Journal of Computational and Applied Mathematics 321 (2017) 532–539

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

A sparse fast Fourier algorithm for real non-negative vectors

Gerlind Plonka *, Katrin Wannenwetsch

University of Göttingen, Institute for Numerical and Applied Mathematics, Lotzestr. 16-18, 37083 Göttingen, Germany

ARTICLE INFO

Article history:
Received 17 February 2016
Received in revised form 16 January 2017

MSC:
65T50
42A38

Keywords:
Discrete Fourier transform
Sparse Fourier reconstruction
Sublinear sparse FFT

ABSTRACT

In this paper we propose a new fast Fourier transform to recover a real non-negative signal $\mathbf{x} \in \mathbb{R}_+^N$ from its discrete Fourier transform $\widehat{\mathbf{x}} = \mathbf{F}_N\mathbf{x} \in \mathbb{C}^N$. If the signal $\mathbf{x}$ appears to have a short support, i.e., vanishes outside a support interval of length $m < N$, then the algorithm has an arithmetical complexity of only $\mathcal{O}(m \log m \log(N/m))$ and requires $\mathcal{O}(m \log(N/m))$ Fourier samples for this computation. In contrast to other approaches there is no a priori knowledge needed about sparsity or support bounds for the vector $\mathbf{x}$. The algorithm automatically recognizes and exploits a possible short support of the vector and falls back to a usual radix-2 FFT algorithm if $\mathbf{x}$ has (almost) full support. The numerical stability of the proposed algorithm is shown by numerical examples.

1. Introduction

Algorithms for fast Fourier transform play a fundamental role in many areas in numerical analysis, particularly in signal and image processing. It is well-known that FFT algorithms for general vectors $\mathbf{x} \in \mathbb{C}^N$ require $\mathcal{O}(N \log N)$ arithmetical operations and that this qualitative bound cannot be improved, see [1]. However, if the discrete Fourier transform is applied to recover vectors with special properties, there is the hope for even faster algorithms.

In recent years, there has been some effort to derive new so-called "sparse FFT" algorithms that exploit the a priori knowledge that the vector to be recovered is sparse or has only a small amount of significant frequencies. Often, further assumptions on the vector appear, as e.g. that the components to be recovered are from a certain quantized range consisting of a finite set of real entries, see e.g. [2]. Most of the proposed algorithms are based on randomization [3–5] and achieve e.g. a complexity of $\mathcal{O}(k \log N)$ [2] for k-sparse signals, or even $\mathcal{O}(k \log k)$, see e.g. [5]. An obvious drawback of randomized approaches is that the algorithms do not always achieve the correct result (or an approximation of it) but only with a certain probability. Another problem is that there exists no sublinear algorithm to check the correctness of the result.

Completely deterministic sparse FFT algorithms for k-sparse signals have been proposed e.g. in [6–10]. The underlying ideas are based on combinatorial approaches employing FFTs of different prime length and the Chinese remainder theorem. These algorithms usually have polynomial costs in $\log N$ and k and only pay off for very large N and strong sparsity.

Sparse FFT algorithms based on Prony's method, see [11–13], for k-sparse signals are usually based on singular value decompositions of size k with a complexity of $\mathcal{O}(k^3)$ and are therefore only efficient for small k. This complexity can be reduced in special cases using a splitting approach [13].

In a recent paper [14], the authors proposed a deterministic sparse FFT algorithm for vectors with short support that is based on usual FFT and is numerically stable with a complexity of $\mathcal{O}(m \log N)$ operations, where here m denotes the support length of the signal.

* Corresponding author.
E-mail addresses: plonka@math.uni-goettingen.de (G. Plonka), k.wannenwetsch@math.uni-goettingen.de (K. Wannenwetsch).

http://dx.doi.org/10.1016/j.cam.2017.03.019
0377-0427/

All algorithms mentioned above require a priori information as e.g. the exact sparsity or the support length of the vector to be recovered, or at least a suitable upper bound for it. They are just not applicable without this information or do not always achieve the correct recovery result by using only a guess for this bound.

However, in practice, while certain structures of the vector (as short support or sparsity) often appear, we do not always have the a priori knowledge on a good upper bound. Therefore it is of high interest to develop deterministic FFT algorithms that are able to automatically recognize certain structures of the vector during the algorithm and to exploit it suitably to reduce complexity and run time.

In this paper, we propose for the first time an algorithm that meets this requirement in the way that no a priori knowledge about the support length of the vector to be recovered is needed beforehand. We present a new deterministic algorithm to recover a real non-negative vector $\mathbf{x} \in \mathbb{R}_+^N$ from its discrete Fourier transform $\widehat{\mathbf{x}}$. If $\mathbf{x}$ has a support with support length m being significantly smaller than N, then the algorithm automatically recognizes this structure and provides the resulting vector with an arithmetical complexity of $\mathcal{O}(m \log m \log(N/m))$ requiring at most $\mathcal{O}(m \log(N/m))$ Fourier data. The idea of the algorithm is based on divide-and-conquer techniques.

Direct applications for the reconstruction of sparse vectors from Fourier data, both with known or unknown support, appear for instance solving phase retrieval problems, where data have to be reconstructed from Fourier intensities. In this case, short support and positivity of the resulting vectors or images are frequently used preconditions in iterated projection algorithms, see e.g. [15].

This paper is structured as follows: After fixing the notations, we introduce the new fast algorithm in Section 2, together with detailed explanations of its structure and complexity. In Section 3, we apply the algorithm to noisy Fourier data and present some numerical results showing the numerical stability of the proposed algorithm in practice.

1.1. Notations

Let $\mathbf{x} \in \mathbb{R}_+^N$ with $N = 2^J$ for some $J > 0$ be a real vector with non-negative entries. We denote the discrete Fourier transform $\widehat{\mathbf{x}}$ of $\mathbf{x}$ by

$$\widehat{\mathbf{x}} = \mathbf{F}_N \mathbf{x},$$

where $\mathbf{F}_N := \left(\omega_N^{jk}\right)_{j,k=0}^{N-1} \in \mathbb{C}^{N\times N}$ is the Fourier matrix and $\omega_N := \mathrm{e}^{\frac{-2\pi \mathrm{i}}{N}}$.

The *support length* $m = |\operatorname{supp} \mathbf{x}|$ of $\mathbf{x} \in \mathbb{R}_+^N$ is defined as the *minimal positive integer* such that $x_k = 0$ for $k \notin I := \{(\mu + \ell) \bmod N \mid \ell = 0, \dots, m-1\}$. We call this index set I the *support index interval of* $\mathbf{x}$. The *first support index of* $\mathbf{x}$ is denoted by μ. Note that the first support index of $\mathbf{x}$ needs not to be the index of the first nonzero entry in $\mathbf{x}$. Considering for example the vector $\mathbf{x} := (13, 21, 0, 0, 0, 10, 31, 0) \in \mathbb{R}_+^8$, we obtain a support length $m = 5$, the support index interval $I = \{5, 6, 7, 0, 1\}$ with the corresponding signal values $(10, 31, 0, 13, 21)$, and $\mu = 5$, i.e., the support starts with $x_5 = 10$. Note further that the support index interval I may contain indices corresponding to zero components of $\mathbf{x}$, as e.g. the index 7 in the small example above. Therefore the support length m is an upper bound of the sparsity, the number of nonzero entries of $\mathbf{x}$. But in any case, it holds that $x_\mu > 0$ and $x_{(\mu+m-1) \bmod N} > 0$, i.e., the first and the last entry of the support of $\mathbf{x}$ are positive.

The support length of a vector $\mathbf{x}$ is always uniquely defined. However, the support index interval and the first support index μ are not necessarily unique. Consider e.g. the vector $\mathbf{x} \in \mathbb{R}_+^N$ with $x_0 = x_{N/2} = 1$ and $x_\ell = 0$ for $\ell \in \{0, \dots, N-1\} \setminus \{0, N/2\}$. For this vector it is possible to choose either $\mu = 0$ or $\mu = N/2$ whereas the support length is $N/2 + 1$ in both cases.

The *periodized vectors* $\mathbf{x}^{(j)} \in \mathbb{R}_+^{2^j}$ of $\mathbf{x}$ are defined by

$$\mathbf{x}^{(j)} = (x_k^{(j)})_{k=0}^{2^j-1} := \left(\sum_{\ell=0}^{2^{J-j}-1} x_{k+2^j\ell} \right)_{k=0}^{2^j-1} \tag{1.1}$$

for $j = 0, \dots, J$. In particular, $\mathbf{x}^{(0)} = \sum_{k=0}^{N-1} x_k$ is the sum of all components of $\mathbf{x}$, $\mathbf{x}^{(1)} = \left(\sum_{k=0}^{N/2-1} x_{2k}, \sum_{k=0}^{N/2-1} x_{2k+1}\right)^T$ and $\mathbf{x}^{(J)} = \mathbf{x}$.

We recall from [14], that the components of the discrete Fourier transforms $\widehat{\mathbf{x}}^{(j)}$ need not to be computed but are already given as a subset of the set of components of $\widehat{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$.

Lemma 1.1. *For the vectors $\mathbf{x}^{(j)} \in \mathbb{R}_+^{2^j}$, $j = 0, \dots, J$, in (1.1), the discrete Fourier transform is given by*

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

where $\widehat{\mathbf{x}} = (\widehat{\mathbf{x}}_k)_{k=0}^{N-1} = \mathbf{F}_N \mathbf{x}$ is the Fourier transform of $\mathbf{x} \in \mathbb{R}_+^N$.