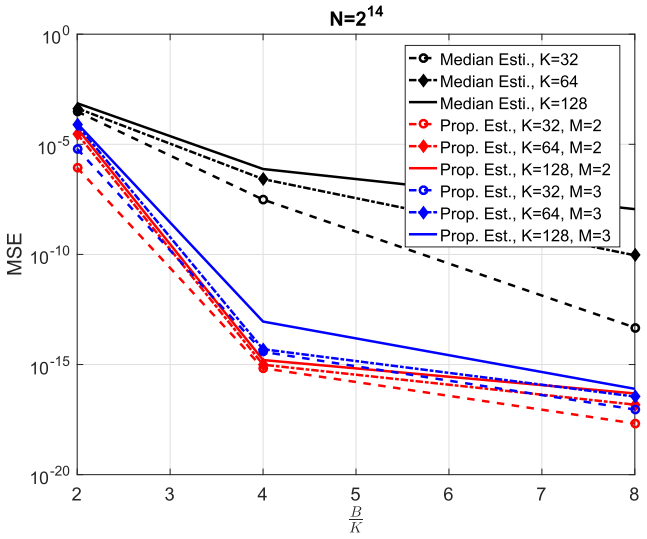
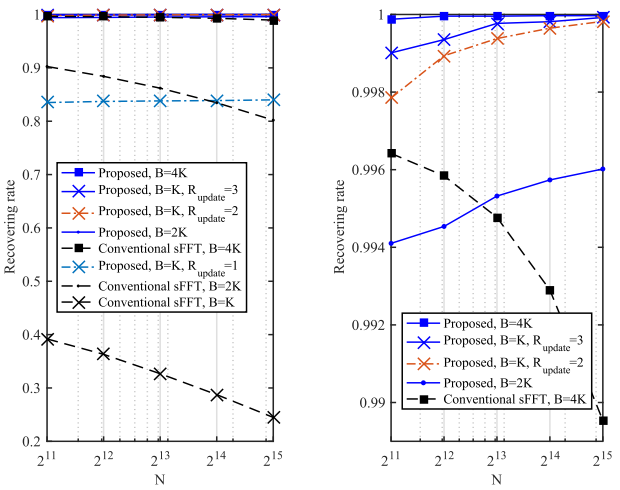


Fig. 4. MSE performance comparison between the conventional median estimator and the proposed mode-mean estimator.



(a) (b) zoom in

Fig. 5. Recovering rate as a function of N.

observe from these two figures that the proposed scheme has better recovering rate than that of the sFFT 1.0 scheme both for $B = K$ and $B = 4K$. Also, from Fig. 5 (b), the performance for $B = K$ with the signal updating iterations approaches that for $B = 4K$ without updating iterations, in this example, when $N = 16384$ the recovering rate for $B = K$ with three signal updating iterations is near the same as that for $B = 4K$. This result shows that with the aid of updating iterations, one can use a smaller size subsampled FFT, *i.e.,* $B = K$ in this example, to achieve comparable performance as that obtained by using a larger subsampled FFT size $B = 4K$.

The average L_1 error as a function of N is shown in Fig. 6. Again, the proposed scheme outperforms the sFFT 1.0 scheme for $B = K$, $B = 2K$ and $B = 4K$. Moreover, it is worth emphasizing that the proposed scheme with $B = K$ and $R_{update} = 3$ can outperform the sFFT 1.0 scheme with $B = 4K$.

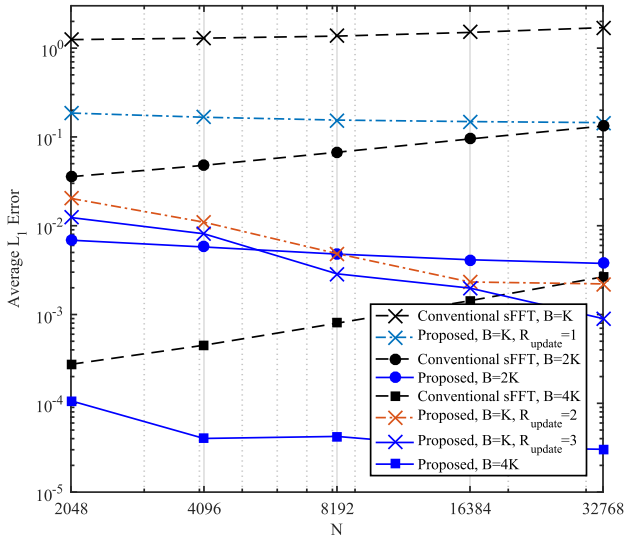


Fig. 6. Average L_1 error as a function of N.

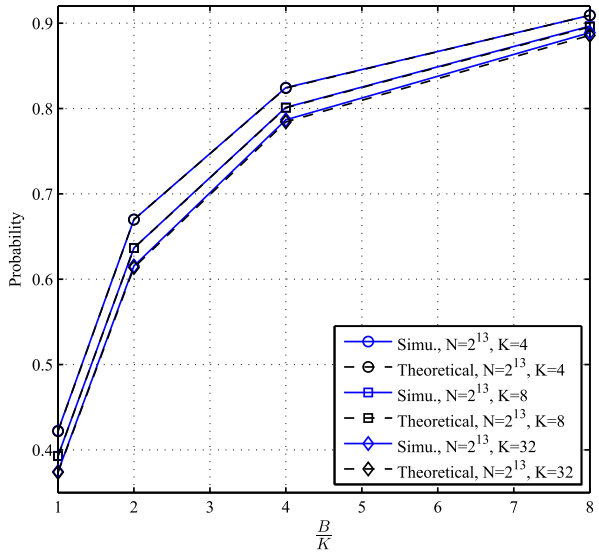


Fig. 7. The probability that a specific sparse element does not overlap with other sparse elements in one iteration.

Experiment 3. Theoretical and simulation results for recovering rate: In this experiment, we show the accuracy of the derived analytical results in Lemma 3 and Propositions 2–4. The number of iterations in the estimation loop is $R_{outer} = 10$.

Let us discuss ideal window first. As discussed in Lemma 3, the probability that a specific sparse element does not overlap with other sparse elements in one iteration has the probability p given in (15). Fig. 7 shows the analytical and simulation results for p as a function of B/K for different values of K. The two results match quite well.

As discussed in Proposition 3, when an element has its rounded values be identical for more than M times, we identify this as a sparse element and assume that it can be recovered. This corresponding probability is expressed in (20).

The corresponding simulation results and analytical results are shown in Figs. 8 and 9 for $K = 4$ and 64, respectively.

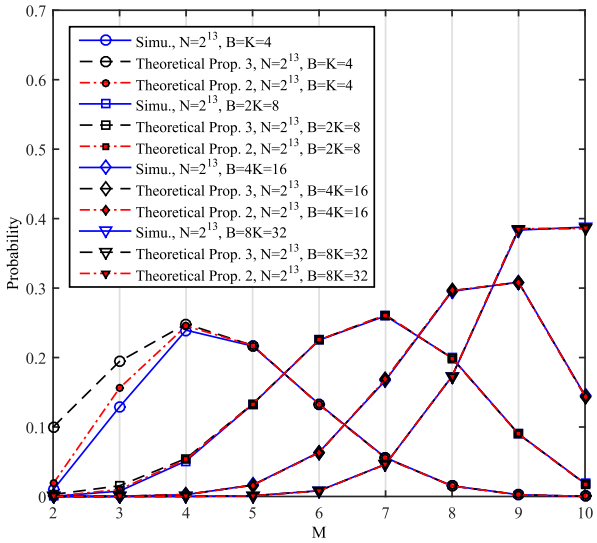


Fig. 8. Probability of the number m of occurrences that the recovered values of a specific sparse satisfy (20) in the 10 iterations: $K = 4$.

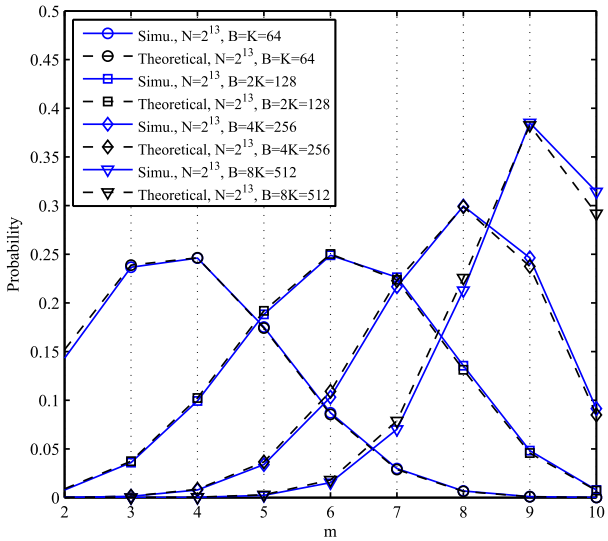


Fig. 9. Probability of the number m of occurrences that the recovered values of a specific sparse satisfy (20) in the 10 iterations: $K = 64$.

Observed from these three figures, in general the theoretical results match to the simulation results closely. The theoretical results in Proposition 3 are more accurate for a large value of K than smaller one. For instance, the theoretical results with $K = 4$ and when $B = K$ in Fig. 8 have some approximation error. This is reasonable because we have explained in Section III-A that when K is small, sometimes the overlapping sparse elements may happen to produce identical values in different iterations when ideal windowing is applied. Fortunately, the corresponding probability decreases as K increase. Thus the derived results become more accurate as K increases.

For a small value of K, the computational complexity is not an issue, using Proposition 2 leads to a more accurate approximation than using Proposition 3. From Fig. 8, we see that for a small value of K the analytical results using Proposition 2 indeed approximate better than that using Proposition 3.

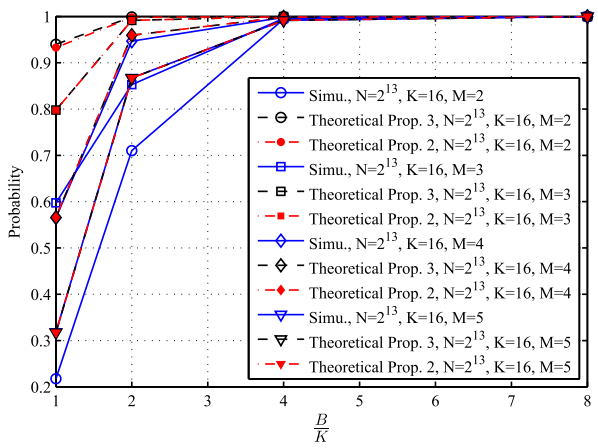


Fig. 10. Recovering rate as a function of B/K with $K = 16$: ideal windows.

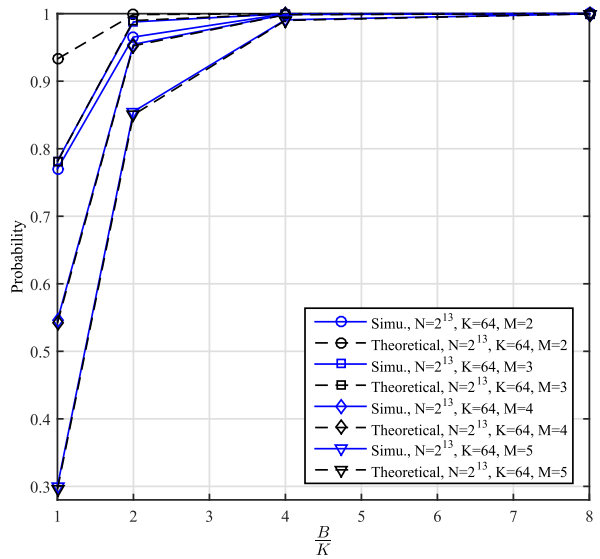


Fig. 11. Recovering rate as a function of B/K with $K = 64$ ideal windows.

In Figs. 10 and 11, we show the recovering rate comparison between the analytical result in (17) and true recovering rate from simulations. We have the following observations: first, setting $M \geq 3$ in (17), the analytical results match the simulation results well. Moreover, comparing these two figures, the theoretical results are more accurate for a larger value of K than a smaller one due to same reason explained above. In general, the original FFT size N is very large (more than 2^{13} according to [15]–[19]). Hence, the interested number K of sparse elements is usually large as well. In this case, the derived recovering rate in Proposition 3 is generally accurate. Nevertheless, as shown in the red circled curve in Figs. 10, if the computation is not an issue, the theoretical result in Proposition 2 can approximate the simulation result more accurately than Proposition 3.

Next, let us show the results for practical window functions. In Figs. 12 and 13, the comparison of the recovering rate between the analytical result in Proposition 4 and the simulation result are provided. Again the two results are quite close, and similar observations obtained in ideal window functions also appear in

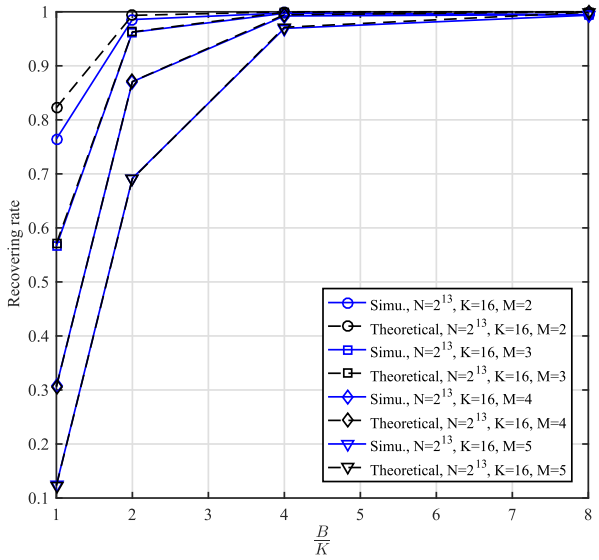


Fig. 12. Recovering rate as a function of B/K with $K = 16$: practical windows.

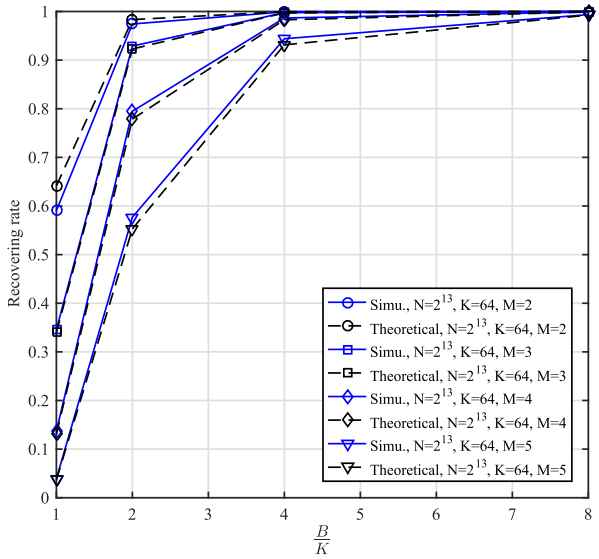


Fig. 13. Recovering rate as a function of B/K with $K = 64$: practical windows.

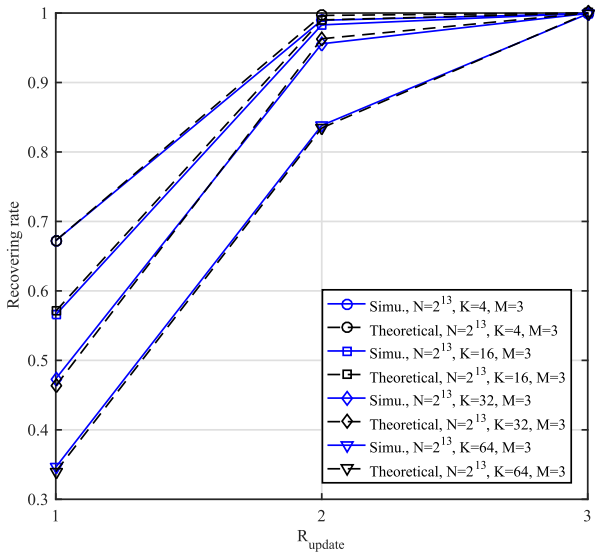


Fig. 14. Recovering rate as a function of the number of signal updating iterations: $B = K$.

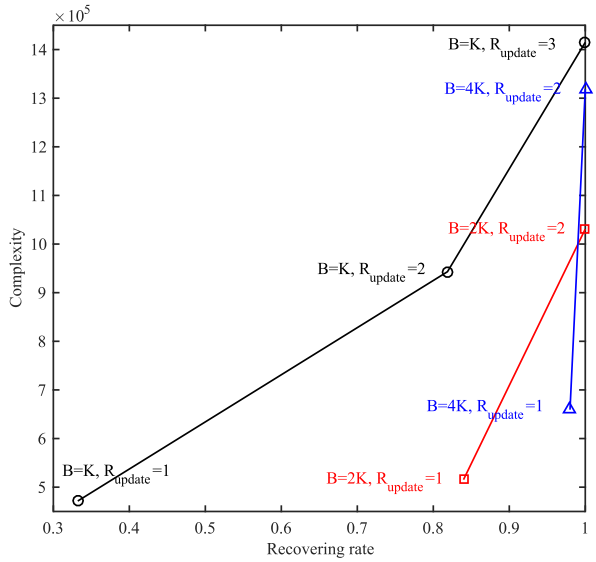


Fig. 15. Complexity as a function of recovering rate for various parameter settings: $K = 2^9$ and $N = 2^{15}$.

practical window function. All these results show the accuracy of the analytical approximations.

Experiment 4. Performance with the aid of signal updating iterations: We show how the performance is improved via signal updating and verify the closed-form expression in Proposition 5.

The theoretical results are obtained by letting $M = 3$ in Proposition 5. Let $B = K$, which demands the minimum subsampled FFT size. Fig. 14 shows the recovering rate as a function of the number R_{update} of signal updating iterations. Again, we see that the theoretical results corroborate the simulation results. Also, for $K \leq 32$, the recovering rate achieves above 95% using two updating iterations. The recovering rate is almost 100% using three updating iterations for all values of K in this simulation.

Experiment 5. Trade-off between performance and complexity with various parameter settings: In this experiment, we see how the derived analytical results help in determining the best parameter setting to achieve a good trade-off between performance and complexity. The philosophy is that by using a small value of B, the subsampled FFT size can be reduced; while its performance degradation can be compensated by using signal updating iterations. For the complexity, it is calculated as follows: The complexity of multiplying a window function with length L is with $\mathcal{O}(L)$, subsampled FFT with size B is with $\mathcal{O}(B \log B)$, and conducting both the location and estimation loop iterations is with $\mathcal{O}(dK\frac{N}{B})$. Because the numbers of iterations is R_{outer} for outer loops and is R_{update} for signal updating,

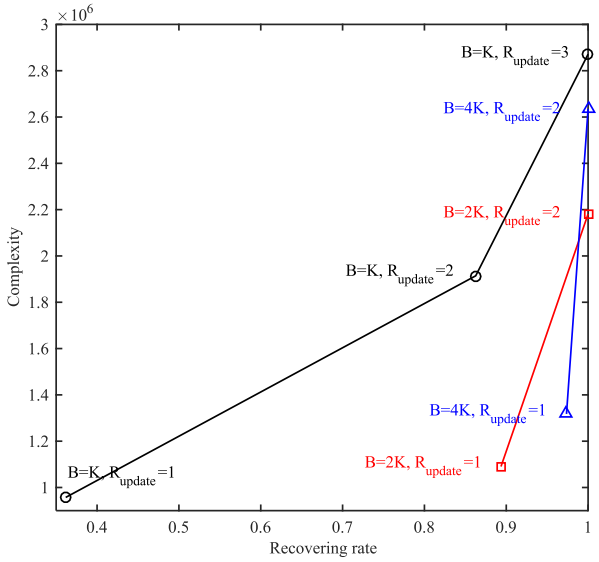


Fig. 16. Complexity as a function of recovering rate for various parameter settings: $K = 2^{10}$ and $N = 2^{16}$.

the total complexity has an order of

$$\mathcal{O}\left(R_{update}R_{outer}\left(L + B\log B + dK\frac{N}{B}\right)\right).$$

The number of iterations in the estimation loop is 10. Figs. 15 and 16 show the complexity as a function of recovering rate for various parameter settings, respectively, for $K = 2^9$, $N = 2^{15}$ and $K = 2^{10}$, $N = 2^{16}$.

We observe from the two figures, if nearly 100% recovering rate is demanded, minimum complexity is achieved by using $B = 2K$ and letting the number of signal updating iterations be two. Because there are a numerous number of parameter settings in practical designs, the theoretical results in Proposition 5 indeed help in determining a suitable parameter setting without running time-consuming simulations.

V. Conclusion

In this paper, we have analyzed the recovering rate performance of the SFFT system and derived the corresponding closed-form expressions. With these accurate closed-form results, the relationships between the performance and complexity are known for various implementations without conducting cumbersome simulations. Also, we have shown how to determine the parameters such as subsampling FFT size, and the numbers of outer loop and signal updating iterations to achieve the lowest computational complexity. Moreover, inspired by the theoretical results, we have proposed a new mode-mean estimator, which has been shown to outperform the conventional median estimator, verified via the simulation results. Since there is an increasing number of emerging applications that use large-size FFT and with sparse signals, these analyses and the new estimation algorithm provide timely engineering references for practical designs.

Appendix

A. Proof of Proposition 1

If two random variables X and Y are uncorrelated, the variance of the product of X and Y is given by

$$\begin{aligned}\mathrm{Var}(XY) &= \mathbb{E}\left\{(XY)^2\right\} - (\mathbb{E}\{XY\})^2 \\ &= \mathbb{E}\left\{X^2\right\}\mathbb{E}\left\{Y^2\right\} - (\mathbb{E}\{X\})^2(\mathbb{E}\{Y\})^2.\end{aligned}$$

Then (6) becomes

$$\begin{aligned}\sigma^2 = \mathbb{E}&\left\{\left(\sum_{j=0,j\neq i}^{K-1} G[d(I_i,I_j)]X[I_j]\right)^2\right\}\mathbb{E}\left\{\left(\frac{1}{G[o(I_i)]}\right)^2\right\} \\ &- \left(\mathbb{E}\left\{\sum_{j=0,j\neq i}^{K-1} G[d(I_i,I_j)]X[I_j]\right\}\right)^2\left(\mathbb{E}\left\{\frac{1}{G[o(I_i)]}\right\}\right)^2.\end{aligned}$$

Since the input sparse signal $\mathbf{X}$ is assumed to have zero mean, it yields $\mathbb{E}\left[\sum_{j=0,j\neq i}^{K-1} G[d(I_i,I_j)]X[I_j]\right] = 0$, and σ^2 becomes

$$\sum_{j=0,j\neq i}^{K-1}\mathbb{E}\left\{\left(G[d(I_i,I_j)]\right)^2\right\}\mathbb{E}\left\{\left(X[I_j]\right)^2\right\}\mathbb{E}\left\{\left(\frac{1}{G[o(I_i)]}\right)^2\right\}. \tag{24}$$

The expression in (24) leads to (7) by using the variance σ_x^2.

B. Proof of Lemma 2

Let $b = h_F(I_i)$ be an index in $\{0,1,,\ldots,B-1\}$ in subsampled FFT. Then the probability of $h_F(I_i)$ is given by

$$\Pr\left[h_F(I_i) = b\right] = \Pr\left[round\left(f_F(I_i)\frac{B}{N}\right) = b\right]. \tag{25}$$

Using the definition of rounding function, (25) becomes

$$\begin{aligned}&\Pr\left[b - \frac{1}{2} \le f_F(I_i)\frac{B}{N} < b + \frac{1}{2}\right] \\ &= \Pr\left[\left(b - \frac{1}{2}\right)\frac{N}{B} \le f_F(I_i) < \left(b + \frac{1}{2}\right)\frac{N}{B}\right].\end{aligned} \tag{26}$$

Using Lemma 1, we obtain that

$$\Pr\left[h_F(I_i) = b\right] = \frac{(b+\frac{1}{2})\frac{N}{B} - (b-\frac{1}{2})\frac{N}{B}}{N} = \frac{1}{B}. \tag{27}$$

C. Proof of Lemma 3

In Lemma 2, the probability that $h_F(I_i)$ falls in a specific index after subsampling is $\frac{1}{B}$. If the number of sparse elements is K, the question that a specific sparse element does not overlap with other sparse elements, *i.e.,* $h_F(I_i) \neq h_F(I_j), j \in \{0,1,\ldots,K-1\}\backslash\{i\}$, can be regarded as a combinatorics problem. This is equivalent to ask: Suppose that there are K independent balls numbered $X_0, X_1, \ldots, X_{K-1}$ thrown into B independent boxes numbered $b_0, b_1, \ldots, b_{B-1}$. What is the

probability that a box has only one ball? Hence this probability is given by:

$$p = B \times \left(\frac{1}{B}\right) \times (B-1)^{K-1} \times \left(\frac{1}{B}\right)^{K-1} = (B-1)^{K-1} \times \left(\frac{1}{B}\right)^{K-1} = \left(1-\frac{1}{B}\right)^{K-1}.$$

D. Proof of Lemma 4

Let p_2 denotes the probability that exactly two sparse elements overlap, *i.e.,* $h_F(I_i) = h_F(I_j) \neq h_F(I_k), I_k \in \mathbf{I}\backslash\{I_i, I_j\}$. From Lemma 3, the probability that a specific sparse element does not overlap with other sparse elements is $\left(1-\frac{1}{B}\right)^{K-1}$. It can be shown that p_2 is given by

$$p_2 = \frac{B \times 1 \times B^{K-2}}{B^K} = \left(\frac{1}{B}\right)^1 \left(1-\frac{1}{B}\right)^{K-2}, \quad (28)$$

and there are $\binom{K-1}{1}$ events with probability p_2.

Let p_3 denotes the probability that exactly three sparse elements overlap, *i.e.,* $h_F(I_i) = h_F(I_j) = h_F(I_k) \neq h_F(I_\ell), I_\ell \in \mathbf{I}\backslash\{I_i, I_j, I_k\}$. Similarly, p_3 can be shown to be

$$p_3 = \frac{B \times 1 \times 1 \times B^{K-3}}{B^K} = \left(\frac{1}{B}\right)^2 \left(1-\frac{1}{B}\right)^{K-3}, \quad (29)$$

and there are $\binom{K-1}{2}$ events with probability p_3.

Let p_i denotes the probability that exactly i sparse elements overlap. Applying the mathematical induction, we have

$$p_i = \frac{B \times 1^{(i-1)} \times B^{K-i}}{B^K} = \left(\frac{1}{B}\right)^{i-1} \left(1-\frac{1}{B}\right)^{K-i}, \quad (30)$$

and there are $\binom{K-1}{i-1}$ events with probability p_i.

E. Proof of Proposition 2

Let p_i be the probability that exact i sparse elements overlap and $x_{i,j}$ be the number of events that exact i sparse elements overlap (the jth situation) in the R_{outer} iterations, where $R_{outer} = \sum_{i=1}^{K} \sum_{j=1}^{\binom{K-1}{i-1}} x_{i,j}$. The probability that in the R_{outer} iterations, there are $x_{1,1}$ times that no sparse elements overlap $\cap$ there are $x_{2,1}$ times that two sparse elements overlaps (the first situation) $\cap \cdots \cap$ there are $x_{2,2}$ times that two sparse elements overlaps (the second situation) $\cap \cdots \cap$ there are $x_{K,1}$ times that K sparse elements overlaps can be expressed as in (19). Since mode operation is used in the proposed estimator, a specific sparse element can be reconstructed if the number of events that no sparse elements overlap is greater than the number of events that two sparse elements overlap, $\cap$ than that of three sparse elements overlap, $\cap \cdots \cap$ than that of K sparse elements overlap. That is, if $x_{1,1} = \alpha$, one needs $\alpha > x_{2,1} \cap \alpha > x_{2,2} \cap \ldots \cap \alpha > x_{K,1}$ to ensure that the mode operation leads to the correct result. This probability is defined in (18), and can be calculated using (19). Since the proposed mode-mean estimator identifies an element as a sparse element when greater than or equal to M equalized values from the R_{outer} iterations have identical rounded values, the identifying probability of the proposed scheme is the summation of S_α, for α from M to R_{outer}, as in (17). The recovering rate can be approximated by this identifying probability if the assumptions that we mentioned between Lemma 3 and Lemma 4 hold.

F. Proof of Proposition 3

When the value of K is sufficiently large, the probability that any two sparse elements overlap become small. The probability that more than two sparse elements overlap become negligible. In this case, we may set $t_{i,j} = 0$ for $i > 2$ in (19), and there are two events; one is no-overlapping and the other is overlapping by two sparse elements. Hence, there are two probabilities corresponding to these two events, *i.e.,* $p_{1,1}$ and $p_{2,i}$. By letting $p_{1,1} = p$ and $\sum_i p_{2,i} = (1-p)$, the probability in (19) can be approximated by (20). Similar approximation was used in [25].

G. Proof of Lemma 5

From Lemmas 1 and 2, $h_F(I_i)$ has a uniform distribution in $[0, 1, 2, \ldots, B-1]$ and $f_F(j)$ has a uniform distribution in $[0, 1, 2, \ldots, N-1]$. Then $X = h(I_i)\frac{N}{B}$ has a uniform distribution in $[0, \frac{N}{B}, 2\frac{N}{B}, \ldots, (B-1)\frac{N}{B}]$ and $Y = f_F(j)$ has a uniform distribution in $[0, 1, 2, \ldots, N-1]$. The difference $Z = X - Y$ can be regarded as the sum of two independent discrete random variables $Z = X + (-Y) = X + Y'$. Let the PMF of X be f_X and the PMF of Y' be f'_Y. Then the PMF of Z is given by

$$f_Z(z) = f_X * f_{Y'}(z) = \sum_{x=0}^{z} f_X(x) f_Y(x-z), \quad x = 0, 1, \ldots, z. \quad (31)$$

From (31), $f_Z(z)$ is with triangular shape. Let the PMF of Z' be $f_{Z'} = f_{(Z)_{mod\ N}}$. After performing modulo operation in the triangular shape PMF of $f_Z(z)$, the PMF of $f_{Z'}$ has a uniform distribution in $[0, 1, 2, \ldots, N-1]$ with probability $1/N$, which is also the PMF of $d(I_i, I_j)$.

H. Proof of Lemma 6

In a practical window, the condition that a specific sparse elements overlaps with other sparse elements when the conditions in (21) hold. From Lemma 5, $d(I_i, I_j)$ is uniformly distributed in $[0, 1, 2, \cdots, N-1]$. Thus the overlapping probability can be expressed as

$$\Pr\{d(I_i, I_j) \leq 2\epsilon_s N\} = \frac{2\epsilon_s N}{N} = 2\epsilon_s, \quad (32)$$

On the other hand, the non-overlapping probability is

$$1 - 2\epsilon_s. \quad (33)$$

Since there are totally K sparse elements, all the elements do not overlap each other in one iteration is thus given by

$$p = (1-2\epsilon_s)^{K-1}. \quad (34)$$

ACKNOWLEDGMENT

The authors would like to thank all the anonymous reviewers for their constructive suggestions, which have significantly improved the quality of this work. They would also like to thank H. Hassanieh *et al.* in MIT for opening the source codes of SFFT that have facilitated the development of this work. Additionally, the support from the Industrial Technology Research Institute (ITRI) is greatly appreciated.

REFERENCES

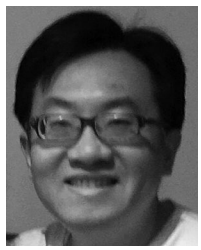
[1] *Very-High-Bit-Rate Digital Subscriber Line Transceiver 2 (VDSL2)*, ITU-T Standard G.993.2, Feb. 2006.
[2] *Digital Framing Structure, Channel Coding and Modulation for a Second Generation Digital Terrestrial Television Broadcasting System (DVB-T2)*, ETSI EN 302 755 V1.1.1, 2009.
[3] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," arXiv preprint arXiv:1312.5851, 2013.
[4] A. Zlateski, K. Lee, and H. S. Seung, "ZNN—A fast and scalable algorithm for training 3D convolutional networks on multi-core and many-core shared memory machines," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2016, pp. 801–811.
[5] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.
[6] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, pp. no. 4, 1289–1306, Apr. 2006.
[7] R. Baraniuk, "Compressive sensing," *IEEE Signal Process. Mag.*, vol. 24, no. 4, pp. 118–121, Jul. 2007.
[8] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 21–30, Mar. 2008.
[9] R. Baraniuk and P. Steeghs, "Compressive radar imaging," in *Proc. IEEE Radar Conf.*, Apr. 2007, pp. 128–133.
[10] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing MRI," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 72–82, Mar. 2008.
[11] T. T. Do, Yi Chen, D. T. Nguyen, N. Nguyen, L. Gan, and T. D. Tran, "Distributed compressed video sensing," in *Proc. IEEE Int. Conf. Image Process.*, Nov. 2009, pp. 1393–1396.
[12] L. Balzano, R. Nowak, and J. Ellenberg, "Compressed sensing audio demonstration," 2012. [Online]. Available: http://sunbeam.ece.wisc.edu/csaudio/
[13] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, "A tutorial on fast Fourier sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 57–66, Mar. 2008.
[14] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "sFFT: Sparse fast Fourier transform," 2015. [Online]. Available: http://groups.csail.mit.edu/netmit/sFFT/
[15] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse Fourier transform," in *Proc. 23rd Annu. ACM-SIAM Symp. Discrete Algo.*, 2012, pp. 1183–1194.
[16] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Nearly optimal sparse Fourier transform," in *Proc. 44th Annu. ACM Symp. Theory of Comput.*, 2012, pp. 563–578.
[17] A. C. Gilbert, P. Indyk, M. Iwen, and L. Schmidt, "Recent developments in the sparse Fourier transform: A compressed Fourier transform for big data," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 91–100, Sep. 2014.
[18] S.-H. Hsieh, C.-S. Lu, and S.-C. Pei, "Sparse fast Fourier transform by downsampling," *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 5637–5641.
[19] S.-H. Hsieh, C.-S. Lu, and S.-C. Pei, "Sparse fast Fourier transform for exactly and generally k-sparse signals by downsampling and sparse recovery," arXiv preprint arXiv:1407.8315, 2014.
[20] S. Heider, S. Kunis, D. Potts, and M. Veit, "A sparse prony FFT," in *Proc. 10th Int. Conf. Sampl. Theory App.*, 2013, pp. 572–575.
[21] A. Rauh and G. R. Arce, "Optimized spectrum permutation for the multidimensional sparse FFT," *IEEE Trans. Signal Process.*, vol. 65, no. 1, pp. 162–172, Jan. 2017.
[22] S. Pawar and K. Ramchandran, "R-FFAST: A robust sub-linear time algorithm for computing a sparse DFT," *IEEE Trans. Inf. Theory*, 2017, preprint, doi: 10.1109/TIT.2017.2679053.
[23] S. A. Talwalkar and S. L. Marple, "Time-frequency scaling property of discrete Fourier transform (DFT)," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2010, pp. 3658–3661.
[24] J. O. Smith, "Fourier theorems for the DFT," in *Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications*, 2nd ed., 2007. [Onine]. Available: http://ccrma.stanford.edu/∼jos/mdft/Fourier_Theorems_DFT.html. Accessed on: 2017.
[25] R. J. Larsen and M. L. Marx, *An Introduction to Mathematical Statistics and Its Applications*, 5th ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2012.

Gui-Lin Chen was born in Taipei, Taiwan, in 1989. He received the B.S. degree from the Department of Electrical Engineering, National Central University, Taoyuan, Taiwan, in 2011, and the M.S. degree in electrical and computer engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 2016. His research interests include signal processing for communications and wireless communications.

Shang-Ho (Lawrence) Tsai (SM'12) was born in Kaohsiung Taiwan. He received the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in August 2005. From June 1999 to July 2002, he was in the Silicon Integrated Systems Corporation, where he participated in the VLSI design for DMT-ADSL systems. From September 2005 to January 2007, he was in the MediaTek Inc., participating in the VLSI design for MIMO-OFDM systems and standard specifications for IEEE 802.11n.

He was a Visiting Fellow in the Department of Electrical Engineering at the Princeton University in June 2013–December 2013. Since February 2007, he has been in the Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan, where he is currently a Professor. His research interests include signal processing for communications, statistical signal processing, and signal processing for VLSI designs.

He was awarded a government scholarship for overseas study from the Ministry of Education, Taiwan, in 2002–2005.



Kai-Jiun Yang received the B.S. degree from Tamkang University, Taipei, Taiwan, R.O.C., in 1999, and the M.S. degree from University of Southern California, Los Angeles, CA, USA, in 2001, both in electrical engineering. From 2001 to 2009, he was in the Trendchip Technologies (now EcoNet Inc.) and developed DMT-ADSL chip-set. He is currently working toward the Ph.D. degree in electrical and control engineering at the National Chiao-Tung University, Hsinchu, Taiwan. He is also in the Industrial Technology Research Institute, Hsinchu, Taiwan, R.O.C., where he participates in developing machine learning platform. His research interests include signal processing, VLSI deign, and hardware–software codesign.