Results Math (2021) 76:53

1422-6383/21/020001-20
published online March 11, 2021
https://doi.org/10.1007/s00025-020-01330-0

Results in Mathematics

Deterministic Sparse Sublinear FFT with Improved Numerical Stability

Gerlind Plonka and Therese von Wulffen

Abstract. In this paper we extend the deterministic sublinear FFT algorithm in Plonka et al. (Numer Algorithms 78:133–159, 2018. https://doi.org/10.1007/s11075-017-0370-5) for fast reconstruction of M-sparse vectors $\mathbf{x}$ of length $N = 2^J$, where we assume that all components of the discrete Fourier transform $\hat{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$ are available. The sparsity of $\mathbf{x}$ needs not to be known a priori, but is determined by the algorithm. If the sparsity M is larger than $2^{J/2}$, then the algorithm turns into a usual FFT algorithm with runtime $\mathcal{O}(N \log N)$. For $M^2 < N$, the runtime of the algorithm is $\mathcal{O}(M^2 \log N)$. The proposed modifications of the approach in Plonka et al. (2018) lead to a significant improvement of the condition numbers of the Vandermonde matrices which are employed in the iterative reconstruction. Our numerical experiments show that our modification has a huge impact on the stability of the algorithm. While the algorithm in Plonka et al. (2018) starts to be unreliable for $M > 20$ because of numerical instabilities, the modified algorithm is still numerically stable for $M = 200$.

Mathematics Subject Classification. 65T50, 42A38.

Keywords. Sparse FFT, discrete Fourier transform, sublinear algorithm, Vandermonde matrices.

1. Introduction

Sparse FFT methods can be used in many different applications, where it is a priori known that the resulting signal in time/space or frequency domain is sparse. Such algorithms have earned a considerable interest within the last years.

Many deterministic sparse FFT algorithms are based on combinatorial approaches or phase shift, see e.g. [1,3,6,9,10,19]. These approaches usually need access to arbitrary values of a given function $f(x) = \sum_{j=1}^{M} a_j \, \mathrm{e}^{2\pi \mathrm{i} w_j x}$ assuming that the unknown frequencies w_j are in $[-N/2, N/2) \cap \mathbb{Z}$. The sparse FFT techniques in [8,17] are based on Prony's method.

By contrast, the deterministic algorithms proposed in [11,13,14,16], or in [15], Section 5.4, consider the fully discrete problem, where for a given vector $\mathbf{x} \in \mathbb{C}^N$, we want to efficiently compute its discrete Fourier transform $\hat{\mathbf{x}}$ under the assumption that $\hat{\mathbf{x}}$ is M-sparse or has a short support of length M. Recently, these techniques have also been transferred to derive sparse fast algorithms for the discrete cosine transform, [4,5].

Problem statement Let $\mathbf{x} = (x_j)_{j=0}^{N-1} \in \mathbb{C}^N$ with $N = 2^J$ for some $J > 1$. Further, let $\mathbf{F}_N := (\omega_N^{jk})_{j,k=0}^{N-1} \in \mathbb{C}^{N \times N}$ with $\omega_N := \mathrm{e}^{-2\pi \mathrm{i}/N}$ denote the Fourier matrix of order N, and $\mathbf{F}_N^{-1} = \frac{1}{N} \overline{\mathbf{F}}_N$. We consider the following two scenarios, which can essentially be treated with the same algorithm.

(a) Assume that $\hat{\mathbf{x}} := \mathbf{F}_N \, \mathbf{x} = (\hat{x}_k)_{k=0}^{N-1}$ is given. How do we, in a sublinear way, determine $\mathbf{x}$ from $\hat{\mathbf{x}}$, if it can be assumed that $\mathbf{x}$ is M-sparse with $M^2 < N$?

(b) Assume that $\mathbf{x} \in \mathbb{C}^N$ is given. How do we, in a sublinear way, determine $\hat{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$ from $\mathbf{x}$, if it can be assumed that $\hat{\mathbf{x}}$ is M-sparse with $M^2 < N$?

In both scenarios, M needs not to be known beforehand. However, if M is known, then this knowledge can be used to simplify the algorithm. Throughout the paper, we say that a vector $\mathbf{x}$ is M-sparse, if only M components have an amplitude that exceeds a predetermined small threshold $\epsilon > 0$.

This paper is organized as follows. In Sect. 2, we summarize the basic multi-scale idea of the algorithm used in [16] for the scenario (a). Section 3 is devoted to the extension of the method in [16]. First, we present the general pseudocode of the sparse FFT algorithm. The numerical stability of this algorithm mainly depends on the condition number of special Vandermonde matrices, which are used at each iteration step for solving a linear system with at most M unknowns. In Sect. 3.1 we give an estimate of the condition number of the occurring Vandermonde matrices, which are partial matrices of the Fourier matrix. This estimate is used in the sequel to determine the two free parameters determining the Vandermonde matrix. One parameter stretches the given nodes generating the Vandermonde matrix, and the second parameter determines the number of its rows. In Sect. 4 we briefly show, how the derived algorithm can be simply adapted to solve the sparse FFT problem (b). Finally, in Sect. 5 we present the large impact of the new approach that allows rectangular Vandermonde matrices. A Python implementation of the new algorithm is available under the link "software" on our homepage http://na.math.uni-goettingen.de.

2. Multi-scale Sparse Sublinear FFT Algorithm from [16]

We consider the problem stated in (a) to derive an iterative stable procedure to reconstruct $\mathbf{x}$ from adaptively chosen Fourier entries of $\hat{\mathbf{x}}$. To state the multi-scale algorithm from [16], we need to define the periodized vectors

$$\mathbf{x}^{(j)} = (x_k^{(j)})_{k=0}^{2^j-1} := \Big(\sum_{l=0}^{2^{J-j}-1} x_{k+2^j l} \Big)_{k=0}^{2^j-1} \in \mathbb{C}^{2^j}, \qquad j = 0, \ldots, J. \tag{1}$$

In particular, $\mathbf{x}^{(J)} = \mathbf{x}$ and $\mathbf{x}^{(0)} = \sum_{k=0}^{N-1} x_k$ is the sum of all components $\mathbf{x}$. Observe that, if the vector $\hat{\mathbf{x}} = (\hat{x}_k)_{k=0}^{N-1}$ is known, then also the Fourier transformed vectors $\hat{\mathbf{x}}^{(j)}$ are immediately known, and we have

$$\hat{\mathbf{x}}^{(j)} = \mathbf{F}_{2^j} \mathbf{x}^{(j)} = (\hat{x}_{2^{J-j}k})_{k=0}^{2^j-1}$$

(see Lemma 2.1 in [13]). Throughout the paper, we assume that no cancellation appears in the periodic vectors, i.e., for each significant component $|x_k| > \epsilon$ of $\mathbf{x}$, $k \in \{0, \ldots, N-1\}$, we have

$$|x_{k'}^{(j)}| > \epsilon \text{ for all } j = 0, \ldots, J-1, \qquad k' = k \bmod 2^j \tag{2}$$

for a fixed shrinkage constant $\epsilon > 0$. Condition (2) is for example satisfied if all components of $\mathbf{x}$ lie in one quadrant of the complex plane, e.g. $\operatorname{Re} x_j \geq 0$ and $\operatorname{Im} x_j \geq 0$ for $j = 1, \ldots, N-1$.

Idea of the algorithm The multi-scale algorithm in [16] iteratively computes $\mathbf{x}^{(j+1)}$ from $\mathbf{x}^{(j)}$, for $j = j_0, \ldots, J-1$. If the sparsity M of $\mathbf{x}$ is unknown, then we start with $j_0 = 0$ and $\mathbf{x}^{(0)} := \hat{x}_0 = \sum_{k=0}^{N} x_k$. If M with $M^2 < N$ is known beforehand, then we fix $j_0 = \lfloor \log_2 M \rfloor + 1$ and compute

$$\mathbf{x}^{(j_0)} := \mathbf{F}_{2^{j_0}}^{-1} \hat{\mathbf{x}}^{(j_0)} = \frac{1}{2^{j_0}} \overline{\mathbf{F}}_{2^{j_0}} \, (\hat{x}_{2^{J-j_0}k})_{k=0}^{2^{j_0}-1}$$

using an FFT algorithm with complexity $\mathcal{O}(j_0\, 2^{j_0}) = \mathcal{O}(M \log M)$. At the j-th iteration step, we assume that $\mathbf{x}^{(j)} \in \mathbb{C}^{2^j}$ with sparsity M_j has already been computed. Then we always have $M_j \leq M$. For $M_j^2 < 2^j$, the computation of $\mathbf{x}^{(j+1)}$ from $\mathbf{x}^{(j)}$ is based on the following theorem (see Theorem 2.2 in [16]).

Theorem 2.1. *Let $\mathbf{x}^{(j)}$, $j = 0, \ldots, J-1$, be the vectors defined in (1) satisfying (2). Then, for each $j = 0, \ldots, J-1$, we have: if $\mathbf{x}^{(j)} \in \mathbb{C}^{2^j}$ is M_j-sparse with support indices $0 \leq n_1 < n_2 < \ldots < n_{M_j} \leq 2^j - 1$, then the vector $\mathbf{x}^{(j+1)}$ can be uniquely recovered from $\mathbf{x}^{(j)}$ and M_j components $\hat{x}_{k_1}, \ldots, \hat{x}_{k_{M_j}}$ of $\hat{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$, where the indices $k_1, \ldots, k_{M_j}$ are taken from the set $\{2^{J-j-1}(2l+1) : l = 0, \ldots 2^j - 1\}$ such that the matrix*

$$\mathbf{A}^{(j)} := \Big(\omega_N^{k_p n_r} \Big)_{p,r=1}^{M_j} \tag{3}$$

is invertible.

The proof of Theorem 2.1 is constructive. With the notation $\mathbf{x}^{(j+1)} = \begin{pmatrix} \mathbf{x}_0^{(j+1)} \\ \mathbf{x}_1^{(j+1)} \end{pmatrix}$, i.e., $\mathbf{x}_0^{(j+1)} := \left(x_\ell^{(j+1)}\right)_{\ell=0}^{2^j-1}$ and $\mathbf{x}_1^{(j+1)} := \left(x_\ell^{(j+1)}\right)_{\ell=2^j}^{2^{j+1}-1}$, we have from (1)

$$\mathbf{x}^{(j)} = \mathbf{x}_0^{(j+1)} + \mathbf{x}_1^{(j+1)}. \tag{4}$$

Thus, if $\mathbf{x}^{(j)}$ is known, it suffices to compute $\mathbf{x}_0^{(j+1)}$, while $\mathbf{x}_1^{(j+1)}$ then follows from (4). We can now use the factorization of the Fourier matrix $\mathbf{F}_{2^{j+1}}$ (see Equation (5.9) in [15]), and obtain

$$\begin{pmatrix} (\hat{x}_{2\ell}^{(j+1)})_{\ell=0}^{2^j-1} \\ (\hat{x}_{2\ell+1}^{(j+1)})_{\ell=0}^{2^j-1} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{2^j} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{2^j} \end{pmatrix} \begin{pmatrix} \mathbf{x}_0^{(j+1)} + \mathbf{x}_1^{(j+1)} \\ \mathbf{W}_{2^j}(\mathbf{x}_0^{(j+1)} - \mathbf{x}_1^{(j+1)}) \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{2^j}\mathbf{x}^{(j)} \\ \mathbf{F}_{2^j}\,\mathbf{W}_{2^j}(2\mathbf{x}_0^{(j+1)} - \mathbf{x}^{(j)}) \end{pmatrix},$$

where $\mathbf{W}_{2^j} := \operatorname{diag}(\omega_{2^{j+1}}^0, \ldots, \omega_{2^{j+1}}^{2^j-1})$, and $\mathbf{0}$ denotes the zero matrix of size $2^j \times 2^j$. Thus, we conclude

$$\mathbf{F}_{2^j}\,\mathbf{W}_{2^j}\left(2\mathbf{x}_0^{(j+1)} - \mathbf{x}^{(j)}\right) = \left(\hat{x}_{2\ell+1}^{(j+1)}\right)_{\ell=0}^{2^j-1}. \tag{5}$$

Further, (4) together with (2) implies that $\mathbf{x}_0^{(j+1)}$ can only have significant entries for the same index set as $\mathbf{x}^{(j)}$, and we have to compute only these M_j entries. Introducing the restricted vectors

$$\tilde{\mathbf{x}}_0^{(j+1)} := \left(x_{n_r}^{(j+1)}\right)_{r=1}^{M_j} \in \mathbb{C}^{M_j}, \quad \tilde{\mathbf{x}}^{(j)} := \left(x_{n_r}^{(j)}\right)_{r=1}^{M_j} \in \mathbb{C}^{M_j},$$

we can also restrict the matrix $\mathbf{F}_{2^j}\,\mathbf{W}_{2^j} \in \mathbb{C}^{2^j \times 2^j}$ in the linear system (5) to its M_j columns with indices n_r. Finally, it suffices to restrict the system in (5) to M_j linear independent rows, and $\mathbf{x}_0^{(j+1)}$ can still be uniquely computed. Therefore a restriction $\mathbf{A}^{(j)} \in \mathbb{C}^{M_j \times M_j}$ of the product $\mathbf{F}_{2^j}\,\mathbf{W}_{2^j}$ can be chosen as

$$\mathbf{A}^{(j)} := \left(\omega_{2^j}^{h_p n_r}\right)_{p,r=1}^{M_j} \operatorname{diag}\left(\omega_{2^{j+1}}^{n_1}, \ldots, \omega_{2^{j+1}}^{n_{M_j}}\right). \tag{6}$$

Here, the matrix $\left(\omega_{2^j}^{h_p n_r}\right)_{p,r=1}^{M_j}$ is a restriction of $\mathbf{F}_{2^j}$ to the the rows $0 \le h_1 < h_2 < \ldots < h_{M_j} \le 2^j$ and columns n_r, $r = 1, \ldots, M_j$ corresponding to support indices of $\mathbf{x}^{(j)}$. The diagonal matrix is the restriction of $\mathbf{W}_{2^j}$ to the rows and columns n_r. Comparison with (3) yields $k_p = 2^{J-j-1}(2h_p + 1)$, $p = 1, \ldots, M_j$. In Algorithm 2.3 in [16], Theorem 2.1 is applied to iteratively compute $\mathbf{x}^{(j+1)}$ from $\mathbf{x}^{(j)}$, if solving the restricted linear system

$$\mathbf{A}^{(j)}\left(2\tilde{\mathbf{x}}_0^{(j+1)} - \tilde{\mathbf{x}}^{(j)}\right) = \left(\hat{x}_{2h_p+1}^{(j+1)}\right)_{p=1}^{M_j} \tag{7}$$

is cheaper than an FFT algorithm for vectors of length 2^j.

The further results in [16] focus on finding good choices of indices $(h_p)_{p=1}^{M_j}$ at each iteration step. Thereby, the paper restricts to matrices $\mathbf{A}^{(j)}$ of the form

$$\mathbf{A}^{(j)} := \left(\omega_{2^j}^{\sigma_j\, p\, n_r}\right)_{p=0,r=1}^{M_j-1,M_j} \operatorname{diag}\left(\omega_{2^{j+1}}^{n_1}, \dots \omega_{2^{j+1}}^{n_{M_j}}\right), \tag{8}$$

i.e., we choose $h_{p+1} = \sigma_j p$ for $p = 0, \dots, M_j - 1$ and some parameter $\sigma_j \in \{1, \dots, 2^j - 1\}$. The first matrix in the factorization (8) is a Vandermonde matrix generated by the roots of unity $w_{2^j}^{\sigma_j n_r}$, $r = 1, \dots, M_j$. The iterative algorithm which is based on Theorem 2.1 will be stable, if the linear system (7) can be efficiently computed in a stable way at each level $j = j_0, \dots, J$. Therefore, [16] tries to find parameters $\sigma_j \in \{1, \dots, 2^j - 1\}$ such that

$$\mathbf{V}_{M_j}(\sigma_j) := \left(\omega_{2^j}^{\sigma_j\, p\, n_r}\right)_{p=0,r=1}^{M_j-1,M_j}$$

is invertible and has a good condition number. Observe that $\mathbf{V}_{M_j}(\sigma_j)$ is always invertible if we choose $\sigma_j = 1$. However, $\sigma_j = 1$ can lead to a very bad condition number of $\mathbf{V}_{M_j}(\sigma_j)$ and $\mathbf{A}^{(j)}$, respectively.

Remark 2.2. Using Theorem 2.1, the reconstruction algorithm is based on the idea to iteratively compute periodizations $\mathbf{x}^{(j)} \in \mathbb{C}^{2^j}$ of $\mathbf{x} \in \mathbb{C}^{2^J}$ of growing length 2^j. At each iteration step, we rigorously exploit the sparsity of these vectors $\mathbf{x}^{(j)}$ and conclude from the support $\{n_1, \dots, n_{M_j}\}$ of $\mathbf{x}^{(j)}$ that the support set of $\mathbf{x}^{(j+1)}$ can only be a subset of $\{n_1, \dots, n_{M_j}\} \cup \{n_1 + 2^j, \dots, n_{M_j} + 2^j\}$. Therefore, the assumption (2) is crucial, since otherwise, not all support indices may be found.

If (2) is not satisfied and if the sparsity M of $\mathbf{x}$ is known beforehand, then the iteration would start by computing the periodization $\mathbf{x}^{(j_0)}$ of length $2^{j_0} > M$ directly, and we can compare the sparsity of $\mathbf{x}^{(j_0)}$ with M to ensure that no cancellation appears. If the sparsity of $\mathbf{x}^{(j_0)}$ is smaller than M, we could then employ a direct FFT algorithm to find the next periodizations $\mathbf{x}^{(j)}$, $j > j_0$, until the sparsity of $\mathbf{x}^{(j)}$ is equal to M. The complexity of the algorithm would then increase and depends on the level, where the last cancellation appears. In the worst case, if cancellation appears already in $\mathbf{x}^{J-1}$, we would get the complexity of a usual FFT algorithm.

3. Extension of the Sparse FFT Algorithm

The main contribution of this paper is an extension of the algorithm proposed in [16], which tremendously improves the stability of that algorithm to make it really applicable.

We will stay with the iterative approach to compute $\mathbf{x}^{(j+1)} \in \mathbb{C}^{2^{j+1}}$ from the M_j-sparse vector $\mathbf{x}^{(j)} \in \mathbb{C}^{2^j}$ via (7) and (4), where we consider only matrices $\mathbf{A}^{(j)}$, which are given as a product of a Vandermonde matrix and a diagonal matrix (with condition number 1) as in (8), and we will also try

to find a suitable parameter $\sigma_j \in \{1, \dots, 2^j - 1\}$ to improve the numerical stability of the system. The Vandermonde structure provides the advantage that the system in (7) can be solved with computational cost of $\mathcal{O}(M^2)$ (see, e.g., [7]).

We however do not insist on a square matrix as in [16], but allow the Vandermonde matrix factor to be a rectangular matrix with more rows than columns of the form

$$\mathbf{V}_{M_j',M_j}(\sigma_j) := \left(\omega_{2^j}^{\sigma_j\, p\, n_r}\right)_{p=0,r=1}^{M_j'-1,M_j}, \qquad M_j' \geq M_j. \tag{9}$$

We will choose the number of rows of the Vandermonde matrix $\mathbf{V}_{M_j',M_j}(\sigma_j)$ adaptively at each iteration step based on the obtained estimate of the condition number of $\mathbf{V}_{M_j',M_j}(\sigma_j)$, where

$$\kappa_2(\mathbf{V}_{M',M}(\sigma)) := \frac{\max_{\mathbf{u}\in\mathbb{C}^M,\|\mathbf{u}\|_2=1} \|\mathbf{V}_{M',M}(\sigma)\,\mathbf{u}\|_2}{\min_{\mathbf{u}\in\mathbb{C}^M,\|\mathbf{u}\|_2=1} \|\mathbf{V}_{M',M}(\sigma)\,\mathbf{u}\|_2}. \tag{10}$$

We start with presenting the general pseudo code for the case of unknown sparsity M. In the further subsections, we will particularly present, how the matrix $\mathbf{A}^{(j)}$ needs to be chosen, where we allow now a rectangular matrix. In Algorithm 3.1, we use the set notation $I^{(j)} + 2^j := \{n + 2^j : n \in I^{(j)}\}$.

Algorithm 3.1. Sparse (inverse) FFT for unknown sparsity M

Input: $N = 2^J$ (length of the vector $\mathbf{x}$),
ϵ (shrinkage constant),
possible access to Fourier values $\hat{x}_k$, $k = 0, \dots, N-1$.

Initialization:

if $|\hat{x}_0| < \epsilon$, Output: $M = 0$, $\mathbf{x} = \mathbf{0}$, $I^{(J)} = \emptyset$.
if $|\hat{x}_0| \geq \epsilon$, then $M := 1$, $I^{(0)} := \{0\}$, and $\tilde{\mathbf{x}}^{(0)} = \hat{x}_0$.

Loop

for $j = 0, \dots, J-1$:
if $M^2 \geq 2^j$, then

Determine $\mathbf{x}_0^{(j+1)}$:

Put $\hat{\mathbf{z}}^{(j+1)} := \left(\hat{x}_{2p+1}^{(j+1)}\right)_{p=0}^{2^j-1} = \left(\hat{x}_{2^{J-j-1}(2p+1)}\right)_{p=0}^{2^j-1} \in \mathbb{C}^{2^j}$.

Compute $\mathbf{x}_0^{(j+1)} := \frac{1}{2}\left(\mathrm{diag}\left((\omega_{2^{j+1}}^k)_{k=0}^{2^j-1}\right)^* (\mathbf{F}_{2^j})^{-1}\,\hat{\mathbf{z}}^{(j+1)} + \mathbf{x}^{(j)}\right)$ using an FFT algorithm.

Determine $\mathbf{x}^{(j+1)}$ and $I^{(j+1)}$:

Compute $\mathbf{x}_1^{(j+1)} := \mathbf{x}^{(j)} - \mathbf{x}_0^{(j+1)}$.

Put $\mathbf{x}^{(j+1)} := \left((\mathbf{x}_0^{(j+1)})^T, (\mathbf{x}_1^{(j+1)})^T\right)^T$.

Determine the index set $I^{(j+1)}$ by deleting all indices in $\left(I^{(j)} \cup (I^{(j)} + 2^j)\right)$ that correspond to entries in $\mathbf{x}^{(j+1)}$ with modulus being smaller

than ϵ.
Set $M := \#I^{(j+1)}$.

else

Set $\tilde{\mathbf{x}}^{(j)} = (\mathbf{x}_l^{(j)})_{l \in I^{(j)}}$.
Determine the Matrix $\mathbf{A}^{(j)} \in \mathbb{C}^{M' \times M}$ and the index set $\{h_{p_1}, \ldots, h_{p_M}\}$: see Sections 3.2 and 3.3.
Determine $\tilde{\mathbf{x}}_0^{(j+1)}$:
Choose the Fourier values $\hat{\mathbf{z}}^{(j+1)} := \left(\hat{x}_{2h_p+1}^{(j+1)}\right)_{p=1}^{M'}$
$= \left(\hat{x}_{2^{J-j-1}(2h_p+1)}\right)_{p=1}^{M'}$.
Compute $\tilde{\mathbf{x}}_0^{(j+1)}$ by solving the system

$$\mathbf{A}^{(j)}\left(2\tilde{\mathbf{x}}_0^{(j+1)} - \tilde{\mathbf{x}}^{(j)}\right) = \hat{\mathbf{z}}^{(j+1)}. \tag{11}$$

Determine $\tilde{\mathbf{x}}^{(j+1)}$ and $I^{(j+1)}$:
Compute $\tilde{\mathbf{x}}_1^{(j+1)} := \tilde{\mathbf{x}}^{(j)} - \tilde{\mathbf{x}}_0^{(j+1)}$.
Put $\tilde{\mathbf{x}}^{(j+1)} := \left((\tilde{\mathbf{x}}_0^{(j+1)})^T, (\tilde{\mathbf{x}}_1^{(j+1)})^T\right)^T$.
Determine the index set $I^{(j+1)}$ by deleting all indices in $\left(I^{(j)} \cup (I^{(j)} + 2^j)\right)$ that correspond to entries in $\tilde{\mathbf{x}}^{(j+1)}$ with modulus being smaller than ϵ.
Set $M := \#I^{(j+1)}$.

Output: $I^{(J)}$, the set of active indices in of $\mathbf{x}$,
$\tilde{\mathbf{x}} = \tilde{\mathbf{x}}^{(J)} = (x_l)_{l \in I^{(J)}}$, the vector restricted to nonzero entries.

To determine the suitable matrix

$$\mathbf{A}^{(j)} = \mathbf{V}_{M_j', M_j}(\sigma_j)\, \mathrm{diag}\left(\omega_{2^{j+1}}^{n_1}, \ldots \omega_{2^{j+1}}^{n_{M_j}}\right),$$

we have to find a well-conditioned Vandermonde matrix $\mathbf{V}_{M_j', M_j}(\sigma_j)$. Our procedure consists of two steps.

1) We compute a suitable parameter σ_j with $\mathcal{O}(M^2)$ operations.
2) We compute the number M_j' of needed rows in the Vandermonde matrix, to achieve a well-conditioned coefficient matrix in the system (11).

As seen already in [16], we can simplify the procedure of determining $\mathbf{V}_{M_j', M_j}(\sigma_j)$, if the number of significant entries M_j of $\mathbf{x}^{(j)}$ did not change in the previous iteration step, i.e., if $M_{j-1} = M_j$. In this case, we can just choose $\sigma_{j+1} := 2\sigma_j$ and stay with the number of columns, i.e., $M_j' := M_{j-1}'$ (see also Sect. 3.4).

3.1. Estimation of the Condition Number of $\mathbf{V}_{M_j',M_j}(\boldsymbol{\sigma_j})$

It is crucial for our algorithm to have a good estimate of the condition number of $\mathbf{V}_{M_j',M_j}(\sigma_j)$. The condition number of $\mathbf{V}_{M_j',M_j}(\sigma_j)$ strongly depends on the minimal distance between its generating nodes $\omega_{2^j}^{\sigma_j n_r}$. More precisely, we have the following theorem (see [12,16] or Theorem 10.23 in [15]).

Theorem 3.2. *Let $0 \le n_1 < n_2 < \ldots < n_{M_j} < 2^j$ be a given set of indices. For a given $\sigma_j \in \{1, \ldots, 2^j - 1\}$ we define*

$$d_j = d(\sigma_j) := \min_{1 \le k < l \le M_j} \left((\pm \sigma_j \, (n_l - n_k)) \bmod 2^j \right) \tag{12}$$

as the smallest (periodic) distance between two indices $\sigma_j \, n_l$ and $\sigma_j \, n_k$, and assume that $d_j > 0$. Then the condition number $\kappa_2(\mathbf{V}_{M_j',M_j}(\sigma_j))$ of the Vandermonde matrix $\mathbf{V}_{M_j',M_j}(\sigma_j) := \left(\omega_{2^j}^{\sigma_j \, p \, n_r}\right)_{p=0,r=1}^{M_j'-1,M_j}$ satisfies

$$\kappa_2(\mathbf{V}_{M_j',M_j}(\sigma_j))^2 \le \frac{M_j' + 2^j/d_j}{M_j' - 2^j/d_j}, \tag{13}$$

provided that $M_j' > \frac{2^j}{d_j}$.

However, this estimate cannot be used for square matrices, i.e., for $M_j = M_j'$, and it is not very sharp for large M_j. Indeed, if $d_j = 2^j/M_j$ which means that the values $\sigma_j \, n_k$ are equidistantly distributed on the periodic interval $[0, 2^j)$, then the square matrix $M_j^{-1/2} \, \mathbf{V}_{M_j,M_j}(\sigma_j)$ (with $M_j' = M_j$) is orthogonal with condition number 1 (see [2]), while the estimate (13) cannot be applied. On the other hand, if $M_j' = 2^j$, then we can simply conclude that $\mathbf{V}_{2^j,M_j}(\sigma_j)^* \mathbf{V}_{2^j,M_j}(\sigma_j) = 2^j \, \mathbf{I}_{M_j}$ such that we again achieve condition number 1, while (13) provides $\frac{2^j(1+1/d_j)}{2^j(1-1/d_j)}$, which again fails for the worst case $d_j = 1$ completely. Therefore, we apply another estimate, which is a simple consequence of the Theorem of Gershgorin, and can be iteratively computed during the iteration steps. It is based on the following Theorem.

Theorem 3.3. *Let $0 \le n_1 < n_2 < \ldots < n_{M_j} < 2^j$ be a given set of indices, and assume that $\sigma_j(n_k - n_\ell) \neq 0 \bmod 2^j$. Further, let for all $k = 1, \ldots, M_j$, $M_j \le M_j' \le 2^j$, and*

$$S_k(\sigma_j) := \sum_{\substack{\ell=1 \\ \ell \neq k}}^{M_j} \left| \frac{\sin\left(\frac{M_j' \pi}{2^j} \, \sigma_j \, (n_k - n_\ell)\right)}{\sin\left(\frac{\pi}{2^j} \, \sigma_j \, (n_k - n_\ell)\right)} \right|. \tag{14}$$

Then the condition number of the Vandermonde matrix $\mathbf{V}_{M_j',M_j}(\sigma_j)$ in (9) is bounded by

$$\kappa_2(\mathbf{V}_{M_j',M_j}(\sigma_j))^2 \le \frac{M_j' + \max_k S_k(\sigma_j)}{M_j' - \max_k S_k(\sigma_j)}. \tag{15}$$

Proof. Considering the matrix product $\mathbf{W} := \mathbf{V}_{M'_j,M_j}(\sigma_j)^* \mathbf{V}_{M'_j,M_j}(\sigma_j) \in \mathbb{C}^{M_j \times M_j}$, it follows for the components $w_{k,\ell}$ of $\mathbf{W}$ that

$$w_{k,k} = \sum_{p=0}^{M'_j-1} \omega_{2^j}^{p\,\sigma_j\,(n_k-n_k)} = M'_j, \qquad k = 0, \ldots, M_j - 1,$$

and for $k \neq \ell$ and $\sigma_j(n_k - n_\ell) \neq 0 \bmod 2^j$,

$$|w_{k,\ell}| = \Big| \sum_{p=0}^{M'_j-1} \omega_{2^j}^{p\,\sigma_j\,(n_k-n_\ell)} \Big| = \Big| \frac{1 - \omega_{2^j}^{M'_j\sigma_j(n_k-n_\ell)}}{1 - \omega_{2^j}^{\sigma_j(n_k-n_\ell)}} \Big| = \Big| \frac{\sin\left(\frac{M'_j\pi}{2^j}\,\sigma_j(n_k - n_\ell)\right)}{\sin\left(\frac{\pi}{2^j}\,\sigma_j(n_k - n_\ell)\right)} \Big|.$$

Thus, $S_k(\sigma_j)$ is the sum of the absolute values of all non-diagonal components in the k-th row of $\mathbf{W}$. The Theorem of Gershgorin implies now that the maximal eigenvalue of $\mathbf{W}$ is bounded from above by $M'_j + \max_k S_k(\sigma_j)$, and the smallest eigenvalue is bounded from below by $M'_j - \max_k S_k(\sigma_j)$. □

While the estimate (15) is quite simple to achieve, it is more accurate than (13). In particular, in the two special cases $M'_j = M_j$, $d_j = 2^j/M_j$ and $M'_j = 2^j$, $d_j = 1$, the estimate is sharp, and we obtain the true condition number 1.

For our computation of σ_j in Sect. 3.2, we will however simplify (14) and will consider instead an approximation of the upper bound of $S_k(\sigma_j)$,

$$\tilde{S}_k(\sigma_j) := \sum_{\substack{\ell=1 \\ \ell\neq k}}^{M_j} \left| \frac{1}{\sin(\frac{\pi}{2^j}\,\sigma_j\,(n_k^{(j)} - n_\ell^{(j)}))} \right| \geq S_k(\sigma_j) \tag{16}$$

which is not longer dependent on M'_j. Note that $\tilde{S}_k(\sigma_j) > 2^j$ can appear, if σ_j is not well chosen.

3.2. Efficient Computation of σ_j

For a given set of indices $0 \leq n_1 < n_2 < \ldots < n_M < 2^j$ we want to find a suitable $\sigma_j \in \{1, \ldots, 2^j - 1\}$ such that an approximation of $\max_k \tilde{S}_k(\sigma_j)$ is minimal. More precisely, as shown in Algorithm 3.4, we compare different possible parameters σ by comparing the sums of four terms in the sum (16), where the largest term is always included.

We surely could just consider all possible sets $\{\sigma n_1, \ldots, \sigma n_M\}$ for $\sigma \in \{1, \ldots, 2^j - 1\}$, compute the maximal sum $\tilde{S}_k^{(j)}(\sigma)$ and compare the results to find the optimal parameter $\tilde{\sigma}_j$. However, this procedure is too expensive. To achieve a sparse FFT algorithm with the desired overall complexity of $\mathcal{O}(M^2 \log N)$, we can spend at most $\mathcal{O}(M^2)$ operations to find a suitable parameter σ_j.

To avoid vanishing distances $\pm\sigma_j(n_k - n_\ell) \bmod 2^j = 0$ for all $n_k \neq n_\ell$, we will only consider odd integers $\sigma_j \geq 1$. We then have that 2^j and σ_j are co-prime such that for each odd σ_j we at least achieve that $\max_k \tilde{S}_k^{(j)}(\sigma_j)$ is

bounded. As our numerical tests show that prime numbers are good candidates for σ_j, we propose the following algorithm to determine σ_j.

Algorithm 3.4. (**Computation of** σ_j if $M_j > M_{j-1}$)
Input:
$N := 2^j$.
Index set $I^{(j)} = \{n_1, \ldots, n_{M_j}\}$.
Initialization:
Set $M_j := \#I^{(j)}$ and choose K with $K \leq M_j / \log_2 M_j$.
Let $\Sigma :=$ be set of K largest prime numbers smaller than $N/2$.
Loop:
For all $\sigma \in \Sigma$:

Compute the set $\sigma I^{(j)} := \{\sigma l \mod N : l \in I^{(j)}\}$.
Order the elements of $\sigma I^{(j)}$ by size to get $\tilde{n}_1 < \ldots < \tilde{n}_{M_j}$.
Compute the sequence of distances $\delta_k := \tilde{n}_k - \tilde{n}_{k-1}$, $k = 1, \ldots, M_j$, where $\tilde{n}_0 := \tilde{n}_{M_j} - N$.
Find the index of the smallest distance $\tilde{k} := \operatorname{argmin}_{k=1,\ldots,M_j} \delta_k$.
Compute

$$D_\sigma := \max\left\{ \left| \frac{1}{\sin(\frac{\delta_{\tilde{k}}\pi}{N})} \right| + \left| \frac{1}{\sin(\frac{\delta_{\tilde{k}-1}\pi}{N})} \right|, \left| \frac{1}{\sin(\frac{\delta_{\tilde{k}}\pi}{N})} \right| + \left| \frac{1}{\sin(\frac{\delta_{\tilde{k}+1}\pi}{N})} \right| \right\}$$

with the convention that $\delta_0 := \delta_{M_j}$ and $\delta_{M_j+1} := \delta_1$.

Completion:
Choose $\sigma \in \Sigma$ with minimal D_σ.
If there are several parameters σ achieving the same value D_σ,

choose the σ which minimizes the sum $\left| \sum_{k=1}^{M_j} \omega_N^{\sigma n_k} \right|$.

Output: $\sigma_j := \sigma$

The most expensive step in Algorithm 3.4 is the sorting of M_j elements in $\sigma I^{(j)}$, which can be done with $M_j \log M_j \leq M \log M$ operations. Since Σ contains $K < M_j / \log_2 M_j$ elements, the algorithm has a computational cost of $\mathcal{O}(M^2)$. Note, that we did not compute the complete sum $\tilde{S}_k(\sigma)$ for all choices of σ in Algorithm 3.4. Instead, for fixed σ, we search for an index $\tilde{k}$ that provides the smallest (periodic) distance $|\sigma(n_{\tilde{k}} - n_{\tilde{k}-1})| = \min_{k \neq \ell} |\sigma(n_k - n_\ell)|$. This index $\tilde{k}$ is a good candidate for $\operatorname{argmax}_k \tilde{S}_k(\sigma)$. We then only compute the sum of the largest component and the neighboring component of $\tilde{S}_{\tilde{k}}(\sigma)$ instead of the full sum, since $\tilde{S}_{\tilde{k}}(\sigma)$ is mainly governed by these components.

Remark 3.5. Using Theorem 3.2 it is of course also possible to determine σ_j by comparing only the minimal distance $d(\sigma)$ in (12) for all $\sigma \in \Sigma$, and to choose $\sigma \in \Sigma$ that maximizes this distance.

There are always enough odd prime numbers available in $[1, \frac{2^j}{2}]$, since $M_j^2 < 2^j$ (see, e.g., [18]).

3.3. Determination of M_j'

Further, we need to fix the number of needed rows $M_j' \geq M_j$ to ensure that the Vandermonde matrix $\mathbf{V}_{M_j',M_j}(\sigma_j)$ is well conditioned. Employing Theorem 3.3, we consider $M_j' = c\,M_j$ for a small set of integers c, e.g. $c \in \{1,2,5\}$. Starting with $c = 1$, we compute $\max_k S_k(\sigma_j)$ in (14) with $\mathcal{O}(M_j^2)$ operations, and check via (15) whether the condition number of $\mathbf{V}_{M_j',M_j}(\sigma_j)$ is acceptable. If it is too large, we enlarge c.

Remark 3.6. We can also use the estimates in Theorem 3.2 for determining M_j'. In this case, we simply fix M_j' such that

$$\left(\frac{M_j' + 2^j/d_j}{M_j' - 2^j/d_j}\right)^{1/2} < C$$

where C is a pre-determined bound for the condition number of $\mathbf{V}_{M_j',M_j}(\sigma_j)$. However, this estimate usually leads to a strong overestimation of M_j'.

In our numerical experiments we achieved good results with the simple bound

$$M_j' = c\,M_j \qquad \text{with} \qquad c := \min\left\{\left\lfloor \frac{2^j/M_j}{d_j}\right\rfloor, c_{\max}\right\}, \tag{17}$$

where $c_{\max}$ is usually an integer with $c_{\max} \leq 5$ (see Sect. 5). This setting can also be understood as a compromise for having a good condition number of the matrix $\mathbf{A}^{(j)}$ in the system (11) on the one hand and the computational cost to solve the linear system on the other hand. Using for example the QR decomposition algorithm in [7] for rectangular Vandermonde matrices of size $cM_j \times M_j$, we obtain a complexity of $(5c + \frac{7}{2})M_j^2 + \mathcal{O}(cM_j)$.

3.4. Choice of $\mathbf{A}^{(j)}$ if $M_{j-1} = M_j$

If $M_j = M_{j-1}$, we apply the following Lemma which is an extension of Theorem 4.2 in [16].

Lemma 3.7. *Let σ_{j-1} and M_{j-1}' be the parameters used in the* Algorithm 3.1 *to determine $\mathbf{V}_{M_{j-1}',M_{j-1}}(\sigma_{j-1})$ in the iteration step $j-1$, where $0 < n_1^{(j-1)} < \ldots < n_{M_{j-1}}^{(j-1)} < 2^{j-1}$ are the support indices of $\boldsymbol{x}^{(j-1)}$. Further, assume that we have found $\mathbf{x}^{(j)}$ with $M_j = M_{j-1}$, and support indices $0 < n_1^{(j)} < \ldots < n_{M_j}^{(j)} < 2^j$. Then we can simply choose $\sigma_j := 2\sigma_{j-1}$ and $M_j' := M_{j-1}'$ to achieve a Vandermonde matrix $\mathbf{V}_{M_j',M_j}(\sigma_j)$ for iteration step j of* Algorithm 3.1. *With this choice, $\mathbf{V}_{M_j',M_j}(\sigma_j)$ coincides with $\mathbf{V}_{M_{j-1}',M_{j-1}}(\sigma_{j-1})$ up to possible permutation of columns. In particular, we have*

$$\kappa_2(\mathbf{V}_{M_j',M_j}(\sigma_j)) = \kappa_2(\mathbf{V}_{M_{j-1}',M_{j-1}}(\sigma_{j-1})).$$

Proof. If $M_j = M_{j-1}$, then it follows that $n_r^{(j)} \in \{n_r^{(j-1)}, n_r^{(j-1)} + 2^{j-1}\}$ for all $r = 1, \ldots, M_{j-1}$. With $\sigma_j = 2\sigma_{j-1}$ we obtain

$$\sigma_j \, n_r^{(j)} \bmod 2^j = 2\sigma_{j-1} n_r^{(j)} \bmod 2^j = 2\sigma_{j-1} n_r^{(j-1)} \bmod 2^j.$$

Thus, for $p = 1, \ldots, M_j'$ (with $M_j' = M_{j-1}'$),

$$\omega_{2^j}^{\sigma_j (p-1) n_r^{(j)}} = \omega_{2^j}^{2\sigma_{j-1}(p-1) n_r^{(j)}} = \omega_{2^j}^{2\sigma_{j-1}(p-1) n_r^{(j-1)}} = \omega_{2^{j-1}}^{\sigma_{j-1}(p-1) n_r^{(j-1)}}.$$

Hence, $\mathbf{V}_{M_{j-1}', M_{j-1}}(\sigma_{j-1})$ and $\mathbf{V}_{M_j', M_j}(\sigma_j)$ have the same columns, and may differ only due to a different ordering of columns. In other words, there is an $M_j \times M_j$ permutation matrix $\mathbf{P}_{M_j}$, such that $\mathbf{V}_{M_j', M_j}(\sigma_j) = \mathbf{V}_{M_{j-1}', M_{j-1}}(\sigma_{j-1})\, \mathbf{P}_{M_j}$. In particular, the two matrices have the same condition number. □

This observation implies that there will be no extra effort to compute the matrix $\mathbf{A}^{(j)}$ at all iteration steps j, where the sparsity M_j has not changed compared to M_{j-1}.

4. The Direct Sparse FFT Algorithm

We consider now the direct sparse FFT problem stated in (b) in Sect. 1. For given $\mathbf{x} \in \mathbb{C}^N$, we want to determine $\mathbf{y} := \hat{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$, assuming that $\mathbf{y}$ possesses unknown sparsity M. We will show that our Algorithm 3.1 can be transferred to this problem.

First, we observe that the Fourier matrix satisfies the property

$$\mathbf{F}_N^{-1} = \frac{1}{N} \overline{\mathbf{F}}_N = \frac{1}{N} \mathbf{J}_N' \, \mathbf{F}_N$$

(see Equation (3.34) in [15]), where $\mathbf{J}_N' := (\delta_{(j+k) \bmod N})_{j,k=0}^{N-1}$ is the so-called flip matrix with $(\mathbf{J}_N')^{-1} = \mathbf{J}_N'$. Here, δ_j denotes the Kronecker symbol, i.e., $\delta_j = 0$ for $j \neq 0$ and $\delta_j = 1$ for $j = 0$. Thus, the relation $\mathbf{x} = \mathbf{F}_N^{-1} \mathbf{y}$ is equivalent to

$$\mathbf{w} := N \, \mathbf{J}_N' \mathbf{x} = \mathbf{F}_N \mathbf{y}.$$

In other words, if we replace the given vector $\mathbf{x}$ by $\mathbf{w}$ in Algorithm 3.1, then $\mathbf{w}$ is the given Fourier transform of the desired vector $\mathbf{y}$, and we can apply Algorithm 3.1 directly to compute $\mathbf{y}$.

5. Numerical Experiments

First, we present some numerical experiments showing that the algorithm in [16] for sparsity $M > 20$ is no longer reliable. We generate randomly chosen sets of support indices $I_M \subset \{0, \ldots, 2^{15} - 1\}$ with different cardinalities $M = 20, 30, \ldots, 100$, and randomly choose values x_k for $k \in I_M$ in double precision arithmetics. Then we apply our Algorithm 3.1, where access to the Fourier transform of $\mathbf{x} \in \mathbb{C}^{2^J}$ is provided. While σ_j is optimally chosen as a

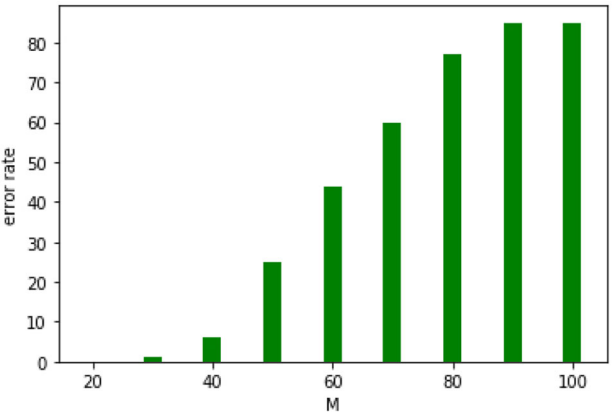

FIGURE 1. Error rate in percentage for the computed set of indices for $c_{max} = 1$ and $J = 15$

prime number according to Algorithm 4.5 in [16], we only consider square Vandermonde matrices (as in [16]), i.e., we set $c_{\max} = 1$. We compare the output index set I_{out} with the generated set I_M of indices and count the failures of 100 tests for each M. The results are presented in Fig. 1. The test shows that the algorithm starts to be unreliable for sparsity $M > 20$.

We now run the test with the same input data as above, but used the criteria in (17) with $c_{\max} = 2$. For any $M = 20, 30, \ldots, 100$, no failures occur for the computed set of indices I_{out}, i.e., we always find $I_M = I_{out}$. Even if we run the tests for $M = 200$, the error rate is still zero.

To understand this strong effect when the number of rows of the Vandermonde matrix is enlarged, we analyze the condition numbers of the Vandermonde matrices occurring in the computations for different values $c_{\max}$. We generate sets I_M of indices and randomly choose the amplitudes of components of $\mathbf{x}$ with support I_M. For Algorithm 3.1, we provide access to the Fourier transformed vector $\hat{\mathbf{x}}$ as an input as before for the tuples (J, M) with $J = 15, 16, \ldots, 22$, and $M = 20, 30, 40, 50$. In this experiment, we vary $c_{\max} \in \{1, 2, 5\}$. In each test we compute the average over all condition numbers of the used Vandermonde matrices and repeat this 20 times for each tuple (J, M). Finally, we take the mean of all the 20 averages, and obtain the results given in the Tables 1 and 2. The results in Table 1 show that a suitable choice of the parameter σ_j, as applied in [16], is not sufficient to ensure moderate condition numbers of the Vandermonde matrices involved in the sparse FFT algorithm for $M \geq 20$.

In Table 2, we provide some further condition numbers for larger numbers M of significant vector entries up to $M = 200$ and $N = 2^{15}, \ldots, 2^{22}$. The experiments show that $c_{\max} = 2$, i.e., doubling the number of rows in the

TABLE 1. Average condition number for $c_{max} = 1$ after 20 tests

$c_{max} = 1$				
J	$M = 20$	$M = 30$	$M = 40$	$M = 50$
15	45587	8959761	826581656	813444189055
16	150932	3541859	41764903	535590260990711
17	502398	1044096	2914884097	719367030204.95
18	103809	674572	1080286999258065	1016723525704275
19	10491	4832052	111942753	12377927191183
20	41983	711412	918528399	93462229700
21	61938	3502253	567002193	143672696329261
22	388062	37168024	259341688	28197228

TABLE 2. Average condition number for $c_{max} = 2$ (left) and $c_{max} = 5$ (right) after 20 tests

$c_{max} = 2$				$c_{max} = 5$			
J	$M = 20$	$M = 100$	$M = 200$	J	$M = 20$	$M = 100$	$M = 200$
15	4.31	128.83	12623.74	15	1.33	4.52	16.42
16	5.57	415.11	167096.38	16	1.36	8.01	38.64
17	7.94	74.23	32290.12	17	1.43	4.97	37.78
18	40.52	591.17	5901.65	18	1.79	8.59	19.76
19	14.76	732.74	154631.91	19	1.44	10.13	38.64
20	14.46	231.35	27979.52	20	1.39	9.56	28.29
21	17.51	259.04	14604.35	21	1.75	7.25	22.41
22	12.86	360.91	17897.02	22	1.63	6.04	23.12

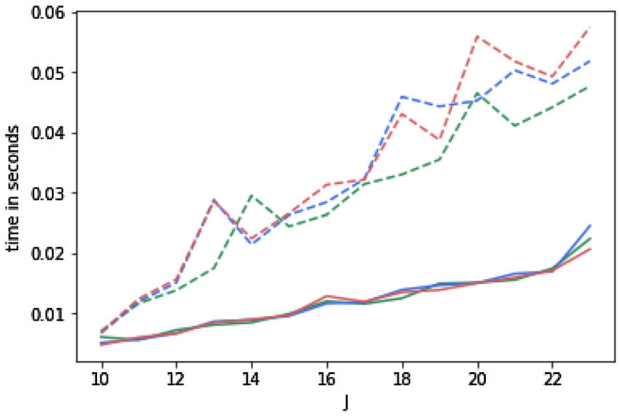

FIGURE 2. Runtime comparison of the Algorithmus 3.1 for $c_{max} = 1$ (green), $c_{max} = 5$ (blue), $c_{max} = 20$ (red) for $M = 10$ (solid line) and $M = 30$ (dashed line) for length $N = 2^J$ with $J = 10, \ldots, 24$. (Color figure online)

matrix $\mathbf{A}^{(j)}$, is usually sufficient for $M \leq 100$. For $M > 100$, we need to take a larger $c_{\max}$.

Now, we investigate how the runtime of the Algorithm depends on $c_{\max}$. In Fig. 2 we present the average runtime for 20 tests with randomly chosen sparse vectors with sparsities $M = 10, 30$ and for $c_{\max} = 1, c_{\max} = 5, c_{\max} = 20$. As we see in Fig. 2, our modifications have only a very small effect on the runtime. Finally, in Fig. 3 we compare the runtime of the Python implemented FFT `numpy.fft.fft` of length 2^J with our algorithm for $c_{\max} = 20$. We can see, that our current Python implementation starts to be faster than the FFT for $M \leq 30$ and $N \geq 2^{20}$. It is available under the link "software" on our homepage http://na.math.uni-goettingen.de.

6. Conclusions

In this paper, we have presented a modification of the sparse FFT algorithm in [16], which is based on the assumption that the wanted vector $\mathbf{x} \in \mathbb{C}^N$ with $N = 2^J$ is M-sparse, and that the components of the discrete Fourier transform $\hat{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$ are available. Our proposed algorithm has the complexity $\mathcal{O}(M^2 \log N)$ and is sublinear in N for small M. As in [16], the reconstruction of $\mathbf{x}$ is based on an iterative reconstruction of 2^j-periodizations of $\mathbf{x}$ for $j = 0, \ldots, J$. At each iteration step, one needs to solve an equation system of size $\mathcal{O}(M)$, where the coefficient matrices are governed by Vandermonde matrices which are submatrices of the Fourier matrix $\mathbf{F}_{2^j}$. Differently from

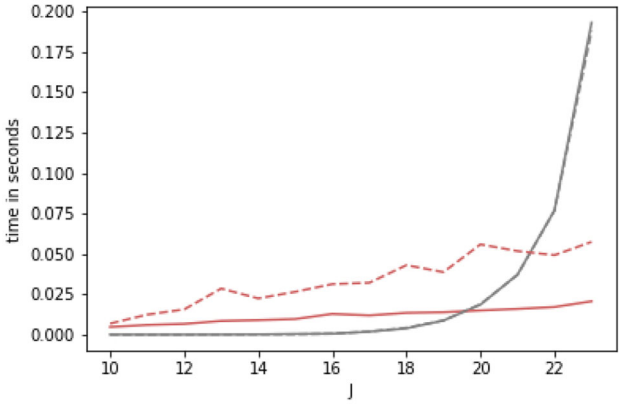

FIGURE 3. Runtime comparison of Algorithmus 3.1 and $c_{\max} = 20$ (red) and the FFT (gray) for $M = 10$ (solid line) and $M = 30$ (dashed line) for length $N = 2^J$ with $J = 10, \ldots, 24$. (Color figure online)

[16], we have considered rectangular Vandermonde matrices, and we have presented efficient methods to determine these matrices in dependence of two parameters, which both have a huge impact on the condition number. The first parameter σ_j changes the nodes $\omega_{2^j}^{n_\ell}$, $\ell = 1, \ldots, M_j$ determining the Vandermonde matrix to $\omega_{2^j}^{\sigma_j n_\ell}$. Here $M_j \leq M$ denotes the found sparsity of $\mathbf{x}^{(j)}$. The second parameter $M_j' \geq M_j$ denotes the number of rows in the Vandermonde matrix. One ingredient to determine suitable parameters σ_j and M_j' is the new estimate for the condition number of the occurring Vandermonde matrices in Theorem 3.3. As shown in the numerical experiments, the presented modification of the sparse FFT algorithm makes it applicable also for larger sparsity values M while the original algorithm in [16] started to be unreliable already for $M > 20$.

Acknowledgements

The authors like to thank the reviewers for very exact reading of the manuscript and many constructive remarks for its improvement. The authors gratefully acknowledge the support by the German Research Foundation in the framework of the RTG 2088.

Funding Open Access funding enabled and organized by Projekt DEAL. The authors gratefully acknowledge the support by the German Research Foundation in the framework of the RTG 2088.

Code Availability A Python implementation of the new algorithm is available under the link "software" on our homepage http://na.math.uni-goettingen.de.

Compliance with Ethical Standards
Conflict of interest The authors declare that they have no conflict of interest.

Human and Animal Rights This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] Akavia, A.: Deterministic sparse Fourier approximation via approximating arithmetic progressions. IEEE Trans. Inf. Theory **60**(3), 1733–1741 (2014). https://doi.org/10.1109/TIT.2013.2290027

[2] Berman, L., Feuer, A.: On perfect conditioning of Vandermonde matrices on the unit circle. Electron. J. Linear Algebra **16**(1), 157–161 (2007). https://doi.org/10.13001/1081-3810.1190

[3] Bittens, S.: Sparse FFT for functions with short frequency support. Dolomites Res. Not. Approx. **10**, 43–55 (2017). https://doi.org/10.14658/pupj-drna-2017-Special_Issue-7

[4] Bittens, S., Plonka, G.: Sparse fast DCT for vectors with one-block support. Numer. Algorithms **82**(2), 663–697 (2019). https://doi.org/10.1007/s11075-018-0620-1

[5] Bittens, S., Plonka, G.: Real sparse fast DCT for vectors with short support. Linear Algebra Appl. **582**, 359–390 (2019). https://doi.org/10.1016/j.laa.2019.08.006

[6] Christlieb, A., Lawlor, D., Yang, W.: A multiscale sub-linear time Fourier algorithm for noisy data. Appl. Comput. Harmon. Anal. **40**(3), 553–574 (2016). https://doi.org/10.1016/j.acha.2015.04.002

[7] Demeure, C.J.: Fast QR factorization of Vandermonde matrices. Linear Algebra Appl. **122–124**, 165–194 (1989). https://doi.org/10.1016/0024-3795(89)90652-6

[8] Heider, S., Kunis, S., Potts, D., Veit, M.: A sparse Prony FFT. In: 10th International Conference on Sampling Theory and Applications (SAMPTA), pp. 572–575. Zenodo, (2013). https://doi.org/10.5281/zenodo.54481

[9] Iwen, M.A.: Combinatorial sublinear-time Fourier algorithms. Found. Comput. Math. **10**, 303–338 (2010). https://doi.org/10.1007/s10208-009-9057-1

[10] Iwen, M.A.: Improved approximation guarantees for sublinear-time Fourier algorithms. Appl. Comput. Harmon. Anal. **34**(1), 57–82 (2013). https://doi.org/10.1016/j.acha.2012.03.007

[11] Merhi, S., Zhang, R., Iwen, M.A., Christlieb, A.: A new class of fully discrete sparse Fourier transforms: Faster stable implementations with guarantees. J. Fourier Anal. Appl. **25**, 751–784 (2019). https://doi.org/10.1007/s00041-018-9616-4

[12] Moitra, A.: Super-resolution, extremal functions and the condition number of Vandermonde matrices. In: STOC '15: Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing, pp. 821–830, (2015). https://doi.org/10.1145/2746539.2746561

[13] Plonka, G., Wannenwetsch, K.: A deterministic sparse FFT algorithm for vectors with small support. Numer. Algorithms **71**(4), 889–905 (2016). https://doi.org/10.1007/s11075-015-0028-0

[14] Plonka, G., Wannenwetsch, K.: A sparse fast Fourier algorithm for real nonnegative vectors. J. Comput. Appl. Math. **321**, 532–539 (2017). https://doi.org/10.1016/j.cam.2017.03.019

[15] Plonka, G., Potts, D., Steidl, G., Tasche, M.: Numerical Fourier Analysis. Birkhäuser, Basel (2018). https://doi.org/10.1007/978-3-030-04306-3

[16] Plonka, G., Wannenwetsch, K., Cuyt, A., Lee, W.-S.: Deterministic sparse FFT for M-sparse vectors. Numer, Algorithms **78**, 133–159 (2018). https://doi.org/10.1007/s11075-017-0370-5

[17] Potts, D., Tasche, M., Volkmer, T.: Efficient spectral estimation by MUSIC and ESPRIT with application to sparse FFT. Front. Appl. Math. Stat. **2**, 1 (2016). https://doi.org/10.3389/fams.2016.00001

[18] Rosser, J.B., Schoenfeld, L.: Approximate formulas for some functions of prime numbers. Illinois J. Math. **6**(1), 64–94 (1962). https://doi.org/10.1215/ijm/1255631807

[19] Segal, B., Iwen, M.A.: Improved sparse Fourier approximation results: faster implementations and stronger guarantees. Numer. Algorithms **63**, 239–263 (2013). https://doi.org/10.1007/s11075-012-9621-7

Gerlind Plonka and Therese von Wulffen
Institute for Numerical and Applied Mathematics
University of Göttingen
Lotzestraße 16-18
37083 Göttingen
Germany
e-mail: plonka@math.uni-goettingen.de;
therese.vonwulffen@stud.uni-goettingen.de

Received: April 21, 2020.
Accepted: December 12, 2020.