Sparse Fast Trigonometric Transforms

Dissertation

zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
„Doctor rerum naturalium“
der Georg-August-Universität Göttingen

im Promotionsprogramm *Mathematical Sciences*
der Georg-August University School of Science (GAUSS)

vorgelegt von
Sina Vanessa Bittens
aus Gehrden

Göttingen, 2019

Betreuungsausschuss:

Prof. Dr. Gerlind Plonka-Hoch
Institut für Numerische und Angewandte Mathematik
Georg-August-Universität Göttingen

Prof. Dr. Russell Luke
Institut für Numerische und Angewandte Mathematik
Georg-August-Universität Göttingen

Mitglieder der Prüfungskommission:

Referentin:

Prof. Dr. Gerlind Plonka-Hoch
Institut für Numerische und Angewandte Mathematik
Georg-August-Universität Göttingen

Korreferent:

Prof. Dr. Felix Krahmer
Fakultät für Mathematik
Technische Universität München

2. Korreferent:

Prof. Dr. Daniel Potts
Fakultät für Mathematik
Technische Universität Chemnitz

Weitere Mitglieder der Prüfungskommission:

Prof. Dr. Thorsten Hohage
Institut für Numerische und Angewandte Mathematik
Georg-August-Universität Göttingen

Jun.-Prof. Dr. Matthew Tam
Institut für Numerische und Angewandte Mathematik
Georg-August-Universität Göttingen

Prof. Dr. Jörg Brüdern
Mathematisches Institut
Georg-August-Universität Göttingen

Tag der mündlichen Prüfung: 13.06.2019

Acknowledgments

This thesis is the result of the work I did in the last three years, during which I had the opportunity to work at the Institute for Numerical and Applied Mathematics at the University of Göttingen.

First of all I owe my deepest thanks to my adviser Gerlind Plonka-Hoch for introducing me to the world of signal and image processing. I could not imagine a more interesting and inspiring topic of research. Furthermore, I would like to thank her, and my co-adviser Russell Luke, for their constantly open doors and their continued support during all stages of my PhD. My sincere thanks also go to the co-referees of my thesis, Daniel Potts and Felix Krahmer. I am particularly grateful to Felix Krahmer for inviting me to Munich in May 2016 to meet Mark Iwen. The collaboration initiated by that visit and finalized during a two-week research stay at Michigan State University in February 2017 resulted in Chapter 3 of my thesis. For the invitation to MSU and also for his support during our collaboration I would like to thank Mark Iwen as well.

Furthermore, I gratefully acknowledge the financial support of the German Research Foundation (DFG) in the framework of the Research Training Group (RTG) 2088 "Discovering structure in complex data: Statistics meets Optimization and Inverse Problems". For the whole duration of my PhD I was either a member or an associated member of this RTG, enabling me to attend many conferences and workshops for broadening my scientific horizon, but also giving me the opportunity to improve my soft skills.

I am particularly indebted to my wonderful colleagues, former colleagues and friends from the "Mathematical Signal and Image Processing" group for the incredible working environment and their emotional support. I would especially like to thank Hanna Knirsch, Inge Keller and Markus Petz for proofreading this thesis.

I am eternally grateful to my family and all my friends for their unconditional support, advice and patience with me during the past three years. This thesis would not exist without you. Finally, I want to express my deepest gratitude to Daniel H.

Contents

Notation

- $\mathbb{N} \coloneqq \{1, 2, 3, \dots\}$ and $\mathbb{N}_0 \coloneqq \mathbb{N} \cup \{0\}$
- p_l: lth prime number
- $C_{2\pi}^k \coloneqq \{f\colon [0, 2\pi] \to \mathbb{C} : f \text{ is } 2\pi\text{-periodic and } k \text{ times continuously differentiable}\}$
- $L_{2\pi}^1 \coloneqq \left\{f\colon [0, 2\pi] \to \mathbb{C} : f \text{ is } 2\pi\text{-periodic with } \int_0^{2\pi} |f(x)|\mathrm{d}x < \infty\right\}$: Banach space with norm $\|f\|_1 \coloneqq \frac{1}{2\pi}\int_0^{2\pi} |f(x)|\mathrm{d}x$
- $L_{2\pi}^2 \coloneqq \left\{f\colon [0, 2\pi] \to \mathbb{C} : f \text{ is } 2\pi\text{-periodic with } \int_0^{2\pi} |f(x)|^2\mathrm{d}x < \infty\right\}$: Hilbert space with inner product $(f, g) \coloneqq \frac{1}{2\pi}\int_0^{2\pi} f(x)\overline{g(x)}\mathrm{d}x$ and norm $\|f\|_2 \coloneqq (f, f)^{\frac{1}{2}}$
- $\ell^1 \coloneqq \left\{\mathbf{c} = (c_k)_{k\in\mathbb{Z}} : \sum_{k\in\mathbb{Z}} |c_k| < \infty\right\}$: Banach space with norm $\|\mathbf{c}\|_1 \coloneqq \sum_{k\in\mathbb{Z}} |c_k|$
- $\ell^2 \coloneqq \left\{\mathbf{c} = (c_k)_{k\in\mathbb{Z}} : \sum_{k\in\mathbb{Z}} |c_k|^2 < \infty\right\}$: Hilbert space with inner product $(\mathbf{c}, \mathbf{d}) \coloneqq \sum_{k\in\mathbb{Z}} c_k\overline{d_k}$ and norm $\|\mathbf{c}\|_2 \coloneqq \left(\sum_{k\in\mathbb{Z}} |c_k|^2\right)^{\frac{1}{2}}$
- $\mathbb{Z}[x]$: the ring of polynomials with integer coefficients
- $\mathbf{0}_n$: $(0, \dots, 0)^T \in \mathbb{R}^n$
- $\mathbf{I}_n$: $n \times n$ identity matrix
- $\mathbf{J}_n$: $n \times n$ counter identity matrix
- $\mathbf{P}_n \coloneqq \begin{pmatrix} (\delta_{2k,l})_{k,l=0}^{\frac{n}{2}-1,n-1} \\ (\delta_{2k+1,l})_{k,l=0}^{\frac{n}{2}-1,n-1} \end{pmatrix}$: $n \times n$ even-odd permutation matrix for even $n \in \mathbb{N}$.
- $\delta_{k,l}$: Kronecker delta with $\delta_{k,l} \coloneqq \begin{cases} 1 & \text{if } k = l, \\ 0 & \text{otherwise,} \end{cases}$ for $k, l \in \mathbb{Z}$.
- $\delta_{k,l}^{(n)}$: n-periodic Kronecker delta with $\delta_{k,l}^{(n)} \coloneqq \begin{cases} 1 & \text{if } k \equiv l \bmod n, \\ 0 & \text{otherwise,} \end{cases}$ for $k, l \in \mathbb{Z}$.
- $\mathbf{F}_N$: Nth Fourier matrix.
- $\widehat{\mathbf{y}} \coloneqq \mathbf{F}_N\mathbf{y}$: DFT of $\mathbf{y} \in \mathbb{C}^N$.
- $\mathbf{C}_N^{\mathrm{II}}, \mathbf{C}_N^{\mathrm{III}}, \mathbf{C}_N^{\mathrm{IV}}$: $N \times N$ cosine matrices of types II, III and IV.
- $\mathbf{S}_N^{\mathrm{IV}}$: $N \times N$ sine matrix of type IV.
- $\mathbf{x}^{\widehat{\mathrm{II}}} \coloneqq \mathbf{C}_N^{\mathrm{II}}\mathbf{x}$, $\mathbf{x}^{\widehat{\mathrm{IV}}} \coloneqq \mathbf{C}_N^{\mathrm{IV}}\mathbf{x}$: DCT-II, DCT-IV of $\mathbf{x} \in \mathbb{R}^N$.
- $I_{a,b}$ for $a \leq b$: interval with $I_{a,b} = \{a, a+1, \dots, b\}$.

- $I_{a,b}^{(j)}$: periodized interval with $I_{a,b}^{(j)} = \{a \mod 2^j, \ldots, b \mod 2^j\} \subseteq I_{0,2^j-1}$.
- $\mathbf{x}_{(0)} \coloneqq (x_k)_{k=0}^{\frac{n}{2}-1}$, $\mathbf{x}_{(1)} \coloneqq (x_k)_{k=\frac{n}{2}}^{n-1}$: first and second half of $\mathbf{x} \in \mathbb{R}^n$, n even.
- $\mathbf{y}^{(j)} \coloneqq \mathbf{y}_{(0)}^{(j+1)} + \mathbf{y}_{(1)}^{(j+1)}$: periodization of length 2^j of $\mathbf{y} = \mathbf{y}^{(J)} \in \mathbb{R}^{2^J}$
- $\mathbf{x}^{[j]} \coloneqq \mathbf{x}_{(0)}^{[j+1]} + \mathbf{J}_{2^j}\mathbf{x}_{(1)}^{[j+1]}$: reflected periodization of length 2^j of $\mathbf{x} = \mathbf{x}^{[J]} \in \mathbb{R}^{2^J}$.
- $\mathbf{A}^{[j]} \coloneqq \mathbf{A}_{(0,0)}^{[j+1]} + \mathbf{J}_{M^{[j]}}\mathbf{A}_{(1,0)}^{[j+1]} + \left(\mathbf{A}_{(0,1)}^{[j+1]} + \mathbf{J}_{M^{[j]}}\mathbf{A}_{(1,1)}^{[j+1]}\right)\mathbf{J}_{N^{[j]}}$: reflected periodization of size $M^{[j]} \times N^{[j]}$ of $\mathbf{A} \in \mathbb{R}^{M \times N}$.

List of Figures

List of Tables

Part I

Introduction

Introduction

Trigonometric transforms are usually understood to be transforms comprised of linear combinations of cosine and sine terms. In a discretized setting, which is necessary for practical applications, the trigonometric transforms are the discrete cosine and sine transforms. They are of immense importance in many areas of signal processing, including the JPEG image compression standard, the AAC audio compression standard, and image and video coding. Furthermore, the discrete cosine and sine transforms can be employed for solving some types of partial differential equations.

Cosine and sine transforms are also closely related to the Fourier transform. As Euler's identity links the natural exponential function with the cosine and sine functions via

$$e^{ix} = \cos(x) + i \sin(x) \qquad \forall\, x \in \mathbb{R},$$

the Fourier transform of an even function reduces to a cosine transform and the Fourier transform of an odd function reduces to a sine transform. Similarly, the Fourier transform in a discretized setting, the so-called discrete Fourier transform, is also connected to the discrete cosine and sine transforms.

Extensive research over the past few decades provided us with essentially runtime-optimal algorithms for the discrete Fourier transform and the discrete cosine and sine transforms. Significant runtime improvements are only possible if there is additional a priori information about the signal. In practice, one usually assumes that the output signal is sparse, meaning that only a few of its components are significant. Thus, it often suffices to only recover those components to obtain a good approximation. Since this can usually be achieved in less time than required by full-length algorithms, the closely connected fields of sparse fast Fourier transforms and sparse fast trigonometric transforms are much investigated areas of research. Many applications in signal and image processing can merit from new methods for sparse fast Fourier transforms and sparse fast trigonometric transforms.

Fourier Transform

The first part of this thesis addresses the problem of sparse fast Fourier transforms, which is closely related to the topic of sparse trigonometric transforms. Like few other mathematical concepts Fourier analysis and its applications have shaped today's world, due to their extensive usage in many areas of signal and image processing, engineering, physics and data processing. Well-known technologies based on Fourier analysis and closely related concepts include, for example, musical signal processing, image and video compression, computer tomography, nuclear magnetic resonance spectroscopy and infrared spectroscopy, as well as mass spectrometry and magnetic resonance imaging.

During the course of his work on heat propagation in solid bodies, the French mathematician Jean Baptiste Joseph Fourier (1768–1830) was able to prove that every periodic function can be approximated well by an expansion into trigonometric functions. For a

2π-periodic function $f\colon [0, 2\pi] \to \mathbb{R}$ this would mean that

$$f(x) \approx a_0 + \sum_{k=1}^{\infty} a_k \cos(kx) + \sum_{k=1}^{\infty} \sin(kx) = \sum_{n=-\infty}^{\infty} c_n e^{inx} \qquad \forall\, x \in [0, 2\pi),$$

where

$$a_0 \coloneqq \frac{1}{2\pi} \int_0^{2\pi} f(x) \mathrm{d}x,$$
$$a_k \coloneqq \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(kx) \mathrm{d}x, \qquad b_k \coloneqq \frac{1}{\pi} \int_0^{2\pi} f(x) \sin(kx) \mathrm{d}x \qquad \forall\, k \in \mathbb{N}.$$

and

$$c_0 \coloneqq a_0 \qquad \text{and} \qquad c_n \coloneqq \begin{cases} \frac{1}{2}(a_n - ib_n) & \text{if } n > 0, \\ \frac{1}{2}(a_n + ib_n) & \text{if } n < 0. \end{cases}$$

Such a series is called Fourier series, and its coefficients a_k, b_k and c_n are known as Fourier coefficients. Fourier series can be utilized to solve certain types of differential equations, particularly linear differential equations with constant coefficients, including the heat equation, the wave equation and Schrödinger's equation. Chapters 1 and 2 in [Fol92] and Chapter 1 in [PPST19] provide detailed derivations of Fourier series from differential equations like the heat equation inspired by Fourier's approach.

Nowadays, the process of computing the coefficients in the Fourier series of a periodic function f is known as the finite Fourier transform. For a periodic function the coefficient c_n can be interpreted as a measure for how much the frequency $n \in \mathbb{Z}$ contributes to the input signal f.

There also exists an extension of the concept of the Fourier transform to non-periodic functions, where the Fourier transform of an absolutely integrable function $f\colon \mathbb{R} \to \mathbb{C}$ is its continuous spectrum. Many physical phenomena can be described using the Fourier transform. One of them is the Fraunhofer diffraction, which approximates the diffraction pattern of a wave at a long distance from the diffracting object, e.g., a single slit or a double slit. The diffraction pattern is approximately given as the Fourier transform of the diffracting object.

The finite Fourier transform for 2π-periodic functions is a powerful theoretical tool, but its computation requires knowledge of the function for a complete period. Thus, it cannot be applied directly in practice, since measurements can only be taken discretely. Discretizing the integrals required for the computation of the Fourier coefficients with the trapezoidal rule on a uniform grid of length $N \in \mathbb{N}$ yields that

$$c_n \approx \frac{1}{N} \sum_{j=0}^{N-1} f\left(\frac{2\pi j}{N}\right) e^{\frac{-2\pi i j n}{N}} \quad \forall\, n \in \mathbb{Z}.$$

The sum on the right-hand side is known as the discrete Fourier transform (DFT) of the vector $\left(f\left(\frac{2\pi j}{N}\right)\right)_{j=0}^{N-1}$. Hence, the coefficient c_n can be approximated by the nth entry of the DFT of this vector of equidistant samples of f. If it is known a priori that the frequencies with significantly large Fourier coefficients are contained in the interval $\left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$, then the above approximation is very accurate. Similarly, the Fourier transform for non-periodic functions can be approximated well by the DFT. The discrete Fourier transform also arises naturally in the context of trigonometric polynomial

interpolation, i.e., the interpolation of data points by a function of the form

$$P(x) = \sum_{k=-n}^{n} c_k e^{ikx} \qquad x \in \mathbb{R}.$$

As a means of discretely approximating the finite Fourier transform, the DFT is already very valuable. However, the definition given above implies that the DFT can be written as the multiplication of a dense $N \times N$ matrix with a vector of length N, which has an arithmetical complexity of $\mathcal{O}(N^2)$. The development of machine computing in the second half of the 20th century and ever-increasing amounts of input data motivated the development of algorithms with significantly lower runtimes. The first algorithm achieving a runtime that is subquadratic in N was published in 1958, see [Goo58], but was not further recognized. In 1965, Cooley and Tukey introduced the first well-known DFT algorithm with a runtime of $\mathcal{O}(N \log N)$, see [CT65]. DFT algorithms with such a runtime are known as fast Fourier transforms (FFT). A detailed compilation of many FFT algorithms can be found in [CG99].

With the definition given as above, FFT algorithms require equidistant samples of a 2π-periodic function. However, the acquisition of equidistant samples is not feasible for all practical applications. This inspired the research of FFT algorithms for non-equispaced data, the so-called NFFTs. There exist NFFT algorithms achieving the same order of runtime as the FFT, see, e.g., [DR93, Bey95, Ste98, PST01]. Chapter 7 in [PPST19] provides an overview of a variety of NFFT methods.

Due to the technological developments of the past 50 years, the amount of data that has to be processed have increased even further. Consequently, faster algorithms than conventional FFTs are desirable for many applications. It has been shown that for arbitrary input vectors of length N the order $N \log N$ of the runtime is optimal. Therefore, research in recent years focused on finding FFT algorithms with lower runtimes. Provided that there is some a priori information about the vector given, runtimes that are sublinear in N could be achieved. Usually, one assumes that the output vector is sparse, meaning that it has only a few significantly large entries. Many such methods are summarized in the survey [GIIS14].

Contribution to Sparse FFTs

In the first part of this thesis we will focus on two different classes of 2π-periodic frequency sparse functions. For both we will introduce deterministic algorithms for computing the Fourier coefficients from as few samples and using as few arithmetical computations as possible. All of our algorithms achieve runtime and sampling complexities that are sublinear in the assumed bandwidth of the function. The first class of frequency sparse functions we will consider are functions with short frequency support, meaning that all frequencies corresponding to significantly large Fourier coefficients are contained in an interval of length B in $\mathbb{Z}$. We introduce two new algorithms arising from different simplifications of Algorithm 2 in [Iwe10], which is a deterministic sublinear-time algorithm for computing the Fourier coefficients of an arbitrary B-sparse function from samples. Algorithm 2 in [Iwe10] requires very complex sampling schemes that can be relaxed in two ways by utilizing the short frequency support, resulting in one algorithm with a runtime of

$$\mathcal{O}\left(B \log B \frac{\log^2 \frac{N}{B}}{\log \log \frac{N}{B}}\right)$$

and another with a runtime of

$$\mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B}\log^2\left(\frac{B+\log N}{\log B}\right)\right).$$

All existing deterministic FFT algorithms for recovering sparse functions from samples have runtimes which scale quadratically in the sparsity B and are thus slower than our algorithms. Both of the algorithms are deterministic and require a priori knowledge of an upper bound on the support length B of the function we aim to recover.

The second class of frequency sparse functions we will investigate in this thesis are functions with polynomially structured sparsity. This means that the frequencies corresponding to significantly large Fourier coefficients are generated by evaluating n polynomials of degree at most d at B consecutive points. Polynomially structured sparsity is a generalization of the concept of short frequency supports, since a short frequency support can be interpreted as being generated by evaluating a single monic linear polynomial at B consecutive points. We will derive a deterministic algorithm that computes the Fourier coefficients of a polynomially structured sparse function from samples in sublinear time. This algorithm, which is, to the best of our knowledge, the first deterministic algorithm for recovering polynomially structured sparse functions, can be seen as a generalization of Algorithm 2 in [Iwe10] and Algorithm 3 in [Iwe13]. The key feature of our new algorithm is the restriction of the input function to frequencies that satisfy certain congruency conditions, which contributes significantly to its

$$\mathcal{O}\left(\frac{Bd^2n^3\log^5 N}{\log^2(2dn)}\right)$$

runtime. Our algorithm needs a priori knowledge of upper bounds on the number of polynomials n, their maximal degree d and the number B of evaluation points. For the special case of block frequency sparse functions, where all generating polynomials are monic, our method can be slightly simplified, yielding an algorithm with runtime of

$$\mathcal{O}\left(\frac{Bn^2\log B\log^4 N}{\log^2(2n)}\right).$$

This runtime scales subquadratically in the sparsity Bn, thus performing better than all previously existing deterministic FFT methods for frequency sparse functions.

Discrete Cosine Transform

The second part of this thesis covers a particular sparse trigonometric transform, the discrete cosine transform of type II (DCT-II). It is a well-known fact that the cosine and sine functions are the eigenfunctions of the homogeneous harmonic oscillator system

$$u'' + \lambda u = 0$$

on the domain $[0, \pi]$. Discretizing this system, which is necessary for any kind of practical application, where all measurements can only be realized discretely, yields discretized cosine and sine functions. As eigenfunctions or eigenvectors of the discretized homogeneous harmonic oscillator they constitute the basis functions for the different types of discrete cosine and sine transforms (DCT and DST). More precisely, it has been shown in [Str99] that the different types of the DCT and DST are "the natural outcome of different combinations of homogeneous boundary conditions applied to the discretized solution of a

simple harmonic oscillator equation", see [BYR06], Section 2.1. For example, the basis functions of the discrete cosine transform of type I (DCT-I) can be obtained by applying Neumann boundary conditions at both ends of N equispaced grid points, and the basis functions of the discrete cosine transform of type II by applying the Neumann boundary conditions mid-grid at $-\frac{1}{2}$ and $N-\frac{1}{2}$. See [Str99] and [BYR06], Section 2.6, for a detailed explanation of this approach.

Using Neumann boundary conditions at 0 and either Neumann or Dirichlet boundary conditions at the far end, yields the DCTs of types I-IV if the conditions are applied either at grid points for both ends or mid-grid for both ends. If one of the boundary conditions is applied at grid points and the other one mid-grid, one obtains the DCTs of types V-VIII, which are used less often in practice.

The closely related DSTs can be found by using Dirichlet boundary conditions at 0 and either Neumann or Dirichlet boundary conditions at the far end. Following the same naming conventions as for the DCTs, the application of the boundary conditions at grid points for both ends or mid-grid for both ends yields the DSTs of types I-IV, and the application of one condition at grid points and the other one mid-grid yields the DSTs of types V-VIII, which are also of little practical importance.

All types of the DCT and the DST, often referred to as the *discrete trigonometric transforms*, possess some very important properties. For example, all of them can be written as a multiplication with an orthogonal transformation matrix. Other properties include linearity, scaling in time, shifts in time, and difference and convolution properties.

Some of the discrete trigonometric transforms have been shown to be very useful for a variety of problems in the area of digital signal processing. From the motivation of their definition as solutions of the discretized homogeneous harmonic oscillator it is not at all apparent why this should be the case. Nevertheless, it can be observed that the DCT-II diagonalizes the correlation matrix of a stationary Markov-1 signal. Since in fact many signals in practical applications are approximately stationary Markov-1 signals, this makes the DCT-II a powerful tool for dealing with such signals.

There even exists a transform that exactly diagonalizes the correlation matrix of any signal, the so-called Karhunen-Loéve transform (KLT), see, e.g., [Kar47, Loé48].

It can be shown that the KLT is an optimal transform with the following properties, cited from [BYR06], Section 3.2.

(i) It completely diagonalizes the signal in the transformation domain.

(ii) It minimizes the mean square error in bandwidth reduction or data compression.

(iii) It contains the most variance (energy) in the fewest number of transform coefficients.

(iv) It minimizes the total representation entropy of the data sequence.

These properties would make the KLT indispensable in signal processing and many other areas if there existed a fast transformation algorithm for it. Unfortunately, as the KLT is a highly signal dependent transform, this is not the case, which makes the KLT impractical for applications. Instead, researchers were interested in finding a predetermined, i.e., signal independent, basis that approximates the KLT well. This actually leads us back to the discrete trigonometric transforms. As proven in [AF82, Kit80, KSK77], the DCT of types I and II can be derived precisely as a solution for the problem of approximating the KLT for stationary Markov-1 signals. Thus, they are asymptotically equivalent to the KLT, which explains their great applicability to signal processing problems. For

the DCT-I the asymptotic behavior is obtained if the matrix size N approaches infinity and the adjacent correlation coefficient ρ does not approach 1. For the DCT-II, whose decorrelation property is independent of N, one has to consider the case where ρ tends to 1. Furthermore, the KLT is also asymptotically equivalent to the DFT for stationary Markov-1 signals if N approaches infinity.

Still, the DCTs and DSTs would never have been so widely used in signal processing if their computation was only possible via the above-mentioned matrix-vector multiplication, as such a multiplication has a runtime of $\mathcal{O}\left(N^2\right)$ for the dense DCT and DST matrices. Fortunately, as for the DFT, extensive research in the past few decades has provided us with a variety of fast algorithms with runtimes of $\mathcal{O}(N \log N)$. See, e.g., [RY90], Chapter 4, and [BYR06], Chapter 4, for an overview of many such methods.

For plenty of applications these improved runtimes are sufficient. However, over the past few decades, in many areas of application the amount of input data that has to be processed has increased faster than the computing power, making even faster, sublinear runtimes necessary. As for the DFT, this is not possible for arbitrary input vectors, though there has been some progress in developing faster algorithms for certain a priori sparsity assumptions.

Contribution to Sparse DCTs

In the second part of this thesis we will investigate the deterministic reconstruction of sparse vectors from their DCT-II transformed vectors. We will assume that the vectors we aim to recover have a short support, meaning that the indices corresponding to significantly large entries are contained in an interval of short length m. We will develop two new algorithms for the inverse discrete cosine transform of type II (IDCT-II) which are, as far as we are aware, the first sparse IDCT-II algorithms specifically tailored to the cosine bases and the short support.

The first algorithm we will derive is based on the fact that the DCT-II of any vector $\mathbf{x} = (x_k)_{k=0}^{N-1} \in \mathbb{R}^N$ can be directly computed from the auxiliary vector

$$\mathbf{y} = (x_0, x_1, \ldots, x_{N-1}, x_{N-1}, x_{N-2}, \ldots, x_0)^T \in \mathbb{R}^{2N}.$$

First, we will develop an algorithm for recovering $\mathbf{y}$, which has a so-called reflected block support if $\mathbf{x}$ has a short support, from its Fourier transformed vector $\widehat{\mathbf{y}}$. In order to do this we will utilize the notion of periodized vectors introduced in [PW16a]. Our new sparse IFFT algorithm can detect the support of $\mathbf{y}$ on the fly; thus, it does not require any a priori knowledge of the support length m. Then we utilize this IFFT method to derive a deterministic, adaptive IDCT-II algorithm for recovering $\mathbf{x}$ from $\mathbf{x}^{\widehat{\mathrm{II}}}$ in

$$\mathcal{O}\left(m \log m \log \frac{2N}{m}\right)$$

time. As our IFFT method is specifically designed for the sparsity structure of $\mathbf{y}$ that is induced by the short support of $\mathbf{x}$, it performs better than previously existing sparse IDCT-II methods employing arbitrary sparse IFFTs.

The DCT-II is a transform that can be computed in a fast way using only real arithmetic, so we will present another IDCT-II algorithm for recovering vectors with short support that only requires real arithmetic. The proposed algorithm employs the notion of reflected periodizations, a DCT-II specific analog to the periodizations arising in the DFT case. Due to slightly different sparsity constraints, this second IDCT-II method

requires a priori knowledge of an upper bound M on the support length m. It achieves a runtime of

$$\mathcal{O}\left(M \log M + m \log \frac{N}{M}\right)$$

for input vectors of length N. To the best of our knowledge, our algorithm is the first existing IDCT-II algorithm for vectors with short support that only uses real arithmetic. Numerical experiments show that it is even faster than our proposed IDCT-II algorithm based on special sparse IFFTs.

As many of the problems in which DCTs are used are actually higher dimensional, e.g., digital image and video compression, there is also a demand for fast algorithms for higher dimensions. Thus, we will introduce a new IDCT-II algorithm for recovering a matrix of size $M \times N$ with block support, meaning that all of its significantly large entries are contained in a rectangle of size $m \times n$, where m and n are small compared to M and N, respectively. The algorithm is based on generalizations of the techniques developed for our 1-dimensional IDCT-II algorithm that only uses real arithmetic. Analogously, the 2-dimensional IDCT-II algorithm requires a priori knowledge of upper bounds on the support sizes m and n. Under the assumption that $N \approx M$ and $n \approx m$ with upper bound $b \geq m$, it has a runtime of

$$\mathcal{O}\left(b^2 \log_2 b + b^2 \log_2 \frac{M}{b} + \frac{M}{2} b \log_2 \left(\frac{M}{2} b\right)\right).$$

As far as we are aware, this is the first 2-dimensional IDCT-II algorithm for block sparse matrices that only requires real arithmetic.

Overview

This thesis is divided into two main parts. In the first part we study fast Fourier transform algorithms for 2π-periodic frequency sparse functions. We begin by giving a brief overview of the discrete Fourier transform (DFT) and a variant of it, the centered discrete Fourier transform (CDFT) in Chapter 1. Further, we sketch one of the most famous algorithms for the fast discrete Fourier transform (FFT), the so-called Sande-Tukey algorithm. As we are interested in FFT algorithms for functions, we then introduce the finite Fourier transform for 2π-periodic functions and highlight its connection to the CDFT.

In Chapter 2 we develop two related algorithms for recovering 2π-periodic functions from samples if the input functions satisfy the simple sparsity constraint of having a short support. We also prove theoretical estimates for their runtime and sampling requirements. We conclude this chapter with a numerical comparison of these two algorithms to other sparse FFT methods regarding both runtime and robustness to noise.

Extending the previous setting, we investigate 2π-periodic functions with polynomially structured sparsity in Chapter 3. We begin by defining this theoretical concept and then derive an algorithm for polynomially structured sparse functions. Additionally, we investigate special cases for which our algorithm can be simplified, most notably block sparsity. For block sparse functions we also provide an adapted version of our algorithm with reduced runtime. Furthermore, we show theoretical runtime and sampling bounds for the algorithm for polynomially structured sparsity and the algorithm for block sparsity. We complete the chapter by numerically investigating the runtime and the robustness to noise of the algorithm for block sparse functions.

The second part of the thesis is concerned with the related topic of sparse fast discrete cosine transforms. Instead of recovering functions from samples, we aim to recover a

real-valued vector from its discrete cosine transform of type II (DCT-II). In Chapter 4 we define the most common types of the DCT and summarize a fast algorithm for the DCT-II. Further, we recall some important results regarding Vandermonde matrices and Chebyshev polynomials.

Starting from this background, we introduce two algorithms for the sparse IDCT-II for vectors with short support. In Chapter 5 we first recall the notions of short support and periodized vectors, which were introduced in [PW16a]. Then we present an algorithm for recovering a vector $\mathbf{x}$ with short support from its DCT-II transformed vector. The method is based on recovering an auxiliary vector $\mathbf{y}$ of double length from its IFFT. The vector $\mathbf{y}$ has a special sparsity structure, the so-called reflected block support, if $\mathbf{x}$ has a short support. We first develop an IDFT algorithm for recovering $\mathbf{y}$ from its FFT transformed vector by iteratively reconstructing its periodizations. Then we will use this method to derive an IDCT-II algorithm for vectors with short support. The chapter closes with theoretical estimates for the runtimes and the number of required samples of both the sparse IDFT and the sparse IDCT-II algorithm.

In Chapter 6 we introduce a sparse IDCT-II algorithm for vectors with short support which only requires real arithmetic. We begin by introducing the concept of reflected periodizations, a DCT-II-specific analog to periodized vectors. Based on them we develop our algorithm and prove its theoretical runtime and sampling complexities. We conclude the chapter with a numerical comparison of our IDFT for vectors with reflected block support and our two IDCT-II algorithms for vectors with short support with other sparse IDFT and sparse IDCT-II methods.

Transferring the techniques from Chapter 6 to the more general setting of matrices, we conclude this thesis by presenting a 2-dimensional IDCT-II algorithm for matrices with block support in Chapter 7. First, we generalize the concepts of short support and reflected periodizations and use them to derive our algorithm. Then we prove estimates on its runtime and number of required samples.

Please Note

Parts of this thesis have already been published in our papers [Bit17c, BP18c, BP18a, BZI19]. I significantly contributed to the publications [BZI19, BP18c, BP18a], which constitute Chapters 3, 5 and 6, and I am the corresponding author for all three. Furthermore, I am the sole author of [Bit17c], included in this thesis as Sections 2.1 to 2.3, and also developed the method introduced in Section 2.4 on my own. Finally, the 2-dimensional IDCT-II for sparse matrices presented in Chapter 7 was also completely developed by myself.

Part II

Sparse Fast Fourier Transform

1 Fourier Transform

The Fourier transform has proven to be one of the most important mathematical transforms, with applications in, e.g., signal and image processing, engineering, physics, and data processing. Hence, the efficient reconstruction of signals from Fourier data or from samples is a problem which has been investigated in great detail over the past few decades. Before presenting new deterministic sparse fast Fourier algorithms for periodic functions with structured Fourier sparsity in Chapters 2 and 3, we will provide the theoretical background for the discrete Fourier transform for complex vectors of length N and the finite Fourier transform for 2π-periodic functions in this chapter.

1.1 Discrete Fourier Transform

Let us begin by defining the discrete Fourier transform (DFT) for complex vectors and stating some of its properties. The following definitions and theorems are based on [CLRS09], Chapter 30.2, [CG99], Chapters 1 and 3, and [PPST19], Chapter 3.2.

Definition 1.1 (Discrete Fourier Transform (DFT)) Let $N \in \mathbb{N}$ and $\mathbf{y} = (y_k)_{k=0}^{N-1} \in \mathbb{C}^N$. Define the *$N$th Fourier matrix* $\mathbf{F}_N \in \mathbb{C}^{N \times N}$ as

$$\mathbf{F}_N \coloneqq \left({\omega_N}^{kl}\right)_{k,\,l=0}^{N-1},$$

where $\omega_N \coloneqq e^{-\frac{2\pi i}{N}}$ is an *Nth primitive root of unity*. Then the *discrete Fourier transform* $\widehat{\mathbf{y}} = (\widehat{y}_k)_{k=0}^{N-1} \in \mathbb{C}^N$ of $\mathbf{y}$ is given by

$$\widehat{\mathbf{y}} \coloneqq \mathbf{F}_N \mathbf{y}.$$

Thus, we can write the entries of $\widehat{\mathbf{y}}$ as

$$\widehat{y}_k = \sum_{l=0}^{N-1} {\omega_N}^{kl} y_l \quad \forall\, k \in \{0, \ldots, N-1\}.$$

It is a well known fact that the Fourier matrix is invertible. For a proof see, e.g., [CLRS09], Theorem 30.7. Consequently, there also exists the inverse discrete Fourier transform (IDFT).

Definition 1.2 (Inverse Discrete Fourier Transform (IDFT)) Let $N \in \mathbb{N}$ and $\widehat{\mathbf{y}} = (\widehat{y}_k)_{k=0}^{N-1} \in \mathbb{C}^N$. Then its *inverse discrete Fourier transform* $\mathbf{y} = (y_k)_{k=0}^{N-1} \in \mathbb{C}^N$ is given by

$$\mathbf{y} \coloneqq {\mathbf{F}_N}^{-1} \widehat{\mathbf{y}},$$

where

$${\mathbf{F}_N}^{-1} \coloneqq \frac{1}{N}\overline{\mathbf{F}_N} = \frac{1}{N}\left({\omega_N}^{-kl}\right)_{k,\,l=0}^{N-1} \in \mathbb{C}^{N \times N}$$

is the inverse of the Nth Fourier matrix. Hence, we find that

$$y_k = \frac{1}{N} \sum_{l=0}^{N-1} {\omega_N}^{-kl} \widehat{y_l} \quad \forall\, k \in \{0, \dots, N-1\}.$$

Note that the Nth Fourier matrix is unitary if the scaling factor in Definition 1.1 is chosen as $\frac{1}{\sqrt{N}}$ instead of 1 and the one in Definition 1.2 as $\frac{1}{\sqrt{N}}$ instead of $\frac{1}{N}$. The DFT has many useful properties, some of which are summarized in the following theorem.

Theorem 1.3 *Let $N \in \mathbb{N}$, $\mathbf{y}, \mathbf{z} \in \mathbb{C}^N$, $\alpha, \beta \in \mathbb{C}$, $n \in \mathbb{Z}$ and define the* flipping matrix *$\mathbf{U}_N$ via*

$$\mathbf{U}_N \coloneqq \left(\delta_{k+l}^{(N)}\right)_{k,\,l=0}^{N-1} \coloneqq \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \iddots & 1 \\ \vdots & \iddots & \iddots & \\ 0 & 1 & & 0 \end{pmatrix},$$

where $\delta^{(N)}$ denotes the N-periodic Kronecker symbol*,*

$$\delta_k^{(N)} \coloneqq \begin{cases} 1 & \text{if } k \equiv 0 \mod N, \\ 0 & \text{if } k \not\equiv 0 \mod N \end{cases} \qquad \forall\, k \in \mathbb{Z}.$$

Further, we define the matrices

$$\mathbf{P}_N \coloneqq \left(\delta_{k-l-1}^{(N)}\right)_{k,\,l=0}^{N-1} \quad \text{and} \quad \mathbf{M}_N = \operatorname{diag}\left(\left({\omega_N}^k\right)_{k=0}^{N-1}\right).$$

Then the following statements are true:

(i) $\widehat{(\alpha\mathbf{y} + \beta\mathbf{z})} = \alpha\widehat{\mathbf{y}} + \beta\widehat{\mathbf{z}}$,

(ii) $\mathbf{y} = {\mathbf{F}_N}^{-1}\widehat{\mathbf{y}} = \frac{1}{N}\mathbf{U}_N\mathbf{F}_N\widehat{\mathbf{y}}$,

(iii) $\widehat{\mathbf{U}_N\mathbf{y}} = \mathbf{U}_N\widehat{\mathbf{y}}$ *and* $\widehat{\overline{\mathbf{y}}} = \mathbf{U}_N\overline{\widehat{\mathbf{y}}}$,

(iv) $\widehat{{\mathbf{P}_N}^n\mathbf{y}} = {\mathbf{M}_N}^n\,\widehat{\mathbf{y}}$ *and* $\widehat{{\mathbf{M}_N}^{-n}\mathbf{y}} = {\mathbf{P}_N}^n\,\widehat{\mathbf{y}}$,

(v) $\frac{1}{N}\,(\widehat{\mathbf{y}}, \widehat{\mathbf{z}}) = (\mathbf{y}, \mathbf{z}) \coloneqq \sum_{k=0}^{N-1} y_k\overline{z_k}$.

For a proof see [PPST19], Chapter 3.2.3, Theorem 3.26.

Remark 1.4 It follows from Theorem 1.3 (ii) that, as the multiplication of $\mathbf{U}_N$ and a vector $\mathbf{y} \in \mathbb{C}^N$ only reorders the entries of $\mathbf{y}$, the DFT and the IDFT can be computed using the same algorithm with runtimes of the same order. ◊

The DFT of a real vector $\mathbf{y}$ and the vector obtained by cyclically shifting all entries by half the vector length are also closely related, as was shown in [PW16a], Lemma 2.2.

Lemma 1.5 Let $\mathbf{u} \in \mathbb{R}^{2^{j+1}}$, $j \geq 0$, and let the shifted vector $\mathbf{u}^1 \coloneqq \left(u_k^1\right)_{k=0}^{2^{j+1}-1} \in \mathbb{R}^{2^{j+1}}$ be given by

$$u_k^1 \coloneqq u_{(k+2^j) \bmod 2^{j+1}}, \qquad k \in \left\{0, \dots, 2^{j+1} - 1\right\}.$$

Then $\widehat{\mathbf{u}^1}$ satisfies

$$\widehat{u^1}_k = (-1)^k \, \widehat{u}_k \qquad \forall \, k \in \left\{0, \ldots, 2^{j+1} - 1\right\}.$$

1.1.1 Fast Fourier Transform

Computing the DFT of a vector $\mathbf{y} \in \mathbb{C}^N$ via the matrix-vector multiplication from Definition 1.1 has a runtime of $\mathcal{O}\left(N^2\right)$. However, employing more efficient strategies, one can develop algorithms with a runtime of $\mathcal{O}(N \log N)$. For arbitrary vectors, where no further a priori knowledge about their entries is given, this runtime can in fact be shown to be optimal.

In this section, which is based on [CG99], Chapter 3 and [PPST19], Chapter 5.2, we will briefly outline one of the most widely known fast DFT (FFT) algorithms, achieving a runtime of $\mathcal{O}(N \log N)$.

By Definition 1.1, the DFT of a vector $\mathbf{y} \in \mathbb{C}^N$ can be computed by multiplying the matrix $\mathbf{F}_N$ by $\mathbf{y}$. However, being a Vandermonde matrix (see Chapter 4.4 for more detailed information on Vandermonde matrices), $\mathbf{F}_N$ has a very special structure which can be exploited using the so-called *divide-and-conquer* paradigm to obtain fast algorithms for the computation of $\mathbf{F}_N\mathbf{y}$. There exists a variety of FFT algorithms based on this approach, see, e.g., [CT65, GS66, Ber68]. In [CG99], Chapter 3, the divide-and-conquer technique is characterized as follows:

Step 1 Divide the problem into two or more subproblems of smaller size.

Step 2 Solve each subproblem recursively by the same algorithm. Apply the boundary condition to terminate the recursion when the sizes of the subproblems are small enough.

Step 3 Obtain the solution for the original problem by combining the solutions to the subproblems.

The most widely known FFT algorithms are *radix-2* algorithms, which are based on separating the computation of the DFT of a vector $\mathbf{y} \in \mathbb{C}^N$ into two DFT computations of half length. Applying this idea recursively implies that these methods are best suited for vector lengths N that are a power of 2, since then one can employ the above steps until the vector length in the subproblems is 1. Let us thus assume that $N = 2^J$.

There are two main possibilities for reducing the problem of the DFT computation to a DFT computation of half length, *decimation in time* and *decimation in frequency*. Recall that by Definition 1.1 we have that

$$\widehat{y}_k = \sum_{l=0}^{N-1} {\omega_N}^{kl} y_l \tag{1.1}$$

for any $\mathbf{y} \in \mathbb{C}^N$. The vector $\mathbf{y}$ is sometimes said to be contained in time-domain and the vector $\widehat{\mathbf{y}}$ in frequency domain, analogously to time and frequency domain for the Fourier transform for square-integrable functions $f\colon \mathbb{R} \to \mathbb{C}$, see, e.g., [PPST19], Chapter 2, and the finite Fourier transform for 2π-periodic functions, see Section 1.2 of this thesis. Decimation in time means that we split the sum in (1.1) into two sums such that each sum only involves half of the entries of the time-domain vector $\mathbf{y}$. To be precise, one sum only depends on the evenly indexed entries of $\mathbf{y}$ and the other one only depends on the oddly indexed entries. This approach yields the so-called Cooley-Tukey FFT algorithm, see [CT65].

We will now explain the second idea of decimation in frequency in more detail. This method, often referred to as the Sande-Tukey algorithm, was first described in [GS66]. In fact, by writing the required operations for decimation in time and decimation in frequency algorithms in matrix form, and factorizing the occurring matrices, one can show that each of the methods can be derived from the other. Analogously to the idea of decimation in time, in each step we restrict the entries of the frequency-domain vector $\widehat{\mathbf{y}}$, which is returned by the method, to the evenly and the oddly indexed ones. Then both the evenly and the oddly indexed entries of $\widehat{\mathbf{y}}$ can be computed with the help of a DFT of length $\frac{N}{2}$. This can be achieved by rewriting (1.1) as follows,

$$\begin{aligned}
\widehat{y}_k &= \sum_{l=0}^{\frac{N}{2}-1} {\omega_N}^{kl} y_l + \sum_{l=\frac{N}{2}}^{N-1} {\omega_N}^{kl} y_l \\
&= \sum_{l=0}^{\frac{N}{2}-1} {\omega_N}^{kl} y_l + \sum_{l=0}^{\frac{N}{2}-1} {\omega_N}^{k\left(l+\frac{N}{2}\right)} y_{l+\frac{N}{2}} \\
&= \sum_{l=0}^{\frac{N}{2}-1} {\omega_N}^{kl} \left(y_l + y_{l+\frac{N}{2}} {\omega_N}^{k\frac{N}{2}} \right) \qquad \forall\, k \in \{0, \dots, N-1\}. \qquad (1.2)
\end{aligned}$$

Now we consider the evenly and the oddly indexed entries of $\widehat{\mathbf{y}}$ separately. First, we focus on indices of the form $k = 2k'$, where $k' \in \left\{0, \dots, \frac{N}{2} - 1\right\}$. Then (1.2) yields

$$\begin{aligned}
\widehat{y}_{2k'} &= \sum_{l=0}^{\frac{N}{2}-1} {\omega_N}^{2k'l} \left(y_l + y_{l+\frac{N}{2}} {\omega_N}^{k'N} \right) \\
&= \sum_{l=0}^{\frac{N}{2}-1} {\omega_{\frac{N}{2}}}^{k'l} \left(y_l + y_{l+\frac{N}{2}} \right). \qquad (1.3)
\end{aligned}$$

If we define the vector $\mathbf{y}^{(N/2),\,\mathrm{e}} \in \mathbb{C}^{\frac{N}{2}}$ via

$$y_l^{(N/2),\,\mathrm{e}} := y_l + y_{l+\frac{N}{2}} \qquad \forall\, l \in \left\{0, \dots, \frac{N}{2} - 1\right\},$$

we find that (1.3) implies

$$\widehat{y}_{2k'} = \widehat{y^{(N/2),\,\mathrm{e}}}_{k'} = \sum_{l=0}^{\frac{N}{2}1-1} {\omega_{\frac{N}{2}}}^{k'l} y_l^{(N/2),\,\mathrm{e}} \qquad \forall\, k' \in \left\{0, \dots, \frac{N}{2} - 1\right\}, \qquad (1.4)$$

so the vector $(\widehat{y}_{2k'})_{k'=0}^{\frac{N}{2}-1}$ can be computed via a DFT of length $\frac{N}{2}$, which is the first subproblem of half size.

For oddly indexed entries of the form $k = 2k'+1$, where $k' \in \left\{0, \dots, \frac{N}{2} - 1\right\}$, it follows

from (1.2) that

$$\begin{aligned}\widehat{y}_{2k'+1} &= \sum_{l=0}^{\frac{N}{2}-1} {\omega_N}^{(2k'+1)l} \left(y_l + y_{l+\frac{N}{2}} {\omega_N}^{(2k'+1)\frac{N}{2}}\right) \\ &= \sum_{l=0}^{\frac{N}{2}-1} {\omega_{\frac{N}{2}}}^{k'l} {\omega_N}^{l} \left(y_l + y_{l+\frac{N}{2}} {\omega_N}^{k'N} {\omega_N}^{\frac{N}{2}}\right) \\ &= \sum_{l=0}^{\frac{N}{2}-1} {\omega_{\frac{N}{2}}}^{k'l} \left(y_l - y_{l+\frac{N}{2}}\right) {\omega_N}^{l}. \end{aligned} \tag{1.5}$$

Defining the vector $\mathbf{y}^{(N/2),\mathrm{o}} \in \mathbb{C}^{\frac{N}{2}}$ via

$$y_l^{(N/2),\mathrm{o}} := \left(y_l - y_{l+\frac{N}{2}}\right) {\omega_N}^{l} \qquad \forall\, l \in \left\{0, \ldots, \frac{N}{2} - 1\right\},$$

(1.5) yields

$$\widehat{y}_{2k'+1} = \widehat{y^{(N/2),\mathrm{o}}}_{k'} = \sum_{l=0}^{\frac{N}{2}-1} {\omega_{\frac{N}{2}}}^{k'l}\, y_l^{(N/2),\mathrm{o}} \qquad \forall\, k \in \left\{0, \ldots, \frac{N}{2} - 1\right\}, \tag{1.6}$$

so the vector $(\widehat{y}_{2k'+1})_{k'=0}^{\frac{N}{2}-1}$ can also be computed using a DFT of length $\frac{N}{2}$, which gives us the second subproblem of half size. By construction of the subproblems, every entry of $\widehat{\mathbf{y}}$ is calculated exactly once if both subproblems (1.4) and (1.6) are solved.

By Step 2 of the divide-and-conquer paradigm, the same idea of splitting the sum for the matrix-vector multiplication is now applied to the DFTs of the two vectors $\mathbf{y}^{(N/2),\mathrm{e}}$ and $\mathbf{y}^{(N/2),\mathrm{o}}$ in (1.4) and (1.6), resulting in two new subproblems of computing the DFT of a vector of length $\frac{N}{4}$ for each of the $\frac{N}{2}$-length vectors. As we assumed that $N = 2^J$, this idea can be applied repeatedly until the final 2^J subproblems have length 1.

It still remains to be shown that this method indeed improves the runtime of $\mathcal{O}\left(N^2\right)$ of the matrix-vector multiplication $\mathbf{F}_N\mathbf{y}$. Computing the two vectors $\mathbf{y}^{(N/2),\mathrm{e}}$ and $\mathbf{y}^{(N/2),\mathrm{o}}$ requires $2 \cdot \frac{N}{2} = N$ complex additions and $\frac{N}{2}$ complex multiplications with the so-called *twiddle factors* ${\omega_N}^{l}$ for $l \in \left\{0, \ldots, \frac{N}{2} - 1\right\}$. Furthermore, two DFTs of length $\frac{N}{2}$ of the vectors $\mathbf{y}^{(N/2),\mathrm{e}}$ and $\mathbf{y}^{(N/2),\mathrm{o}}$ have to be computed. Instead of calculating these DFTs directly, we split each of the two vectors again into two vectors of length $\frac{N}{4}$. Thus, the second step also requires $2 \cdot 2 \cdot \frac{N}{4} = N$ additions and $2 \cdot \frac{N}{4} = \frac{N}{2}$ complex multiplications, in addition to 4 DFTs of length $\frac{N}{4}$. Repeatedly applying this idea until subproblem vectors of length 1 are achieved, the Sande-Tukey algorithm performs J steps altogether, using

$$\sum_{j=0}^{J-1} N = NJ = N \log_2 N$$

complex additions and

$$\sum_{j=0}^{J-1} \frac{N}{2} = \frac{N}{2} \log_2 N$$

complex multiplications. This yields an overall runtime of $\mathcal{O}(N \log N)$.

The closely related Cooley-Tukey algorithm, based on decimation in time, also has a

runtime of $\mathcal{O}(N \log N)$. Furthermore, it can be shown that for arbitrary input vectors $\mathbf{y} \in \mathbb{C}^N$ the order $N \log N$ of the runtime of these FFT methods is optimal, i.e., that there cannot exist a general FFT algorithm with a runtime below $\mathcal{O}(N \log N)$, see [Mor73].

There also exist *radix-4* algorithms for vector lengths $N = 4^J$, see, e.g., [Ber68, Nus82], and so-called *split-radix* algorithms, which combine radix-4 and radix-2 methods, see, e.g., [Yav68, DH84].

Remark 1.6 FFT algorithms also exist for vectors whose lengths are not powers of 2. For vectors with small prime length see, e.g., [Rad68], and for vectors with lengths that are powers of a single prime see, e.g., [Win78]. If the vector length is the product of two relatively prime numbers, [Goo58, Nus82], among others, provide fast algorithms. There also exist FFT algorithms if the vector length is the product of several small primes, see, e.g., [Ber67].

Even for completely arbitrary vector lengths, e.g., [Blu70, RSR69] presented FFT procedures with a runtime of $\mathcal{O}(N \log N)$. These methods utilize that the DFT of a vector can be written as a convolution. The computation of this convolution can be embedded into a circulant matrix whose size is a power of 2. For an efficient calculation of the DFT this matrix has to be diagonalized, which only needs a DFT that can be found using, e.g., a radix-2 algorithm like the Sande-Tukey algorithm sketched above. See [CG99], Chapter 13, for a more detailed explanation of this approach.

Thus, even for arbitrary vector lengths N, there always exist methods with a runtime of $\mathcal{O}(N \log N)$ and a sampling complexity of $\mathcal{O}(N)$. Hence, we assume throughout this thesis that any N-length DFT computation requires $\mathcal{O}(N \log N)$ arithmetical operations. ◊

1.1.2 Centered Discrete Fourier Transform

For certain applications it is more convenient not to index the entries of a Fourier transformed vector $\widehat{\mathbf{y}} \in \mathbb{C}^N$ from 0 to $N-1$, as in Definition 1.1, but from $-\lceil \frac{N}{2} \rceil + 1$ to $\lfloor \frac{N}{2} \rfloor$. This usually makes sense if one is referring to *frequencies* in Fourier domain, as we will do in Chapters 2 and 3. For this reason we also define the centered discrete Fourier transform (CDFT). As we will never use both the DFT and the CDFT for the same problem, we denote both of them by $\widehat{\mathbf{y}}$ and clarify at the beginning of each section which transform will be used in the following.

Definition 1.7 (Centered Discrete Fourier Transform (CDFT)) Let $N \in \mathbb{N}$ and $\mathbf{y} \coloneqq (y_k)_{k=0}^{N-1} \in \mathbb{C}^N$. Its *centered discrete Fourier transform* $\widehat{\mathbf{y}} = (\widehat{y}_\nu)_{\nu=-\lceil \frac{N}{2} \rceil + 1}^{\lfloor \frac{N}{2} \rfloor} \in \mathbb{C}^N$ is given by

$$\widehat{\mathbf{y}} \coloneqq \widetilde{\mathbf{F}}_N \mathbf{y},$$

where the *Nth centered Fourier matrix* $\widetilde{\mathbf{F}}_N \in \mathbb{C}^{N \times N}$ is defined as

$$\begin{aligned} \widetilde{\mathbf{F}}_N &\coloneqq \frac{1}{N} \left({\omega_N}^{\nu k} \right)_{\nu=-\lceil \frac{N}{2} \rceil + 1, k=0}^{\lfloor \frac{N}{2} \rfloor, N-1} \\ &= \frac{1}{N} \left({\omega_N}^{\nu k} \cdot {\omega_N}^{k\left(-\lceil \frac{N}{2} \rceil + 1\right)} \right)_{\nu, k=0}^{N-1} \\ &= \frac{1}{N} \mathbf{F}_N \operatorname{diag} \left(\left({\omega_N}^{k\left(-\lceil \frac{N}{2} \rceil + 1\right)} \right)_{k=0}^{N-1} \right). \end{aligned} \tag{1.7}$$

Consequently, we can write the entries of $\widehat{\mathbf{y}}$ as

$$\widehat{y}_\nu = \frac{1}{N} \sum_{k=0}^{N-1} {\omega_N}^{\nu k} y_k \qquad \forall\, \nu \in \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}.$$

Lemma 1.8 The Nth centered Fourier matrix $\widetilde{\mathbf{F}}_N$ is invertible with inverse matrix

$$\begin{aligned} \widetilde{\mathbf{F}}_N^{-1} &= N \operatorname{diag}\left(\left(\omega_N^{-k\left(-\left\lceil \frac{N}{2} \right\rceil + 1\right)} \right)_{k=0}^{N-1} \right) {\mathbf{F}_N}^{-1} \\ &= \left({\omega_N}^{k\left(\left\lceil \frac{N}{2} \right\rceil - 1\right)} {\omega_N}^{-k\nu} \right)_{k,\,\nu=0}^{N-1} \\ &= \left({\omega_N}^{-k\nu} \right)_{k=0,\,\nu=-\left\lceil \frac{N}{2} \right\rceil + 1}^{N-1,\,\left\lfloor \frac{N}{2} \right\rfloor}. \end{aligned}$$

Proof. The proof follows immediately from Definition 1.2 and (1.7), since the diagonal matrix in said equation is invertible. □

Remark 1.9 Due to (1.7), the transformation matrices $\widetilde{\mathbf{F}}_N$ and $\mathbf{F}_N$ only differ by the multiplication with a diagonal matrix whose entries are Nth roots of unity. Thus, the CDFT can be computed using the same fast algorithms as the DFT after performing the $\mathcal{O}(N)$ operations necessary for the multiplication. We also obtain a sampling complexity of N and a theoretical runtime of $\mathcal{O}(N \log N)$ for the CDFT using the FFT mentioned in Section 1.1.1 for arbitrary vector lengths N. Consequently, we will henceforth say that we *apply the FFT* to a vector without specifying whether we used Definition 1.1 or 1.7, i.e., the CDFT or the standard DFT, as for all our purposes these two methods have the same order runtime. From the context it will always be clear whether we are employing the DFT or the CDFT. Furthermore, the multiplication with the diagonal matrix in (1.7) does not change the absolute values of the entries of the vector it is applied to. Hence, for any $\mathbf{y} \in \mathbb{C}^N$, $\mathbf{F}_N\mathbf{y}$ and $\widetilde{\mathbf{F}}_N\mathbf{y}$ have the same number of non-zero entries and also the same number of entries with small absolute value. ◊

1.2 Finite Fourier Transform

The concept of Fourier transforms does not only exist for vectors, but also for periodic functions, and for absolutely integrable functions from $\mathbb{R}$ into $\mathbb{C}$. All of these transforms are related, but there is a very close connection between the discrete Fourier transform and the one for periodic functions. For the remainder of this thesis we only consider 2π-periodic functions. However, the concepts discussed herein can also be extended to general P-periodic functions. The following section is based on [Fol92], Chapter 2.1 and [PPST19], Chapter 1.

Definition 1.10 (Fourier Coefficients and Fourier Series) Let $f \in L^1_{2\pi}$. The *Fourier coefficients* of f are given by

$$c_\nu(f) = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-i\nu x} \mathrm{d}x \qquad \forall\, \nu \in \mathbb{Z}.$$

Then the *Fourier series* of f is defined as

$$\sum_{\nu\in\mathbb{Z}} c_\nu(f)e^{i\nu x} \qquad \forall\, x \in [0, 2\pi).$$

With the help of the Fourier coefficients we can now define the finite Fourier transform for 2π-periodic square-integrable functions.

Definition 1.11 (Finite Fourier Transform) The mapping $F_{2\pi}\colon L^2_{2\pi} \to \ell^2$ with

$$F_{2\pi} f \coloneqq \mathbf{c}(f) \coloneqq (c_\nu(f))_{\nu\in\mathbb{Z}}$$

is called *Finite Fourier transform*. The vector $\mathbf{c}(f)$ is called the *finite spectrum* of f. The domain of $F_{2\pi}$ is also referred to as *time domain*, and its range as *frequency domain*.

The mapping $F_{2\pi}\colon L^2_{2\pi} \to \ell^2$ is linear, bounded and bijective, and its inverse, the *inverse finite Fourier transform*, is the mapping $F^{-1}_{2\pi}\colon \ell^2 \to L^2_{2\pi}$ with

$$\left(F^{-1}_{2\pi}\mathbf{c}\right)(x) \coloneqq \sum_{\nu\in\mathbb{Z}} c_\nu e^{i\nu x} \qquad \forall\, x \in [0, 2\pi).$$

Similar to the DFT, the finite Fourier transform has many useful properties.

Theorem 1.12 *Let $f, g \in L^1_{2\pi}$, $\alpha, \beta \in \mathbb{R}$, $\eta \in \mathbb{Z}$ and $x_0 \in [0, 2\pi)$. Then the following statements are true.*

(i) $\mathbf{c}(\alpha f + \beta g) = \alpha\mathbf{c}(f) + \beta\mathbf{c}(g)$,

(ii) $\mathbf{c}\left(f(\cdot - x_0)\right) = \left(e^{-i\nu x_0} c_\nu(f)\right)_{\nu\in\mathbb{Z}}$,

(iii) $(c_{\nu-\eta}(f))_{\nu\in\mathbb{Z}} = \left(c_\nu\left(e^{i\eta\cdot} f\right)\right)_{\nu\in\mathbb{Z}}$.

For a proof see [PPST19], Chapter 1.2, Lemma 1.6.

Theorem 1.13 *Let $f, g \in L^2_{2\pi}$. Then the following statements are true.*

(i) f has a unique representation of the form

$$f(x) = \sum_{\nu\in\mathbb{Z}} c_\nu(f)e^{i\nu x},$$

(ii) $$(f, g) \coloneqq \frac{1}{2\pi}\int_0^{2\pi} f(x)\overline{g(x)}dx = \sum_{\nu\in\mathbb{Z}} c_\nu(f)\overline{c_\nu(g)} =: (\mathbf{c}(f), \mathbf{c}(g)).$$

A proof of these claims can be found in [PPST19], Chapter 1.2, Theorem 1.3.

One can show that, under certain conditions, the Fourier series of a function f converges in the $L^2_{2\pi}$-norm (see [PPST19], Chapter 1.3), pointwise (see [PPST19], Chapter 1.4.1, Theorem 1.34) or uniformly (see [PPST19], Chapter 1.4.2, Remark 1.38) to f.

Theorem 1.14 *Let $f\colon \mathbb{R} \to \mathbb{C}$ be 2π-periodic.*

(i) If $f \in L^2_{2\pi}$, then its Fourier series converges to f w.r.t. the $L^2_{2\pi}$-norm, i.e.,

$$\lim_{n\to\infty} \|f - S_n f\|_2 = 0,$$

where

$$(S_n f)(x) \coloneqq \sum_{\nu=-n}^{n} c_\nu(f) e^{i\nu x} \qquad \forall\, x \in [0, 2\pi).$$

(ii) If f is piecewise differentiable, then for every $x_0 \in \mathbb{R}$ the Fourier series of f converges as

$$\lim_{n\to\infty} (S_n f)(x_0) = \frac{1}{2}\left(f(x_0 + 0) + f(x_0 - 0)\right).$$

(iii) If $f \in C^1_{2\pi}$, then its Fourier series converges uniformly to f.

Theorems 1.13 and 1.14 enable us to identify f with its Fourier series. For the remainder of this thesis we consider functions that only have finitely many significantly large Fourier coefficients, i.e., that satisfy

$$|c_\nu(f)| < \varepsilon \quad \forall\, \nu \notin \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}$$

for some threshold value $\varepsilon > 0$ and large $N \in \mathbb{N}$. Then we obtain that

$$f(x) \approx \sum_{\nu=-\left\lceil \frac{N}{2} \right\rceil + 1}^{\left\lfloor \frac{N}{2} \right\rfloor} c_\nu(f) e^{i\nu x} \qquad \forall\, x \in [0, 2\pi).$$

1.2.1 Connection between Finite and Centered Discrete Fourier Transform

For bandlimited 2π-periodic functions the finite Fourier transform and the CDFT are closely related. Let us first formally define the notion of bandlimited functions.

Definition 1.15 (Bandlimited Function) A function $f \in L^2_{2\pi}$ is called *bandlimited* on $\left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}$ for some $N \in \mathbb{N}$ if

$$|c_\nu(f)| = 0 \quad \forall\, \nu \notin \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}.$$

The natural number N is called the *bandwidth* of N.

Let us now consider a function $f \in L^2_{2\pi}$ with bandwidth N, i.e.,

$$f(x) = \sum_{\nu=-\left\lceil \frac{N}{2} \right\rceil + 1}^{\left\lfloor \frac{N}{2} \right\rfloor} c_\nu(f) e^{i\nu x} \qquad \forall\, x \in [0, 2\pi). \tag{1.8}$$

Note that a function f as in (1.8) is a trigonometric polynomial, see, e.g., [PPST19], Section 1.2, and as such contained in $C_{2\pi}$. We would like to obtain a connection between the finite spectrum $\mathbf{c}(f)$ of f and the CDFT. As the input argument of the CDFT has to be a vector rather than a function, we need to discretize f by constructing a suitable vector of evaluations of f. Since f is bandlimited with bandwidth N, its finite spectrum is determined by N Fourier coefficients. Because of the linearity of the DFT and the CDFT, it is natural to evaluate f at N equidistant points.

We define the following vector of N equidistant samples of f on $[0, 2\pi)$,

$$\mathbf{a}^N \coloneqq \left(f\left(\frac{2\pi j}{N}\right)\right)_{j=0}^{N-1} \in \mathbb{C}^N.$$

If we now apply the CDFT to $\mathbf{a}^N$, we obtain for any $\eta \in \left\{-\left\lceil \frac{N}{2}\right\rceil + 1, \dots, \left\lfloor \frac{N}{2}\right\rfloor\right\}$ that

$$\begin{aligned}
\widehat{a^N}_\eta &= \frac{1}{N}\sum_{j=0}^{N-1} {\omega_N}^{\eta j} f\left(\frac{2\pi j}{N}\right) \\
&= \frac{1}{N}\sum_{j=0}^{N-1} e^{\frac{-2\pi i j\eta}{N}} \sum_{\nu=-\left\lceil \frac{N}{2}\right\rceil+1}^{\left\lfloor \frac{N}{2}\right\rfloor} c_\nu(f) e^{\frac{2\pi i j\nu}{N}} \\
&= \sum_{\nu=-\left\lceil \frac{N}{2}\right\rceil+1}^{\left\lfloor \frac{N}{2}\right\rfloor} c_\nu(f) \underbrace{\frac{1}{N}\sum_{j=0}^{N-1} {\omega_N}^{j(\eta-\nu)}}_{=\begin{cases} 1 & \text{if } \nu = \eta, \\ 0 & \text{if } \nu \neq \eta. \end{cases}} \\
&= c_\eta(f). \qquad (1.9)
\end{aligned}$$

Consequently, by calculating the CDFT of the vector $\mathbf{a}^N \in \mathbb{C}^N$ of N equidistant samples of f, we obtain a vector that contains precisely the N Fourier coefficients $c_\nu(f)$ of f for $\nu \in \left\{-\left\lceil \frac{N}{2}\right\rceil + 1, \dots, \left\lfloor \frac{N}{2}\right\rfloor\right\}$. This vector is just a restriction of the finite spectrum $\mathbf{c}(f) \in \mathbb{C}^{\mathbb{Z}}$ of f to the frequencies contained in $\left\{-\left\lceil \frac{N}{2}\right\rceil + 1, \dots, \left\lfloor \frac{N}{2}\right\rfloor\right\}$,

$$\widehat{\mathbf{a}^N} = \left(c_\nu(f)\right)_{\nu \in \left\{-\left\lceil \frac{N}{2}\right\rceil+1, \dots, \left\lfloor \frac{N}{2}\right\rfloor\right\}} \in \mathbb{C}^N.$$

This means that we can compute the finite spectrum of a bandlimited function, which by (1.8) and Theorem 1.12 (i) completely defines f, if we calculate the CDFT of the vector $\mathbf{a}^N$ of N equidistant samples of f. Thus, we can recover f from N discrete, equidistant samples via the FFT in $\mathcal{O}(N \log N)$ time.

For functions that are only approximately bandlimited, i.e.,

$$|c_\nu(f)| < \varepsilon \quad \forall\, \nu \notin \left\{-\left\lceil \frac{N}{2}\right\rceil + 1, \dots, \left\lfloor \frac{N}{2}\right\rfloor\right\}$$

for some threshold value $\varepsilon > 0$ and large bandwidth $N \in \mathbb{N}$, with

$$f(x) \approx \sum_{\nu=-\left\lceil \frac{N}{2}\right\rceil+1}^{\left\lfloor \frac{N}{2}\right\rfloor} c_\nu(f) e^{i\nu x} \qquad \forall\, x \in [0, 2\pi),$$

the FFT of $\mathbf{a}^N$ provides a good approximation of the function.

Remark 1.16 If we apply the CDFT to a vector of equidistant samples of f of length $s < N$, i.e., to

$$\mathbf{a}^s \coloneqq \left(f\left(\frac{2\pi j}{s}\right)\right)_{j=0}^{s-1} \in \mathbb{C}^s,$$

we find that

$$\begin{aligned}\widehat{a^s}_\eta &= \frac{1}{s}\sum_{j=0}^{s-1} e^{\frac{-2\pi i j\eta}{s}} \sum_{\nu=-\lceil\frac{N}{2}\rceil+1}^{\lfloor\frac{N}{2}\rfloor} c_\nu e^{\frac{2\pi i j\nu}{s}} \\ &= \frac{1}{s}\sum_{\nu=-\lceil\frac{N}{2}\rceil+1}^{\lfloor\frac{N}{2}\rfloor} c_\nu \sum_{j=0}^{s-1} {\omega_s}^{j(\eta-\nu)} \\ &= \sum_{\substack{\nu=-\lceil\frac{N}{2}\rceil+1 \\ \nu\equiv\eta \bmod s}}^{\lfloor\frac{N}{2}\rfloor} c_\nu\end{aligned}$$

for all $\eta \in \left\{-\left\lceil\frac{s}{2}\right\rceil+1,\ldots,\left\lfloor\frac{s}{2}\right\rfloor\right\}$. ◊

Since the $\mathcal{O}(N\log N)$ runtime of the FFT is, as noted in Section 1.1.1, optimal for arbitrary N-length vectors, we can only hope to improve the runtime of fast Fourier algorithms for 2π-periodic functions if their Fourier coefficients satisfy additional a priori known conditions. The by far most interesting case is the one of functions with sparse frequency support, meaning that most of the corresponding Fourier coefficients are insignificantly small and that only few of them actually contribute to the Fourier series of the function. We will investigate different types of sparsity in Chapters 2 and 3.

2 Sparse FFT for 2π-Periodic Functions with Short Support

Sparse Fourier transforms have many applications in signal processing, for example analog-to-digital conversion, see, e.g., [LKM$^+$06, YRR$^+$12], GPS signal acquisition, see, e.g., [HAKI12], and wideband communication or spectrum sensing, see, e.g., [HSA$^+$14, YG12]. Thus, in the first part of this thesis we are interested in deterministically recovering 2π-periodic functions f from samples. By Theorem 1.14 it suffices to know the significantly large Fourier coefficients and the frequencies corresponding to them in order to obtain a good approximation of f. We can only hope to do this in a more efficient way than by directly applying the CDFT to the vector of N equidistant samples of f as in Section 1.2.1 if the number of Fourier coefficients we need to recover is small compared to the assumed bandwidth N of f.

Most of the existing sparse Fourier transform methods do not assume any further structure of the sparsity. The first sparse methods which achieved runtimes that are sublinear in the bandwidth or vector length N were randomized algorithms. This means that with a small, usually tunable probability the returned vector is not a good approximation of the correct solution. Such algorithms have runtimes of $\mathcal{O}\left(B \log^{\mathcal{O}(1)} N\right)$, see, e.g., [AGS03, GMS05, GGI$^+$02, Man92, HIKP12a, HIKP12c, IKP14, IGS07, CLW16, CCW16, LWC13, MZIC18, SI13, CIK18]. More information and implementations can be found in a survey about randomized sparse FFT algorithms, see [GIIS14].

There also exist deterministic sparse Fourier algorithms where the probability of failure is zero. These methods include techniques arising from modifications of Prony's method with a runtime of $\mathcal{O}\left(B^3\right)$, see, e.g., [HKPV13, PT14, PTV16]. As many Prony-based techniques suffer from numerical instabilities for noisy input data, they cannot be applied to all problems. Other deterministic method utilize arithmetic progressions or the Chinese Remainder Theorem, see, e.g., [Aka10, Aka14, Iwe10, Iwe13], or other properties of the DFT, see, e.g., [Mor16, PWCW18]. All of these non-Prony-based methods have in common that their runtime is $\mathcal{O}\left(B^2 \log^{\mathcal{O}(1)} N\right)$. Thus, they are sublinear in the vector length or bandwidth N, but quadratic in the sparsity B. For general B-sparsity it seems to be extremely difficult to reduce the quadratic dependence of the runtime on B, see, e.g., [BDF$^+$11, CI16, FR13].

However, if there is some additional a priori information about the sparsity structure, runtimes scaling subquadratically in the sparsity can indeed be achieved, see, e.g., [PW16a, PW17a] with runtimes of $\mathcal{O}(B \log N)$ and $\mathcal{O}\left(B \log B \log \frac{N}{B}\right)$. Consequently, we will focus on two types of structured sparsity in the first part of this thesis. To be more precise, we will always assume that the frequencies associated with significantly large Fourier coefficients are contained in a small number, n, of support sets $S_1, \ldots, S_n \subsetneq \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$, and that each of the unknown sets S_j has a "simple" structure.

In this chapter we will investigate the special case that there is only one such set S

and that S is an interval in $\mathbb{Z}$, i.e., that $n = 1$ and

$$S = \{\omega_1, \omega_1 + 1, \dots, \omega_1 + B - 1\}$$

for some starting frequency $\omega_1 \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor - B + 1\right\}$. In Chapter 3 we will extend our methods for deterministically recovering 2π-periodic functions to the case of frequency supports consisting of several sets with more complex structures.

In the vector setting the special case of $n = 1$ and S being an interval corresponds to the case that the vector we aim to recover has a short support. This is precisely the sparsity assumption of the deterministic sparse FFT algorithms [PW16a, PW17a]. See Section 5.3 for a more detailed explanation of these two methods.

Sections 2.1 to 2.3 in this chapter are based on my paper [Bit17c] and are in part identical with the representations therein. Section 2.4 presents completely new, previously unpublished results that I developed on my own.

2.1 Sparsity and Short Support

Throughout the next two chapters we will always consider a 2π-periodic function $f \in C_{2\pi}$ with finite spectrum $\mathbf{c}(f) \in \mathbb{C}^{\mathbb{Z}}$. We will assume that f has approximately a large bandwidth $N \in \mathbb{N}$. This means that the Fourier coefficients corresponding to the frequencies that are not contained in $\left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$ have an absolute value which is so small that it can be disregarded. Hence, f is of the form

$$f(x) \approx \sum_{\omega \in -\left\lceil \frac{N}{2} \right\rceil + 1}^{\left\lfloor \frac{N}{2} \right\rfloor} c_\omega(f) e^{i\omega x}.$$

As in practical applications the given data is usually noisy, we will assume that the function f is perturbed by a 2π-periodic function $\eta \in C_{2\pi}$ with $\mathbf{c}(\eta) \in \ell^1$ satisfying $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$ for some suitably chosen noise threshold $\varepsilon > 0$. Since we aim to recover f from finitely many samples of noisy data, our main object of interest for now are the Fourier coefficients of $f + \eta$.

Using a threshold parameter $\varepsilon > 0$, we can now formally define the notion of significantly large Fourier coefficients.

Definition 2.1 Let $f \in C_{2\pi}$, let $\varepsilon > 0$ be a suitably chosen noise threshold and $\omega \in \mathbb{Z}$. A Fourier coefficient $c_\omega(f) \in \mathbb{C}$ is called *significantly large* if $|c_\omega(f)| > \varepsilon$. A frequency ω is called *energetic* if its corresponding Fourier coefficient $c_\omega(f)$ is significantly large.

As already mentioned above, due to the fact that the runtime of the FFT is optimal for arbitrary N-length input vectors, we can only expect to improve its runtime of $\mathcal{O}(N \log N)$ if it is known a priori that many of the Fourier coefficients are insignificantly small. This motivates the following formal definition of the concept of sparsity.

Definition 2.2 (Sparsity) Let $f \in C_{2\pi}$ and let $\varepsilon > 0$ be a suitably chosen noise threshold. Then f is called *B-sparse* if it has only B energetic frequencies.

In this chapter we are interested in functions whose energetic frequencies are contained in a short interval in $\mathbb{Z}$.

Definition 2.3 (Short Support) Let $f \in C_{2\pi}$ and let $\varepsilon > 0$ be a suitably chosen noise threshold. If f is B-sparse such that all energetic frequencies are contained in a support interval

$$S \coloneqq \{\omega_1, \omega_1 + 1, \ldots, \omega_1 + B - 1\} \subsetneq \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}$$

of length B, we say that f has a *short (frequency) support of length B*.

A short support is, depending on the context, also called *block* or *one-block support*. We will employ a related concept for vectors in Chapters 5 and 6.

Remark 2.4 Since we denote the energetic frequencies by $\omega_1, \ldots, \omega_B$ in order to use the same notation as in [Iwe10, Bit17c, BZI19], we need to be able to distinguish the kth energetic frequency from the kth primitive root of unity. Consequently, in the few equations where we require to explicitly write down the Nth primitive root of unity in Chapter 2, we will denote it by $\boldsymbol{\omega}_N$ to avoid ambiguities. This will not be necessary in any other chapter of this thesis. ◊

In this and the following chapter, we will always consider the samples to be taken from the perturbed function $f + \eta$, where $\eta \in C_{2\pi}$ satisfies $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. Consequently, we are recovering a function which might be non-sparse in Fourier domain and could have an unstructured frequency support. However, if, for example, the nonzero Fourier coefficients of f all satisfy $|c_\omega(f)| > 2\varepsilon$, then the structured frequency sparsity of f guarantees that

$$\{\omega : |c_\omega(f + \eta)| > \varepsilon\} \cap R_B^{\text{opt}}(f + \eta) \subseteq S,$$

where $R_B^{\text{opt}}(f + \eta) \subsetneq \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}$ contains the indices of B entries of $\mathbf{c}(f + \eta)$ with largest magnitudes.

2.2 Methodical Background

Our aim in this chapter is to develop a fast Fourier algorithm that deterministically reconstructs a function $f \in C_{2\pi}$ from noisy samples if it is known that f has a short frequency support. In [Iwe10, Iwe13] a deterministic algorithm for the reconstruction of general unstructured B-sparse functions from samples was introduced, achieving a runtime that is sublinear in the assumed bandwidth N of f. Using our support set notation, general B-sparsity means that there are B support sets $S_1, \ldots, S_B$, each containing only one frequency,

$$S_j = \{\omega_j\} \subsetneq \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}.$$

Employing the stronger condition of a short frequency support, we will simplify the methods from [Iwe10, Iwe13] in two ways. Thus, we obtain two algorithms using significantly less samples which also have a much shorter runtime.

In order to be able to adapt the procedures from [Iwe10, Iwe13] to the setting of a short frequency support, we will first sketch the ideas of Algorithm 2 in [Iwe10] that are required for reconstructing a function $f \in C_{2\pi}$ with short frequency support of length at most B. Note that Algorithm 2 in [Iwe10] and Algorithm 3 in [Iwe13] are essentially the same method and are just written down using different notation. For now we will assume that all samples are known exactly, i.e., that $\eta \equiv 0$.

2.2.1 Reconstruction Procedure for One Frequency

We begin by supposing that the function f possesses only a single energetic frequency $\omega \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$ with corresponding Fourier coefficient $c_\omega \coloneqq c_\omega(f) \in \mathbb{C} \setminus \{0\}$. We will later extend the reconstruction procedure to several energetic frequencies. Then f is of the form

$$f(x) = c_\omega \cdot e^{i\omega x}.$$

Thus, f is completely determined if we can recover ω and c_ω. We will do this utilizing vectors $\mathbf{a}^M$ of equidistant samples of f, where we keep the slightly unintuitive notation of $\mathbf{a}^M$ used in [Iwe10, Iwe13, BZI19, Bit17c].

Definition 2.5 (Vector of Equidistant Samples) Let $f \in C_{2\pi}$ and $M \in \mathbb{N}$. By $\mathbf{a}^M \in \mathbb{C}^M$ we denote the vector of M equidistant samples of f, i.e.,

$$\mathbf{a}^M \coloneqq \left(a_j^M\right)_{j=0}^{M-1} \coloneqq \left(f\left(\frac{2\pi j}{M}\right)\right)_{j=0}^{M-1}.$$

Recall that we learned in Section 1.2.1 that we can recover f by applying the FFT to the vector $\mathbf{a}^N$ of N equidistant samples of f in $\mathcal{O}(N \log N)$ time. The central idea of Algorithm 2 in [Iwe10] is to obtain a method with shorter runtime than the FFT by applying the CDFT to several vectors of equidistant samples with lengths that are small compared to the bandwidth N. We choose the CDFT here so that we can consider functions with energetic frequencies contained in the interval $\left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$ centered around 0. In order for this idea to work, the vector lengths have to be chosen in a very specific way, which we will discuss later on. First, we will investigate what happens if we apply the CDFT to the vector $\mathbf{a}^M$ of $M \ll N$ equidistant samples of f. We find that

$$\begin{aligned}
\widehat{a^M}_\nu &= \frac{1}{M} \sum_{j=0}^{M-1} e^{\frac{-2\pi i j \nu}{M}} \cdot c_\omega e^{\frac{2\pi i j \omega}{M}} \\
&= \frac{c_\omega}{M} \sum_{j=0}^{M-1} {\boldsymbol{\omega}_M}^{j(\nu-\omega)} \\
&= \begin{cases} c_\omega & \text{if } \nu \equiv \omega \mod M, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall\, \nu \in \left\{-\left\lceil \frac{M}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{M}{2} \right\rfloor\right\}. \qquad (2.1)
\end{aligned}$$

Hence, $\widehat{\mathbf{a}^M}$ has exactly one nonzero entry, namely the Fourier coefficient corresponding to ω, and its index is the residue of ω modulo M,

$$\widehat{a^M}_{\omega \bmod M} = c_\omega = \frac{1}{2\pi} \int_0^{2\pi} c_\omega e^{i\omega x} \cdot e^{-i\omega x} \mathrm{d}x.$$

Consequently, (2.1) gives us the value of the Fourier coefficient c_ω and the residue of the frequency ω modulo M while actually knowing neither c_ω nor ω. If we compute the CDFTs of several such vectors with different lengths, we obtain a system of simultaneous congruencies. Under certain conditions on the occurring moduli, i.e., the lengths of the vectors of equidistant samples, such a system can be solved via the well-known Chinese Remainder Theorem. For a proof and more details, see [Lan05], Chapter I, §4.

Theorem 2.6 (Chinese Remainder Theorem (CRT)) *Let $M_1, \ldots, M_L$ be pairwise relatively prime integers and $N \leq \prod_{l=1}^{L} M_l$. Further, let $r_l \in \left\{ -\left\lceil \frac{M_l}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{M_l}{2} \right\rfloor \right\}$ for all $l \in \{1, \ldots, L\}$. Then there exists a unique solution modulo N of the system of simultaneous congruencies*

$$\begin{aligned} x &\equiv r_1 \mod M_1, \\ &\vdots \\ x &\equiv r_L \mod M_L. \end{aligned}$$

The unique solution of such a system can be computed using the following algorithm adapted from the implementation `ChineseRem` in GAP, an open source system for computational discrete algebra (see [The18a]).

Algorithm 1 CRT Reconstruction Algorithm

Input: Residues $r_1, \ldots, r_L$ modulo pairwise relatively prime integers $M_1, \ldots, M_L$.
Output: Integer $\nu \in \{0, \ldots, M-1\}$ such that $\nu \equiv r_l \mod M_l \ \forall l \in \{1, \ldots, L\}$ and $M \geq \prod_{l=1}^{L} M_l$.

1: Set $l = 1$, $\nu = r_1$ and $M = M_1$.
2: **while** $l < L$ **do**
3: Set $l = l + 1$.
4: $(g, u, v) \leftarrow$ `extended_gcd`(M, M_l)
5: **if** $g \neq 1$ and $r_l - \nu \mod g \neq 0$ **then**
6: Error ▷ The residues have to be equal modulo g.
7: **end if**
8: $\nu = M \left(\left(\frac{r_l - \nu}{g} \cdot u \right) \mod M_l \right) + \nu$
9: $M = \frac{M_l}{g} \cdot M$
10: **end while**
11: $\nu = \nu \mod M$

Output: ν, M.

The function `extended_gcd` in line 4 of Algorithm 1 computes the greatest common divisor g of two integers a and b using the extended Euclidean algorithm, as well as two integers u and v satisfying Bézout's identity, see [Bos06], Chapter 2.4, Theorem 15.

Theorem 2.7 (Euclidean Algorithm) *Let R be a Euclidean ring. For two elements $a, b \in R \setminus \{0\}$ consider the sequence $z_0, z_1, \ldots \in R$ which is given inductively by*

$$\begin{aligned} z_0 &= a, \\ z_1 &= b, \\ z_{k+1} &= \begin{cases} z_{k-1} \mod z_k & \text{if } z_k \neq 0, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Then there exists a smallest index $n \in \mathbb{N}$ such that $z_{n+1} = 0$. It satisfies $z_n = \gcd(a, b)$. Furthermore, there exists an explicit representation of the greatest common divisor of a and b in the form

$$g = \gcd(a, b) = ua + vb$$

for some $u, v \in R$, which is known as Bézout's identity.

It can be shown that for $a > b$ the extended Euclidean algorithm has a runtime of $\mathcal{O}(\log b)$; see, e.g., [CLRS09], Chapter 31.2. Thus, Algorithm 1 requires $\mathcal{O}\left(\sum_{l=1}^{L} \log M_l\right)$ arithmetical operations, as we will show in the proof of Theorem 2.22.

If $\widehat{\mathbf{a}^M}$ is known for sufficiently many pairwise relatively prime moduli $M_l \ll N$, $l \in \{1, \ldots, L\}$, with $N \leq \prod_{l=1}^{L} M_l$, the CRT implies that we can uniquely recover the energetic frequency ω. The corresponding Fourier coefficient c_ω is already given by (2.1),

$$\widehat{a^{M_l}}_{\omega \bmod M_l} = c_\omega \qquad \forall\, l \in \{1, \ldots, L\}.$$

This means that, instead of computing the CDFT of length N of $\widehat{\mathbf{a}^N}$, it suffices to calculate L CDFTs of length $M_1, \ldots, M_L \ll N$ and reconstruct ω from its residues modulo $M_1, \ldots, M_L$.

Let us illustrate the reconstruction procedure from Algorithm 1 by an example.

Example 2.8 Let $f \in C_{2\pi}$, $f(x) = e^{i \cdot 210x}$ with bandwidth $N = 1{,}000$. According to the CRT, the single energetic frequency ω is uniquely determined modulo N by its residues modulo $M_1 = 10, M_2 = 11$ and $M_3 = 13$, as M_1, M_2 and M_3 are pairwise relatively prime and their product is 1,430. Locating the nonzero entry of $\widehat{\mathbf{a}^{10}}, \widehat{\mathbf{a}^{11}}$ and $\widehat{\mathbf{a}^{13}}$, we find

$$\begin{aligned}
\widehat{a^{10}}_0 = 1 \quad &\Rightarrow \quad \omega \equiv 0 \bmod 10 \quad \Rightarrow \quad r_1 \coloneqq 0, \\
\widehat{a^{11}}_1 = 1 \quad &\Rightarrow \quad \omega \equiv 1 \bmod 11 \quad \Rightarrow \quad r_2 \coloneqq 1 \quad \text{and} \\
\widehat{a^{13}}_2 = 1 \quad &\Rightarrow \quad \omega \equiv 2 \bmod 13 \quad \Rightarrow \quad r_3 \coloneqq 2.
\end{aligned}$$

Now we can recover ω from its residues r_1, r_2 and r_3 using Algorithm 1. We begin by setting $M \coloneqq M_1 = 10$ and $\omega \coloneqq r_1 = 0$. Then, with

$$\gcd(M, M_2) = 1 = -1 \cdot 10 + 1 \cdot 11,$$

it follows that $u = -1$, and thus obtain

$$\begin{aligned}
\omega &\coloneqq M \cdot (((r_2 - \omega) \cdot u) \bmod M_2) + \omega \\
&= 10 \cdot (((1 - 0) \cdot (-1)) \bmod 11) + 0 \\
&= 10 \cdot 10 \\
&= 100.
\end{aligned}$$

The frequency $\omega = 100$ satisfies that $\omega \equiv 0 \bmod 10$ and $\omega \equiv 1 \bmod 11$. Now we update $M \coloneqq M_2 \cdot M = 110$. Since

$$\gcd(M, M_3) = 1 = -2 \cdot 110 + 17 \cdot 13,$$

we have that $u = -2$, so we redefine

$$\begin{aligned}
\omega &\coloneqq 110 \cdot (((2 - 100) \cdot (-2)) \bmod 13) + 100 \\
&= 110 \cdot (196 \bmod 13) + 100 \\
&= 210.
\end{aligned}$$

Finally, we set $M \coloneqq M_3 \cdot M = 1{,}430$. As $\omega \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\} = \{-499, \ldots, 500\}$, the output of Algorithm 1 is $\omega = 210$ and $M = 1{,}430$. The obtained frequency ω indeed satisfies the required congruencies $\omega \equiv 0 \bmod 10$, $\omega \equiv 1 \bmod 11$ and $\omega \equiv 2 \bmod 13$.

The Fourier coefficient c_ω is given by (2.1) as, e.g.,

$$c_\omega = \widehat{a^{10}}_0 = 1.$$

For the above computation three CDFTs of lengths 10, 11 and 13 were necessary; hence, only 34 instead of $N = 1{,}000$ samples of f were used. Moreover, as we discussed in Section 1.1.1, there exist fast algorithms for the DFT of vectors of arbitrary length, and, by Remark 1.9, also for the CDFT, so the three CDFTs can be computed in $\mathcal{O}\left(\sum_{l=1}^{3} M_l \log M_l\right)$ time, instead of calculating one CDFT with complexity $\mathcal{O}(N \log N)$. We will show in the proof of Theorem 2.22 that the frequency reconstruction needs $\mathcal{O}\left(\sum_{l=1}^{3} \log M_l\right)$ arithmetic operations, which is insignificant compared to the computational costs of the CDFTs. ◊

However, as soon as the function we aim to recover has more than one energetic frequency, the residues of these frequencies can coincide modulo various integers. If we choose the moduli M_l arbitrarily, it can happen that $\omega_1 \equiv \omega_2 \mod M_1$, so it is impossible to distinguish these two frequencies modulo M_1. Furthermore, we cannot determine the values of the Fourier coefficients from $\widehat{\mathbf{a}^{M_1}}$. This means that in the case of a B-sparse function, we have to choose the moduli M_l carefully in order to avoid ambiguities. Without a priori knowledge of the energetic frequencies, guaranteeing unique recoverability requires a more involved reconstruction procedure.

2.2.2 Reconstruction Procedure for Several Frequencies

Let us now examine a 2π-periodic function $f \in C_{2\pi}$ with B distinct energetic frequencies $\omega_1, \ldots, \omega_B \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$ and their Fourier coefficients $c_{\omega_1}, \ldots, c_{\omega_B} \in \mathbb{C} \setminus \{0\}$,

$$f(x) = \sum_{k=1}^{B} c_{\omega_k} \cdot e^{i\omega_k x}.$$

Note that for an arbitrary $s \in \mathbb{N}$ the CDFT of the vector $\mathbf{a}^s$ of s equidistant samples of f given by Definition 2.5 satisfies

$$\begin{aligned} \widehat{a^s}_\nu &= \frac{1}{s} \sum_{j=0}^{s-1} e^{\frac{-2\pi i j \nu}{s}} \sum_{k=1}^{B} c_{\omega_k} e^{\frac{2\pi i j \omega_k}{s}} \\ &= \frac{1}{s} \sum_{k=1}^{B} c_{\omega_k} \sum_{j=0}^{s-1} {\boldsymbol{\omega}_s}^{j(\nu-\omega_k)} \\ &= \sum_{\substack{k=1 \\ \omega_k \equiv \nu \bmod s}}^{B} c_{\omega_k} \end{aligned} \tag{2.2}$$

for all $\nu \in \left\{-\left\lceil \frac{s}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{s}{2} \right\rfloor\right\}$, see also Remark 1.16. Since the B energetic frequencies are distinct, there exists an $s \in \mathbb{N}$ such that their residues modulo s do not coincide. This motivated the use of the notion of separation in [Iwe10].

Definition 2.9 (Separation) Let $s, B \in \mathbb{N}$ and $\omega_1, \ldots, \omega_B \in \mathbb{Z}$ be distinct. Then s *separates* the integers $\omega_1, \ldots, \omega_B$ if

$$\omega_k \bmod s \neq \omega_l \bmod s \quad \forall\, k, l \in \{1, \ldots, B\},\ k \neq l.$$

It is intuitively clear that we need the chosen moduli M_l to separate all energetic frequencies. We assume for the moment that such a separating s is known. If we apply the CDFT to $\mathbf{a}^s$, (2.2) yields that

$$\widehat{a^s}_\nu = \begin{cases} c_{\omega_1} & \text{if } \nu \equiv \omega_1 \bmod s, \\ \vdots & \vdots \\ c_{\omega_B} & \text{if } \nu \equiv \omega_B \bmod s, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall\, \nu \in \left\{ -\left\lceil \frac{s}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{s}{2} \right\rfloor \right\}. \tag{2.3}$$

Hence, $\widehat{\mathbf{a}^s}$ has exactly B nonzero entries and their indices are the residues of the energetic frequencies ω_k modulo s, as the frequencies' residues cannot coincide due to the separation property of s. However, this is still not sufficient for unique recovery of the energetic frequencies and their corresponding Fourier coefficients. If the Fourier coefficients of some of the frequencies are equal, (2.3) yields that it is impossible to directly match their residues modulo any separating integer uniquely to the frequencies.

In order to be able to apply the CRT reconstruction from Algorithm 1 for finding $\omega_1, \ldots, \omega_B$, we have to choose the moduli M_l in a way that allows us to determine for each frequency the residue modulo M_l just from entries of the vector $\widehat{\mathbf{a}^{M_l}}$ for all $l \in \{1, \ldots, L\}$. We define the required moduli using that for all $p \in \mathbb{N}$ and $a, b \in \mathbb{Z}$ the following holds,

$$\begin{aligned} & (a \equiv b \bmod ps \quad \Rightarrow \quad a \equiv b \bmod s) \\ \Leftrightarrow\ & (a \not\equiv b \bmod ps \quad \Leftarrow \quad a \not\equiv b \bmod s). \end{aligned} \tag{2.4}$$

Consequently, if s separates the frequencies $\omega_1, \ldots, \omega_B$, so does ps for all $p \in \mathbb{N}$, which means that we can generate infinitely many separating natural numbers if one such number is known. Then, as in (2.3), we obtain that the residues of all energetic frequencies modulo ps can be obtained from $\widehat{\mathbf{a}^{ps}}$. However, for several distinct values of p, the numbers ps are of course not pairwise relatively prime anymore, so we cannot apply Algorithm 1 directly to the residues modulo ps. Instead, we will use the residues modulo the p, choosing finitely many such that the prerequisites of the CRT are satisfied for them and s. If we tried to obtain the residues modulo p directly from $\widehat{\mathbf{a}^p}$, we would still have the problem of uniquely matching the energetic frequencies to the residues. Computing the residues modulo p from $\widehat{\mathbf{a}^{ps}}$ instead solves this problem. Hence, the moduli we use for the CRT reconstruction are not the same as the lengths of the vectors of equidistant samples of f.

The residue of any integer a modulo p can be computed from its residue modulo ps by

$$a \bmod p = (a \bmod ps) \bmod p \qquad \forall\, a \in \mathbb{Z}. \tag{2.5}$$

The simplest way to ensure the prerequisites of the CRT is to take the L smallest prime numbers $t_1, \ldots, t_L$ that are relatively prime to s such that

$$s \cdot \prod_{l=1}^{L-1} t_l < N \leq s \cdot \prod_{l=1}^{L} t_l, \qquad \gcd(t_l, s) = 1 \quad \forall\, l \in \{1, \ldots, L\}.$$

We show now how to correctly find the residues of the unknown frequencies modulo all the $t_l s$. Let us fix an arbitrary residue r_0 modulo s such that $\widehat{a^s}_{r_0} \neq 0$. By ω we denote its still unknown corresponding energetic frequency. In order to simplify the notation, we just consider an arbitrary prime p, but the same procedure works for all primes t_l. How can we find the residue of ω modulo p? From (2.3) we know that

$$\widehat{a^{ps}}_{\omega \bmod ps} = c_\omega = \widehat{a^s}_{\omega \bmod s},$$

so the residue of ω modulo ps can be found by comparing it to its residue modulo s. Since $\omega \equiv r_0 \mod s$, it is of the form

$$\omega = r_0 + a \cdot s$$

for an $a \in \mathbb{Z}$, and its residue modulo ps satisfies

$$\omega \bmod ps = (r_0 + as) \bmod ps = r_0 + (a \bmod p) \cdot s =: r_0 + b_{\min} \cdot s \tag{2.6}$$

for $b_{\min} := a \bmod p \in \left\{-\left\lceil \frac{p}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{p}{2} \right\rfloor\right\}$. We do not know a yet, but by (2.6) we only have to check p possible values in order to find the correct residue of the frequency ω modulo ps, instead of checking ps possibilities. Recall that due to the separation property of s we have that

$$r_0 = \omega \bmod s \neq \omega_k \bmod s \quad \forall\, k \in \{1, \ldots, B\} \text{ with } \omega \neq \omega_k.$$

Then

$$(r_0 + bs) \bmod s \neq \omega_k \bmod s \quad \forall\, k \in \{1, \ldots, B\} \text{ with } \omega \neq \omega_k,$$

and therefore (2.4) yields that

$$(r_0 + bs) \bmod ps \neq \omega_k \bmod ps \quad \forall\, k \in \{1, \ldots, B\} \text{ with } \omega \neq \omega_k \tag{2.7}$$

for all $b \in \left\{-\left\lceil \frac{p}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{p}{2} \right\rfloor\right\}$. Consequently, none of the p possible values $r_0 + bs$ for the residue of ω modulo ps from (2.6) can coincide with the residue of another energetic frequency $\omega_k \neq \omega$ modulo ps. Therefore, we cannot match a wrong energetic residue to ω if we restrict ourselves to the p possible residues from (2.6). Exactly one of the p values $\widehat{a^{ps}}_{r_0+bs}$, where $b \in \left\{-\left\lceil \frac{p}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{p}{2} \right\rfloor\right\}$, is not zero but equal to $\widehat{a^s}_{r_0} = c_\omega$. Hence, we can determine $\omega \bmod ps$ by comparing $\widehat{a^s}_{r_0}$ and $\widehat{a^{ps}}_{r_0+bs}$ for all possible values of b, i.e.,

$$\begin{aligned} & \omega \bmod ps = r_0 + b_{\min} \cdot s \\ \Leftrightarrow \quad & \left| \widehat{a^s}_{r_0} - \widehat{a^{ps}}_{r_0 + b_{\min} \cdot s} \right| = \min_{b \in \left\{-\left\lceil \frac{p}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{p}{2} \right\rfloor\right\}} \left| \widehat{a^s}_{r_0} - \widehat{a^{ps}}_{r_0+bs} \right|. \end{aligned} \tag{2.8}$$

Having found the residue of ω modulo ps from (2.8), its residue modulo p can be calculated with the help of (2.5) via

$$\omega \bmod p = (\omega \bmod ps) \bmod p. \tag{2.9}$$

Recall that this procedure can be used to determine the residues of ω modulo t_l for all $l \in \{1, \ldots, L\}$. After computing the residues modulo all t_l, we can uniquely reconstruct ω from its residues modulo $s, t_1, \ldots, t_L$ using Algorithm 1. Since s separates all occurring

frequencies, the Fourier coefficient c_ω is given by (2.3) as

$$c_\omega = \widehat{a^s}_{\omega \bmod s}. \tag{2.10}$$

The remaining frequencies and their coefficients can be found analogously. Due to the separation property of s, all residues are matched to the right frequency and thus all energetic frequencies and their corresponding Fourier coefficients are found correctly.

We also demonstrate the reconstruction procedure for 2π-periodic functions with several energetic frequencies by an example.

Example 2.10 Let $f \in C_{2\pi}$, $f(x) = e^{-i\cdot 105x} - e^{i\cdot 42x} + e^{i\cdot 210x}$ with bandwidth $N = 1{,}000$ and assume that it is known a priori that $s = 10$ separates the frequencies $\omega_1 = -105$, $\omega_2 = 42$ and $\omega_3 = 210$. Indeed, we have that

$$\omega_1 \equiv 5 \bmod s, \qquad \omega_2 \equiv 2 \bmod s \qquad \text{and} \qquad \omega_3 \equiv 0 \bmod s.$$

Computing the CDFT of the vector $\mathbf{a}^{10}$ of 10 equidistant samples of f, we obtain

$$\widehat{\mathbf{a}^{10}} = (0, 0, 0, 0, 1, 0, -1, 0, 0, 1)^T.$$

This vector already shows that we cannot uniquely match the frequencies ω_1 and ω_3 to their residues modulo 10, since $\widehat{a^{10}}_0 = \widehat{a^{10}}_5 = 1 = c_{\omega_1} = c_{\omega_3}$. Instead, we choose an arbitrary residue r_0 modulo 10 such that $\widehat{a^{10}}_{r_0} \neq 0$, e.g., $r_0 = 5$ with $\widehat{a^{10}}_{r_0} = 1$. Now we reconstruct the frequency corresponding to this residue modulo s. In practice the frequencies are not known a priori, so let us just denote the frequency corresponding to the residue $r_0 = 5$ modulo s by ω, as we cannot tell yet that it is ω_1.

In order to ensure the prerequisites of the CRT, we set $t_1 := 3$, $t_2 := 7$ and $t_3 := 11$, since $3, 7, 11$ are relatively prime to $s = 10$, and $10 \cdot 3 \cdot 7 = 210$ and $10 \cdot 3 \cdot 7 \cdot 11 = 2{,}310$. It follows from (2.6) that the t_l possible residues of ω modulo $10 \cdot t_l$ satisfy

$$\omega \bmod (10 \cdot t_l) \in \left\{ r_0 + b \cdot 10 : b \in \left\{ -\left\lceil \frac{t_l}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{t_l}{2} \right\rfloor \right\} \right\}$$

for $l \in \{1, 2, 3\}$. For $t_1 = 3$ this yields

$$\omega \bmod 30 \in \{-5, 5, 15\}.$$

Since

$$\widehat{a^{30}}_{-5} = \widehat{a^{30}}_5 = 0 \qquad \text{and} \qquad \widehat{a^{30}}_{15} = 1,$$

using (2.8) we find that $b_{\min} = 1$, so we have

$$\omega \bmod 30 = r_0 + b_{\min} \cdot s = 15.$$

By (2.9) we obtain for the residue of ω modulo 3 that

$$\omega \bmod 3 = (\omega \bmod 30) \bmod 3 = 0.$$

Analogously, we find for $t_2 = 7$ that

$$\omega \bmod 70 \in \{-25, -15, -5, 5, 15, 25, 35\}.$$

It follows from

$$\widehat{a^{70}}_{-25} = \cdots = \widehat{a^{70}}_{25} = 0 \qquad \text{and} \qquad \widehat{a^{70}}_{35} = 1$$

that $b_{\min} = 3$, so

$$\omega \mod 70 = r_0 + b_{\min} \cdot s = 35.$$

Consequently, we have that

$$\omega \mod 7 = (\omega \mod 70) \mod 7 = 0.$$

For $t_3 = 11$ the residue of ω modulo 110 has to satisfy that

$$\omega \mod 110 \in \{-45, -35, -25, -15, -5, 5, 15, 25, 35, 45, 55\}.$$

Since

$$\widehat{a^{110}}_{-45} = \cdots = \widehat{a^{110}}_{-5} = \widehat{a^{110}}_{15} = \cdots = \widehat{a^{110}}_{55} = 0 \qquad \text{and} \qquad \widehat{a^{110}}_{5} = 1,$$

we find that $b_{\min} = 0$, and thus

$$\omega \mod 110 = r_0 + b_{\min} \cdot s = 5.$$

Hence, we obtain for the residue of ω modulo 11 that

$$\omega \mod 11 = (\omega \mod 110) \mod 11 = 5.$$

Now we can reconstruct the frequency ω from its residues $\omega \equiv 5 \mod 10$, $\omega \equiv 0 \mod 3$, $\omega \equiv 0 \mod 7$ and $\omega \equiv 5 \mod 11$ via the CRT procedure from Algorithm 1, which yields

$$\omega = -105 \in \{-499, \ldots, 500\}.$$

By (2.3), the corresponding Fourier coefficient is given as

$$c_\omega = \widehat{a^{10}}_{-105 \mod 10} = \widehat{a^{10}}_5 = 1.$$

Thus, we have correctly recovered the frequency $\omega_1 = -105$ and its Fourier coefficient $c_{\omega_1} = 1$, even though ω_1 is not the only frequency with Fourier coefficient 1. The two remaining frequencies ω_2 and ω_3 can be found analogously, starting with the other two significantly large entries of $\widehat{\mathbf{a}^{10}}$. As a last step let us compare the residues of the three energetic frequencies modulo st_l. We have that

$$\begin{aligned}
&\omega_2 \equiv 12 \mod 30 && \text{and} && \omega_3 \equiv 0 \mod 30,\\
&\omega_2 \equiv -28 \mod 70 && \text{and} && \omega_3 \equiv 0 \mod 70,\\
&\omega_2 \equiv 42 \mod 110 && \text{and} && \omega_3 \equiv -10 \mod 110.
\end{aligned}$$

Since we know from (2.6) that

$$\begin{aligned}
&\omega_1 \mod 30 \in \{-5, 5, 10\},\\
&\omega_1 \mod 70 \in \{-25, -15, \ldots, 35\},\\
&\omega_1 \mod 110 \in \{-45, -35, \ldots, 55\},
\end{aligned}$$

we can discern that for a fixed $l \in \{1, 2, 3\}$ indeed none of the t_l possible values for the residue of ω_1 modulo st_l collides with the residue of another energetic frequency modulo

st_l, thus illustrating what we already proved in (2.7).

In order to recover the function f with this method, we essentially calculated four CDFTs of lengths $10, 30, 70$ and 110, which together require 220 samples of f instead of the $N = 1{,}000$ samples needed by the single CDFT of length N for finding the Fourier coefficients from $\widehat{\mathbf{a}^N}$ as in Section 1.2.1. The shorter CDFTs can be computed in $\mathcal{O}\left(\sum_{l=0}^{3} st_l \log(st_l)\right)$ time instead of $\mathcal{O}(N \log N)$ time for the N-length CDFT. As we remarked in Example 2.8, the effort of the frequency reconstruction via Algorithm 1 is insignificant compared to the computational effort of the CDFTs. $\lozenge$

So far, we just assumed that a natural number s separating all energetic frequencies is known. However, for arbitrary at most B-sparse functions with unknown frequencies, as in [Iwe10,Iwe13], guaranteeing that any s separates all energetic frequencies is impossible. With some combinatorial constructions it could be shown in [Iwe10] that for a suitable $\widetilde{K}$ depending on B and N at least more than half of $\widetilde{K}$ integers $\tilde{s}_1, \ldots, \tilde{s}_{\widetilde{K}}$, satisfying certain additional properties, separate all energetic frequencies. Applying median techniques then yields the correct frequencies and coefficient estimates. In Chapter 3.2, we will go into more detail about these aspects of the algorithm. For now, we just note that, for a function with B-sparse frequency support and bandwidth N, Algorithm 2 in [Iwe10] uses the $\tilde{L}$ smallest primes $\tilde{t}_1, \ldots, \tilde{t}_{\tilde{L}}$ such that

$$\prod_{l=1}^{\tilde{L}-1} \tilde{t}_l < \frac{N}{B} \leq \prod_{l=1}^{\tilde{L}} \tilde{t}_l.$$

Then $\tilde{s}_1$ is chosen as the smallest prime that is greater than both B and $\tilde{t}_{\tilde{L}}$. Setting $\widetilde{K} \coloneqq 8B \left\lfloor \log_{\tilde{s}_1} N \right\rfloor + 1$, one additionally chooses $\tilde{s}_2, \ldots, \tilde{s}_{\widetilde{K}}$ as the $\widetilde{K} - 1$ smallest primes that are greater than $\tilde{s}_1$. Then the algorithm requires the sampling vectors $\mathbf{a}^{\tilde{s}_k \tilde{t}_l}$ and computes the $\widetilde{K} \cdot \left(\tilde{L} + 1\right)$ CDFTs $\widehat{\mathbf{a}^{\tilde{s}_k \tilde{t}_l}}$ for all $k \in \left\{1, \ldots, \widetilde{K}\right\}$ and $l \in \left\{0, \ldots, \tilde{L}\right\}$.

Employing some combinatorial constructions, the algorithm returns the B most energetic frequencies and accurate estimates for their corresponding Fourier coefficients if the input data is noisy. Note that the sparsity of the function we aim to recover does not have to be known a priori; a good upper bound B on it suffices. Both Algorithm 2 in [Iwe10] and Algorithm 3 in [Iwe13] have a runtime of

$$\mathcal{O}\left(\frac{B^2}{\log^2 B} \cdot \frac{\log^4 N \log^2(B \log N)}{\log \log \frac{N}{B}}\right)$$

and require

$$\mathcal{O}\left(\frac{B^2}{\log^2 B} \cdot \frac{\log^4 N \log(B \log N)}{\log \log \frac{N}{B}}\right)$$

samples.

Remark 2.11 Note that, as the distance between increasing prime numbers is often rather large, it might be possible to omit some of the smaller prime numbers $\tilde{t}_l$ while still satisfying that their product is greater than $\frac{N}{B}$. For example, if $N = 1{,}000$ and $B = 3$, choosing the $\tilde{t}_l$ as above would yield $\tilde{t}_1 = 2$, $\tilde{t}_2 = 3$, $\tilde{t}_3 = 5$, $\tilde{t}_4 = 7$ and $\tilde{t}_5 = 11$, since

$$2 \cdot 3 \cdot 5 \cdot 7 = 210 < \frac{N}{B} \leq 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 2{,}310.$$

Then one could actually omit $\tilde{t}_1 = 2$ and thus work with fewer vectors, since

$$3 \cdot 5 \cdot 7 = 105 < \frac{N}{B} \leq 3 \cdot 5 \cdot 7 \cdot 11 = 1{,}155,$$

so $\tilde{t}_2 = 3, \tilde{t}_3 = 5, \tilde{t}_4 = 7$ and $\tilde{t}_5 = 11$ already satisfy the requirements of the CRT. This does not affect the order of the theoretical runtime and sampling complexities, but in practice it slightly improves the performance of the algorithm. Being closely related to the frequency reconstruction approach sketched above, similar improvements are possible for Algorithms 2, 3, 4, 5 and 6 in this thesis and could thus be incorporated into future improved implementations. ◊

2.3 Sparse FFT for Functions with Short Frequency Support I

Before investigating frequency supports with more complex structures in Chapter 3, we will focus on functions with short frequency support for the remainder of this chapter, which is based directly on [Bit17c] and is in parts identical with the representations therein. In this case, the reconstruction approach from [Iwe10, Iwe13], which we sketched in Section 2.2.2, can be simplified. We now consider functions $f \in C_{2\pi}$ of the form

$$f(x) = \sum_{k=0}^{B-1} c_{\omega_1+k} \cdot e^{i(\omega_1+k)x},$$

such that all energetic frequencies are contained in the B-length support interval

$$S \coloneqq \{\omega_1, \ldots, \omega_1 + B - 1\} \subsetneq \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}.$$

Consequently, B already separates the B consecutive energetic frequencies. In the procedure outlined in Section 2.2.2 the moduli we used were a separating s and the L smallest primes t_l that do not divide B such that

$$N \leq B \cdot \prod_{l=1}^{L} t_l.$$

To simplify the estimation of the runtime and sample bounds, and to avoid collision with the other moduli, we set s as the smallest power of 2 that is greater than B,

$$s := 2^\alpha, \quad \text{where} \quad \alpha := \lfloor \log_2 B \rfloor + 1.$$

Then $t_1, \ldots, t_L$ can be chosen as the L smallest odd primes satisfying

$$B \cdot \prod_{l=1}^{L-1} t_l < N \leq B \cdot \prod_{l=1}^{L} t_l, \tag{2.11}$$

and, for inductive purposes, we set $t_0 \coloneqq 1$. As s is greater than B, it still separates the B energetic frequencies $\omega_1, \omega_1 + 1, \ldots, \omega_1 + B - 1$. Furthermore, s is relatively prime to all small odd primes $t_1, \ldots, t_L$, so we can indeed uniquely recover the frequencies from their residues modulo $s, t_1, \ldots, t_L$ with the help of Algorithm 1.

Remark 2.12 Using the reconstruction procedure with s is just applying Algorithm 2 in [Iwe10] to the first element of a B-majority selective collection of sets, $\mathscr{S}$, whose elements do not have to be primes (see [Iwe10], Section 3). The combinatorial considerations necessary for the general case of B energetic frequencies are rendered redundant by the fact that for functions with short frequency support of length B we always find an $s = 2^\alpha > B$ that separates all energetic frequencies, so, unlike in [Iwe10], we do not require s to separate any B-element subset of $\{-\lceil \frac{N}{2} \rceil + 1, \ldots, \lfloor \frac{N}{2} \rfloor\}$. ◊

Due to the block structure of the energetic frequencies, it suffices to perform the CRT reconstruction procedure for a single energetic frequency $\tilde{\omega}$ and find the remaining ones by examining whether the absolute values of the Fourier coefficients corresponding to the $2B-1$ frequencies in $\{\tilde{\omega}-B+1, \tilde{\omega}-B+2, \ldots, \tilde{\omega}+B-1\}$ are significantly large. All of the at most B energetic frequencies have to be contained in this set, as the distance of any energetic frequency ω to $\tilde{\omega}$ can be at most $B-1$. We can find such a frequency $\tilde{\omega}$ by reconstructing it via Algorithm 1 from the index of the largest magnitude entry of $\widehat{\mathbf{a}^s}$ and the corresponding residues modulo the smallest odd primes $t_1, \ldots, t_L$, as the indices of the significantly large entries of $\widehat{\mathbf{a}^s}$ are precisely the residues of the energetic frequencies modulo s.

The Fourier coefficients are then given without any further computation, since, by (2.3), they are just the significantly large entries of $\widehat{\mathbf{a}^{st_l}}$ for any l. The $2B-1$ possibly energetic, consecutive frequencies $\tilde{\omega}-B+1, \ldots, \tilde{\omega}+B-1$ cannot be distinct modulo s, but they are separated by $st_1 = 3s$. For exact data their Fourier coefficients are given by

$$c_\omega = \widehat{a^{3s}}_{\omega \bmod 3s}$$

for all $\omega \in \{\tilde{\omega}-B+1, \ldots, \tilde{\omega}+B-1\}$. Note that it is not necessary here to know the block length exactly; it suffices that an upper bound B on it is known. If B is not the exact block length, some of the reconstructed Fourier coefficients will just be zero or, in the case of noisy data, insignificantly small.

Remark 2.13 It was recently pointed out that, choosing $s > B$, one can even detect the remaining at most $B-1$ energetic frequencies by looking at the nonzero entries of $\widehat{\mathbf{a}^s}$, since one energetic frequency $\tilde{\omega}$ is already known and f has a short frequency support of length at most B. Then, using $\tilde{\omega}$, the energetic frequencies can be uniquely found from their residues modulo s. Consequently, the method presented hereafter can be further improved. The orders of theoretical runtime and sampling complexities of both approaches are not affected by this, as $2B = \mathcal{O}(B)$. ◊

Before presenting the detailed algorithm, we illustrate the procedure sketched above by an example.

Example 2.14 Let $f \in C_{2\pi}$, $f(x) = e^{i\cdot 210x} - e^{i\cdot 211x} + 2e^{i\cdot 212x} - e^{i\cdot 213x} - 2e^{i\cdot 214x}$ with bandwidth $N = 1{,}000$ and block length 5. Let us assume that we only know the upper bound $B = 6$ on the true block length a priori. Then we set

$$s := 2^{\lfloor \log_2 B \rfloor + 1} = 2^3 = 8.$$

Since $B \cdot 3 \cdot 5 \cdot 7 = 630$ and $B \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 6{,}930$, we set $t_1 := 3$, $t_2 := 5$, $t_3 := 7$ and $t_4 := 11$. Computing the CDFT of $\widehat{\mathbf{a}^8}$, we find that

$$\widehat{\mathbf{a}^8} = (-1, -2, 0, 0, 0, 1, -1, 2)^T.$$

We begin by recovering the frequency corresponding to the residue modulo $s = 8$ given by the index of one of the largest magnitude entries of $\widehat{\mathbf{a}^8}$, i.e., to -2 or 4 modulo 8. We choose the residue $r_0 := 4 \bmod 8$ and denote the corresponding frequency by $\tilde{\omega}$. Applying the reconstruction procedure for several frequencies detailed in Section 2.2.2 to $\mathbf{a}^{24}, \mathbf{a}^{40}, \mathbf{a}^{56}$ and $\mathbf{a}^{88}$, we obtain that

$$\tilde{\omega} \equiv 2 \bmod 3, \quad \tilde{\omega} \equiv 2 \bmod 5, \quad \tilde{\omega} \equiv 2 \bmod 7 \quad \text{and} \quad \tilde{\omega} \equiv 3 \bmod 11.$$

Then Algorithm 1 yields that $\tilde{\omega} = 212$. Since $\tilde{\omega}$ is an arbitrary energetic frequency, we now have to determine the remaining energetic frequencies by examining the Fourier coefficients of the $2B - 1 = 11$ frequencies contained in the set $S = \{207, 208, \ldots, 217\}$. Of course, these frequencies are not separated by $s = 8$ anymore; however, $st_1 = 24$ does separate them. It follows that

$$\widehat{\mathbf{a}^{24}} = (0, 0, \underbrace{0}_{=c_{207}}, 0, 0, \underbrace{1}_{=c_{210}}, -1, 2, -1, \underbrace{-2}_{=c_{214}}, 0, 0, \underbrace{0}_{=c_{217}}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T.$$

Thus, we find that there are 5 energetic frequencies, $210, 211, 212, 213$ and 214, with corresponding coefficients $c_{210} = 1, c_{211} = -1, c_{212} = 2, c_{213} = -1$ and $c_{214} = -2$, which means that we have completely recovered the function f. This approach requires five CDFTs of lengths $8, 24, 40, 56$ and 88, with a total runtime of $\mathcal{O}\left(\sum_{l=0}^{4} st_l \log(st_l)\right)$ and a sampling complexity of 216, instead of a single N-length CDFT with a runtime of $\mathcal{O}(N \log N)$ and $N = 1{,}000$ samples.

$\Diamond$

Remark 2.15 In Example 2.14, similar to the comment made in Remark 2.11, the runtime of the procedure could be reduced by omitting one of the primes $t_1 = 3$ or $t_2 = 5$, since the requirements of the CRT are then still satisfied. Furthermore, we could actually have restricted our search of the remaining energetic frequencies to the frequencies in $S = \{207, 208, \ldots, 217\}$ whose residues modulo 8 correspond to nonzero entries of $\widehat{\mathbf{a}^8}$, as no other frequencies can have significantly large Fourier coefficients. This would have decreased the runtime of the method slightly. However, not realizing this was possible at that time, the implementation of this method used for the numerical experiments in [Bit17c] and Section 2.5 does not incorporate these improvements. $\Diamond$

2.3.1 Algorithm for Functions with Short Frequency Support I

We summarize the procedure introduced above in Algorithm 2, which finds the energetic frequencies and the Fourier coefficients of a function $f \in C_{2\pi}$ with bandwidth N and short frequency support of length at most B if N and an upper bound B on the support length are known a priori. We will investigate its performance with respect to runtime and noisy input data in numerical experiments in Section 2.5, where we will also compare Algorithm 2 to other methods.

As in Section 2.2.1, the function `extended_gcd` in line 16 of Algorithm 2 finds a representation of the greatest common divisor g of two integers a and b of the form

$$g = \gcd(a, b) = ua + vb,$$

where $u, v \in \mathbb{Z}$. By definition of $s, t_1, \ldots, t_L$, we always have $g = 1$ in line 16.

Algorithm 2 Algorithm for Functions with Short Frequency Support I (Algorithm 1 in [Bit17c])

Input: $f \in C_{2\pi}$, N, B, where f has bandwidth N and a short frequency support of length at most $B < N$, and noise threshold $\varepsilon > 0$.

Output: The set R of at most B energetic frequencies of f and the vector $\mathbf{x}$ of estimates for their Fourier coefficients.

Initialize $R \leftarrow \emptyset$.
Find L and the smallest odd primes $t_1, \ldots, t_L$ s.t. $B \cdot \prod_{l=1}^{L-1} t_l < N \leq B \cdot \prod_{l=1}^{L} t_l$.
Set $s := 2^\alpha$, where $\alpha := \lfloor \log_2 B \rfloor + 1$.
for l from 0 to L **do**
    $\mathbf{a}^{st_l} \leftarrow \left(f\left(\frac{2\pi j}{st_l}\right)\right)_{j=0}^{st_l-1}$
    $\widehat{\mathbf{a}^{st_l}} \leftarrow \mathbf{CDFT}\left[\mathbf{a}^{st_l}\right]$
end for

Identification of one of the Energetic Frequencies

$r_0 \leftarrow \operatorname{argmax}\left\{\left|\widehat{a^s}_\nu\right| : \nu \in \left\{-\frac{s}{2}+1, \ldots, \frac{s}{2}\right\}\right\}$.
for l from 1 to L **do**
    $b_{\min} \leftarrow \underset{b \in \left\{-\left\lceil \frac{t_l}{2}\right\rceil+1, \ldots, \left\lfloor \frac{t_l}{2}\right\rfloor\right\}}{\operatorname{argmin}} \left(\left| \widehat{a^s}_{r_0} - \widehat{a^{st_l}}_{b\cdot s+r_0} \right| \right)$
    $r_l \leftarrow (b_{\min} \cdot s + r_0) \mod t_l$
end for

Reconstruction of $\tilde{\omega}$ from its Residues

Set $l = 0$, $\tilde{\omega} = r_0$ and $n = s$.
while $l < L$ **do**
    Set $l = l + 1$
    $(g, u, v) \leftarrow \texttt{extended\_gcd}(n, t_l)$
    $\tilde{\omega} = n\left(((r_l - \tilde{\omega}) \cdot u) \mod t_l\right) + \tilde{\omega}$
    $n = t_l \cdot n$
end while
Set $\tilde{\omega} = \tilde{\omega} \mod n$ and shift $\tilde{\omega}$ into the range $\left\{-\left\lceil \frac{N}{2}\right\rceil + 1, \ldots, \left\lfloor \frac{N}{2}\right\rfloor\right\}$, since $N \leq n$.

Identification of the Remaining Frequencies and Coefficients

for ω from $\tilde{\omega} - B + 1$ to $\tilde{\omega} + B - 1$ **do**
    if $\left|\widehat{a^{st_1}}_{\omega \bmod st_1}\right| > \varepsilon$ **then**
        $R \leftarrow R \cup \{\omega\}$
        $x_\omega \leftarrow \widehat{a^{st_1}}_{\omega \bmod st_1}$
    end if
end for

Output: R, $\mathbf{x}$.

Remark 2.16 Until now we considered the samples of the input function $f \in C_{2\pi}$ to be noiseless. In practice, however, this is rarely the case. If we assume that instead of f we can only measure the perturbed function $f+\eta$, where $\eta \in C_{2\pi}$ satisfies that $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$ for some suitable noise threshold $\varepsilon > 0$, then $f + \eta$ has still approximately a short support of length B. As Algorithm 2 reconstructs the frequency $\tilde{\omega}$ with largest magnitude Fourier coefficient of $f + \eta$ in lines 8 to 20, $\tilde{\omega}$ is also an energetic frequency of f if $\varepsilon > 0$ is not chosen too large, e.g., if

$$|c_\omega(f)| > 2\varepsilon$$

for all energetic frequencies ω of f. The remaining energetic frequencies are determined by examining whether the corresponding Fourier coefficients of $f + \eta$ are significantly large, i.e., whether

$$\left|\widehat{a^{st_1}}_{\omega \bmod st_1}\right| > \varepsilon$$

for all $\omega \in \{\tilde{\omega} - B + 1, \ldots, \tilde{\omega} + B - 1\}$. Consequently, the thus obtained coefficients are good approximations of the true Fourier coefficients $c_\omega(f)$ of f, even if the samples are only obtained from the noisy function $f + \eta$. This behavior is also supported empirically by the numerical results presented in Section 2.5.

Recall that, as mentioned in Remarks 2.11 and 2.15, the runtime of Algorithm 2 could be decreased by checking whether the condition in line 2 is still satisfied if some of the small odd primes $t_1, \ldots, t_L$ are omitted, as then fewer samples have to be used and fewer DFTs have to be computed. Additionally, one can also restrict the identification of the energetic frequencies in lines 21 to 26 to the frequencies in $\{\tilde{\omega} - B + 1, \ldots, \tilde{\omega} + B - 1\}$ whose residues modulo s correspond to nonzero entries of $\widehat{a^s}$.

The performance of Algorithm 2 for noisy data can also be improved by choosing the Fourier coefficient approximates in line 24 via

$$x_\omega := \frac{1}{L} \sum_{l=1}^{L} \widehat{a^{st_l}}_{\omega \bmod st_l}.$$

Note that none of these improvements change the order of the theoretical runtime and sample bound we will show in Section 2.3.2. For the numerical experiments in Section 2.5 we did not incorporate these improvements and stabilizations into the implementation of the algorithm. ◊

2.3.2 Runtime and Sampling Bounds

Proving runtime and sample bounds for Algorithm 2 requires some preliminary results about the occurring sums of prime numbers. Recall that we have to compute CDFTs of length st_l for all $l \in \{0, \ldots, L\}$. By Section 1.1.1 and Remark 1.9, a CDFT of length st_l requires st_l samples and $\mathcal{O}(st_l \log(st_l))$ arithmetical operations. Hence, we have to estimate the number of necessary samples,

$$\sum_{l=0}^{L} st_l, \tag{2.12}$$

of the input function f, and the runtime of the computation of the CDFTs,

$$\mathcal{O}\left(\sum_{l=0}^{L} st_l \log(st_l)\right) = \mathcal{O}\left(s\sum_{l=0}^{L} t_l \log(t_l) + s\log(s)\sum_{l=0}^{L} t_l\right). \tag{2.13}$$

First, we will estimate the largest of the small odd primes, t_L.

Definition 2.17 For $l \in \mathbb{N}$ we denote by p_l the lth prime. Further, we set $p_0 \coloneqq 1$.

The following result about the smallest M primes $p_1, \ldots, p_M$ has been shown in [IS08], Lemma 4.

Lemma 2.18 Let $B, N \in \mathbb{N}$ with $B \leq N$. If $M \in \mathbb{N}$ satisfies

$$\prod_{l=1}^{M-1} p_l < \frac{N}{B} \leq \prod_{l=1}^{M} p_l,$$

there exists a constant $a > 0$ with

$$p_M = \log\frac{N}{B} + \mathcal{O}\left(\frac{\log\frac{N}{B}}{\exp\left(a\sqrt{\log\log\frac{N}{B}}\right)}\right).$$

Using Lemma 2.19, we obtain an estimate for the prime t_L required by Algorithm 2.

Lemma 2.19 (Lemma 3.2 in [Bit17c]) Denote by t_l the lth odd prime. Let $B, N \in \mathbb{N}$ with $B \leq N$. If $L \in \mathbb{N}$ satisfies

$$\prod_{l=1}^{L-1} t_l < \frac{N}{B} \leq \prod_{l=1}^{L} t_l,$$

there exists a constant $a > 0$ with

$$t_L = \log\frac{2N}{B} + \mathcal{O}\left(\frac{\log\frac{2N}{B}}{\exp\left(a\sqrt{\log\log\frac{2N}{B}}\right)}\right).$$

Proof. Multiplying the presumed inequalities by 2 yields that

$$\prod_{l=1}^{L-1} t_l < \frac{N}{B} \leq \prod_{l=1}^{L} t_l \quad \Leftrightarrow \quad \prod_{l=1}^{L} p_l < \frac{2N}{B} \leq \prod_{l=1}^{L+1} p_l.$$

By Lemma 2.18 there exists a constant $a > 0$ such that

$$t_L = p_{L+1} = \log\frac{2N}{B} + \mathcal{O}\left(\frac{\log\frac{2N}{B}}{\exp\left(a\sqrt{\log\log\frac{2N}{B}}\right)}\right) = \mathcal{O}\left(\log\frac{N}{B}\right).$$

□

Additionally, we require estimates about general sums of prime numbers, which were provided in Lemmas 5 and 6 in [IS08].

Lemma 2.20 For all $R \in \mathbb{N}$ the following estimates hold true,

(i) $$\sum_{\substack{p \leq R \\ p \text{ prime}}} p = \frac{R^2}{2 \log R} + \mathcal{O}\left(\frac{R^2}{\log^2 R}\right),$$

(ii) $$\sum_{\substack{p \leq R \\ p \text{ prime}}} p \log p = \frac{R^2}{2} + \mathcal{O}\left(\frac{R^2}{\log R}\right).$$

With the help of Lemma 2.20 we can now estimate the sums in (2.12) and (2.13).

Lemma 2.21 (Lemma 3.4 in [Bit17c]) Denote by t_l the lth odd prime. Let $B, N \in \mathbb{N}$ with $B \leq N$ and let $L \in \mathbb{N}$ such that

$$\prod_{l=1}^{L-1} t_l < \frac{N}{B} \leq \prod_{l=1}^{L} t_l.$$

Then we obtain the following estimates,

(i) $$\sum_{l=0}^{L} t_l = \mathcal{O}\left(\frac{\log^2 \frac{N}{B}}{\log \log \frac{N}{B}}\right),$$

(ii) $$\sum_{l=0}^{L} t_l \log t_l = \mathcal{O}\left(\log^2 \frac{N}{B}\right).$$

Proof. (i) In order to prove the first claim, we apply the estimate from Lemma 2.20 (i) and find that

$$\begin{aligned} \sum_{l=0}^{L} t_l &\leq \sum_{\substack{p \leq t_L \\ p \text{ prime}}} p \\ &= \frac{{t_L}^2}{2 \log t_L} + \mathcal{O}\left(\frac{{t_L}^2}{\log^2 t_L}\right). \end{aligned}$$

Using the estimate

$$t_L = \log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right), \quad \text{where} \quad A \coloneqq \exp\left(a \sqrt{\log \log \frac{2N}{B}}\right),$$

from Lemma 2.19, we find

$$\begin{aligned} \sum_{l=0}^{L} t_l &= \frac{\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)^2}{2 \log\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)} + \mathcal{O}\left(\frac{\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)^2}{\log^2\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)}\right) \\ &= \mathcal{O}\left(\frac{\log^2 \frac{N}{B}}{\log \log \frac{N}{B}}\right). \end{aligned}$$

(ii) For the second claim we employ the result from Lemma 2.20 (ii), which yields

$$\begin{aligned}\sum_{l=0}^{L} t_l \log t_l &\leq \sum_{\substack{p \leq t_L \\ p \text{ prime}}} p \log p \\ &= \frac{{t_L}^2}{2} + \mathcal{O}\left(\frac{{t_L}^2}{\log t_L}\right) \\ &= \frac{\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)^2}{2} + \mathcal{O}\left(\frac{\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)^2}{\log\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)}\right) \\ &= \mathcal{O}\left(\log^2 \frac{N}{B}\right).\end{aligned}$$

□

Combining all of these estimates, we can prove the following main result about the runtime and sampling complexity of Algorithm 2.

Theorem 2.22 (Theorem 3.5 in [Bit17c]) *Let B and $N \in \mathbb{N}$ with $B < N$ and $\omega_1 \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor - B + 1\right\}$. Let $f \in C_{2\pi}$ have a short frequency support of length at most B and bandwidth N, i.e.,*

$$f(x) = \sum_{k=0}^{B-1} c_{\omega_1+k} \cdot e^{i(\omega_1+k)x}.$$

Then Algorithm 2 returns the energetic frequencies of f and their corresponding Fourier coefficients in

$$\mathcal{O}\left(B \log B \cdot \frac{\log^2 \frac{N}{B}}{\log \log \frac{N}{B}}\right)$$

time, and has a sampling complexity of

$$\mathcal{O}\left(B \cdot \frac{\log^2 \frac{N}{B}}{\log \log \frac{N}{B}}\right).$$

Proof. It is evident from the construction of Algorithm 2 that it correctly returns the energetic frequencies and their corresponding Fourier coefficients, disregarding numerical errors, for exact data. We can now calculate the runtimes of the different parts of Algorithm 2 using the observations made above.

The costs of computing the small odd primes $t_1, \ldots, t_L$ are insignificant. Even if we choose a bandwidth $N = 10^{10}$ and a support length $B = 1$, the largest required prime, $t_L = \mathcal{O}\left(\log \frac{N}{B}\right)$, is 31. Usually one would consider greater support lengths, which means that even fewer t_l sufficed. Hence, the t_l can easily be found from precomputed lists of small primes in $\mathcal{O}\left(\log \frac{N}{B}\right)$ time.

By Section 1.1.1 and Remark 1.9, calculating the CDFT of a vector of arbitrary length

M has a runtime of $\mathcal{O}(M \log M)$, so the CDFTs in lines 4 to 7 require

$$\mathcal{O}\left(\sum_{l=0}^{L} st_l \log(st_l)\right) = \mathcal{O}\left(s \sum_{l=0}^{L} t_l \log(t_l) + s\log(s) \sum_{l=0}^{L} t_l\right)$$

arithmetical operations. With Lemma 2.21 and $s = \mathcal{O}(B)$, we obtain that

$$\begin{aligned}
&\mathcal{O}\left(\sum_{l=0}^{L} st_l \log(st_l)\right) \\
&= \mathcal{O}\left(B \log^2 \frac{N}{B} + B \log B \cdot \frac{\log^2 \frac{N}{B}}{\log\log \frac{N}{B}}\right) \\
&= \mathcal{O}\left(B \log^2 \frac{N}{B}\left(1 + \frac{\log B}{\log\log \frac{N}{B}}\right)\right).
\end{aligned}$$

Finding the largest magnitude entry of $\widehat{\mathbf{a}^s}$ in line 8 needs $\mathcal{O}(s)$ operations. The computation of the residues of an energetic frequency in lines 9 to 12 has a complexity of

$$\mathcal{O}\left(\sum_{l=0}^{L} t_l\right) = \mathcal{O}\left(\frac{\log^2 \frac{N}{B}}{\log\log \frac{N}{B}}\right).$$

As already mentioned in Section 2.2.1, the runtime of the extended Euclidean algorithm `extended_gcd`(n, t_l) in line 16 is $\mathcal{O}(\log t_l)$, since $t_l < n$ for all l. Therefore, the CRT reconstruction procedure in lines 13 to 20, see Algorithm 1, requires $\mathcal{O}\left(\sum_{l=1}^{L} \log t_l\right)$ operations, which is insignificant compared to the runtime of lines 9 to 12. Identifying the remaining frequencies in lines 21 to 26 has an arithmetical complexity of $\mathcal{O}(B)$.

Combining all these considerations yields an overall runtime of

$$\begin{aligned}
&\mathcal{O}\left(\sum_{l=0}^{L} st_l \log(st_l)\right) + \mathcal{O}(s) + \mathcal{O}\left(\sum_{l=0}^{L} t_l\right) + \mathcal{O}(B) \\
&= \mathcal{O}\left(B \log^2 \frac{N}{B}\left(1 + \frac{\log B}{\log\log \frac{N}{B}}\right) + B + \log \frac{N}{B} + B\right) \\
&= \mathcal{O}\left(B \log B \cdot \frac{\log^2 \frac{N}{B}}{\log\log \frac{N}{B}}\right).
\end{aligned}$$

Further, we obtain with Lemma 2.21 (i) that Algorithm 2 has a sampling complexity of

$$s \sum_{l=0}^{L} t_l = \mathcal{O}\left(B \cdot \frac{\log^2 \frac{N}{B}}{\log\log \frac{N}{B}}\right).$$

□

2.4 Sparse FFT for Functions with Short Frequency Support II

The simplification of Algorithm 2 in [Iwe10] which we developed in Section 2.3 is not the only feasible way to adapt said method to the setting of a function $f \in C_{2\pi}$ with short frequency support of length at most B. Instead of choosing the L smallest odd primes satisfying

$$B \cdot \prod_{l=1}^{L-1} t_l < N \leq B \cdot \prod_{l=1}^{L} t_l$$

and the smallest power of 2, s, that is greater than B in order to reconstruct the energetic frequencies of f from their residues modulo s and $t_1, \ldots, t_L$, one can also consider K primes $s_1, \ldots, s_K$ greater than $2B$ such that they satisfy the prerequisites of the CRT, and reconstruct the frequencies from their residues modulo $s_1, \ldots, s_K$, thus completely omitting the small primes t_l. This approach was suggested to us by the anonymous reviewer of [Bit17c].

If one can guarantee that the residue of the, e.g., smallest energetic frequency modulo s_k is identified correctly for all k, this choice of $s_1, \ldots, s_K$ yields an algorithm which the anonymous reviewer expected to have an even smaller theoretic runtime. We will prove in Section 2.4.3 that it has a runtime of

$$\mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B} \log\left(\frac{B+\log N}{\log B}\right)\right),$$

whereas Algorithm 2 has a runtime of

$$\mathcal{O}\left(B \log B \cdot \frac{\log^2(N/B)}{\log\log(N/B)}\right).$$

Since it is not obvious which of the algorithms is faster for usual choices for the support length B and the bandwidth N, and impossible to tell which of the algorithms performs better with respect to noisy data, we will investigate the suggested approach in detail hereafter and compare it numerically to Algorithm 2 in Section 2.5.

Due to the block structure of the energetic frequencies, it suffices to reconstruct, for example, the smallest energetic frequency ω_1 of the function

$$f(x) = \sum_{k=0}^{B-1} c_{\omega_1+k} \cdot e^{i(\omega_1+k)x}.$$

The remaining energetic frequencies can then be identified by examining whether the Fourier coefficients corresponding to the frequencies contained in the support interval $S \coloneqq \{\omega_1, \omega_1+1, \ldots, \omega_1+B-1\}$ are significantly large. All of the at most B possibly energetic frequencies have to be contained in this set, which is completely determined by ω_1 and the bound B on the support length. Consequently, we will now focus on the correct identification of ω_1, for which the concept of first support indices can be employed.

Definition 2.23 (First Support Index) Let $f \in C_{2\pi}$ have a short frequency support of length at most B and let $M \in \mathbb{N}$. The *first support index* of $\widehat{\mathbf{a}^M}$ is defined as

$$\nu_M \coloneqq \max\Bigg\{\nu \in \left\{-\left\lceil \frac{M}{2}\right\rceil + 1, \dots, \left\lfloor \frac{M}{2}\right\rfloor\right\} : \widehat{a^M}_\omega = 0 \\ \forall \omega \notin \{\nu, (\nu+1) \bmod M, \dots, (\nu+B-1) \bmod M\}\Bigg\},$$

i.e., ν_M is the largest index for which all entries of $\widehat{\mathbf{a}^M}$ that are not the B periodically consecutive entries $\widehat{a^M}_{\nu_M}, \widehat{a^M}_{(\nu_M+1) \bmod M}, \dots, \widehat{a^M}_{(\nu_M+B-1) \bmod M}$ vanish.

Remark 2.24 Note that Definition 2.23 is only meaningful if $M > 2B$. For $M \leq 2B$, there might not be an intuitive first support index. Consider for example the function

$$f(x) = e^{i\cdot 213x} + e^{i\cdot 217x} \in C_{2\pi}.$$

We find that $\omega_1 \coloneqq 213 \equiv -3 \bmod 8$, $\omega_5 \coloneqq 217 \equiv 1 \bmod 8$ and

$$\widehat{\mathbf{a}^8} = (1,0,0,0,1,0,0,0)^T.$$

By Definition 2.23, the first support index of $\widehat{\mathbf{a}^8}$ is 1, even though -3, which corresponds to ω_1, makes more sense. As we want to use the first support index of $\mathbf{a}^M$ such that it corresponds to the smallest energetic frequency of f, we will restrict ourselves to the case $M > 2B$ from now on.

Recall that by (1.9)

$$\widehat{a^N}_\nu = \begin{cases} c_\nu & \text{if } \nu \in \{\omega_1, \omega_1+1, \dots, \omega_1+B-1\}, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the first support index of $\widehat{\mathbf{a}^N}$ is precisely the smallest energetic frequency ω_1. If we consider an integer $s > 2B$, then s separates all energetic frequencies. The CDFT of the vector of s equidistant samples, $\widehat{\mathbf{a}^s}$, satisfies that

$$\widehat{a^s}_\nu = \sum_{\substack{\omega=-\left\lceil \frac{N}{2}\right\rceil+1 \\ \omega \equiv \nu \bmod s}}^{\left\lfloor \frac{N}{2}\right\rfloor} c_\omega$$

by (2.2). Thus, it follows from the separation property of s and (2.3) that the block structure of the short support of $\widehat{\mathbf{a}^N}$ remains intact in $\widehat{\mathbf{a}^s}$, though it might be periodically wrapped around the boundary of $\mathbf{a}^s$, i.e., around $\left\lfloor \frac{s}{2}\right\rfloor$ and $-\left\lceil \frac{s}{2}\right\rceil + 1$. Figure 2.1 shows the essentially two possibilities for the support of $\widehat{\mathbf{a}^s}$ if f has a short frequency support. Hence, $\widehat{\mathbf{a}^s}$ is also said to have a *short support*, which can be of the form

$$\widehat{\mathbf{a}^M} = \left(\widehat{a^M}_{-\left\lceil \frac{M}{2}\right\rceil+1}, \dots, \widehat{a^M}_{-\left\lceil \frac{M}{2}\right\rceil+B-b-1}, 0, \dots, 0, \widehat{a^M}_{\left\lfloor \frac{M}{2}\right\rfloor-b}, \dots, \widehat{a^M}_{\left\lfloor \frac{M}{2}\right\rfloor}\right)^T.$$

The concept of short supports for vectors is not relevant for this chapter, but will be used extensively in Chapters 5 and 6. ◊

By (2.3), the first support index ν_s of $\widehat{\mathbf{a}^s}$ is the residue of the smallest energetic fre-

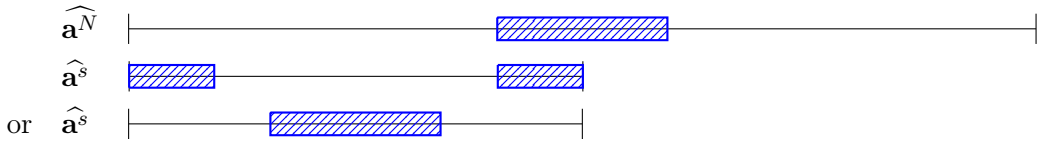

Figure 2.1: Short support of $\widehat{\mathbf{a}^N}$ and periodized blocks in $\widehat{\mathbf{a}^s}$

quency ω_1 modulo s if $s > 2B$. Consequently, if we can determine the first support indices of $\widehat{\mathbf{a}^{s_1}}, \dots, \widehat{\mathbf{a}^{s_K}}$ for K primes $s_1, \dots, s_K > 2B$ satisfying the requirements of the CRT, we can uniquely reconstruct ω_1 from its thus obtained residues using Algorithm 1. The remaining energetic frequencies can be identified by checking for all frequencies ω in the set $S = \{\omega_1, \omega_1 + 1, \dots, \omega_1 + B - 1\}$ of possibly energetic frequencies whether the corresponding Fourier coefficient c_ω is significantly large. Hence, besides N, this method also requires an upper bound on the support length B, though it does not have to be known exactly.

The simplest way to ensure that the CRT reconstruction can be used is to choose the $K \coloneqq \lfloor \log_{s_1} N \rfloor + 1$ smallest primes $s_1, \dots, s_K$ that are greater than $2B$. They are pairwise relatively prime and satisfy

$$\prod_{k=1}^{K} s_k \geq \prod_{k=1}^{K} s_1 = s_1^{\lfloor \log_{s_1} N \rfloor + 1} \geq s_1^{\log_{s_1} N} = N.$$

Choosing $s_1, \dots, s_K > 2B$ guarantees that the first support indices of $\widehat{\mathbf{a}^{s_1}}, \dots, \widehat{\mathbf{a}^{s_K}}$ are unique and correspond to the smallest energetic frequency ω_1, which allows us to recover ω_1 from its residues modulo $s_1, \dots, s_K$ as in Section 2.2.2. All remaining possibly energetic frequencies then have to be contained in the set $S = \{\omega_1, \dots, \omega_1 + B - 1\}$. For exact data, their Fourier coefficients are given without any further computation, since for any k the prime s_k is separating, so the Fourier coefficients are just the significantly large entries of $\widehat{\mathbf{a}^{s_k}}$, see (2.3). For noisy data we can stabilize this method by using the mean of these values as an estimate for the coefficients, i.e.,

$$c_\omega = \frac{1}{K} \sum_{k=1}^{K} \widehat{a^{s_k}}_{\omega \bmod s_k}$$

for all $\omega \in \{\omega_1, \dots, \omega_1 + B - 1\}$. Thus, we can correctly identify the energetic frequencies and find good estimates for the corresponding Fourier coefficients.

Remark 2.25 Similarly to Remark 2.11, this method can be further improved by checking whether some of the primes $s_1, \dots, s_K$ can be omitted while still satisfying the requirements of the CRT. Again, we did not incorporate this enhancement in the implementation of our algorithm used for the numerical experiments in Section 2.5. ◊

However, in order to obtain an efficient algorithm, we have to be able to find the first support index of the CDFT of a vector of equidistant samples in a fast and stable way.

2.4.1 Detecting the First Support Index

In the following section we will discuss two methods for efficiently and stably finding the first support index of $\widehat{\mathbf{a}^s}$ for an $s > 2B$.

The first one, taken from [PW16a], relies on the computation of the *local energies* $e_{s,\omega}$ for $\omega \in \left\{-\left\lceil \frac{s}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{s}{2} \right\rfloor\right\}$, defined as

$$e_{s,\omega} \coloneqq \sum_{\nu=\omega}^{(\omega+B-1) \bmod s} \left|\widehat{a^s}_{\nu \bmod s}\right|^2.$$

The local energies satisfy

$$e_{s,\omega} = e_{s,\omega-1} - \left|\widehat{a^s}_{\omega-1}\right|^2 + \left|\widehat{a^s}_{(\omega+B-1) \bmod s}\right|^2$$

for all $\omega \in \left\{-\left\lceil \frac{s}{2} \right\rceil + 2, \ldots, \left\lfloor \frac{s}{2} \right\rfloor\right\}$. The first support index ν_s of $\widehat{\mathbf{a}^s}$ is given as

$$\nu_s = \operatorname{argmax}\left\{ \left|\widehat{a^s}_{\nu}\right| : \nu \in \operatorname*{argmax}_{\omega \in \left\{-\left\lceil \frac{s}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{s}{2} \right\rfloor\right\}} \{e_{s,\omega}\} \right\},$$

i.e., as the index corresponding to the largest magnitude entry, $\left|\widehat{a^s}_{\nu_s}\right|$, of $\widehat{\mathbf{a}^s}$ of the set of indices maximizing the local energies. This guarantees that, in the case where several indices result in the same maximal local energy, an index corresponding to a significantly large entry of $\widehat{\mathbf{a}^s}$ is chosen. The maximizer of $e_{s,\omega}$ is only not unique if the bound B is not the exact support length, so we avoid using a non-energetic index as the first support index by the above choice.

However, this method is not well-suited for noisy data. In [PW16a], the authors suggested a stabilized version for finding the first support index. This stabilization cannot be applied in our setting, since the available samples are obtained from $\mathbf{a}^s$ and not from the full-length vector $\mathbf{a}^N$. Further, we do not consider successive periodizations of the same vector. The input vectors we require for our approach are structured differently; thus, we cannot utilize similar redundancies in order to stabilize the detection.

The second method for detecting the first support index we want to employ is a simple *block search* algorithm. If $s > 2B$ and the data is exact, the first support index ν_s is uniquely determined as the largest index of $\widehat{\mathbf{a}^s}$ with nonzero entry which is preceded by at least $B + 1$ entries that are zero. In the case of noisy data, it is the largest index with significantly large entry that is preceded by at least $B + 1$ insignificant entries. Here, analogously to the definition for frequencies, we call an entry of $\widehat{\mathbf{a}^s}$ *significantly large* and its corresponding index *energetic* if the absolute value of the entry is greater than or equal to some threshold $\varepsilon > 0$ depending on the noise level. For exact data this approach works well, but for noisy data one has to choose a suitable threshold $\varepsilon > 0$, usually without knowing the noise level a priori.

Note that both methods require a priori knowledge of a good upper bound B on the support length of f. With either of these methods we can now correctly identify the first support index of $\widehat{\mathbf{a}^s}$ for any $s > 2B$. If we compute the residues of ω_1 modulo some integers $s_1, \ldots, s_K$ such that they satisfy the prerequisites of the CRT, we can uniquely reconstruct ω_1 via Algorithm 1.

Again, we provide an example for the method presented above before giving the detailed algorithm. We apply our second procedure for functions with short frequency support to the same function as in Example 2.14.

Example 2.26 Let $f \in C_{2\pi}$, $f(x) = e^{i\cdot 210x} - e^{i\cdot 211x} + 2e^{i\cdot 212x} - e^{i\cdot 213x} - 2e^{i\cdot 214x}$ with bandwidth $N = 1{,}000$ and block length 5. Let us again assume that we only know the upper bound $B = 6$ on the true block length a priori. Then we set $s_1 \coloneqq 13 > 2 \cdot B$,

$$K \coloneqq \left\lfloor \log_{s_1} N \right\rfloor + 1 = 3,$$

$s_2 \coloneqq 17$ and $s_3 \coloneqq 19$. Indeed, we have that $B \cdot 13 \cdot 17 = 221$ and $B \cdot 13 \cdot 17 \cdot 19 = 4{,}199$. Computing the CDFT of $\mathbf{a}^{13}$, we obtain

$$\widehat{\mathbf{a}^{13}} = (0,0,0,0,0,0,0,0,1,-1,2,-1,-2)^T.$$

We aim to recover the smallest energetic frequency ω_1 of f from its residues modulo $13, 17$ and 19, which are precisely the first support indices of the vectors $\widehat{\mathbf{a}^{13}}$, $\widehat{\mathbf{a}^{17}}$ and $\widehat{\mathbf{a}^{19}}$. For this example we want to detect the first support indices using local energies. First, we have to compute $e_{13,\,\omega}$ for all $\omega \in \{-6,\dots,6\}$. We find that

$$e_{13,\,-6} = \sum_{\nu=-6}^{-1} \left| \widehat{a^{13}}_\nu \right|^2 = 0,$$

and obtain the following vector of local energies,

$$(e_{13,\,\omega})_{\omega=-6}^{6} = (0,0,0,1,2,6,7,11,11,10,9,5,4)^T.$$

Consequently, since $e_{13,\,1} = e_{13,\,2} = 11$ and $\left|\widehat{a^{13}}_1\right| = 0 < \left|\widehat{a^{13}}_2\right| = 1$, the first support index ν_{13} of $\widehat{\mathbf{a}^{13}}$ is 2. Analogously, we can compute the first support indices of $\widehat{\mathbf{a}^{17}}$ and $\widehat{\mathbf{a}^{19}}$. With

$$\begin{aligned} \widehat{\mathbf{a}^{17}} &= (-1,-2,0,0,0,0,0,0,0,0,0,0,0,0,1,-1,2)^T && \text{and} \\ \widehat{\mathbf{a}^{19}} &= (0,0,0,0,0,0,0,0,0,0,0,1,-1,2,-1,-2,0,0,0,0)^T, \end{aligned}$$

we find that

$$\begin{aligned} (e_{17,\,\omega})_{\omega=-8}^{8} &= (5,4,0,0,0,0,0,0,0,1,2,6,7,11,11,10,9)^T && \text{and} \\ (e_{19,\,\omega})_{\omega=-9}^{9} &= (0,0,0,0,0,1,2,6,7,11,11,10,9,5,4,0,0,0,0)^T. \end{aligned}$$

It follows from $e_{17,\,5} = e_{17,\,6} = 11$ and $\left|\widehat{a^{17}}_5\right| = 0 < \left|\widehat{a^{17}}_6\right| = 1$ that the first support index ν_{17} of $\widehat{\mathbf{a}^{17}}$ is 6. Since $e_{19,\,0} = e_{19,\,1} = 11$ and $\left|\widehat{a^{19}}_0\right| = 0 < \left|\widehat{a^{19}}_1\right| = 1$, the first support index ν_{19} of $\widehat{\mathbf{a}^{19}}$ is 1. Consequently, the smallest energetic frequency ω_1 of f satisfies

$$\omega_1 \equiv 2 \bmod 13, \quad \omega_1 \equiv 6 \bmod 17 \quad \text{and} \quad \omega_1 \equiv 1 \bmod 19.$$

Reconstructing ω_1 from these residues via Algorithm 1 yields that $\omega_1 = 210$. The remaining energetic frequencies can be identified by computing the Fourier coefficients of the $B = 6$ frequencies contained in the set $S = \{210, 211, \dots, 215\}$ via

$$c_\omega = \frac{1}{3} \sum_{k=1}^{3} \widehat{a^{s_k}}_{\omega \bmod s_k} \qquad \forall\, \omega \in S.$$

Thus, we obtain that $c_{210} = 1$, $c_{211} = -1$, $c_{212} = 2$, $c_{213} = -1$ and $c_{214} = -2$. Since $c_{215} = 0$, we find that 215 is not an energetic frequency and that f has a short frequency support of length 5. This method requires three CDFTs of lengths 13, 17 and 19 with a runtime of $\mathcal{O}\left(\sum_{k=1}^{3} s_k \log s_k\right)$ and a sampling complexity of 49 instead of a single CDFT of length 1,000. Note that for this example the second method for reconstructing a function with short frequency support needs significantly less samples than the 220 samples required by the procedure described in Section 2.3. We will compare the runtimes and sampling complexities of the two methods more thoroughly in Section 2.5.

$\Diamond$

2.4.2 Algorithm for Functions with Short Frequency Support II

We summarize the procedure introduced above in Algorithm 3, which deterministically finds the energetic frequencies and the Fourier coefficients of a function $f \in C_{2\pi}$ with bandwidth N and short frequency support if N and an upper bound B on the support length are known a priori. We will investigate its performance with respect to runtime and noisy input data in numerical experiments in Section 2.5, where we will also compare Algorithm 3 to Algorithm 2 and other sparse FFT methods.

As in Section 2.2.1, the function `extended_gcd` in line 13 of Algorithm 3 finds a representation of the greatest common divisor g of two integers a and b of the form

$$g = \gcd(a, b) = ua + vb,$$

where $u, v \in \mathbb{Z}$. As in Algorithm 2, we always have $g = 1$ in line 13 by choice of $s_1, \ldots, s_K$.

2.4.3 Runtime and Sampling Bounds

In order to prove runtime and sample bounds for Algorithm 3, we need some estimates involving the required primes $s_1, \ldots, s_K$. This can be done using two equivalent formulations of the *Prime Number Theorem*.

Theorem 2.27 (Prime Number Theorem) *For $x \in \mathbb{R}$ define the* prime-counting function

$$\pi(x) \coloneqq \sum_{\substack{p \leq x \\ p \text{ prime}}} 1.$$

Then the following estimates hold.

(i) $\pi(x) = \dfrac{x}{\log x} + \mathcal{O}\left(\dfrac{x}{\log^2 x}\right)$,

(ii) $p_l = l \log l + \mathcal{O}(l \log \log l)$.

See [MV07], Chapter 6.2, Theorem 6.9 for a proof of (i) and [HW60], Chapter 1, Theorem 8 for a proof of (ii). Recall that in line 5 of Algorithm 3 we have to compute CDFTs of length s_k for all $k \in \{1, \ldots, K\}$. Each of them needs s_k equidistant samples of f and requires $\mathcal{O}(s_k \log s_k)$ arithmetical operations, as already discussed in Section 1.1.2. Thus, we have to find estimates for

$$\sum_{k=1}^{K} s_k \quad \text{and} \quad \sum_{k=1}^{K} s_k \log s_k,$$

Algorithm 3 Algorithm for Functions with Short Frequency Support II

Input: $f \in C_{2\pi}$, B, N, where f has bandwidth N and a short frequency support of length at most $B < N$, and noise threshold $\varepsilon > 0$.

Output: The set R of at most B energetic frequencies of f and the vector $\mathbf{x}$ of estimates for their Fourier coefficients.

Set s_1 as the smallest prime with $s_1 > 2B$.
Let $K = \lfloor \log_{s_1} N \rfloor + 1$ and $s_2 < \cdots < s_K$ the $K-1$ smallest primes greater than s_1.
for k from 1 to K **do**
    $\mathbf{a}^{s_k} \leftarrow \left(f\left(\frac{2\pi j}{s_k}\right)\right)_{j=0}^{s_k-1}$
    $\widehat{\mathbf{a}^{s_k}} \leftarrow \mathbf{CDFT}[\mathbf{a}^{s_k}]$
end for

Identification of the Smallest Energetic Frequency ω_1

for k from 1 to K **do**
    $\nu_{s_k} \leftarrow$ first support index of $\widehat{\mathbf{a}^{s_k}}$
end for

Reconstruction of ω_1 from its Residues

Set $k = 1$, $\omega_1 = \nu_{s_1}$ and $n = s_1$.
while $k < K$ **do**
    Set $k = k + 1$
    $(g, u, v) \leftarrow \texttt{extended\_gcd}(n, s_k)$
    $\omega_1 = n\left(((\nu_{s_k} - \omega_1) \cdot u) \mod s_k\right) + \omega_1$
    $n = s_k \cdot n$
end while
Set $\omega_1 = \omega_1 \mod n$ and shift ω_1 into the range $\left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$, since $N \leq n$.

Identification of the Remaining Frequencies and Coefficients

for ω from ω_1 to $\omega_1 + B - 1$ **do**
    if $\left| \frac{1}{K} \sum_{k=1}^{K} \widehat{a^{s_k}}_{\omega \bmod s_k} \right| > \varepsilon$ **then**
        $R \leftarrow R \cup \{\omega\}$
        $x_\omega \leftarrow \frac{1}{K} \sum_{k=1}^{K} \widehat{a^{s_k}}_{\omega \bmod s_k}$
    end if
end for

Output: R, $\mathbf{x}$.

in order to obtain bounds for the number of required samples of the function we aim to recover and the runtime of the computation of the CDFTs.

Lemma 2.28 Let $B, N \in \mathbb{N}$ with $B < N$ and let s_1 be the smallest prime greater than $2B$. Set $K = \lfloor \log_{s_1} N \rfloor + 1$ and $s_2, \ldots, s_K$ as the $K-1$ smallest primes greater than s_1. Then the following estimates are satisfied.

(i) $\displaystyle\sum_{k=1}^{K} s_k = \mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B} \log\left(\frac{B+\log N}{\log B}\right)\right),$

(ii) $\displaystyle\sum_{k=1}^{K} s_k \log s_k = \mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B} \log^2\left(\frac{B+\log N}{\log B}\right)\right).$

Proof. (i) There exists an index $q \in \mathbb{N}$ such that $s_1 = p_q$, i.e., s_1 is the qth prime. Consequently, p_{q-1} is the largest prime that is smaller than $2B$. Theorem 2.27 (i) yields

$$q - 1 = \pi(2B) = \frac{2B}{\log(2B)} + \mathcal{O}\left(\frac{2B}{\log^2(2B)}\right) = \mathcal{O}\left(\frac{B}{\log B}\right).$$

Note that $s_K = p_{q+K-1}$. Further, by Betrand's postulate, see [HW60], Chapter 22.3, Theorem 417, there exists at least one prime number between $2B$ and $2 \cdot 2B$. Thus, we know that $2B < s_1 < 4B$, and $s_1 = \mathcal{O}(B)$. With $K = \lfloor \log_{s_1} N \rfloor + 1$, we obtain

$$\begin{aligned} q + K - 1 &= \mathcal{O}\left(\frac{B}{\log B}\right) + \lfloor \log_{s_1} N \rfloor + 1 \\ &= \mathcal{O}\left(\frac{B}{\log B} + \frac{\log N}{\log B}\right) \\ &= \mathcal{O}\left(\frac{B+\log N}{\log B}\right). \end{aligned}$$

Hence, the second formulation of the Prime Number Theorem (Theorem 2.27 (ii)) implies

$$\begin{aligned} s_K &= p_{q+K-1} \\ &= \mathcal{O}\left((q+K-1)\log(q+K-1)\right) \\ &= \mathcal{O}\left(\frac{B+\log N}{\log B} \log\left(\frac{B+\log N}{\log B}\right)\right). \end{aligned} \tag{2.14}$$

Using (2.14), we find for the number of required samples that

$$\begin{aligned} \sum_{k=1}^{K} s_k &= \mathcal{O}\left(K s_K\right) \\ &= \mathcal{O}\left(\frac{\log N}{\log B} \cdot \frac{B+\log N}{\log B} \log\left(\frac{B+\log N}{\log B}\right)\right) \\ &= \mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B} \log\left(\frac{B+\log N}{\log B}\right)\right). \end{aligned}$$

(ii) In order to deal with the estimates for the runtime of the CDFT computation, we first recall a property of the logarithm. For any $a > 0$, the logarithm satisfies that

$$\log(a \cdot \log(a)) = \log a + \log\log(a) = \mathcal{O}(\log a). \tag{2.15}$$

Consequently, combining (2.14) and (2.15) yields

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K} s_k \log s_k\right) \\
&= \mathcal{O}\left(K s_K \log s_K\right) \\
&= \mathcal{O}\left(\frac{\log N}{\log B} \cdot \frac{B+\log N}{\log B} \log\left(\frac{B+\log N}{\log B}\right) \cdot \log\left(\frac{B+\log N}{\log B} \log\left(\frac{B+\log N}{\log B}\right)\right)\right) \\
&= \mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B} \log^2\left(\frac{B+\log N}{\log B}\right)\right).
\end{aligned}$$

□

Employing the estimates proven in Lemma 2.28, we can now show the main result about the runtime and sampling complexities of Algorithm 3.

Theorem 2.29 *Let $B, N \in \mathbb{N}$ with $B < N$ and $\omega_1 \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$. Let $f \in C_{2\pi}$ have a short frequency support of length at most B with bandwidth N, i.e.,*

$$f(x) = \sum_{k=0}^{B-1} c_{\omega_1+k} \cdot e^{i(\omega_1+k)x}.$$

Then Algorithm 3 returns the energetic frequencies of f and their corresponding Fourier coefficients in

$$\mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B} \log^2\left(\frac{B+\log N}{\log B}\right)\right)$$

time, and has a sampling complexity of

$$\mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B} \log\left(\frac{B+\log N}{\log B}\right)\right).$$

Proof. By construction of Algorithm 3, we know that it returns the correct frequencies and Fourier coefficients, apart from numerical errors, for exact data. We will now examine the runtime of the different parts of the algorithm using the considerations made above.

Using precomputed lists of the first, e.g., 10,000 primes, the computational costs of finding the $K = \lfloor \log_{s_1} N \rfloor + 1$ smallest primes greater than $2B$ in lines 1 and 2 are

$$\mathcal{O}\left(\pi(s_K)\right) = \mathcal{O}\left(q + K - 1\right) = \mathcal{O}\left(\frac{B+\log N}{\log B}\right),$$

as one has to check all primes less than or equal to $s_K = p_{q+K-1}$.

By Section 1.1.1 and Remark 1.9, the computation of a CDFT of length M has a runtime of $\mathcal{O}(M \log M)$. Consequently, it follows from Lemma 2.28 (ii) that the CDFTs in lines 3 to 6 require

$$\mathcal{O}\left(\sum_{k=1}^{K} s_k \log s_k\right) = \mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B} \log^2\left(\frac{B+\log N}{\log B}\right)\right) \tag{2.16}$$

arithmetical operations.

The runtime of line 8 depends on whether we find the first support indices ν_{s_k} of the vector $\widehat{\mathbf{a}^{s_k}}$ via local energies or via block search.

(i) If we use the local energies method to detect the first support index, computing the first local energy

$$e_{s_k,0} = \sum_{\nu=-\lceil \frac{s_k}{2} \rceil+1}^{\lfloor \frac{s_k}{2} \rfloor+B-1} \left| \widehat{a^{s_k}}_\nu \right|^2$$

requires $\mathcal{O}(B)$ additions. As for $j \in \{1, \ldots, s_k - 1\}$ the jth local energy is given as

$$e_{s_k,j} = e_{s_k,j-1} - \left| \widehat{a^{s_k}}_{-\lceil \frac{s_k}{2} \rceil+1+j-1} \right|^2 + \left| \widehat{a^{s_k}}_{\left(-\lceil \frac{s_k}{2} \rceil+1+j+B-1\right) \bmod s_k} \right|^2,$$

we have to execute $\mathcal{O}(B + s_k)$ additions in order to calculate all s_k local energies for a fixed k. Thus, employing this method, lines 7 to 9 have a runtime of

$$\mathcal{O}\left(\sum_{k=1}^{K}(B + s_k)\right) = \mathcal{O}\left(K \cdot (B + s_K)\right) = \mathcal{O}(K \cdot s_K),$$

since $B < s_K$.

(ii) If we want to identify the first support index ν_{s_k} of $\widehat{\mathbf{a}^{s_k}}$ by looking for $B+1$ consecutive entries that have an absolute value less than the noise threshold $\varepsilon > 0$, we need to check each entry at most twice to also allow for blocks that are wrapped periodically around the boundary of the vector. Hence, finding the first support index of $\widehat{\mathbf{a}^{s_k}}$ requires $\mathcal{O}(s_k)$ arithmetical operations, and lines 7 to 9 need

$$\mathcal{O}\left(\sum_{k=1}^{K} s_k\right) = \mathcal{O}\left(K \cdot s_K\right)$$

operations.

Consequently, the theoretical runtime of lines 7 to 9 is always dominated by the computational effort of the CDFT computations in lines 3 to 6.

As we discussed in Section 2.2.1, the runtime of the extended Euclidean algorithm `extended_gcd`(n, s_k) in line 13 is $\mathcal{O}(\log s_k)$ if $k \geq 3$, since then $n = \prod_{l=1}^{k-1} s_l > s_k$. If $k = 2$, we can still estimate its runtime with $\mathcal{O}(\log s_2)$, as $s_2 > n = s_1$. Consequently, the CRT reconstruction procedure in lines 10 to 17, i.e., Algorithm 1, has a runtime of

$$\mathcal{O}\left(\sum_{k=1}^{K} \log s_k\right) = \mathcal{O}(K \cdot \log s_K),$$

which is again dominated by the runtime of the CDFT computations. Finally, the calculation of the coefficient estimates for the B possibly energetic frequencies contained in $\{\omega_1, \ldots, \omega_1 + B - 1\}$ in lines 18 to 23 needs $\mathcal{O}(K \cdot B) = \mathcal{O}(K \cdot s_K)$ arithmetical operations.

Hence, we obtain that the runtime of Algorithm 3 is dominated by the runtime of the CDFT computations in lines 3 to 6, so, by (2.16), we obtain an overall runtime of

$$\mathcal{O}\left(\sum_{k=1}^{K} s_k \log s_k\right) = \mathcal{O}\left(\frac{(B + \log N) \log N}{\log^2 B} \log^2\left(\frac{B + \log N}{\log B}\right)\right).$$

Further, Lemma 2.28 (i) yields that Algorithm 3 has a sampling complexity of

$$\sum_{k=1}^{K} s_k = \mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B}\log\left(\frac{B+\log N}{\log B}\right)\right),$$

which completes the proof. □

2.5 Numerical Results for Algorithms 2 and 3

We now present some numerical results regarding the runtimes of Algorithms 2 and 3, their performances for noisy input data and their sampling complexities. Additionally, we compare them to the deterministic sparse inverse FFT algorithm for vectors with short support presented by Plonka and Wannenwetsch as Algorithm 2 in [PW16a], and to MATLAB 2016a's `fft` function.

Algorithm 2 in [PW16a] recovers a vector $\mathbf{y} \in \mathbb{C}^{2^J}$ with short support of length B from its Fourier transform $\widehat{\mathbf{y}}$ by considering periodizations $\mathbf{y}^{(j)} \in \mathbb{C}^{2^j}$ of $\mathbf{y}$,

$$\mathbf{y}^{(j)} \coloneqq \left(\sum_{l=0}^{2^{J-j}-1} y_{k+2^j l}\right)_{k=0}^{2^j-1} \qquad \forall\, j \in \{\lceil \log_2 B\rceil + 1, \dots, J\}.$$

For a more detailed description of this method see Section 5.3. In the case of noisy input data, the algorithm has an arithmetical complexity of $\mathcal{O}(B\log N)$, where $\mathcal{O}(B+\log N)$ samples of the input vector $\widehat{\mathbf{y}}$ are being used. Algorithm 2 in [PW16a] is actually an IFFT algorithm that recovers a 2^J-length vector $\mathbf{y}$ from its Fourier transformed vector $\widehat{\mathbf{y}} \in \mathbb{C}^{2^J}$, whereas Algorithms 2 and 3 find the finite spectrum, i.e., the vector of Fourier coefficients of a 2π-periodic function, from several vectors of equidistant samples of the form

$$\mathbf{a}^{st_l} = \left(f\left(\frac{2\pi j}{st_l}\right)\right)_{j=0}^{st_l-1} \quad \text{or} \quad \mathbf{a}^{s_k} = \left(f\left(\frac{2\pi j}{s_k}\right)\right)_{j=0}^{s_k-1}.$$

Thus, Algorithm 2 in [PW16a] cannot be applied to the same data as our new methods for functions with short support. The more complex sampling schemes for IDFT methods for functions rely heavily on the fact that $f \in C_{2\pi}$ can be evaluated at any $x \in [0, 2\pi)$, whereas Algorithm 2 in [PW16a], which can be modified to become a DFT method for recovering a vector $\widehat{\mathbf{y}} \in \mathbb{C}^{2^J}$ with short support from $\mathbf{y}$, see [PPST19], Section 5.4.2, always requires 2^J equidistant samples of f.

MATLAB's `fft` routine is a fast and highly optimized implementation of the fast Fourier transform, based on the FFTW library, see [FJ17,The18b]. In order to use it meaningfully in our setting, we sample the input function f at N equidistant points, i.e., we determine

$$\mathbf{a}^N = \left(f\left(\frac{2\pi j}{N}\right)\right)_{j=0}^{N-1}.$$

Then we compute an approximation of $\mathbf{c}(f)$ by applying `fft` to $\mathbf{a}^N$, as we discussed in Section 1.2.1. Vector-based and function-based algorithms are not really comparable from a sampling perspective, as for vector-based algorithms we can only use the given vector entries, which correspond to equidistantly sampling the function at N points. For function-based algorithms, however, one can use much more complex sampling schemes. In Algorithms 2 and 3, for example, we sample equidistantly at st_l points for $L+1$ values

for t_l, or at s_k points for K different primes s_k.

As we do not know of any other, more suitable algorithms to which we can compare Algorithms 2 and 3, we decided to use Algorithm 2 in [PW16a] anyway. It requires that the length of $\mathbf{y}$ is a power of 2, whereas Algorithms 2 and 3 and `fft` can be applied to arbitrary bandwidths N of f or input vector lengths, respectively. In order to be able to better compare these four algorithms, we always consider bandwidths that are of the form $N = 2^J$ in the following numerical experiments.

For sake of completeness we also include the average runtimes of Algorithm 2 in [Iwe10] for the support lengths $B = 10$ and $B = 100$. However, since Algorithms 2 and 3 are different simplifications of the method used therein, we expect them to have significantly shorter runtimes. The primes considered in our two algorithms for functions with short frequency support are essentially slightly altered subsets of the $\widetilde{K} + \tilde{L}$ primes $\tilde{s}_1, \ldots, \tilde{s}_{\widetilde{K}}, \tilde{t}_1, \ldots, \tilde{t}_{\tilde{L}}$ used by Algorithm 2 in [Iwe10] for general B-sparse functions, which computes $\widetilde{K} \cdot \tilde{L}$ CDFTs of length $\tilde{s}_k \tilde{t}_l$, as mentioned in Section 2.2.2. Algorithm 2, on the other hand, needs $L + 1$ CDFTs of length $s \cdot t_l$, where $t_l \approx \tilde{t}_l$ and $s \approx \tilde{s}_1$, whereas Algorithm 3 requires $K < \widetilde{K}$ CDFTs of prime length $s_k \approx \tilde{s}_k$. Thus, for both methods for functions with short frequency support there are less and shorter CDFTs to be computed. This is also supported by the theoretical runtimes

$$\mathcal{O}\left(B \log B \cdot \frac{\log^2 \frac{N}{B}}{\log \log \frac{N}{B}}\right)$$

of Algorithm 2,

$$\mathcal{O}\left(\frac{(B + \log N) \log N}{\log^2 B} \log^2\left(\frac{B + \log N}{\log B}\right)\right)$$

of Algorithm 3 and

$$\mathcal{O}\left(\frac{B^2}{\log^2 B} \cdot \frac{\log^4 N \log^2(B \log N)}{\log \log \frac{N}{B}}\right)$$

of Algorithm 2 in [Iwe10], as well as by the numerical results we will present later on.

All algorithms have been implemented in MATLAB R2016a, and the code is freely available in [Bit17a, Bit17b, PW16b]. For Algorithm 3 we test both methods for finding the first support index of $\widehat{\mathbf{a}^{s_k}}$ discussed in Section 2.4.1. Note that the publicly available code for Algorithm 2 in [PW16a], which we used for the numerical experiments in [Bit17c] and in this section, is not implemented optimally, as its runtime is worse than that of `fft`, which can be seen in Figure 2.2. Theoretically, though, this algorithm has a runtime of $\mathcal{O}(B \log N)$ with a small constant. The numerical experiments in [Wan16], where the runtime performance was not investigated, suggest that the implementation is very stable with respect to noisy input data, which is also supported by our experiments. In order to achieve this level of stability, a higher runtime seems to have been accepted.

Figure 2.2 depicts the average runtimes of Algorithm 2, both versions of Algorithm 3, Algorithm 2 in [PW16a], Algorithm 2 in [Iwe10] and MATLAB's `fft` for 100 randomly generated input functions or vectors, where the absolute values of the real and imaginary parts of the Fourier coefficients or entries, respectively, are bounded by 10. We choose a noise threshold of $\varepsilon = 10^{-4}$ for Algorithm 2 and and the version of Algorithm 3 using block search.

Of course, any comparison of the first four algorithms with the highly optimized implementation `fft` of the FFT must be flawed; however, we note that Algorithm 2 and Algorithm 3 are much faster than both `fft` and Algorithm 2 in [PW16a] for support

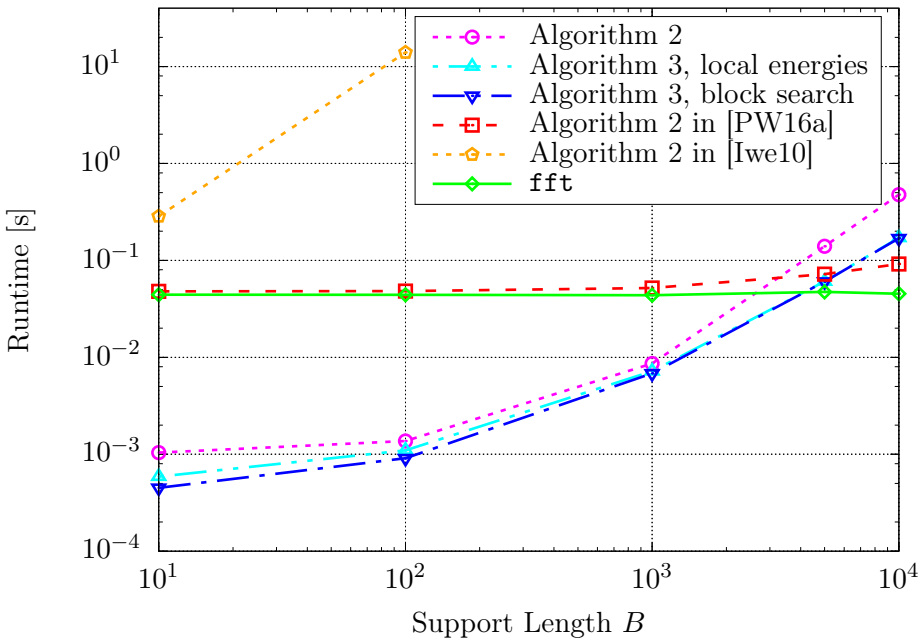

Figure 2.2: Average runtimes of Algorithm 2, Algorithm 3 using local energies and block search, Algorithm 2 in [PW16a], Algorithm 2 in [Iwe10] and MATLAB's `fft` for 100 random input functions with support length B and bandwidth $N = 2^{20}$

lengths up to $B = 1{,}000 \approx \sqrt{N}$, and that Algorithm 2 in [PW16a] is as fast as `fft` for the same support lengths. For greater support lengths, all four methods for functions or vectors with short support perform much slower than `fft`, whose runtime of $\mathcal{O}(N \log N)$ is independent of the support length, whereas the algorithms for short supports have runtimes that are almost linear or linear in B.

Additionally, it can be seen that the runtimes of Algorithm 3 and especially Algorithm 2 increase much faster in the support length B than the runtime of Algorithm 2 in [PW16a]. One can clearly discern that Algorithm 3 slightly improves the runtime of the related Algorithm 2. Even though the block search approach for detecting the first support index ν_{s_k} of $\widehat{\mathbf{a}^{s_k}}$ is somewhat faster up to $B = 1{,}000$, the runtime of Algorithm 3 is basically the same for both methods for finding the first support index if the support length is increased further. If the stabilizing step of computing the Fourier coefficient estimate x_ω in line 21 by averaging is omitted, the runtime of Algorithm 3 can be reduced further. However, as this negatively affects its performance with respect to noise, we choose the stabilized variant for all numerical experiments.

As expected, even for a support of length $B = 100$, the runtime of Algorithm 2 in [Iwe10] is several orders of magnitude greater than the runtime of any of the other considered algorithms, and it also increases much faster in B. Hence, we do not investigate its runtime for greater support lengths and also do not consider its performance with respect to noise. To highlight the fact that Algorithms 2 and 3 are simplifications of Algorithm 2 in [Iwe10], we will still include it in our study of the sampling requirements later on.

Next, we examine the quality of the frequency and coefficient reconstructions for noisy input data. For Algorithm 2, Algorithm 3 and `fft` we assume that we can only sample the perturbed function $f + \eta$, where f has a short frequency support of length at most

B and $\eta \in C_{2\pi}$. Further, η has to satisfy that its vector of Fourier coefficients $\mathbf{c}(\eta) \in \mathbb{C}^N$ is uniformly distributed noise with $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$ for some suitable noise threshold $\varepsilon > 0$. Then Algorithms 2 and 3 and `fft` reconstruct the restriction $\mathbf{c}(N) \in \mathbb{C}^N$ of $\mathbf{c}(f+\eta) \in \mathbb{C}^{\mathbb{Z}}$ to the frequencies contained in $\left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$.

For Algorithm 2 in [PW16a], we create disturbed Fourier data $\widehat{\mathbf{z}} \in \mathbb{C}^N$ by adding uniform noise $\widetilde{\boldsymbol{\eta}} \in \mathbb{C}^N$ to $\widehat{\mathbf{y}}$,

$$\widehat{\mathbf{z}} := \widehat{\mathbf{y}} + \widetilde{\boldsymbol{\eta}}.$$

We measure the noise with the *signal-to-noise ratio (SNR)*,

$$\mathrm{SNR} := 20 \cdot \log_{10} \frac{\|\mathbf{c}(f)\|_2}{\|\mathbf{c}(\eta)\|_2}, \quad \text{and} \quad \mathrm{SNR} := 20 \cdot \log_{10} \frac{\|\widehat{\mathbf{y}}\|_2}{\|\widetilde{\boldsymbol{\eta}}\|_2},$$

respectively. Recall that Algorithm 2 and Algorithm 3 reconstruct the Fourier coefficients of f from function values, and that `fft` is applied to the vector $\mathbf{a}^N$ of N equidistant function values. Algorithm 2 in [PW16a], on the other hand, recovers a vector $\mathbf{y}$ from its Fourier transform $\widehat{\mathbf{y}}$, which means that the output of this algorithm is contained in a different domain. Figures 2.3 and 2.4 depict the average reconstruction errors

$$\frac{\|\mathbf{x} - \mathbf{x}'\|_2}{N},$$

where $\mathbf{x}$ denotes the restriction $\mathbf{c}(N, f)$ of the original finite spectrum $\mathbf{c}(f) \in \mathbb{C}^{\mathbb{Z}}$ to the frequencies contained in $\left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$, or the original vector $\mathbf{y}$, respectively. By $\mathbf{x}'$, we denote the reconstruction by the corresponding algorithm applied to noisy input data for support lengths $B = 100$ and $B = 1{,}000$.

The threshold parameter ε for the block search method for finding the first support index ν_{s_k} of $\widehat{\mathbf{a}^{s_k}}$ is chosen according to Table 2.1. The values for ε, depending on the

SNR	0	10	20	30	40	50
ε	70	35	20	15	2	0.5

Table 2.1: Parameter ε for the block search method in Algorithm 3

SNR and the fact that $|\operatorname{Re}(c_\omega)|$ and $|\operatorname{Im}(c_\omega)|$ are bounded by 10 for all frequencies, were obtained via an attempt to minimize the approximation error. As the afore-mentioned dependencies are nontrivial, we cannot recommend good heuristics for finding ε. If the SNR is known approximately a priori, one can determine good values for ε by applying the algorithm to synthetic data with similar noise levels. If ε is too small, the found support blocks will be too long, as, due to the noise, Fourier coefficients corresponding to non-energetic frequencies will also be included. If ε is too large, the found support block tends to be too short, because coefficients corresponding energetic frequencies might be cut off. In the case of Algorithm 2 we always choose $\varepsilon = 10^{-4}$.

Algorithm 2 in [PW16a] achieves the lowest average reconstruction errors for all considered SNR values and both support lengths. This very high stability was achieved at the cost of an increased runtime. For higher SNR values the average reconstruction errors for both variants of Algorithm 3 are smaller than the one of Algorithm 2, due to taking the mean of the K possible coefficient estimates in line 21 of Algorithm 3. However, for lower SNR values, the error of Algorithm 3 increases up to the error level of `fft`, especially for the block search method, which is only well-suited for low-level

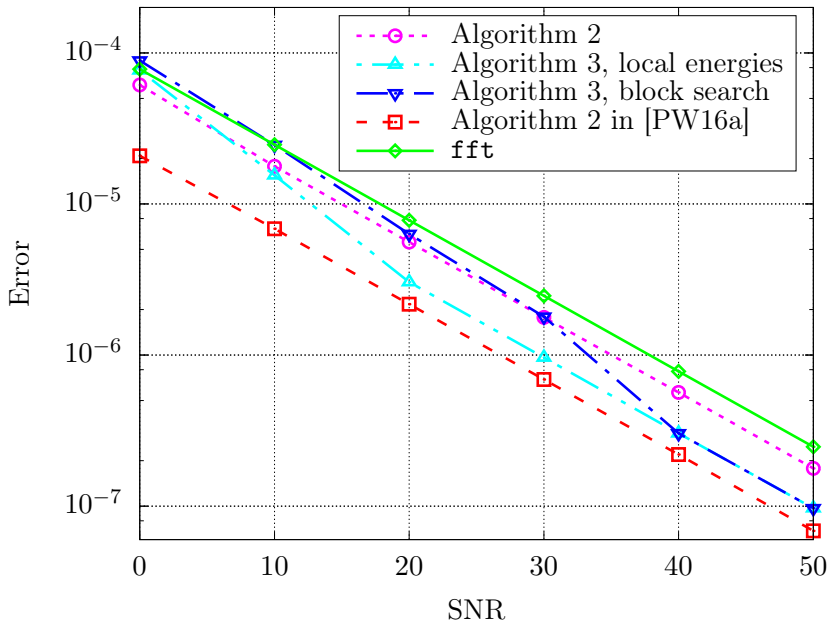

Figure 2.3: Average reconstruction errors $\|\mathbf{x}-\mathbf{x}'\|_2/N$ of Algorithm 2, Algorithm 3 using local energies and block search, Algorithm 2 in [PW16a] and `fft` for 100 random input functions with uniformly distributed noise, $B = 100$, $N = 2^{20}$

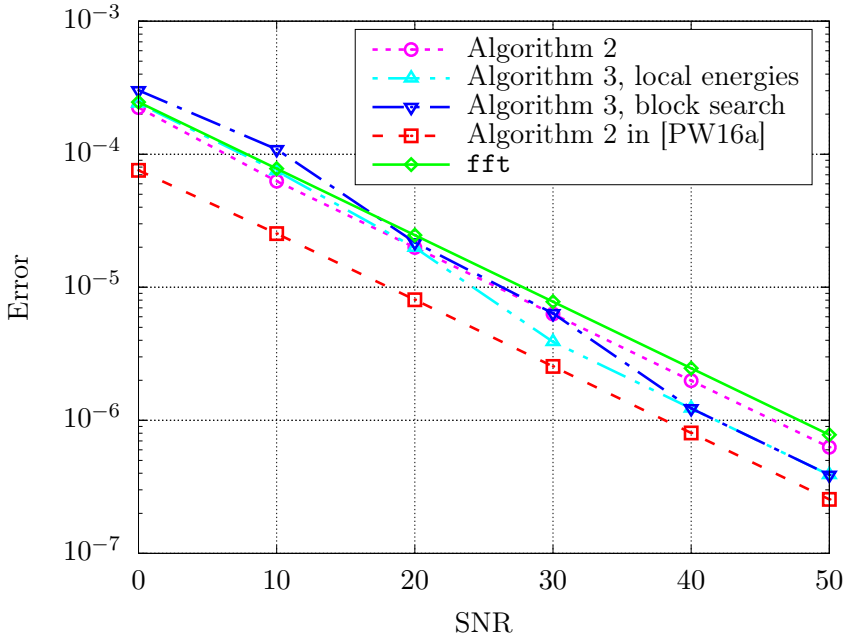

Figure 2.4: Average reconstruction errors $\|\mathbf{x}-\mathbf{x}'\|_2/N$ of Algorithm 2, Algorithm 3 using local energies and block search, Algorithm 2 in [PW16a] and `fft` for 100 random input functions with uniformly distributed noise, $B = 1{,}000$, $N = 2^{20}$

noise. If the first support index of $\widehat{\mathbf{a}^{s_k}}$ is detected using local energies, Algorithm 3 has an approximation error slightly greater than the one of Algorithm 2 in [PW16a] up to an SNR of 20 if $B = 100$, and 30 if $B = 1{,}000$. If one did not take the mean in line 21

of Algorithm 3, its average reconstruction error would be slightly greater than the one of Algorithm 2 and about the size of the error of `fft` for all considered SNR values.

As for some applications it might be important to correctly recover the short support of the function or vector in question, we also investigate whether the considered methods are able to do so. Tables 2.2 and 2.3 show the percentage of correctly found smallest energetic frequencies ω_1 for Algorithm 2, Algorithm 3 using block search and local energies, and Algorithm 2 in [PW16a].

SNR	Rate of Correct Recovery in % Using Algorithm 2	Algorithm 3		Algorithm 2 in [PW16a]
		Local Energies	Block Search	
0	91	18	0	84
10	100	92	65	100
20	100	100	89	100
30	100	100	95	100
40	100	100	100	100
50	100	100	100	100

Table 2.2: Rate of correct recovery of ω_1 in percent for Algorithm 2, Algorithm 3 using local energies and block search and Algorithm 2 in [PW16a] for the 100 random input functions with support length $B = 100$ from Figure 2.3

SNR	Rate of Correct Recovery in % Using Algorithm 2	Algorithm 3		Algorithm 2 in [PW16a]
		Local Energies	Block Search	
0	81	27	0	86
10	99	83	53	100
20	100	97	86	100
30	100	100	91	100
40	100	100	100	100
50	100	100	100	100

Table 2.3: Rate of correct recovery of ω_1 in percent for Algorithm 2, Algorithm 3 using local energies and block search and Algorithm 2 in [PW16a] for the 100 random input functions with support length $B = 1{,}000$ from Figure 2.4

It can be seen that for an SNR of 0, Algorithm 3 using the block search method for detecting the first support index of $\widehat{\mathbf{a}^{s_k}}$ fails to recover ω_1 in all 100 test runs for both support lengths B, and has a rate of correct recovery of 65% if $B = 100$, and 53% if $B = 1{,}000$, for an SNR of 10. If one employs the local energies method, ω_1 will still be found in 18% and 27%, respectively, of the cases for an SNR of 0 and even in 92% and 83% for an SNR of 10.

The problem for both the block search and local energies method is that, in order to compute the smallest energetic frequency ω_1, we have to find its residues modulo the s_k by finding K first support indices. Hence, if the first support index ν_{s_k} of just one vector $\widehat{\mathbf{a}^{s_k}}$ is found incorrectly, the reconstructed ω_1 might deviate much from the true frequency,

resulting in higher reconstruction errors for Algorithm 3. In this respect, Algorithm 2 is much more stable; in our experiments it always recovers the correct frequencies except for an SNR of 0, where the rates of correct recovery are 91% and 81%, respectively, and, for $B = 1{,}000$, for an SNR of 10, with a rate of 99%. For Algorithm 2 in [PW16a], the procedure for identifying the first support index via local energies has to be applied only once and can also be stabilized in a way that is infeasible for Algorithm 3. This results in always correctly identified first support indices, except for an SNR of 0, where the rates of correct recovery are 84% and 86%, respectively. Even though the runtime of its implementation is not optimal, our numerical experiments show that it is highly stable for noisy input data.

Another aspect regarding to which we want to compare Algorithms 2 and 3 are the sampling requirements. If obtaining samples of the input function f requires a lot of resources, e.g., time, money or measuring equipment in practical applications, reducing the number of samples might be more important than minimizing the runtime of the algorithm. The theoretical sampling requirements of

$$\mathcal{O}\left(B \cdot \frac{\log^2 \frac{N}{B}}{\log\log \frac{N}{B}}\right)$$

for Algorithm 2 and

$$\mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B} \log\left(\frac{B+\log N}{\log B}\right)\right)$$

for Algorithm 3 indicate that we should expect Algorithm 3 to use significantly less samples than Algorithm 2. We also investigate this numerically. Figure 2.5 shows the ratio between the number of used samples and the bandwidth $N = 2^{20}$ for varying support lengths for Algorithm 2, Algorithm 3, whose sampling requirements do not depend on the selected method for detecting the first support indices, Algorithm 2 in [PW16a], Algorithm 2 in [Iwe10] and `fft`. One can see that, compared to Algorithm 2, the sampling requirements of Algorithm 3 are an order of magnitude smaller, and of the same size as the ones of Algorithm 2 in [PW16a], which needs $\mathcal{O}(B+\log N)$ equidistant samples of f. Figure 2.5 also illustrates that both Algorithm 2 and 3 require significantly less samples than Algorithm 2 in [Iwe10], of which they are special simplifications.

Summing up the insights gained from the presented numerical examples, we can conclude that Algorithm 3 performs better than Algorithm 2 if the costs for obtaining the samples of the input function are significant or if it is known that the input data is only perturbed by noise with a high SNR. In the second case the runtime can also be slightly decreased by removing the computation of the mean, though this increases the average reconstruction error of the method. As using the block search method only gives a small runtime advantage for support lengths up to $B = 100$ while resulting in less stability for noisy data, we recommend to find the first support indices via local energies, which also do not require any a priori knowledge about the precise noise level.

Being more stable with respect to noise, Algorithm 2 is a better choice for higher noise levels, i.e., lower SNR values, despite its slightly greater runtime. As its runtime is also lower than that of Algorithm 2 in [PW16a] while being similarly robust with respect to noise, Algorithm 2 should also be preferred over the latter method if the sampling requirements are not crucial.

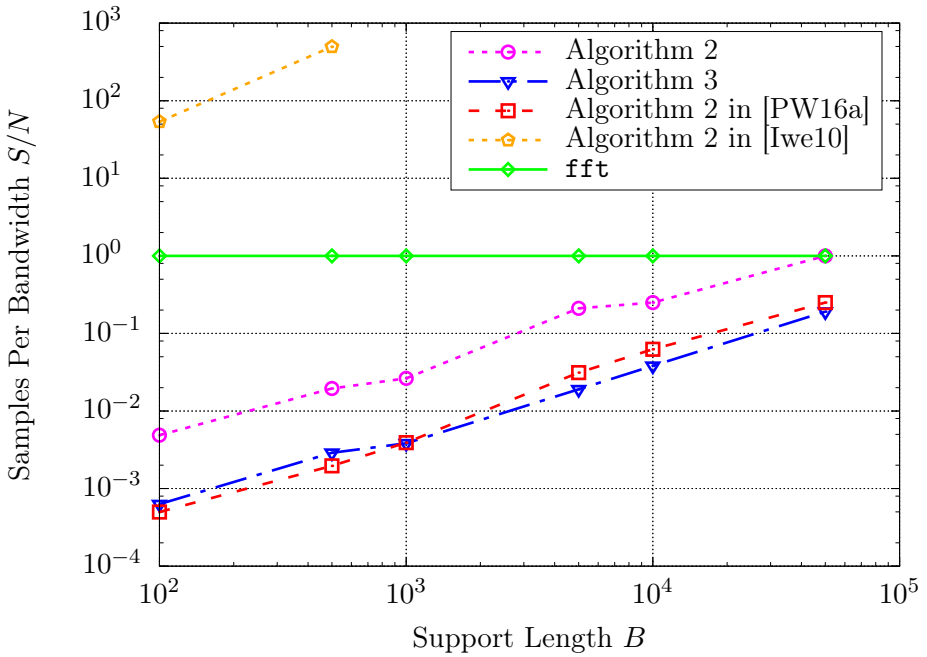

Figure 2.5: Number of used samples per bandwidth $N = 2^{20}$ for varying support lengths B for Algorithm 2, Algorithm 3, Algorithm 2 in [PW16a], Algorithm 2 in [Iwe10] and `fft`

3 Sparse FFT for 2π-Periodic Functions with Polynomially Structured Sparsity

In Sections 2.3 and 2.4 we introduced two algorithms for deterministically recovering a function $f \in C_{2\pi}$ with short support $S = \{\omega_1, \omega_1 + 1, \ldots, \omega_1 + B - 1\}$ of length at most B from samples. In this chapter, which is based on our paper [BZI19] and is in parts identical with the representations therein, we want to focus on more general structures. To be more precise, we assume that the energetic frequencies of f are contained in a small number, n, of support sets $S_1, \ldots, S_n \subsetneq \{-\lceil \frac{N}{2} \rceil + 1, \ldots, \lfloor \frac{N}{2} \rfloor\}$, and that each of the unknown sets S_j is generated by evaluating a polynomial P_j of degree at most d at B points. This means that f is of the form

$$f(x) = \sum_{j=1}^{n} \sum_{\omega \in S_j} c_\omega(f) e^{i\omega x},$$

where

$$S_j = \{P_j(x) : x \in \{1, \ldots, B\}\}$$

for some polynomial P_j of degree at most d with integer coefficients. The perhaps simplest class of functions with such a frequency structure are the functions with short frequency support of length B studied in Chapter 2, which can be obtained by setting $n = 1$ and generating the set S_1 by evaluating the polynomial $P_1(x) = x + \omega_1 - 1$ at $x \in \{1, \ldots, B\}$.

The generalization to block sparse functions, where each of the n support sets is generated by evaluating a linear, monic polynomial at B points, is of importance in many signal processing problems, for example the reconstruction of multiband signals via blind sampling at sub-Nyquist rates, see, e.g., [FB96, ME10, MET08, MEDS11, ME09]. Another application is the fast and efficient evaluation of functions that can be represented as a sparse expansion of other orthonormal basis functions, like Legendre or Gegenbauer, see, e.g. [PT16]. For example, functions that are a sparse combination of high-degree Legendre polynomials can be approximated via computing DFTs of samples of an auxiliary periodic function, which can be shown to be block frequency sparse, see [HIK17].

Our aim in this chapter is to develop an algorithm that deterministically recovers a general polynomially structured sparse function from samples by generalizing the reconstruction ideas introduced in [Iwe10, Iwe13], which we also briefly outlined in Section 2.1. Unlike in Sections 2.3 and 2.4, we cannot hope to obtain such a method by simplifying Algorithm 2 in [Iwe10].

3.1 Polynomially Structured Sparsity

As in Chapter 2, we always consider a 2π-periodic function $f \in C_{2\pi}$ with finite spectrum $\mathbf{c}(f) \in \mathbb{C}^{\mathbb{Z}}$ and a large bandwidth $N \in \mathbb{N}$. Again, we assume that we can only evaluate the perturbed function $f + \eta$, where $\eta \in C_{2\pi}$ satisfies $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$ for some suitable noise threshold $\varepsilon > 0$. We denote by $\mathbf{c}(N) \in \mathbb{C}^N$ the restriction of the finite spectrum $\mathbf{c}(f + \eta) \in \mathbb{C}^{\mathbb{Z}}$ of $f + \eta$ to the frequencies contained in $\{-\lceil \frac{N}{2} \rceil + 1, \ldots, \lfloor \frac{N}{2} \rfloor\}$,

and by $\mathbf{c}(N,\mathbb{Z}) \in \mathbb{C}^{\mathbb{Z}}$ the embedding of $\mathbf{c}(N)$ into $\mathbb{C}^{\mathbb{Z}}$,

$$(\mathbf{c}(N,\mathbb{Z}))_\omega = \begin{cases} c_\omega(f+\eta) & \text{if } \omega \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}, \\ 0 & \text{otherwise.} \end{cases}$$

In the case of exact data, i.e., $\eta \equiv 0$, we denote by $\mathbf{c}(N,f) \in \mathbb{C}^N$ the restriction of $\mathbf{c}(f)$ to the frequencies in $\left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$.

We begin by formally defining the concept of polynomially structured frequency sparsity mentioned above.

Definition 3.1 ($P(n,d,B)$-structured Sparsity (Definition 2.3 in [BZI19])) Let B, d, n and $N \in \mathbb{N}$ such that $d < B < N$. Let $P_1, \dots, P_n \in \mathbb{Z}[x]$ be non-constant polynomials of degree at most d with

$$P_j(x) = \sum_{k=0}^{d} a_{jk} x^k,$$

where $a_{jk} \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$ such that for all $j \in \{1,\dots,n\}$ and $x \in \{1,\dots,B\}$ we have that $P_j(x) \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$. The n *support sets* are defined as

$$S_j \coloneqq \{P_j(x) : x \in \{1,\dots,B\}\},$$

and we set $S \coloneqq \bigcup_{j=1}^{n} S_j$. Let $\varepsilon > 0$ be a suitably chosen noise threshold. If $f \in C_{2\pi}$ is Bn-sparse such that all of its energetic frequencies are contained in S, it is called *P(n,d,B)-structured sparse (polynomially structured sparse)*. Then f is of the form

$$f(x) = \sum_{\omega \in S} c_\omega(f) e^{i\omega x}$$

for some vector of Fourier coefficients $(c_\omega(f))_{\omega \in S} \in \mathbb{C}^{Bn}$.

Polynomially structured sparsity means that the at most Bn energetic frequencies of the function f are generated by evaluating n polynomials with integer coefficients of degree at most d at B points. The following example illustrates this concept.

Example 3.2 Let $N = 1{,}024$, $n = d = 2$ and $B = 9$. We choose the polynomials

$$P_1(x) = 11x^2 - 22x - 200 \qquad \text{and} \qquad P_2(x) = -13x^2 + 26x + 350.$$

The support sets generated by P_1 and P_2 are

$$S_1 = \{-211, -200, -167, -112, -35, 64, 185, 328, 493\} \qquad \text{and}$$
$$S_2 = \{-469, -274, -105, 38, 155, 246, 311, 350, 363\}.$$

Then, with $S \coloneqq S_1 \cup S_2$, the function

$$f(x) = \sum_{\omega \in S} e^{i\omega x}$$

is $P(2,2,9)$-structured sparse with $c_\omega(f) = 1$ for all frequencies $\omega \in S$ and $c_\omega(f) = 0$ for all $\omega \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\} \setminus S$. ◊

As general polynomially structured sparsity is a more complex construct than the short support considered in Chapter 2, using separating primes does not suffice here. Instead, we introduce the concept of a *good hashing prime*; a prime modulo which not all frequencies in a support set S_j have the same residue.

Definition 3.3 (Definition 2.3 in [BZI19]) Let f be $P(n, d, B)$-structured sparse with bandwidth N, noise threshold $\varepsilon > 0$ and support set $S = \bigcup_{j=1}^{n} S_j$ generated by the polynomials $P_1, \ldots, P_n$. A prime $p > B$ *hashes a support set* S_j *well* if the cardinality of the set obtained by taking the residues modulo p of all elements of S_j is greater than 1, i.e., if

$$|\{\omega \mod p : \omega \in S_j\}| > 1.$$

Example 3.4 (Example 3.2 continued) We consider the same support sets

$$S_1 = \{-211, -200, -167, -112, -35, 64, 185, 328, 493\} \quad \text{and}$$
$$S_2 = \{-469, -274, -105, 38, 155, 246, 311, 350, 363\}$$

as in Example 3.2. Then we have that

$$\{\omega \mod 11 : \omega \in S_1\} = \{9\} \quad \text{and} \quad \{\omega \mod 11 : \omega \in S_2\} = \{0, 1, 3, 4, 5, 9\}.$$

Consequently, 11 does not hash S_1 well, but it hashes S_2 well. ◊

Whether a prime hashes a support set well or not can be easily deduced from the coefficients of the generating polynomial using the following well-known generalization of the fundamental theorem of algebra.

Theorem 3.5 *Let k be a field and P a polynomial in one variable x in $k[x]$, of degree $d \geq 0$. Then P has at most d roots in k.*

For a proof see [Lan05], Chapter IV, §1, Theorem 1.4. With the help of Theorem 3.5 we can now prove the following result.

Lemma 3.6 (Lemma 2.5 in [BZI19]) Let f be $P(n, d, B)$-structured sparse with bandwidth N, noise threshold $\varepsilon > 0$ and support set $S = \bigcup_{j=1}^{n} S_j$ generated by the polynomials $P_1, \ldots, P_n$. Then a prime $p > B$ hashes a support set S_j with generating polynomial

$$P_j(x) = \sum_{k=0}^{d} a_{jk} x^k \in \mathbb{Z}[x]$$

well if and only if there exists a coefficient a_{jk} with $k \neq 0$ such that $p \nmid a_{jk}$.

Proof. Assume $p | a_{jk}$ for all $k \in \{1, \ldots, d\}$. Then we have for all $x \in \{1, \ldots, B\}$ that

$$P_j(x) = \sum_{k=0}^{d} a_{jk} x^k \equiv a_{j0} \mod p.$$

As S_j is generated by evaluating P_j at $x \in \{1, \ldots, B\}$, we obtain that

$$|\{\omega \mod p : \omega \in S_j\}| = 1,$$

so p does not hash S_j well.

If, on the other hand, p does not hash S_j well, then

$$|\{\omega \bmod p : \omega \in S_j\}| = 1.$$

This implies that

$$P_j(y) \equiv P_j(z) \bmod p \quad \forall\, y, z \in \{1, \dots, B\}.$$

Hence, for fixed $y \in \{1, \dots, B\}$, the polynomial

$$Q(x) \coloneqq P_j(x) - P_j(y) = \sum_{k=0}^{d} a_{jk} x^k - P_j(y)$$

of degree d has $B > d$ zeroes modulo p. By Theorem 3.5, Q is the zero polynomial modulo p, so

$$p \mid (a_{j0} - P_j(y)) \quad \text{and} \quad p \mid a_{jk} \quad \forall\, j \in \{1, \dots, d\},$$

which is a contradiction. □

For a good hashing prime and a $P(n, d, B)$-structured sparse function we can bound the number of energetic frequencies that are hashed to the same residue by the maximal polynomial degree d.

Lemma 3.7 (Lemma 2.6 in [BZI19]) Let f be $P(n, d, B)$-structured sparse with bandwidth N, noise threshold $\varepsilon > 0$ and support set $S = \bigcup_{j=1}^{n} S_j$ generated by the polynomials $P_1, \dots, P_n \in \mathbb{Z}[x]$. If a support set S_j is hashed well by a prime $p > B$, then

(i) P_j is not a constant polynomial modulo p,

(ii) $|\{\omega \in S_j : \omega \equiv \nu \bmod p\}| \leq d$ for all residues $\nu \in \{0, \dots, p-1\}$.

Proof. It is clear that P_j cannot be constant modulo p if $|\{\omega \bmod p : \omega \in S_j\}| > 1$. Assume now that $|\{\omega \in S_j : \omega \equiv \nu \bmod p\}| > d$ for some $\nu \in \{0, \dots, p-1\}$. Since all elements of S_j are generated by evaluating P_j at some $x \in \{1, \dots, B\}$, we find for a $y \in \{1, \dots, B\}$ with $P_j(y) \equiv \nu \bmod p$ that

$$P_j(y) \equiv P_j(z) \bmod p$$

for d distinct choices of $z \in \{1, \dots, B\} \setminus \{y\}$. Then the polynomial $Q(x) \coloneqq P_j(x) - P_j(y)$ has at least $d + 1$ zeroes modulo p. By Theorem 3.5 this is a contradiction. □

Example 3.8 (Example 3.2 continued) We consider the same support sets as in Example 3.2. Recall that

$$S_2 = \{-469, -274, -105, 38, 155, 246, 311, 350, 363\},$$

the support set generated by P_2, is well-hashed by 11. Indeed, P_2 is not a constant polynomial modulo 11, as

$$P_2(x) \equiv \left(9x^2 + 4x + 9\right) \bmod 11 \not\equiv c \bmod 11$$

for all $c \in \{0, \dots, 10\}$. Further, we obtain the following vector of residues modulo 11 of the elements of S_2,

$$(\omega \bmod 11)_{\omega \in S_2} = (4, 1, 5, 5, 1, 4, 3, 9, 0)^T \in \mathbb{N}_0^{|S_2|}.$$

Hence, for each residue ν modulo 11 there are at most two elements in S_2 that are congruent to ν,

$$|\{\omega \in S_2 : \omega \equiv \nu \mod 11\}| = \begin{cases} 0 & \text{if } \nu \in \{2,6,7,8,10\}, \\ 1 & \text{if } \nu \in \{0,3,9\}, \\ 2 & \text{if } \nu \in \{1,4,5\}. \end{cases}$$

$\Diamond$

Let us now assume that there exists a prime $p > B$ that hashes all support sets $S_1, \dots, S_n$ of a $P(n,d,B)$-structured sparse function f well. Then the restriction of any support set S_j to the frequencies congruent to ν modulo p contains at most d elements by Lemma 3.7 (ii) and this holds for all residues $\nu \in \{0, \dots, p-1\}$. Consequently, the restriction of S to these frequencies contains at most dn elements, and a function whose support set is the restriction of S is dn-sparse. Thus, we will from now on also refer to support sets as *sparse* if the corresponding functions are sparse.

Instead of developing a completely new algorithm, we can now employ existing methods for reconstructing the restricted functions, which are much sparser than f. Due to the structure of the problem, Algorithm 2 in [Iwe10] and Algorithm 3 in [Iwe13] are especially well-suited for this task. In theory, though, any sparse FFT algorithm for recovering a function $f \in C_{2\pi}$ from samples could be considered. It can be shown both theoretically and numerically that Algorithm 2 in [Iwe10] and Algorithm 3 in [Iwe13], which are essentially the same method, are very efficient for very small sparsities. This is also apparent from the numerical experiments we performed in Section 2.5, see Figure 2.2. The main idea of our approach for reconstructing a function with polynomially structured sparsity from samples is to apply Algorithm 3 in [Iwe13] to the restrictions to frequencies congruent to ν modulo u for all residues ν, where u is a prime that hashes all support sets well, since these restrictions are at most dn-sparse. If this procedure is performed for all residues, we find all energetic frequencies and their Fourier coefficients. By applying Algorithm 3 in [Iwe13] only to the dn-sparse restriction instead of the Bn-sparse function, we can indeed reduce the overall runtime, as the runtime of Algorithm 3 in [Iwe13] scales quadratically in the sparsity, which we already mentioned in Section 2.2.2. We will prove theoretical bounds on the runtime and sampling complexities of our new algorithm in Section 3.3.3.

However, finding a single prime u that hashes all n support sets well is, in general, not possible without further information on the generating polynomials. In the remainder of this section we will show how to find M primes such that the majority of them hashes all support sets well. The existence of such primes can be proven with the help of the CRT, see Theorem 2.6, and the concept of separation, which we already introduced in Definition 2.9. Recall that $u \in \mathbb{N}$ separates the integers $\omega_1, \dots, \omega_B$ if their residues modulo u are all distinct. The correct energetic frequencies can then be found using median arguments.

Example 3.9 (Example 3.2 continued) For the support sets from Example 3.2 we find, e.g., that

$$(\omega \mod 17)_{\omega \in S_2} = (7, 15, 14, 4, 2, 8, 5, 10, 6)^T,$$

so 17 separates the elements of S_2. $\Diamond$

The following result about separating primes has been shown in Lemma 1 in [Iwe10]. Note that, according to Definition 2.17, we denote the rth prime by p_r.

Lemma 3.10 Let $E, N \in \mathbb{N}$, $E < N$, and $u_1 \coloneqq p_r$ for some $r \in \mathbb{N}$. We define $M = 2 \cdot E \cdot \lfloor \log_{u_1} N \rfloor + 1$. Choose $M-1$ further primes with $u_1 < \cdots < u_M$, and let $T \subsetneq \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}$ with $|T| \leq E$. Then more than $\frac{M}{2}$ of the u_m separate every $x \in \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}$ from all $t \in T \setminus \{x\}$.

In the next lemma we prove that, for a suitable M, it suffices to find M primes such that more than half of them separate the leading coefficients of the frequency generating polynomials from 0 at the same time in order to guarantee that more than half of these primes hash all support sets well.

Lemma 3.11 (Lemma 2.13 in [BZI19]) Let f be $P(n, d, B)$-structured sparse with bandwidth N, noise threshold $\varepsilon > 0$ and support set $S = \bigcup_{j=1}^{n} S_j$ generated by the polynomials $P_1, \ldots, P_n$, and set $E = n+1$. Let M primes $B < u_1 < \cdots < u_M$ satisfying Lemma 3.10 be given. Then more than $\frac{M}{2}$ of the u_m hash all n support sets $S_1, \ldots, S_n$ well.

Proof. Let T be the set consisting of the leading coefficients of the polynomials $P_1, \ldots, P_n$ that generate the support sets $S_1, \ldots, S_n$, i.e.,

$$T \coloneqq \left\{ a_{j,\deg(P_j)} : P_j(x) = \sum_{k=0}^{\deg(P_j)} a_{jk} x^k,\ j \in \{1, \ldots, n\} \right\}.$$

By definition $a_{j,\deg(P_j)} \neq 0$ for all polynomials and, since $|T \cup \{0\}| \leq E$, by Lemma 3.10 more than $\frac{M}{2}$ of the u_m separate every element of $\left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}$ from all other elements of $T \cup \{0\}$, i.e., from all distinct leading polynomial coefficients and from 0.

Let $p = u_m$ be one of these separating primes for an $m \in \{1, \ldots, M\}$. Assume that there exists a support set S_j that is not well hashed by p, so that we have

$$\{\omega \bmod p : \omega \in S_j\} = \{\nu\}$$

for some residue $\nu \in \{0, \ldots, p-1\}$. Then the polynomial P_j that generates S_j satisfies

$$P_j(x) - \nu \equiv 0 \bmod p \qquad \forall\, x \in \{1, \ldots, B\}.$$

Consider now the polynomial $Q(x) \coloneqq P_j(x) - \nu$ modulo p. It is a polynomial of degree at most d with $B > d$ zeroes modulo p, so by Theorem 3.5 it has to be the zero polynomial modulo p. Consequently, we obtain that

$$p | a_{jk} \quad \forall\, k \in \{1, \ldots, d\} \quad \text{and} \quad p | (a_{j0} - \nu),$$

so $a_{j,\deg(P_j)} \equiv 0 \bmod p$. Since p separates $a_{j,\deg(P_j)}$ from 0 and the other leading coefficients, we also find that

$$a_{j,\deg(P_j)} \not\equiv t \bmod p \quad \forall\, t \in (T \cup \{0\}) \setminus \left\{ a_{j,\deg(P_j)} \right\}.$$

This is only possible if $a_{j,\deg(P_j)} = 0$, which is a contradiction. Thus, it follows that

$$|\{\omega \bmod p : \omega \in S_j\}| > 1,$$

so p hashes all S_j well. Hence, all of the more than $\frac{M}{2}$ primes $u_1, \ldots, u_M$ that separate the leading coefficients from one another and from 0 also hash all support sets well. □

Example 3.12 (Example 3.2 continued) For the $P(2,2,9)$-structured sparse function f from Example 3.2 we can choose $u_1 = 11 > B$ and require

$$M = 2 \cdot 3 \lfloor \log_{11} N \rfloor + 1 = 13$$

primes in total, e.g., $u_1 = 11$, $u_2 = 13, \ldots, u_{13} = 59$. According to Lemma 3.11, at least seven of these primes have to hash S_1 and S_2 well. In fact, all u_m except $u_1 = 11$ and $u_2 = 13$ hash both support sets well. ◊

Lemmas 3.10 and 3.11 imply that, for M suitably chosen primes $u_1, \ldots, u_M$, the restriction of the input function to frequencies congruent to ν modulo u_m is at most dn-sparse for the majority of the u_m. Consequently, since Algorithm 3 in [Iwe13] recovers the dn most energetic frequencies and gives accurate estimates for their Fourier coefficients if the restriction is at most dn-sparse, applying it to the restriction to the frequencies congruent to ν modulo u_m yields the correct frequencies and Fourier coefficients for the majority of the u_m.

Having thus established a foundation for our main idea, we now have to formalize the method in order to be able to determine the required samples and the runtime of the method. First, however, we will look at Algorithm 3 in [Iwe13] more closely.

3.2 Methodical Background

In this section we will explain Algorithm 2 in [Iwe10] and Algorithm 3 in [Iwe13] in more detail, and summarize some of the results proven in the respective papers. Note that both algorithms are essentially the same method, but use different notation. Algorithm 3 in [Iwe13] transfers the approach from Algorithm 2 in [Iwe10], which we briefly outlined in Section 2.2, to a matrix setting, thus allowing for more efficient computations and better error bounds. The main ideas remain unchanged.

Both algorithms reconstruct the B most energetic frequencies and produce accurate estimates for the corresponding Fourier coefficients of a sparse function $f \in C_{2\pi}$ with bandwidth N from the CDFTs of the vectors $\mathbf{a}^{\tilde{s}_k \tilde{t}_l}$, $k \in \left\{1, \ldots, \widetilde{K}\right\}$, $l \in \left\{0, \ldots, \tilde{L}\right\}$ of equidistant samples of f, where $\tilde{s}_k$ and $\tilde{t}_l$ are small primes depending on the bandwidth and sparsity of the function. The energetic frequencies are reconstructed from their residues modulo $\tilde{s}_k$ and $\tilde{t}_1, \ldots, \tilde{t}_{\tilde{L}}$ for all k with the help of the CRT reconstruction, see Algorithm 1, which is why the primes have to satisfy

$$\prod_{l=1}^{\tilde{L}-1} \tilde{t}_l < \frac{N}{\tilde{s}_1} \leq \prod_{l=1}^{\tilde{L}} \tilde{t}_l.$$

For a general B-sparse input function the method introduced in [Iwe10, Iwe13] cannot be guaranteed to work for a single prime $\tilde{s}_k$. However, setting $\widetilde{K} = 8B \lfloor \log_{\tilde{s}_1} N \rfloor + 1$ and choosing $\tilde{s}_1, \ldots, \tilde{s}_{\widetilde{K}}$ as the $\widetilde{K}$ smallest primes greater than B and $\tilde{t}_{\tilde{L}}$, all energetic frequencies are correctly reconstructed from their residues for more than $\widetilde{K}/2$ of them. The residues modulo $\tilde{t}_l$ can be found by comparing the entries of $\widehat{\mathbf{a}^{\tilde{s}_k}}$ and $\widehat{\mathbf{a}^{\tilde{s}_k \tilde{t}_l}}$ that correspond to the same frequency. The coefficient estimates are then obtained by taking the medians over the $\widetilde{K}$ coefficient estimates found for each of the $\tilde{s}_k$.

3.2.1 Measurement Matrices I

In [Iwe13], the concept of measurement matrices was introduced in order to facilitate the notation of the computations briefly outlined in Section 2.2.2. As we want to use the same notation for our method later on, we explain this construction here in more detail. For the definition of measurement matrices we first require the definition of the row-wise Hadamard tensor product.

Definition 3.13 (Row-wise Hadamard Product) Let $\mathbf{A} = (a_{k,l})_{k,l=0}^{\kappa-1,m-1} \in \mathbb{C}^{\kappa\times m}$, $\mathbf{B} = (b_{k,l})_{k,l=0}^{\lambda-1,m-1} \in \mathbb{C}^{\lambda\times m}$. Then the row-wise Hadamard product $\mathbf{A} \circledast \mathbf{B} \in \mathbb{C}^{(\kappa\cdot\lambda)\times m}$ is given by

$$(\mathbf{A} \circledast \mathbf{B})_{k,l} \coloneqq a_{k \bmod \kappa, l} \cdot b_{\frac{k-(k \bmod \kappa)}{\kappa}, l}, \quad k \in \{0,\dots,\kappa\lambda-1\},\, l \in \{0,\dots,m-1\},$$

i.e., the first κ rows of $\mathbf{A} \circledast \mathbf{B}$ are given as the Hadamard product of all κ rows of $\mathbf{A}$ with the first row of $\mathbf{B}$, the second κ rows as the Hadamard product of all κ rows of $\mathbf{A}$ with the second row of $\mathbf{B}$ and so forth.

The following property of the Hadamard product follows directly from Definition 3.13.

Lemma 3.14 Let $\mathbf{A} \in \mathbb{C}^{\kappa\times m}$, $\mathbf{B} \in \mathbb{C}^{\lambda\times m}$. Then every row of $\mathbf{A} \circledast \mathbf{B}$ is given as the row tensor product of a row of $\mathbf{A}$ with a row of $\mathbf{B}$.

Now we can define the measurement matrices required for Algorithm 3 in [Iwe13]. We choose the necessary primes $\tilde{s}_k$ and $\tilde{t}_l$ in the way we already indicated at the beginning of Section 3.2. We want to apply the CRT reconstruction from Algorithm 1 to the residues modulo $\tilde{t}_1,\dots,\tilde{t}_{\tilde{L}}$ and $\tilde{s}_k$ for all k. Hence, we need that

$$\tilde{s}_k \prod_{l=1}^{\tilde{L}-1} \tilde{t}_l < N \le \tilde{s}_k \prod_{l=1}^{\tilde{L}} \qquad \forall\, k \in \left\{1,\dots,\widetilde{K}\right\}.$$

In order to be able to choose the $\tilde{t}_l$ independently of the $\tilde{s}_k$, we use that $\tilde{s}_k$ has to be greater than B for all k.

Definition 3.15 (Measurement Matrices I ((5) in [Iwe13])) Let $B, N, \epsilon^{-1} \in \mathbb{N}\backslash\{1\}$ with $B < N$. Let $\tilde{t}_1,\dots,\tilde{t}_{\tilde{L}}$ be the smallest primes such that

$$B \prod_{l=1}^{\tilde{L}-1} \tilde{t}_l < N \le B \prod_{l=1}^{\tilde{L}} \tilde{t}_l.$$

For algorithmic purposes we additionally set $t_0 \coloneqq 1$. Let $\tilde{s}_1$ be the smallest prime greater than $\max\left\{\tilde{t}_{\tilde{L}}, B\right\}$ and let $\widetilde{K} = 4\frac{B}{\epsilon}\lfloor \log_{\tilde{s}_1} N \rfloor + 1$. Set $\tilde{s}_2,\dots,\tilde{s}_{\widetilde{K}}$ as the smallest $\widetilde{K}-1$ primes greater than $\tilde{s}_1$. Furthermore, we define $\tilde{\kappa} \coloneqq \sum_{k=1}^{\widetilde{K}} \tilde{s}_k$, $\tilde{\lambda} \coloneqq \sum_{l=0}^{\tilde{L}} \tilde{t}_l$ and $\tilde{q} \coloneqq \operatorname{lcm}\{N, \tilde{s}_1,\dots,\tilde{s}_{\widetilde{K}}, \tilde{t}_1,\dots,\tilde{t}_{\tilde{L}}\}$, where $\operatorname{lcm}\{a,b\}$ denotes the least common multiple of a and b for any $a, b \in \mathbb{Z}$.

We define a special $\tilde{\kappa}\times N$ *measurement matrix*. All entries of this matrix will be either one or zero. It is built up row-wise, where an entry of the jth row is one if and only if its column index l is congruent to a certain residue modulo $\tilde{s}_k$. For a formal definition we fix an index $k \in \{1,\dots,\widetilde{K}\}$ and a residue $\nu \in \left\{-\left\lceil \frac{\tilde{s}_k}{2}\right\rceil + 1,\dots,\left\lfloor \frac{\tilde{s}_k}{2}\right\rfloor\right\}$ modulo $\tilde{s}_k$. Then

we define the row $\mathbf{r}_{\tilde{s}_k,\nu} \in \{0,1\}^{1\times N}$ corresponding to the residue ν modulo $\tilde{s}_k$ by

$$\left(\mathbf{r}_{\tilde{s}_k,\nu}\right)_j \coloneqq \delta_{(j-\nu) \bmod \tilde{s}_k,0} \coloneqq \begin{cases} 1 & \text{if } j \equiv \nu \bmod \tilde{s}_k, \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

for $j \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$, where $\delta_{k,l}$ denotes the Kronecker delta. Combining for each prime $\tilde{s}_k$ the $\tilde{s}_k$ possible rows, we set

$$\mathcal{M}_{\tilde{s}_1,\widetilde{K}} \coloneqq \begin{pmatrix} \mathbf{r}_{\tilde{s}_1,-\left\lceil \frac{\tilde{s}_1}{2} \right\rceil+1} \\ \vdots \\ \mathbf{r}_{\tilde{s}_1,\left\lfloor \frac{\tilde{s}_1}{2} \right\rfloor} \\ \mathbf{r}_{\tilde{s}_2,-\left\lceil \frac{\tilde{s}_2}{2} \right\rceil+1} \\ \vdots \\ \mathbf{r}_{\tilde{s}_{\widetilde{K}},\left\lfloor \frac{\tilde{s}_{\widetilde{K}}}{2} \right\rfloor} \end{pmatrix} \in \{0,1\}^{\tilde{\kappa}\times N}.$$

In order to be able to utilize $\mathcal{M}_{\tilde{s}_1,\widetilde{K}}$ in our setting, its number of columns has to be divisible by all $\tilde{s}_k$ and $\tilde{t}_l$. Hence, we define the extension $\mathcal{E}_{\widetilde{K}}$ of $\mathcal{M}_{\tilde{s}_1,\widetilde{K}}$ to a $\tilde{\kappa} \times \tilde{q}$ matrix by extending all rows $\mathbf{r}_{\tilde{s}_k,\nu}$ to columns indexed by $j \in \left\{-\left\lceil \frac{\tilde{q}}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{\tilde{q}}{2} \right\rfloor\right\}$, as in (3.1). Then $\mathcal{E}_{\widetilde{K}}$ is the first measurement matrix employed by the algorithm. The middle N columns of $\mathcal{E}_{\widetilde{K}}$ are just $\mathcal{M}_{\tilde{s}_1,\widetilde{K}}$, and, since $\tilde{s}_1, \ldots, \tilde{s}_{\widetilde{K}}$ all divide $\tilde{q}$, we have that

$$\mathcal{E}_{\widetilde{K}} = \left(\begin{array}{cc|cccc|cc} \ldots & \ldots & \multicolumn{4}{c|}{\mathbf{I}_{\tilde{s}_1} \quad \mathbf{I}_{\tilde{s}_1} \quad \mathbf{I}_{\tilde{s}_1} \quad \mathbf{I}_{\tilde{s}_1}} & \ldots & \ldots \\ \hline \ldots & \ldots & \multicolumn{2}{c}{\mathbf{I}_{\tilde{s}_2}} & \multicolumn{2}{c}{\mathbf{I}_{\tilde{s}_2} \quad \mathbf{I}_{\tilde{s}_2}} & \ldots & \ldots \\ \hline \vdots & & \multicolumn{2}{c}{\vdots} & \multicolumn{2}{c}{\vdots \quad \vdots} & & \vdots \\ \vdots & & \multicolumn{2}{c}{\vdots} & \multicolumn{2}{c}{\vdots \quad \vdots} & & \vdots \\ \hline \ldots & \ldots & \multicolumn{4}{c}{\mathbf{I}_{\tilde{s}_{\widetilde{K}}}} & \ldots & \ldots \end{array}\right),$$

where $\mathbf{I}_{\tilde{s}_k}$ denotes the identity matrix of size $\tilde{s}_k \times \tilde{s}_k$.

Further, we analogously define the $(\tilde{\lambda} - 1) \times N$ matrix $\mathcal{M}_{\tilde{t}_1,\tilde{L}}$, consisting of the rows corresponding to the possible residues modulo $\tilde{t}_1, \ldots, \tilde{t}_{\tilde{L}}$,

$$\mathcal{M}_{\tilde{t}_1,\tilde{L}} \coloneqq \begin{pmatrix} \mathbf{r}_{\tilde{t}_1,-\left\lceil \frac{\tilde{t}_1}{2} \right\rceil+1} \\ \vdots \\ \mathbf{r}_{\tilde{t}_{\tilde{L}},\left\lfloor \frac{\tilde{t}_{\tilde{L}}}{2} \right\rfloor} \end{pmatrix}.$$

Let us denote by $\mathbf{1}_M \in \mathbb{C}^M$ the vector consisting of M ones. We set $\mathcal{N}_{\tilde{t}_1,\tilde{L}}$ to be the $\tilde{\lambda}\times N$ matrix whose first row contains only ones and whose other rows are given by $\mathcal{M}_{\tilde{t}_1,\tilde{L}}$,

$$\mathcal{N}_{\tilde{t}_1,\tilde{L}} \coloneqq \begin{pmatrix} \mathbf{1}_N \\ \mathcal{M}_{\tilde{t}_1,\tilde{L}} \end{pmatrix},$$

and define the $(\tilde{\kappa}\cdot\tilde{\lambda})\times N$ row-wise Hadamard product $\mathcal{R}_{\tilde{L},\widetilde{K}} \coloneqq \mathcal{M}_{\tilde{s}_1,\widetilde{K}} \circledast \mathcal{N}_{\tilde{t}_1,\tilde{L}}$ of $\mathcal{M}_{\tilde{s}_1,\widetilde{K}}$ and $\mathcal{N}_{\tilde{t}_1,\tilde{L}}$ and its extension $\mathcal{G}_{\tilde{L},\widetilde{K}}$ to a $(\tilde{\kappa}\cdot\tilde{\lambda})\times\tilde{q}$ matrix, which is the second measurement matrix used by the algorithm. Recall that by definition, all $\tilde{s}_k$ and $\tilde{t}_l$ divide $\tilde{q}$. Later on, $\tilde{q}$ will be the length of the vector of equidistant samples of f from which the algorithm chooses only a few. Note that $\tilde{q}$ is even since $\tilde{t}_1 = 2$.

The measurement matrices have been chosen as above in order to be able to write down all congruencies occurring in the reconstruction procedure described in Section 2.2 in an easy way. The following property has been shown in Lemma 5 in [Iwe13].

Lemma 3.16 Let $f \in C_{2\pi}$ have bandwidth N and let $B \in \mathbb{N}$ with $B < N$. By the CRT, every row of $\mathcal{G}_{\tilde{L},\widetilde{K}}$ is of the form

$$\left(\mathbf{r}_{\tilde{s}_k\tilde{t}_l,h}\right)_j = \left(\mathbf{r}_{\tilde{s}_k,h \bmod \tilde{s}_k} \circledast \mathbf{r}_{\tilde{t}_l,h \bmod \tilde{t}_l}\right)_j = \begin{cases} 1 & \text{if } j \equiv h \bmod \tilde{s}_k\tilde{t}_l, \\ 0 & \text{otherwise,} \end{cases}$$

for some indices $h \in \left\{-\left\lceil \frac{\tilde{s}_k\tilde{t}_l}{2}\right\rceil + 1,\dots,\left\lfloor\frac{\tilde{s}_k\tilde{t}_l}{2}\right\rfloor\right\}$, $j \in \left\{-\frac{\tilde{q}}{2}+1,\dots,\frac{\tilde{q}}{2}\right\}$, $k \in \left\{1,\dots,\widetilde{K}\right\}$ and $l \in \left\{0,\dots,\tilde{L}\right\}$. With $\tilde{t}_0 = 1$ we can use the same notation for rows of the form $\mathbf{r}_{\tilde{s}_k,h \bmod \tilde{s}_k} \circledast \mathbf{1}_N$, which are generated as the row-wise Hadamard product of a row of $\mathcal{E}_{\widetilde{K}}$ with the first row of the extension of $\mathcal{N}_{\tilde{t}_1,\tilde{L}}$ to a $\tilde{\lambda}\times\tilde{q}$ matrix.

Remark 3.17 With the help of Lemma 3.16 we can use $\mathcal{G}_{\tilde{L},\widetilde{K}}$ in order to obtain the required samples of f as a few measurements of $\mathbf{a}^{\tilde{q}}$, where

$$\mathbf{a}^{\tilde{q}} = \left(f\left(\frac{2\pi j}{\tilde{q}}\right)\right)_{j=0}^{\tilde{q}-1}$$

is, as in Definition 2.5, the vector of $\tilde{q}$ equidistant samples of f. Then $\mathbf{a}^{\tilde{q}}$ can be considered to be the generating sampling vector, since any vector $\mathbf{a}^{\tilde{s}_k\tilde{t}_l}$ of equidistant samples required for Algorithm 3 in [Iwe13] can be obtained from it via

$$\mathbf{a}^{\tilde{s}_k\tilde{t}_l} = \left(a^{\tilde{q}}_{j\cdot\frac{\tilde{q}}{\tilde{s}_k\tilde{t}_l}}\right)_{j=0}^{\tilde{s}_k\tilde{t}_l-1}.$$

Of course we will not sample f at $\tilde{q}$ equidistant points; the vector $\mathbf{a}^{\tilde{q}}$ is only used theoretically in [Iwe13] to show that the concepts introduced therein work, by embedding all necessary samples into one vector of equidistant samples. Lemma 3.16 yields for every $h \in \left\{-\left\lceil\frac{\tilde{s}_k\tilde{t}_l}{2}\right\rceil+1,\dots,\left\lfloor\frac{\tilde{s}_k\tilde{t}_l}{2}\right\rfloor\right\}$ that by multiplying the row $\mathbf{r}_{\tilde{s}_k\tilde{t}_l,h} \in \{0,1\}^{1\times\tilde{q}}$ of $\mathcal{G}_{\tilde{L},\widetilde{K}}$

with the CDFT of the sampling vector $\mathbf{a}^{\tilde{q}}$, we obtain the corresponding entry of $\widehat{\mathbf{a}^{\tilde{s}_k\tilde{t}_l}}$,

$$\begin{aligned}
\mathbf{r}_{\tilde{s}_k\tilde{t}_l,h}\cdot\widehat{\mathbf{a}^{\tilde{q}}} &= \sum_{j=-\frac{\tilde{q}}{2}+1}^{\frac{\tilde{q}}{2}} \left(\mathbf{r}_{\tilde{s}_k\tilde{t}_l,h}\right)_j \widehat{a^{\tilde{q}}}_j \\
&= \sum_{j=-\frac{\tilde{q}}{2}+1}^{\frac{\tilde{q}}{2}} \delta_{(j-h)\bmod \tilde{s}_k\tilde{t}_l,0}\cdot\widehat{a^{\tilde{q}}}_j \\
&= \sum_{j'=-\left\lceil\frac{\tilde{q}}{2\tilde{s}_k\tilde{t}_l}\right\rceil+1}^{\left\lfloor\frac{\tilde{q}}{2\tilde{s}_k\tilde{t}_l}\right\rfloor} \frac{1}{\tilde{q}}\sum_{b=0}^{\tilde{q}-1} e^{\frac{-2\pi ib(h+j'\tilde{s}_k\tilde{t}_l)}{\tilde{q}}} a_b^{\tilde{q}} \\
&= \sum_{b=0}^{\tilde{q}-1}\frac{1}{\tilde{q}}a_b^{\tilde{q}}\cdot e^{\frac{-2\pi ibh}{\tilde{q}}} \sum_{j'=-\left\lceil\frac{\tilde{q}}{2\tilde{s}_k\tilde{t}_l}\right\rceil+1}^{\left\lfloor\frac{\tilde{q}}{2\tilde{s}_k\tilde{t}_l}\right\rfloor} e^{\frac{-2\pi ibj'\tilde{s}_k\tilde{t}_l}{\tilde{q}}} \\
&= \sum_{b=0}^{\tilde{q}-1}\frac{1}{\tilde{s}_k\tilde{t}_l} f\left(\frac{2\pi b}{\tilde{q}}\right) e^{\frac{-2\pi ibh}{\tilde{q}}}\delta_{b\bmod\frac{\tilde{q}}{\tilde{s}_k\tilde{t}_l},0} \\
&= \sum_{b'=0}^{\tilde{s}_k\tilde{t}_l-1}\frac{1}{\tilde{s}_k\tilde{t}_l} f\left(\frac{2\pi b'\frac{\tilde{q}}{\tilde{s}_k\tilde{t}_l}}{\tilde{q}}\right) e^{\frac{-2\pi ib'h\frac{\tilde{q}}{\tilde{s}_k\tilde{t}_l}}{\tilde{q}}} \\
&= \frac{1}{\tilde{s}_k\tilde{t}_l}\sum_{b'=0}^{\tilde{s}_k\tilde{t}_l-1} f\left(\frac{2\pi b'}{\tilde{s}_k\tilde{t}_l}\right) e^{\frac{-2\pi ib'h}{\tilde{s}_k\tilde{t}_l}} \\
&= \widehat{a^{\tilde{s}_k\tilde{t}_l}}_h,
\end{aligned}$$

where we have $j = h + j'\tilde{s}_k\tilde{t}_l$ for $j' \in \left\{-\left\lceil\frac{\tilde{q}}{2\tilde{s}_k\tilde{t}_l}\right\rceil+1,\ldots,\left\lfloor\frac{\tilde{q}}{2\tilde{s}_k\tilde{t}_l}\right\rfloor\right\}$ and $b = b'\frac{\tilde{q}}{\tilde{s}_k\tilde{t}_l}$ for some $b' \in \{0,\ldots,\tilde{s}_k\tilde{t}_l-1\}$.

Hence, the vector $\mathcal{G}_{\tilde{L},\tilde{K}}\cdot\mathbf{F}_{\tilde{q}}\cdot\mathbf{a}^{\tilde{s}_k\tilde{t}_l}$ contains precisely the CDFTs of the necessary vectors of equidistant samples of f, and is of the form

$$\mathcal{G}_{\tilde{L},\tilde{K}}\mathbf{F}_{\tilde{q}}\cdot\mathbf{a}^{\tilde{q}} = \left(\widehat{\mathbf{a}^{\tilde{s}_1}}^T, \widehat{\mathbf{a}^{\tilde{s}_1\tilde{t}_1}}^T,\ldots,\widehat{\mathbf{a}^{\tilde{s}_1\tilde{t}_{\tilde{L}}}}^T, \widehat{\mathbf{a}^{\tilde{s}_2}}^T, \widehat{\mathbf{a}^{\tilde{s}_2\tilde{t}_1}}^T,\ldots,\widehat{\mathbf{a}^{\tilde{s}_{\widetilde{K}}\tilde{t}_{\tilde{L}}}}^T\right)^T.$$

Analogously, one can show that

$$\mathcal{E}_{\widetilde{K}}\mathbf{F}_{\tilde{q}}\cdot\mathbf{a}^{\tilde{q}} = \left(\widehat{\mathbf{a}^{\tilde{s}_1}}^T, \widehat{\mathbf{a}^{\tilde{s}_2}}^T,\ldots,\widehat{\mathbf{a}^{\tilde{s}_{\widetilde{K}}}}^T\right)^T.$$

This implies that we do not need all $\tilde{q}$ equidistant samples of f in order to obtain the CDFTs of the required sampling vectors $\mathbf{a}^{\tilde{s}_k\tilde{t}_l}$. These CDFTs can be found by computing CDFTs of the vectors of $\tilde{s}_k\tilde{t}_l$ equidistant samples of f for all $k \in \{1,\ldots,\widetilde{K}\}$ and $l \in \{0,\ldots,\tilde{L}\}$, which can be done in much less time than computing the CDFT of the vector $\mathbf{a}^{\tilde{q}}$ of $\tilde{q}$ equidistant samples of f. ◊

3.2.2 Algorithm 3 in [Iwe13]

Using the notion of measurement matrices, we can now summarize Algorithm 3 in [Iwe13] as Algorithm 4.

Algorithm 4 Algorithm 3 in [Iwe13]

Input: $f+\eta$, $B, N, \epsilon^{-1} \in \mathbb{N} \setminus \{1\}$ with $B < N$, where f is at most B-sparse.

Output: R, $\mathbf{x}_R$, where R contains the $2B$ frequencies ω with greatest magnitude coefficient estimates x_ω.

1: Find the $\tilde{L}$ smallest primes $\tilde{t}_1, \dots, \tilde{t}_{\tilde{L}}$ such that $\prod_{l=1}^{\tilde{L}-1} \tilde{t}_l < \frac{N}{B} \leq \prod_{l=1}^{\tilde{L}} \tilde{t}_l$. Set $\tilde{t}_0 = 1$.

2: Let $s_1 > \max\{B, \tilde{t}_{\tilde{L}}\}$ be the smallest prime, $\widetilde{K} = 4\frac{B}{\epsilon}\left\lfloor \log_{\tilde{s}_1} N \right\rfloor + 1$ and $\tilde{s}_2, \dots, \tilde{s}_{\widetilde{K}}$ be the smallest primes greater than $\tilde{s}_1$.

3: Initialize $R = \emptyset$, $\mathbf{x}_R = \mathbf{0}_N$, $\tilde{q} = \operatorname{lcm}\left(N, \tilde{s}_1, \dots, \tilde{s}_{\widetilde{K}}, \tilde{t}_1, \tilde{t}_{\tilde{L}}\right)$.

4: $\mathcal{G}_{\tilde{L},\widetilde{K}} \cdot \widehat{\mathbf{a}^{\tilde{q}}} \leftarrow \left(\widehat{\mathbf{a}^{\tilde{s}_1}}^T, \widehat{\mathbf{a}^{\tilde{s}_1\tilde{t}_1}}^T, \dots, \widehat{\mathbf{a}^{\tilde{s}_1\tilde{t}_{\tilde{L}}}}^T, \widehat{\mathbf{a}^{\tilde{s}_2}}^T, \dots, \widehat{\mathbf{a}^{\tilde{s}_{\widetilde{K}}\tilde{t}_{\tilde{L}}}}^T\right)^T$

5: $\mathcal{E}_{\widetilde{K}} \cdot \widehat{\mathbf{a}^{\tilde{q}}} \leftarrow \left(\widehat{\mathbf{a}^{\tilde{s}_1}}^T, \dots, \widehat{\mathbf{a}^{\tilde{s}_{\widetilde{K}}}}^T\right)^T$

6: **for** k from 1 to $\widetilde{K}$ **do**

7: **for** h from $-\left\lceil \frac{\tilde{s}_k}{2} \right\rceil + 1$ to $\left\lfloor \frac{\tilde{s}_k}{2} \right\rfloor$ **do**

8: **for** l from 1 to $\tilde{L}$ **do**

9: $b_{\min} \leftarrow \underset{b \in \left\{-\left\lceil \frac{\tilde{t}_l}{2} \right\rceil + 1, \dots, \left\lfloor \frac{\tilde{t}_l}{2} \right\rfloor\right\}}{\operatorname{argmin}} \left| \left(\mathcal{E}_{\widetilde{K}} \cdot \widehat{\mathbf{a}^{\tilde{q}}}\right)_{\mathbf{r}_{\tilde{s}_k, h}} - \left(\mathcal{G}_{\tilde{L},\widetilde{K}} \cdot \widehat{\mathbf{a}^{\tilde{q}}}\right)_{\mathbf{r}_{\tilde{s}_k\tilde{t}_l, h+b\cdot\tilde{s}_k}} \right|$

10: $r_l^{k,h} \leftarrow (h + b_{\min} \cdot \tilde{s}_k) \bmod \tilde{t}_l$

11: **end for**

12: Recover $\omega^{k,h}$ from $\omega^{k,h} \equiv h \bmod \tilde{s}_k$, $\omega^{k,h} \equiv a_l^{k,h} \bmod \tilde{t}_l$ for $l \in \{1, \dots, \tilde{L}\}$.

13: **end for**

14: **end for**

15: **for each** $\omega^{k,h}$ value reconstructed more than $\frac{\widetilde{K}}{2}$ times **do**

16: $\operatorname{Re}(x_\omega) \leftarrow \underset{\substack{h \in \left\{-\left\lceil \frac{\tilde{s}_k}{2} \right\rceil + 1, \dots, \left\lfloor \frac{\tilde{s}_k}{2} \right\rfloor\right\} \\ k \in \{1,\dots,\widetilde{K}\} \\ l \in \{1,\dots,\tilde{L}\}}}{\operatorname{median}} \left\{ \operatorname{Re}\left(\left(\mathcal{G}_{\tilde{L},\widetilde{K}} \cdot \widehat{\mathbf{a}^{\tilde{q}}}\right)_{\mathbf{r}_{\tilde{s}_k\tilde{t}_l, h}} \right) : \omega = \omega^{k,h} \right\}$

17: $\operatorname{Im}(x_\omega) \leftarrow \underset{\substack{h \in \left\{-\left\lceil \frac{\tilde{s}_k}{2} \right\rceil + 1, \dots, \left\lfloor \frac{\tilde{s}_k}{2} \right\rfloor\right\} \\ k \in \{1,\dots,\widetilde{K}\} \\ l \in \{1,\dots,\tilde{L}\}}}{\operatorname{median}} \left\{ \operatorname{Im}\left(\left(\mathcal{G}_{\tilde{L},\widetilde{K}} \cdot \widehat{\mathbf{a}^{\tilde{q}}}\right)_{\mathbf{r}_{\tilde{s}_k\tilde{t}_l, h}} \right) : \omega = \omega^{k,h} \right\}$

18: **end for**

19: Sort the coefficients by magnitude s.t. $|x_{\omega_1}| \geq |x_{\omega_2}| \geq \cdots$.

Output: $R = \{\omega_1, \dots, \omega_{2B}\}$, $\mathbf{x}_R$.

In lines 9 and 10 of Algorithm 4, the residues of the possibly energetic frequencies modulo $\tilde{t}_l$ are computed, similar to (2.6) and (2.9). Line 12 reconstructs the frequencies from their previously computed residues via the CRT procedure from Algorithm 1. For all frequencies that have been reconstructed more than $\frac{\widetilde{K}}{2}$ times by Algorithm 4, the Fourier coefficient estimates x_ω are computed by taking the median of the real and the imaginary parts of the corresponding entries of $\widehat{\mathbf{a}^{\tilde{s}_k\tilde{t}_l}}$ for all $\tilde{s}_k$ and $\tilde{t}_l$. If ω is found for

more than $\frac{\tilde{K}}{2}$ values for $\tilde{s}_k$, its coefficient estimate, which is the corresponding Fourier coefficient of $f+\eta$, is accurate for the same primes $\tilde{s}_k$, so by taking the medians, one obtains the real and imaginary parts of the correct coefficient estimates. By additionally considering all $\tilde{s}_k\tilde{t}_l$ instead of just the $\tilde{s}_k$, as in (2.10), the method becomes more stable with respect to noise, since we want to find good estimates for the Fourier coefficients of the original function f.

The following lemma summarizes the main results shown for Algorithm 3 in [Iwe13] that are relevant for our method. Here, for any vector $\mathbf{y} \in \mathbb{C}^{|I|}$ with index set I, and a subset $R \subseteq I$, we denote by $\mathbf{y}_R \in \mathbb{C}^{|I|}$ the vector

$$(\mathbf{y}_R)_j = \begin{cases} y_j & \text{if } j \in R, \\ 0 & \text{otherwise,} \end{cases}$$

for all $j \in I$. For any $s < |I|$ we will let the subset $R_s^{\text{opt}} \subseteq I$ be the, in lexicographical order, first s-element subset such that $|y_j| \geq |y_k|$ for all $j \in R_s^{\text{opt}}$ and $k \in I \setminus R_s^{\text{opt}}$. Thus, R_s^{opt} contains the indices of s entries of $\mathbf{y}$ with largest magnitudes. While choosing s entries with largest magnitudes might not be unique, R_s^{opt} is unique by definition. To simplify notation we set $\mathbf{y}_s^{\text{opt}} \coloneqq \mathbf{y}_{R_s^{\text{opt}}}$.

Lemma 3.18 Let $f \in C_{2\pi}$ have bandwidth N and let $B \in \mathbb{N}$ with $B < N$.

(i) (Lemma 6 in [Iwe13]) If $\omega \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$ satisfies

$$|c_\omega| > 4 \cdot \left(\frac{1}{2B} \left\| \mathbf{c}(N) - \mathbf{c}_{2B}^{\text{opt}}(N) \right\|_1 + \left\| \mathbf{c}(f) - \mathbf{c}(N,\mathbb{Z}) \right\|_1 \right),$$

then ω will be reconstructed more than $\frac{\widetilde{K}}{2}$ times.

(ii) (Proof of Theorem 7 in [Iwe13]) If ω is reconstructed more than $\frac{\widetilde{K}}{2}$ times, then

$$|x_\omega - c_\omega| \leq \sqrt{2} \left(\frac{1}{2B} \left\| \mathbf{c}(N) - \mathbf{c}_{2B}^{\text{opt}}(N) \right\|_1 + \left\| \mathbf{c}(f) - \mathbf{c}(N,\mathbb{Z}) \right\|_1 \right).$$

(iii) (Theorem 7 in [Iwe13]) Choosing $\epsilon^{-1} = 2$, Algorithm 3 in [Iwe13] will output an $\mathbf{x}_R \in \mathbb{C}^N$ satisfying

$$\begin{aligned} &\left\| \mathbf{c}(N) - \mathbf{x}_R \right\|_2 \\ \leq\ &\left\| \mathbf{c}(N) - \mathbf{c}_B^{\text{opt}}(N) \right\|_2 + \frac{11}{\sqrt{B}} \left\| \mathbf{c}(N) - \mathbf{c}_{2B}^{\text{opt}}(N) \right\|_1 + 22\sqrt{B} \left\| \mathbf{c}(f) - \mathbf{c}(N,\mathbb{Z}) \right\|_1 \end{aligned}$$

in a runtime of

$$\mathcal{O}\left(\frac{B^2 \log^2 N \log^2(2B \log N) \log^2 \frac{N}{2B}}{\log^2(2B) \log\log \frac{N}{2B}} \right),$$

and using

$$\mathcal{O}\left(\frac{B^2 \log^2 N \log(2B \log N) \log^2 \frac{N}{2B}}{\log^2(2B) \log\log \frac{N}{2B}} \right).$$

samples of f.

Remark 3.19 (i) Heuristically, the assertions of Lemma 3.18 (i) imply that if ω is sufficiently energetic, if the perturbed function $f+\eta$ is approximately B-sparse and if N

is a good estimate on the bandwidth of f, ω will be reconstructed for more than half of the $\tilde{s}_k$. Thus, its coefficient estimate will be computed by taking the medians in lines 16 and 17.

(ii) It follows from Lemma 3.18 (ii) that if a frequency ω is reconstructed for more than half of the $\tilde{s}_k$, then its coefficient estimate x_ω, calculated in lines 16 and 17, will be accurate if the perturbed function $f + \eta$ is still approximately B-sparse and if N is a good estimate on the bandwidth of f.

(iii) Lemma 3.18 (iii) implies that, if the perturbed function $f + \eta$ is still approximately B-sparse and N is a good estimate on the bandwidth of f, the reconstruction of $\mathbf{c}(N)$ by the vector $\mathbf{x}_R$ given by Algorithm 4 is accurate.

Further, it follows that the runtime and sampling complexity of Algorithm 4 are sublinear in the bandwidth N and quadratic in the sparsity B.

◊

3.3 Polynomially Structured Sparse Functions

3.3.1 Measurement Matrices II

In Section 3.1 we already mentioned that the main idea of our algorithm for polynomially structured sparse functions is to apply Algorithm 3 in [Iwe13] to all restrictions of the input function to frequencies congruent to a residue modulo the primes $u_1, \dots, u_M$. In order to be able to do so, we first have to introduce the measurement matrices and primes required for our approach. Analogously to Algorithm 4, we want to apply the CRT reconstruction procedure from Algorithm 1 to the residues modulo $s_k u_m$ and $t_1, \dots, t_L$ for all $k \in \{1, \dots, K\}$ and $m \in \{1, \dots, M\}$. Consequently, we require that

$$s_k u_m \prod_{l=1}^{L-1} t_l < N \leq s_k u_m \prod_{l=1}^{L} t_l \qquad \forall\, k \in \{1, \dots, K\} \quad \text{and} \quad \forall\, m \in \{1, \dots, M\}.$$

In order to be able to choose the t_l independently of the s_k and u_m, we use that u_m will be greater than B and s_k will be greater than dn for all $k \in \{1, \dots, K\}$ and $m \in \{1, \dots, M\}$.

Definition 3.20 (Definition 3.1 in [BZI19]) Let $f \in C_{2\pi}$ be $P(n, d, B)$-structured sparse with bandwidth N and noise threshold $\varepsilon > 0$. Let $t_1, \dots, t_L$ be the L smallest primes satisfying

$$Bdn \prod_{l=1}^{L-1} t_l < N \leq Bdn \prod_{l=1}^{L} t_l.$$

For algorithmic purposes we let $t_0 \coloneqq 1$. Further, we set s_1 to be the smallest prime that is greater than both dn and t_L, i.e.,

$$s_1 \coloneqq p_a > \max\{dn, t_L\} \geq p_{a-1}.$$

The minimal K for guaranteeing correct recovery of the restricted functions for more than half of the s_k by Algorithm 3 in [Iwe13] would now be

$$K = 8dn \left\lfloor \log_{s_1} \frac{N}{u_1} \right\rfloor + 1.$$

However, since we did not choose the hashing primes $u_1, \ldots, u_M$ yet, which, similarly to Definition 3.15, have to be distinct from the s_k, we increase K slightly using that the unknown u_1 will have to be strictly greater than B, i.e.,

$$K \coloneqq 8dn \left\lfloor \log_{s_1} \frac{N}{B} \right\rfloor + 1 \geq 8dn \left\lfloor \log_{s_1} \frac{N}{u_1} \right\rfloor + 1.$$

Hence, we can now choose the remaining s_k independently from the hashing primes $u_1, \ldots, u_M$ to be $s_k \coloneqq p_{a-1+k}$ for $k \in \{1, \ldots, K\}$. The u_m can be found by setting

$$u_1 \coloneqq p_b > \max\{B, s_K\} \geq p_{b-1},$$

$M \coloneqq 2(n+1) \lfloor \log_{u_1} N \rfloor + 1$ and $u_m \coloneqq p_{b-1+m}$ for $m \in \{1, \ldots, M\}$. With these definitions $s_1, \ldots, s_K, t_1, \ldots, t_L, u_1, \ldots, u_M$ are pairwise relatively prime and satisfy that

$$s_k u_m \cdot \prod_{l=1}^{L} t_l \geq N$$

for all $k \in \{1, \ldots, K\}$ and $m \in \{1, \ldots, M\}$, so the frequencies of the restriction can be reconstructed from their residues modulo $s_k \cdot u_m, t_1, \ldots, t_L$ for all k and m via the CRT reconstruction method in Algorithm 1. By choice of M and Lemmas 3.10 and 3.11, more than $\frac{M}{2}$ of the u_m hash all support sets S_j of the input function f well. Furthermore, we have that $s_k > dn$ and $u_m > B$ for all k and m.

We also set $\kappa \coloneqq \sum_{k=1}^{K} s_k$, $\lambda \coloneqq \sum_{l=0}^{L} t_l$ and $\mu \coloneqq \sum_{m=1}^{M} u_m$. Analogously to $\tilde{q}$ defined in Section 3.2.1, we set

$$q = \operatorname{lcm}(N, s_1, \ldots, s_K, t_1, \ldots, t_L, u_1, \ldots, u_M),$$

because all s_k, t_l and u_m have to divide q. Again, q will be the length of the generating sample vector later on, i.e., all required samples of f can be selected as entries of $\mathbf{a}^q$, but of course we do not need all entries of $\mathbf{a}^q$. Note that q is even, as $t_1 = 2$.

From now on we always assume that the occurring natural numbers $q, s_1, \ldots, s_K$, $t_1, \ldots, t_L, u_1, \ldots, u_M$ comply with Definition 3.20. Recall that in Definition 2.5 we set the vector of M equidistant samples of $f \in C_{2\pi}$ to be

$$\mathbf{a}^M = \left(f\left(\frac{2\pi j}{M}\right) \right)_{j=0}^{M-1} \in \mathbb{C}^M.$$

In order to apply Algorithm 3 in [Iwe13] to the restrictions of f to frequencies that are congruent to ν modulo u_m for all residues $\nu \in \left\{ -\left\lceil \frac{u_m}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{u_m}{2} \right\rfloor \right\}$ and all $m \in \{1, \ldots, M\}$, we need to transform the vector $\widehat{\mathbf{a}^q} = \mathbf{F}_q \mathbf{a}^q$ into a matrix with sparse columns whose entries correspond to the frequencies that are congruent to ν modulo u_m.

Definition 3.21 (Measurement Matrices II (Definition 3.6 in [BZI19])) For $t_1, \ldots, t_L$, $s_1, \ldots, s_K, u_1, \ldots, u_M$ and q as in Definition 3.20, we construct a special $\mu \times N$ measurement matrix $\mathcal{M}_{u_1, M}$, analogously to the measurement matrices used in [Iwe13], given by Definition 3.15. As before, an entry of a row is one if and only if its column index is congruent to a certain residue modulo u_m. Let $m \in \{1, \ldots, M\}$ and $\nu \in \left\{ -\left\lceil \frac{u_m}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{u_m}{2} \right\rfloor \right\}$ be a fixed residue modulo u_m. Then we define the row

$\mathbf{r}_{u_m,\nu} \in \{0,1\}^{1\times N}$ by

$$(\mathbf{r}_{u_m,\nu})_j \coloneqq \delta_{(j-\nu) \bmod u_m, 0} = \begin{cases} 1, & \text{if } j \equiv \nu \bmod u_m, \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

for $j \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$, and set

$$\mathcal{M}_{u_1,M} \coloneqq \begin{pmatrix} \mathbf{r}_{u_1,-\left\lceil \frac{u_1}{2} \right\rceil+1} \\ \vdots \\ \mathbf{r}_{u_1,\left\lfloor \frac{u_1}{2} \right\rfloor} \\ \mathbf{r}_{u_2,-\left\lceil \frac{u_2}{2} \right\rceil+1} \\ \vdots \\ \mathbf{r}_{u_M,\left\lfloor \frac{u_M}{2} \right\rfloor} \end{pmatrix}.$$

We define the extension $\mathcal{H}_{M,L,K}$ of $\mathcal{M}_{u_1,M}$ to a $\mu \times q$ matrix by extending all rows $\mathbf{r}_{u_m,\nu}$ to columns indexed by $j \in \left\{-\frac{q}{2}+1, \dots, \frac{q}{2}\right\}$, as given in (3.2). Thus, the middle N columns of $\mathcal{H}_{M,L,K}$ are just $\mathcal{M}_{u_1,M}$. Further, as in Section 3.2.1, we define the $\kappa \times N$ matrix $\mathcal{M}_{s_1,K}$ and the $(\lambda-1)\times N$ matrix $\mathcal{M}_{t_1,L}$, consisting of the rows corresponding to the possible residues modulo all the s_k and t_l, respectively,

$$\mathcal{M}_{s_1,K} \coloneqq \begin{pmatrix} \mathbf{r}_{s_1,-\left\lceil \frac{s_1}{2} \right\rceil+1} \\ \vdots \\ \mathbf{r}_{s_K,\left\lfloor \frac{s_K}{2} \right\rfloor} \end{pmatrix} \quad \text{and} \quad \mathcal{M}_{t_1,L} \coloneqq \begin{pmatrix} \mathbf{r}_{t_1,-\left\lceil \frac{t_1}{2} \right\rceil+1} \\ \vdots \\ \mathbf{r}_{t_L,\left\lfloor \frac{t_L}{2} \right\rfloor} \end{pmatrix}.$$

We set $\mathcal{N}_{t_1,L}$ to be the $\lambda \times N$ matrix whose first row contains only ones and whose other rows are given by $\mathcal{M}_{t_1,L}$,

$$\mathcal{N}_{t_1,L} \coloneqq \begin{pmatrix} \mathbf{1}_N \\ \mathcal{M}_{t_1,L} \end{pmatrix},$$

and define the $(\kappa\cdot\lambda)\times N$ row-wise Hadamard product $\mathcal{R}_{L,K} \coloneqq \mathcal{M}_{s_1,K} \circledast \mathcal{N}_{t_1,L}$ of $\mathcal{M}_{s_1,K}$ and $\mathcal{N}_{t_1,L}$ and its extension $\mathcal{G}_{L,K}$ to a $(\kappa\cdot\lambda)\times q$ matrix. The measurement matrices required for our algorithm are $\mathcal{E}_K$, $\mathcal{G}_{L,K}$ and $\mathcal{H}_{M,L,K}$.

Remark 3.22 (Restriction of $\widehat{\mathbf{a}^q}$ (Remark 3.7 in [BZI19])) If we compute the row-wise Hadamard product of $\mathcal{H}_{M,L,K}$ with $(\widehat{\mathbf{a}^q})^T \in \mathbb{C}^{1\times q}$, every row of $\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T$ is, by Lemma 3.14, given as the row-wise Hadamard product of a row of $\mathcal{H}_{M,L,K}$ and $(\widehat{\mathbf{a}^q})^T$. Let us denote the columns of $(\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T \in \mathbb{C}^{q\times\mu}$ by $\boldsymbol{\rho}^T_{u_m,\nu}$ for all residues $\nu \in \left\{-\left\lceil \frac{u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u_m}{2} \right\rfloor\right\}$ and $m \in \{1,\dots,M\}$. Then the column $\boldsymbol{\rho}^T_{u_m,\nu}$ corresponds to a residue ν modulo the hashing prime u_m. This column only contains nonzero entries at frequencies that are congruent to ν modulo u_m,

$$\begin{aligned} \left(\boldsymbol{\rho}^T_{u_m,\nu}\right)_j &\coloneqq \left(\mathbf{r}_{u_m,\nu} \circledast \left(\widehat{\mathbf{a}^q}\right)^T\right)^T_j \\ &= (\mathbf{r}_{u_m,\nu})^T_j \cdot \left(\widehat{a^q}\right)_j \\ &= \delta_{(j-\nu) \bmod u_m, 0} \cdot \widehat{a^q}_j \end{aligned}$$

$$= \begin{cases} \widehat{a^q}_j, & \text{if } j \equiv \nu \mod u_m, \\ 0, & \text{otherwise} \end{cases}$$

for $j \in \left\{-\frac{q}{2}+1, \dots, \frac{q}{2}\right\}$. Hence, the column $\boldsymbol{\rho}^T_{u_m,\nu}$ of $(\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T$ is the restriction of $\widehat{\mathbf{a}^q}$ to the frequencies congruent to ν modulo u_m, which is at most dn-sparse for a good hashing prime u_m. Thus, for more than $\frac{M}{2}$ of the u_m, we can apply Algorithm 3 in [Iwe13] with sparsity dn to all columns. ◊

3.3.2 Algorithm for Polynomially Structured Sparse Functions

By Remark 3.22, Algorithm 3 in [Iwe13] can be applied with sparsity dn to all columns $\boldsymbol{\rho}^T_{u_m,\nu}$ of $(\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T$. Recall that the column $\boldsymbol{\rho}^T_{u_m,\nu}$ is only guaranteed to be at most dn-sparse if u_m hashes all support sets $S_1, \dots, S_n$ well. This means that only for the columns corresponding to those primes the algorithm will return all energetic frequencies and provide good estimates for their Fourier coefficients. Consequently, we have to apply Algorithm 3 in [Iwe13] to every single column of $(\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T$ and choose the frequencies and coefficient estimates that appear for more than $\frac{M}{2}$ of the u_m.

In Algorithm 3 in [Iwe13] estimates for the Fourier coefficients c_ω of the input function f are calculated from certain entries of $\mathcal{G}_{L,K} \cdot \widehat{\mathbf{a}^q}$. These entries can be obtained in a fast way from f by computing CDFTs of the vectors $\mathbf{a}^{s_k t_l}$. A similar property is true for the entries of $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T$ required for polynomially structured sparse functions, as the following remark shows.

Remark 3.23 (Remark 3.9 in [BZI19]) For polynomially structured sparse input functions we now prove that the entries of $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{A}_q})^T)^T$ can be calculated fast. As we want to use the residues modulo the hashing primes $u_1, \dots, u_M$ for the reconstruction as well, an idea similar to the one from [Iwe13] leads to CDFTs of the $s_k t_l u_m$-length vectors $\mathbf{a}^{s_k t_l u_m}$ of equidistant samples from Definition 2.5.

Consider an entry of $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T \in \mathbb{C}^{\kappa\lambda \times \mu}$. By construction, it is the product of a row of $\mathcal{G}_{L,K}$, which is by Lemma 3.16 of the form $\mathbf{r}_{s_k t_l, h}$ for a residue $h \in \left\{-\left\lceil \frac{s_k t_l}{2} \right\rceil + 1, \dots, \left\lfloor \frac{s_k t_l}{2} \right\rfloor\right\}$ modulo $s_k t_l$, with a column $\boldsymbol{\rho}^T_{u_m,\nu}$ of $(\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T$ for a residue ν modulo u_m for some $m \in \{1, \dots, M\}$. Remark 3.22 yields that

$$\begin{aligned} \mathbf{r}_{s_k t_l, h} \cdot \boldsymbol{\rho}^T_{u_m,\nu} &= \sum_{j=-\frac{q}{2}+1}^{\frac{q}{2}} \left(\mathbf{r}_{s_k t_l, h}\right)_j \left(\boldsymbol{\rho}^T_{u_m,\nu}\right)_j \\ &= \sum_{j=-\frac{q}{2}+1}^{\frac{q}{2}} \delta_{(j-h) \bmod s_k t_l, 0} \cdot \delta_{(j-\nu) \bmod u_m, 0} \cdot \widehat{a^q}_j. \end{aligned} \tag{3.3}$$

As there can only be nonzero summands in (3.3) if $j \equiv h \mod s_k t_l$ and $j \equiv \nu \mod u_m$, we find with the CRT that j has to be of the form

$$j = \tau + j' s_k t_l u_m, \quad j' \in \left\{ -\left\lceil \frac{q}{2 s_k t_l u_m} \right\rceil + 1, \dots, \left\lfloor \frac{q}{2 s_k t_l u_m} \right\rfloor \right\},$$

where $\tau \in \left\{-\left\lceil \frac{s_k t_l u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{s_k t_l u_m}{2} \right\rfloor\right\}$ is the residue of j modulo $s_k t_l u_m$. Then it

follows that

$$\begin{aligned}
\mathbf{r}_{s_k t_l, h} \cdot \boldsymbol{\rho}_{u_m, \nu}^T &= \sum_{j'=-\left\lceil \frac{q}{2 s_k t_l u_m} \right\rceil + 1}^{\left\lfloor \frac{q}{2 s_k t_l u_m} \right\rfloor} \widehat{a^q}_{\tau + j' \cdot s_k t_l u_m} \\
&= \sum_{j'=-\left\lceil \frac{q}{2 s_k t_l u_m} \right\rceil + 1}^{\left\lfloor \frac{q}{2 s_k t_l u_m} \right\rfloor} \frac{1}{q} \sum_{b=0}^{q-1} e^{\frac{-2\pi i b(\tau + j' s_k t_l u_m)}{q}} a_b^q \\
&= \sum_{b=0}^{q-1} \frac{1}{q} a_b^q \cdot e^{\frac{-2\pi i b \tau}{q}} \sum_{j'=-\left\lceil \frac{q}{2 s_k t_l u_m} \right\rceil + 1}^{\left\lfloor \frac{q}{2 s_k t_l u_m} \right\rfloor} e^{\frac{-2\pi i b j' s_k t_l u_m}{q}} \\
&= \sum_{b=0}^{q-1} \frac{1}{s_k t_l u_m} f\left(\frac{2\pi b}{q}\right) e^{\frac{-2\pi i b \tau}{q}} \cdot \delta_{b \bmod \frac{q}{s_k t_l u_m}, 0} \\
&= \sum_{b'=0}^{s_k t_l u_m - 1} \frac{1}{s_k t_l u_m} f\left(\frac{2\pi b' \frac{q}{s_k t_l u_m}}{q}\right) e^{\frac{-2\pi i \tau b' \frac{q}{s_k t_l u_m}}{q}} \\
&= \widehat{a^{s_k t_l u_m}}_\tau.
\end{aligned}$$

By Bézout's identity, see Theorem 2.7, $1 = \gcd(s_k t_l, u_m) = v \cdot s_k t_l + w \cdot u_m$ for some $v, w \in \mathbb{Z}$, and we obtain that τ satisfies

$$\tau = ((h-\nu) w \bmod s_k t_l) \cdot u_m + \nu \in \left\{ -\left\lceil \frac{s_k t_l u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{s_k t_l u_m}{2} \right\rfloor \right\}. \tag{3.4}$$

Thus, in the column corresponding to the residue ν modulo u_m, for fixed $s_k t_l$ only the $s_k t_l$ different values $\widehat{a^{s_k t_l u_m}}_\tau$ with τ depending on h as in (3.4) are contained. Hence, the column of $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T$ corresponding to ν modulo u_m is of the form

$$\left(\widehat{\mathbf{v}_{s_1}^{m,\nu}}^T, \widehat{\mathbf{v}_{s_1 t_1}^{m,\nu}}^T, \dots, \widehat{\mathbf{v}_{s_1 t_L}^{m,\nu}}^T, \widehat{\mathbf{v}_{s_2}^{m,\nu}}^T, \widehat{\mathbf{v}_{s_2 t_1}^{m,\nu}}^T, \dots, \widehat{\mathbf{v}_{s_K t_L}^{m,\nu}}^T \right)^T,$$

where

$$\left(\widehat{v_{s_k t_l}^{m,\nu}} \right)_j := \widehat{a^{s_k t_l u_m}}_{((j-\nu) w \bmod s_k t_l) \cdot u_m + \nu} \tag{3.5}$$

for all $j \in \left\{ -\left\lceil \frac{s_k t_l}{2} \right\rceil + 1, \dots, \left\lfloor \frac{s_k t_l}{2} \right\rfloor \right\}$. The entries of $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T$ can be calculated in a fast way, using KLM CDFTs of the vectors of $s_k t_l u_m$ equidistant samples, which has a runtime $\mathcal{O}(s_k t_l u_m \cdot \log(s_k t_l u_m))$ for all k, l, m, as we know from Section 1.1.1 and Remark 1.9. ◊

We still have to specify how we apply Algorithm 4 (Algorithm 3 in [Iwe13]) to the columns of the matrix $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T$. Until now, we considered fixed residues h modulo $s_k t_l$ and ν modulo u_m. However, in line 7 of Algorithm 4, we fix the residue h' modulo s_k of a frequency ω and find the residues modulo the $s_k t_l$ of ω in line 9. The following remark shows how these residues can be combined.

Remark 3.24 Choose $s_1, \dots, s_K, t_1, \dots, t_L, u_1, \dots, u_M$ as in Definition 3.20 and let $\omega \in \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor \right\}$ satisfy $\omega \equiv \nu \bmod u_m$ and $\omega \equiv h' \bmod s_k$ for some residues

$\nu \in \left\{-\left\lceil \frac{u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u_m}{2} \right\rfloor\right\}$ and $h' \in \left\{-\left\lceil \frac{s_k}{2} \right\rceil + 1, \dots, \left\lfloor \frac{s_k}{2} \right\rfloor\right\}$, where $k \in \{1, \dots, K\}$ and $m \in \{1, \dots, M\}$. For the fixed frequency ω we have to find the corresponding residue modulo $s_k t_l u_m$ for every $l \in \{1, \dots, L\}$. Bézout's identity implies that

$$1 = \gcd(s_k, u_m) = v' \cdot s_k + w' \cdot u_m$$

for some $v', w' \in \mathbb{Z}$. Then the residue τ' of ω modulo $s_k u_m$ has to satisfy

$$\tau' \coloneqq \omega \mod s_k u_m = \left(\left(h' - \nu\right) w' \mod s_k\right) \cdot u_m + \nu,$$

and the residue of ω modulo $s_k t_l u_m$ is of the form

$$\omega \mod s_k t_l u_m = \tau' + b_{\min} \cdot s_k u_m \tag{3.6}$$

for a $b_{\min} \in \left\{-\left\lceil \frac{t_l}{2} \right\rceil + 1, \dots, \left\lfloor \frac{t_l}{2} \right\rfloor\right\}$, which is given as

$$b_{\min} \coloneqq \operatorname*{argmin}_{b \in \left\{-\left\lceil \frac{t_l}{2} \right\rceil + 1, \dots, \left\lfloor \frac{t_l}{2} \right\rfloor\right\}} \left| \widehat{a^{s_k u_m}}_{\tau'} - \widehat{a^{s_k t_l u_m}}_{\tau' + b \cdot s_k u_m} \right|. \tag{3.7}$$

If s_k and u_m are separating, then $b_{\min}$ is unique, since no other energetic frequency can have a residue of the form in (3.6). This is analogous to (2.8) in Section 2.2.2. If $b_{\min}$ is not unique, we can choose any minimizer, resulting in a possibly wrong residue for the frequency ω modulo t_l and an incorrectly recovered ω. However, since more than half of the s_k and more than half of the u_m guarantee separation, we will still reconstruct ω correctly in more than half of the cases. Thus, we can find ω by employing median techniques in both Algorithm 4 and our new Algorithm 5.

Finally, it follows from (3.6) that the residue of ω modulo t_l is

$$a_l \coloneqq \omega \mod t_l = \left(\tau' + b_{\min} \cdot s_k u_m\right) \mod t_l,$$

and ω can be reconstructed from its residues $\omega \equiv \tau' \mod s_k u_m$, $\omega \equiv a_1 \mod t_1$, …, $\omega \equiv a_L \mod t_L$. Recall the vectors $\widehat{\mathbf{v}_{s_k t_l}^{(m,\nu)}}$ introduced in (3.5). These vectors are defined such that if $\omega \equiv \nu \mod u_m$, $\omega \equiv h' \mod s_k$ and $\omega \equiv h \mod s_k t_l$, we have

$$\left(\widehat{v_{s_k t_l}^{m,\nu}}\right)_{\omega \bmod s_k t_l} = \widehat{a^{s_k t_l u_m}}_{\omega \bmod s_k t_l u_m}.$$

To use this notation, we take the residues modulo $s_k t_l u_m$ again modulo $s_k t_l$, and obtain

$$\begin{aligned}
b_{\min} &= \operatorname*{argmin}_{b \in \left\{-\left\lceil \frac{t_l}{2} \right\rceil + 1, \dots, \left\lfloor \frac{t_l}{2} \right\rfloor\right\}} \left| \widehat{a^{s_k u_m}}_{\tau'} - \widehat{a^{s_k t_l u_m}}_{\tau' + b \cdot s_k u_m} \right| \\
&= \operatorname*{argmin}_{b \in \left\{-\left\lceil \frac{t_l}{2} \right\rceil + 1, \dots, \left\lfloor \frac{t_l}{2} \right\rfloor\right\}} \left| \left(\widehat{v_{s_k}^{m,\nu}}\right)_{h'} - \left(\widehat{v_{s_k t_l}^{m,\nu}}\right)_{(\tau' + b \cdot s_k u_m) \bmod s_k t_l} \right| \\
&= \operatorname*{argmin}_{b \in \left\{-\left\lceil \frac{t_l}{2} \right\rceil + 1, \dots, \left\lfloor \frac{t_l}{2} \right\rfloor\right\}} \left| \left(\mathcal{E}_K \cdot \widehat{\mathbf{a}^q}\right)_{\mathbf{r}_{s_k, h'}, \boldsymbol{\rho}_{m,\nu}^T} - \left(\mathcal{G}_{L,K} \cdot \widehat{\mathbf{a}^q}\right)_{\mathbf{r}_{s_k t_l, (\tau' + b s_k u_m) \bmod s_k t_l}, \boldsymbol{\rho}_{m,\nu}^T} \right|
\end{aligned}$$

This shows that we can indeed apply Algorithm 4 to the restriction to the frequencies congruent to ν modulo u_m while also utilizing this congruency information. ◊

Remark 3.25 Similarly to Remarks 2.11 and 2.25, Algorithms 4 and 5 can achieve even faster runtimes if some of the primes $s_1, \dots, s_K$, $t_1, \dots, t_L$, $u_1, \dots, u_M$ are omitted and the preconditions of the CRT are still met. We also did not incorporate these improvements in the implementation of our algorithm for block sparse functions we used for the numerical experiments in Section 3.5. ◊

Algorithm 5 presents itself as a summary of the preceding considerations.

Algorithm 5 Algorithm for Functions with Polynomially Structured Frequency Support (Algorithm 1 in [BZI19])

Input: $f + \eta$, $n, d, B, N \in \mathbb{N}$ with $d < B < N$, where $f \in C_{2\pi}$ has bandwidth N and is $P(n, d, B)$-structured sparse.

Output: $R, \mathbf{x}_R$, where R contains the nB frequencies ω with greatest magnitude coefficient estimates x_ω.

Find the L smallest primes $t_1, \dots, t_L$ such that $\prod_{l=1}^{L-1} t_l < \frac{N}{Bdn} \leq \prod_{l=1}^{L} t_l$. Set $t_0 = 1$.

Let $s_1 > \max\{dn, t_L\}$ be the smallest prime, $K = 8dn \left\lfloor \log_{s_1} \frac{N}{B} \right\rfloor + 1$ and $s_2, \dots, s_K$ the smallest primes greater than s_1.

Let $u_1 > \max\{B, s_K\}$ be the smallest prime, $M = 2(n+1) \cdot \lfloor \log_{u_1} N \rfloor + 1$ and $u_2, \dots, u_M$ the smallest primes greater than u_1.

Initialize $R = \emptyset$, $\mathbf{x}_R = \mathbf{0}_N$, $q = \operatorname{lcm}(N, s_1, \dots, s_K, t_1, \dots, t_L, u_1, \dots, u_M)$.

$\mathcal{G}_{L,K} \cdot \left(\mathcal{H}_{M,L,K} \circledast \left(\widehat{\mathbf{a}^q}\right)^T\right)^T \leftarrow \left(\left(\widehat{\mathbf{v}_{s_1}^{m,\nu}}^T, \widehat{\mathbf{v}_{s_1 t_1}^{m,\nu}}^T, \dots, \widehat{\mathbf{v}_{s_K t_L}^{m,\nu}}^T\right)^T\right)_{m=1,\nu=0}^{M,u_m-1}$

$\mathcal{E}_K \cdot \left(\mathcal{H}_{M,L,K} \circledast \left(\widehat{\mathbf{a}^q}\right)^T\right)^T \leftarrow \left(\left(\widehat{\mathbf{v}_{s_1}^{m,\nu}}^T, \dots, \widehat{\mathbf{v}_{s_K}^{m,\nu}}^T\right)\right)_{m=1,\nu=0}^{M,u_m-1}$

for m from 1 to M **do**

  for ν from $-\left\lceil \frac{u_m}{2} \right\rceil + 1$ to $\left\lfloor \frac{u_m}{2} \right\rfloor$ **do**

    $\left(R^{(m,\nu)}, \mathbf{x}^{(m,\nu)}\right) \leftarrow 2dn$ frequencies with largest magnitude coefficient estimates returned by Algorithm 4 applied to $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T_{\boldsymbol{\rho}^T_{m,\nu}}$ and $\mathcal{E}_K \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T_{\boldsymbol{\rho}^T_{m,\nu}}$ with sparsity dn.

  end for

end for

for each $\omega \in \bigcup_{m=1}^{M} \bigcup_{\nu=0}^{u_m-1} R^{(m,\nu)}$ found more than $\frac{M}{2}$ times **do**

$$\operatorname{Re}(x_\omega) = \underset{\substack{\nu=\left\{-\left\lceil \frac{u_m}{2} \right\rceil+1,\dots,\left\lfloor \frac{u_m}{2} \right\rfloor\right\} \\ m\in\{1,\dots,M\}}}{\operatorname{median}} \left\{\operatorname{Re}\left(x_{\tilde{\omega}}^{(m,\nu)}\right) : \tilde{\omega} = \omega, \tilde{\omega} \in R^{(m,\nu)}\right\}$$

$$\operatorname{Im}(x_\omega) = \underset{\substack{\nu=\left\{-\left\lceil \frac{u_m}{2} \right\rceil+1,\dots,\left\lfloor \frac{u_m}{2} \right\rfloor\right\} \\ m\in\{1,\dots,M\}}}{\operatorname{median}} \left\{\operatorname{Im}\left(x_{\tilde{\omega}}^{(m,\nu)}\right) : \tilde{\omega} = \omega, \tilde{\omega} \in R^{(m,\nu)}\right\}$$

end for

Sort the coefficients by magnitude s.t. $|x_{\omega_1}| \geq |x_{\omega_2}| \geq \cdots$.

$R \leftarrow \{\omega_1, \dots, \omega_{Bn}\}$

Output: : $R, \mathbf{x}_R$.

3.3.3 Error, Runtime and Sampling Bounds

We will now prove runtime and sampling bounds for Algorithm 5. Furthermore, we will give theoretical bounds on the accuracy of the returned Fourier coefficient estimates. In order to obtain such bounds, we can utilize some of the results developed in [Iwe13]

for the more than $\frac{M}{2}$ primes u_m that hash all support sets $S_1, \dots, S_n$ well. These are precisely the primes for which the corresponding columns $\boldsymbol{\rho}^T_{u_m,\nu}$ of $(\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}}^q)^T)^T$ are guaranteed to be at most dn-sparse.

Analogously to our previous notation, we denote by $\mathbf{c}(N, u_m, \nu)$, $\mathbf{c}(N, \mathbb{Z}, u_m, \nu)$ and $\mathbf{c}(u_m, \nu)$ the restrictions of $\mathbf{c}(N)$, $\mathbf{c}(N, \mathbb{Z})$ and $\mathbf{c}(f + \eta)$, respectively, to the frequencies congruent to ν modulo u_m for a residue $\nu \in \left\{-\left\lceil \frac{u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u_m}{2} \right\rfloor\right\}$ modulo u_m for some $m \in \{1, \dots, M\}$. Further, recall that $\mathbf{c}^{\text{opt}}_{2dn}(N, u_m, \nu)$ is the optimal $2dn$-term representation of $\mathbf{c}(N, u_m, \nu)$.

The following lemma guarantees that any sufficiently significant frequency, i.e., any frequency whose corresponding Fourier coefficient has a sufficiently large absolute value, will be found by Algorithm 5. Further, it also implies that the Fourier coefficient estimate given by the algorithm is accurate.

Lemma 3.26 (Lemma 3.10 in [BZI19]) Let $f \in C_{2\pi}$ be $P(n, d, B)$-structured sparse with bandwidth N and noise threshold $\varepsilon > 0$. Let $\eta \in C_{2\pi}$ such that $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. Let u_1 be a prime and $s_1, t_1 \in \mathbb{N}$ such that with $K = 8dn \left\lfloor \log_{s_1} \frac{N}{u_1} \right\rfloor + 1$ and $M = 2(n+1) \left\lfloor \log_{u_1} N \right\rfloor + 1$ we have that $B < u_1 < \dots < u_M$ are prime numbers, $t_1 < \dots < t_L < s_1 < \dots < s_K$, and $s_1, \dots, s_K, t_1, \dots, t_L, u_1, \dots, u_M$ are pairwise relatively prime with $\prod_{l=1}^L t_l \geq \frac{N}{s_1 u_1}$. Set

$$\begin{aligned}
\delta &\coloneqq \max_{\substack{\nu \in \left\{-\left\lceil \frac{u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u_m}{2} \right\rfloor\right\} \\ u_m \text{ hashes well}}} \left\{ \delta^{(m,\nu)} \right\} \\
&= \max_{\substack{\nu = \left\{-\left\lceil \frac{u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u_m}{2} \right\rfloor\right\} \\ u_m \text{ hashes well}}} \left\{ \frac{1}{2dn} \left\| \mathbf{c}(N, u_m, \nu) - \mathbf{c}^{\text{opt}}_{2dn}(N, u_m, \nu) \right\|_1 \right. \\
&\qquad \left. + \left\| \mathbf{c}(N, \mathbb{Z}, u_m, \nu) - \mathbf{c}(u_m, \nu) \right\|_1 \right\}.
\end{aligned}$$

Then each $\omega \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$ with $|c_\omega| > \varepsilon + 4\delta$ is added to the output R of Algorithm 5 in line 17, and its coefficient estimate from lines 13 and 14 satisfies

$$|x_R(\omega) - c_\omega| \leq 2\delta.$$

Proof. Let u_m be a good hashing prime, $\nu \in \left\{-\left\lceil \frac{u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u_m}{2} \right\rfloor\right\}$ a residue modulo u_m and assume that $\omega \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$ is contained in $R^{(m,\nu),\,\text{opt}}_{dn} \setminus R^{(m,\nu)}$, where $R^{(m,\nu)}$ are the frequencies returned in line 9 of Algorithm 5 by applying Algorithm 4 with sparsity dn to $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}}^q)^T)^T_{\boldsymbol{\rho}^T_{m,\nu}}$. This means that ω is one of the dn frequencies congruent to ν modulo u_m with largest magnitude Fourier coefficients, but it is not contained in $R^{(m,\nu)}$. Since the residue of ω modulo u_m is unique, ω cannot be contained in any of the sets $R^{(m,\tilde{\nu})}$ for $\tilde{\nu} \in \left\{-\left\lceil \frac{u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u_m}{2} \right\rfloor\right\} \setminus \{\nu\}$.

Let us first have a look at the error caused by frequencies with small Fourier coefficients that are not included in the set R of returned frequencies, even though this is not part of the statement of the lemma. Let $\omega \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$ with $|c_\omega| \leq \varepsilon + 4\delta$ and assume that it is not included in the reconstruction R in line 17. Then we have that

$$|x_R(\omega) - c_\omega| \leq \varepsilon + 4\delta.$$

Recall that δ is defined as the maximum over the values $\delta^{(m,\nu)}$, where

$$\delta^{(m,\nu)} = \frac{1}{2dn} \left\| \mathbf{c}(N, u_m, \nu) - \mathbf{c}^{\mathrm{opt}}_{2dn}(N, u_m, \nu) \right\|_1 + \left\| \mathbf{c}(N, \mathbb{Z}, u_m, \nu) - \mathbf{c}(u_m, \nu) \right\|_1$$

for all good hashing primes u_m. The first summand measures the distance of the restriction of $f + \eta$ to frequencies congruent to ν modulo u_m to being a $2dn$-sparse function, and the second summand measures how good the assumed bandwidth N is for $f + \eta$. As for all good hashing primes the restriction of f to the frequencies congruent to ν modulo u_m is at most dn-sparse, the error caused by omitting ω from the reconstruction R is small if the noise is not too dominant and the bandwidth is chosen well.

If $|c_\omega| > \varepsilon + 4 \cdot \delta$ and $\omega \notin R^{(m,\nu)}$, it also follows that $|c_\omega| > 4 \cdot \delta^{(m,\nu)}$ for all good hashing primes u_m and all residues $\nu \in \left\{ -\left\lceil \frac{u_m}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{u_m}{2} \right\rfloor \right\}$. By Lemma 3.18 (i), ω will be reconstructed more than $\frac{K}{2}$ times by Algorithm 4 in line 9 of Algorithm 5. Hence, ω can only not be contained in $R^{(m,\nu)}$ if there exist $dn + 1$ frequencies $\tilde{\omega}$ in $R^{(m,\nu)} \setminus R^{(m,\nu),\,\mathrm{opt}}_{dn}$ that satisfy $\left| x^{(m,\nu)}_{\tilde{\omega}} \right| \geq \left| x^{(m,\nu)}_{\omega} \right|$, i.e., that have coefficient estimates with greater magnitude than the one of ω. Recall that u_m hashes all support sets $S_1, \ldots, S_n$ well, so f has at most dn energetic frequencies that are congruent to ν modulo u_m. Suppose that all frequencies with this residue modulo u_m are ordered by magnitude of their Fourier coefficients, i.e.,

$$\left| c_{\omega^{(m,\nu)}_1} \right| \geq \left| c_{\omega^{(m,\nu)}_2} \right| \geq \cdots \geq \left| c_{\omega^{(m,\nu)}_{dn}} \right| \geq \underbrace{\left| c_{\omega^{(m,\nu)}_{dn+1}} \right|}_{\leq \varepsilon} \geq \cdots \quad .$$

Then $|c_{\tilde{\omega}}| \leq \left| c_{\omega^{(m,\nu)}_{dn+1}} \right| \leq |c_\omega|$ for all $\tilde{\omega} \in R^{(m,\nu)} \setminus R^{(m,\nu),\,\mathrm{opt}}_{dn}$. By Lemma 3.18 (ii) we have for all $\bar{\omega}$ that are reconstructed more than $\frac{K}{2}$ times if Algorithm 4 is applied to $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T_{\boldsymbol{\rho}^T_{m,\nu}}$ that

$$\left| x^{(m,\nu)}_{\bar{\omega}} - c_{\bar{\omega}} \right| \leq \sqrt{2}\delta^{(m,\nu)}. \tag{3.8}$$

It follows from (3.8) that

$$\left| x^{(m,\nu)}_{\bar{\omega}} \right| \leq |c_{\bar{\omega}}| + \sqrt{2}\delta^{(m,\nu)} \qquad \text{and} \tag{3.9}$$

$$\left| x^{(m,\nu)}_{\bar{\omega}} \right| \geq |c_{\bar{\omega}}| - \sqrt{2}\delta^{(m,\nu)}. \tag{3.10}$$

As $\bar{\omega}$ was chosen arbitrarily from the frequencies reconstructed more than $\frac{K}{2}$ times, (3.8) to (3.10) also hold for the frequencies ω and $\tilde{\omega}$ from above. Thus, we find for all $\omega \in R^{(m,\nu),\,\mathrm{opt}}_{dn} \setminus R^{(m,\nu)}$ that

$$\begin{aligned}
\left| c_{\omega^{(m,\nu)}_{dn+1}} \right| + \sqrt{2}\delta^{(m,\nu)} &\geq |c_{\tilde{\omega}}| + \sqrt{2}\delta^{(m,\nu)} \\
&\geq \left| x^{(m,\nu)}_{\tilde{\omega}} \right| \geq |x_\omega| \\
&\geq |c_\omega| - \sqrt{2}\delta^{(m,\nu)} \\
&\geq \left| c_{\omega^{(m,\nu)}_{dn+1}} \right| - \sqrt{2}\delta^{(m,\nu)}.
\end{aligned}$$

Hence,

$$|c_\omega| \leq \left| c_{\omega_{dn+1}^{(m,\nu)}} \right| + 2\sqrt{2}\delta^{(m,\nu)} \leq \varepsilon + 2\sqrt{2}\delta^{(m,\nu)},$$

which contradicts $|c_\omega| > \varepsilon + 4\delta$. Consequently, we obtain that ω was indeed reconstructed by Algorithm 4, so $\omega \in R^{(m,\nu)}$. Since u_m was an arbitrary good hashing prime, this holds for all more than $\frac{M}{2}$ good hashing primes. Thus, ω is contained in more than $\frac{M}{2}$ sets $R^{(m,\nu)}$ and will be considered from line 12 of Algorithm 5 onward.

Before we can show that ω will indeed be added to R in line 17, we first have to prove the accuracy of the corresponding coefficient estimate x_ω. From (3.8) it follows that

$$\left| \mathrm{Re}\left(x_\omega^{(m,\nu)} \right) - \mathrm{Re}(c_\omega) \right| \leq \left| x_\omega^{(m,\nu)} - c_\omega \right| \leq \sqrt{2}\delta^{(m,\nu)} \leq \sqrt{2}\delta, \tag{3.11}$$

and, analogously, the same estimate holds for the imaginary parts. As the inequalities are true for all more than $\frac{M}{2}$ good hashing primes u_m, they also hold for the medians in lines 13 and 14 of Algorithm 5. These are taken over at most M coefficient estimates $x_{\tilde{\omega}}^{(m,\nu)}$ for ω, since for each prime u_m the frequency ω can be contained in at most one set $R^{(m,\nu)}$ with $\nu \in \left\{ -\left\lceil \frac{u_m}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{u_m}{2} \right\rfloor \right\}$. Thus, we obtain that

$$|\mathrm{Re}\,(x_\omega) - \mathrm{Re}(c_\omega)| \leq \sqrt{2}\delta \quad \text{and} \quad |\mathrm{Im}\,(x_\omega) - \mathrm{Im}(c_\omega)| \leq \sqrt{2}\delta.$$

Combining these two estimates yields that

$$|x_\omega - c_\omega| = \sqrt{(\mathrm{Re}\,(x_\omega - c_\omega))^2 + (\mathrm{Im}\,(x_\omega - c_\omega))^2} \leq \sqrt{\left(\sqrt{2}\delta\right)^2 + \left(\sqrt{2}\delta\right)^2} = 2\delta.$$

All that remains to be shown is that ω with $|c_\omega| > \varepsilon + 4\delta$ will actually be added to R in line 17. Similarly to (3.10) we find that $|x_\omega| \geq |c_\omega| - 2\delta$. Together with $|c_\omega| > \varepsilon + 4\delta$ this implies that $|x_\omega| > \varepsilon + 2\delta$. Then it is only possible that ω is not included in the output set R if x_ω is not among the Bn largest magnitude coefficient estimates, i.e., if there exist Bn other frequencies $\tilde{\omega}$ that satisfy $|x_{\tilde{\omega}}| \geq |x_\omega|$. We know that ω is energetic, which means that at least one of these $\tilde{\omega}$ must have a Fourier coefficient with $|c_{\tilde{\omega}}| \leq \varepsilon$. Then an analog to (3.9) yields

$$|x_\omega| \leq |x_{\tilde{\omega}}| \leq |c_{\tilde{\omega}}| + 2\delta \leq \varepsilon + 2\delta,$$

which contradicts $|x_\omega| > \varepsilon + 2\delta$. Hence, ω will be added to R in line 17. □

Example 3.27 (Example 3.2 continued) Recall the $P(2,2,9)$-structured sparse function f with bandwidth $N = 1{,}024$ from Example 3.2. Let us now find the required primes for this example. Since

$$2 \cdot 3 < \frac{N}{Bnd} \leq 2 \cdot 3 \cdot 5,$$

we set $L \coloneqq 3$, $t_1 \coloneqq 2$, $t_2 \coloneqq 3$ and $t_3 \coloneqq 5$. Then s_1 can be chosen as the smallest prime greater than both $dn = 4$ and $t_L = 5$,

$$s_1 \coloneqq 7 > \max\{dn, t_L\}.$$

Then $K = 8dn \left\lfloor \log_{s_1} \frac{N}{B} \right\rfloor + 1 = 65$ and we find that

$$\{s_1, \ldots, s_K\} = \{7, 11, 13, \ldots, p_{68} = 337\}.$$

Setting $u_1 \coloneqq p_{69} = 347 > \max\{B, s_K\}$, we obtain $M = 2(n+1)\left\lfloor \log_{u_1} N \right\rfloor + 1 = 7$ and

$$\{u_1, \dots, u_M\} = \{347, 349, \dots, p_{75} = 379\}.$$

Note that since B and N are very small with $B \not\gg d^2 n \log N$ to keep the example simple, we have to choose many primes. Algorithm 5 is not efficient for the chosen parameters, as the bandwidth needs to be significantly larger than the sparsity in order for our method to be fast. We only computed the required primes for the example in order to illustrate the prime-choosing procedure. ◊

Using the s_k, t_l and u_m from Definition 3.20, the following theorem provides us with the runtime and error bounds of Algorithm 5.

Theorem 3.28 (Theorem 3.13 in [BZI19]) *Let $f \in C_{2\pi}$ be $P(n,d,B)$-structured sparse with bandwidth $N \in \mathbb{N}$ and noise threshold $\varepsilon > 0$. Let $\eta \in C_{2\pi}$ such that $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. Let $t_1, \dots, t_L$ be the smallest primes with $\prod_{l=1}^{L} t_l \geq \frac{N}{Bdn}$. Set s_1 as the smallest prime greater than $\max\{dn, t_L\}$, $K = 8dn\left\lfloor \log_{s_1} \frac{N}{B} \right\rfloor + 1$ and $s_2, \dots, s_K$ as the first $K-1$ primes greater than s_1. Let u_1 be the smallest prime greater than $\max\{B, s_K\}$, $M = 2(n+1)\left\lfloor \log_{u_1} N \right\rfloor + 1$ and $u_2, \dots, u_M$ the first $M-1$ primes greater than u_1. Let further δ be defined as*

$$\delta \coloneqq \max_{\substack{\nu \in \left\{-\left\lceil \frac{u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u_m}{2} \right\rfloor\right\} \\ u_m \text{ hashes well}}} \left\{ \frac{1}{2dn} \left\| \mathbf{c}(N, u_m, \nu) - \mathbf{c}^{\mathrm{opt}}_{2dn}(N, u_m, \nu) \right\|_1 + \left\| \mathbf{c}(N, \mathbb{Z}, u_m, \nu) - \mathbf{c}(u_m, \nu) \right\|_1 \right\}.$$

Then the output $(R, \mathbf{x}_R)$ of Algorithm 5 satisfies

$$\|\mathbf{c}(N) - \mathbf{x}_R\|_2 \leq \left\| \mathbf{c}(N) - \mathbf{c}^{\mathrm{opt}}_{Bn}(N) \right\|_2 + \sqrt{Bn} \cdot (\varepsilon + 6\delta).$$

If $B > s_K$, the output can be computed in a runtime of

$$\mathcal{O}\left(\frac{d^2 n^3 (B + n \log N) \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log N \log\left(2dn \log \frac{N}{B}\right) \log^2\left(\frac{B + n \log N}{\log B}\right)}{\log^2 B \log^2(2dn) \log\log \frac{N}{2Bdn}} \right),$$

and the algorithm has a sampling complexity of

$$\mathcal{O}\left(\frac{d^2 n^3 (B + n \log N) \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log N \log\left(2dn \log \frac{N}{B}\right) \log\left(\frac{B + n \log N}{\log B}\right)}{\log^2 B \log^2(2dn) \log\log \frac{N}{2Bdn}} \right).$$

Proof. Recall that the entries of $\mathbf{c}_R(N) \in \mathbb{C}^N$ are the coefficients $c_\omega(f + \eta)$ of the perturbed function $f + \eta$ for the frequencies contained in the output R of Algorithm 5 and zero for the frequencies not contained in R, whereas the entries of $\mathbf{c}(N)$ are the Fourier coefficients of $f + \eta$ for all $\omega \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$. The triangle inequality yields

$$\|\mathbf{c}(N) - \mathbf{x}_R\|_2 \leq \|\mathbf{c}(N) - \mathbf{c}_R(N)\|_2 + \|\mathbf{c}_R(N) - \mathbf{x}_R\|_2. \tag{3.12}$$

The square of first summand in (3.12) can be written as

$$\begin{aligned}
&\|\mathbf{c}(N) - \mathbf{c}_R(N)\|_2^2 \\
&= \sum_{\omega=-\lceil \frac{N}{2} \rceil+1}^{\lfloor \frac{N}{2} \rfloor} |c_\omega - c_R(\omega)|^2 \\
&= \sum_{\omega \notin R} |c_\omega|^2 + \sum_{\omega \in R} \underbrace{|c_\omega - c_R(\omega)|^2}_{=0} \\
&= \sum_{\omega \notin R_{Bn}^{\mathrm{opt}}} |c_\omega|^2 + \sum_{\omega \in R_{Bn}^{\mathrm{opt}} \setminus R} |c_\omega|^2 - \sum_{\omega \in R \setminus R_{Bn}^{\mathrm{opt}}} |c_\omega|^2 \\
&= \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right\|_2^2 + \sum_{\omega \in R_{Bn}^{\mathrm{opt}} \setminus R} |c_\omega|^2 - \sum_{\omega \in R \setminus R_{Bn}^{\mathrm{opt}}} |c_\omega|^2.
\end{aligned}$$

For every frequency $\omega \in R_{Bn}^{\mathrm{opt}} \setminus R$ we know by Lemma 3.26 that $|c_\omega| \leq \varepsilon + 4\delta$, because otherwise it would be contained in R. As $R_{Bn}^{\mathrm{opt}} \setminus R$ contains at most Bn elements, we find

$$\begin{aligned}
&\|\mathbf{c}(N) - \mathbf{c}_R(N)\|_2^2 \\
&= \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right\|_2^2 + \underbrace{\sum_{\omega \in R_{Bn}^{\mathrm{opt}} \setminus R} |c_\omega|^2}_{\leq Bn(\varepsilon+4\delta)^2} - \underbrace{\sum_{\omega \in R \setminus R_{Bn}^{\mathrm{opt}}} |c_\omega|^2}_{\geq 0} \\
&\leq \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right\|_2^2 + Bn(\varepsilon + 4\delta)^2.
\end{aligned}$$

For the second summand in (3.12) we consider consider a frequency $\omega \in R$. For each of the more than $\frac{M}{2}$ good hashing primes ω has to be contained in exactly one of the sets $R^{(m,\nu)}$ returned in line 9 of Algorithm 5 by applying Algorithm 4 with sparsity dn to $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}}^q)^T)^T_{\boldsymbol{\rho}^T_{m,\nu}}$. Thus, ω must have been reconstructed more than $\frac{K}{2}$ times by Algorithm 4. Hence, it follows from Lemma 3.18 that

$$\left| x_\omega^{(m,\nu)} - c_\omega \right| \leq \sqrt{2}\delta^{(m,\nu)},$$

which, analogously to (3.8) and (3.11) in the proof of Lemma 3.26, yields

$$|x_R(\omega) - c_\omega| \leq 2\delta.$$

Since R contains at most Bn elements, we find that

$$\|\mathbf{c}_R(N) - \mathbf{x}_R\|_2^2 = \sum_{\omega \in R} \underbrace{|c_\omega - x_R(\omega)|^2}_{\leq 4\delta^2} \leq 4Bn\delta^2.$$

Combining all these estimates we obtain that

$$\begin{aligned}
\|\mathbf{c}(N) - \mathbf{x}_R\|_2 &\leq \sqrt{\|\mathbf{c}(N) - \mathbf{c}_R(N)\|_2^2 + \|\mathbf{c}_R(N) - \mathbf{x}_R\|_2^2} \\
&\leq \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right\|_2 + \sqrt{Bn}(\varepsilon + 4\delta) + 2\sqrt{Bn}\delta \\
&= \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right\|_2 + \sqrt{Bn}(\varepsilon + 6\delta).
\end{aligned}$$

In order to determine the runtime of the algorithm, let us first consider the runtime of the calculation of the CDFTs in line 5 of Algorithm 5. We have to calculate the CDFTs of length $s_k t_l u_m$ of the vectors $\mathbf{a}^{s_k t_l u_m}$ of equidistant samples of $f+\eta$ for all $k \in \{1,\dots,K\}$, $l \in \{1,\dots,L\}$ and $m \in \{1,\dots,M\}$. By Section 1.1.1 and Remark 1.9, a CDFT of length $s_k t_l u_m$ has runtime $\mathcal{O}(s_k t_l u_m \log(s_k t_l u_m))$. It was shown in Lemma 4 and Section IV in [IS08], and Section 2 in [Iwe13] that

$$t_L = \mathcal{O}\left(\log \frac{N}{2Bdn}\right) \quad \text{and} \quad s_K = \mathcal{O}\left(dn \log_{2dn} \frac{N}{B} \log\left(2dn \log \frac{N}{B}\right)\right).$$

If $s_K \le B$, we can set $u_1 \coloneqq p_b > B \ge p_{b-1}$ to be the first prime greater than B, so by the first formulation of the Prime Number Theorem, see Theorem 2.27 (i), we find that

$$b - 1 = \pi(B) = \mathcal{O}\left(\frac{B}{\log B}\right)$$

and, since $M = 2(n+1)\left\lfloor \log_{u_1} N \right\rfloor + 1 = \mathcal{O}(n \log_B N)$,

$$b - 1 + M = \mathcal{O}\left(\frac{B}{\log B} + n \log_B N\right) = \mathcal{O}\left(\frac{B + n \log N}{\log B}\right),$$

where π is the prime counting function. The second formulation of the Prime Number Theorem, see Theorem 2.27 (ii), yields for $u_M = p_{b-1+M}$ that

$$u_M = \mathcal{O}((b-1+M)\log(b-1+M)) = \mathcal{O}\left(\frac{B+n\log N}{\log B} \log\left(\frac{B+n\log N}{\log B}\right)\right).$$

Recall that by Lemma 2.20 (i)

$$\sum_{\substack{2 \le p \le R \\ p \text{ prime}}} p = \mathcal{O}\left(\frac{R^2}{\log R}\right).$$

However, if we estimated $\sum_{m=1}^{M} u_m \log u_m$ by adding $p \log p$ for all primes that are at most u_M, we would take into account many primes that do not contribute to the sum, as the u_m are rather large because $u_m > s_k > t_l$ for all k, l and m. Instead, we estimate the hashing primes by $u_m \le u_M$ for all $m \in \{1,\dots,M\}$, i.e.,

$$\sum_{m=1}^{M} u_m \log u_m = \mathcal{O}(M \cdot u_M \log u_M).$$

Since $u_1 > s_K$, we obtain for the runtime of the CDFT calculations in line 5 that

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L}\sum_{m=1}^{M} s_k t_l u_m \log(s_k t_l u_m)\right) \\
&= \mathcal{O}\left(\sum_{k=1}^{K} s_k \sum_{l=0}^{L} t_l \sum_{m=1}^{M} u_m \log u_m\right) \\
&= \mathcal{O}\left(\frac{t_L^2}{\log t_L} \cdot \frac{s_K^2}{\log s_K} \cdot M \cdot u_M \log u_M\right)
\end{aligned}$$

$$= \mathcal{O}\left(\frac{\log^2 \frac{N}{2Bdn}}{\log\log \frac{N}{2Bdn}} \cdot \frac{d^2n^2 \log^2 \frac{N}{B} \log^2\left(2dn \log \frac{N}{B}\right)}{\log^2(2dn) \log\left(dn \log_{2dn} \frac{N}{B} \log\left(2dn \log \frac{N}{B}\right)\right)}\right.$$
$$\left. \cdot \frac{n \log N}{\log B} \cdot \frac{B + \log N}{\log B} \log\left(\frac{B + n \log N}{\log B}\right) \log\left(\frac{B + \log N}{\log B} \log\left(\frac{B + n \log N}{\log B}\right)\right)\right)$$
$$= \mathcal{O}\left(\frac{d^2n^3(B + n \log N) \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log N \log(2dn \log \frac{N}{B}) \log^2\left(\frac{B+n \log N}{\log B}\right)}{\log^2 B \log^2(2dn) \log\log \frac{N}{2Bdn}}\right).$$

Now we can estimate the runtime of the remaining steps of the algorithm. The least common multiple q in line 4 does not actually have to be computed, it is just defined there in order to introduce more convenient notation. The algorithm also does not need all q samples of f; rather, any of the required samples can be written as an entry of $\mathbf{a}^q$.

In line 9, we apply Algorithm 4 to the column of $\mathcal{G}_{L,K} \cdot \left(\mathcal{H}_{M,L,K} \circledast \left(\widehat{\mathbf{a}^q}\right)^T\right)^T$ corresponding to the residue ν modulo u_m. We know from Lemma 3.18 (iii) that the runtime of Algorithm 4 is dominated by the computation of the CDFTs. Since the CDFTs in line 5 have even greater lengths than the CDFTs required for Algorithm 4, the runtime of lines 7 to 11 of Algorithm 5 is insignificant compared to runtime of line 5. In order to find out for which frequencies lines 12 to 15 have to be executed, we can sort the $2dn \sum_{m=1}^{M} u_m$ frequencies that are returned by all the calls of Algorithm 4 by size and count how often each distinct frequency appears. This can be done in

$$\mathcal{O}\left(2dn\left(\sum_{m=1}^{M} u_m\right) \cdot \log\left(2dn \sum_{m=1}^{M} u_m\right)\right) = \mathcal{O}\left(dnMu_M \cdot \log\left(dnMu_M\right)\right)$$

time, so its computational effort is also dominated by the effort of the CDFT computation. There are at most

$$\frac{2}{M} \cdot \sum_{m=1}^{M} \sum_{\nu=-\left\lceil \frac{u_m}{2} \right\rceil + 1}^{\left\lfloor \frac{u_m}{2} \right\rfloor} 2dn = \frac{4dn}{M} \cdot \sum_{m=1}^{M} u_m = \mathcal{O}\left(4dn \cdot u_M\right)$$

frequencies that have been found more than $\frac{M}{2}$ times. If we fix one of these frequencies, ω, then for each u_m there exists exactly one residue $\nu^{(m)} \in \left\{-\left\lceil \frac{u_m}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{u_m}{2} \right\rfloor\right\}$ with $\omega \equiv \nu^{(m)} \mod u_m$. Since the $2dn$ frequencies recovered for any fixed residue modulo some hashing prime u_m are distinct, each frequency can be reconstructed at most M times by all hashing primes together. This means that the medians in lines 13 and 14 are taken over at most M elements. As medians can be computed with the help of a sorting algorithm, both lines have a runtime of $\mathcal{O}(M \log M)$. Combining these considerations, we obtain that lines 12 to 15 require

$$\mathcal{O}\left(4dn \cdot u_M \cdot M \log M\right)$$

arithmetical operations, which is dominated by the effort of the CDFT computations in line 5. Finally, sorting the $\mathcal{O}\left(4dn \cdot u_M\right)$ coefficient estimates in line 16 has a runtime of

$$\mathcal{O}\left(4dnu_M \log\left(4dnu_M\right)\right).$$

Consequently, the runtime of Algorithm 5 is determined by that of line 5, which yields

an overall runtime of

$$\mathcal{O}\left(\frac{d^2n^3(B+n\log N)\log^2\frac{N}{2Bdn}\log^2\frac{N}{B}\log N\log(2dn\log\frac{N}{B})\log^2\left(\frac{B+n\log N}{\log B}\right)}{\log^2 B\log^2(2dn)\log\log\frac{N}{2Bdn}}\right).$$

Each of the KLM CDFTs of size $s_kt_lu_m$ requires $s_kt_lu_m$ samples of f, so we find for the sampling complexity of Algorithm 5 that

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L}\sum_{m=1}^{M}s_kt_lu_m\right)\\
=&\,\mathcal{O}\left(\sum_{k=1}^{K}s_k\sum_{l=0}^{L}t_l\sum_{m=1}^{M}u_m\right)\\
=&\,\mathcal{O}\left(\frac{t_L^2}{\log t_L}\cdot\frac{s_K^2}{\log s_K}\cdot M\cdot u_M\right)\\
=&\,\mathcal{O}\left(\frac{d^2n^3(B+n\log N)\log^2\frac{N}{2Bdn}\log^2\frac{N}{B}\log N\log(2dn\log\frac{N}{B})\log\left(\frac{B+n\log N}{\log B}\right)}{\log^2 B\log^2(2dn)\log\log\frac{N}{2Bdn}}\right).
\end{aligned}$$

□

If not only f, but also $f+\eta$ is bandlimited with bandwidth N, the error bound from Theorem 3.28 can be simplified.

Corollary 3.29 (Corollary 3.14 in [BZI19]) Let $f\in C_{2\pi}$ be $P(n,d,B)$-structured sparse with bandwidth N and noise threshold $\varepsilon>0$. Let $\eta\in C_{2\pi}$ such that $f+\eta$ also has bandwidth N with $\mathbf{c}(\eta)\in\ell^1$ and $\|\mathbf{c}(\eta)\|_\infty\leq\varepsilon$. Choosing $s_1,\dots,s_K$, $t_1,\dots,t_L$, $u_1,\dots,u_M$ as in Theorem 3.28, the output $(R,\mathbf{x}_R)$ of Algorithm 5 satisfies

$$\|\mathbf{c}(N)-\mathbf{x}_R\|_2\leq\left\|\mathbf{c}(N)-\mathbf{c}_{Bn}^{\mathrm{opt}}(N)\right\|_2+\sqrt{Bn}\left(\varepsilon+\frac{3}{dn}\left\|\mathbf{c}(N)-\mathbf{c}_{2Bn}^{\mathrm{opt}}(N)\right\|_1\right).$$

Proof. By definition of δ we have that

$$\begin{aligned}
\delta=&\max_{\substack{\nu=\{-\lceil\frac{u_m}{2}\rceil+1,\dots,\lfloor\frac{u_m}{2}\rfloor\}\\ u_m\text{ hashes well}}}\left\{\delta^{(m,\nu)}\right\}=:\delta^{(m',\nu')}\\
=&\,\frac{1}{2dn}\left\|\mathbf{c}\left(N,u_{m'},\nu'\right)-\mathbf{c}_{2dn}^{\mathrm{opt}}\left(N,u_{m'},\nu'\right)\right\|_1+\left\|\mathbf{c}\left(N,\mathbb{Z},u_{m'},\nu'\right)-\mathbf{c}\left(u_{m'},\nu'\right)\right\|_1 \quad (3.13)
\end{aligned}$$

for some residue $\nu'\in\left\{-\lceil\frac{u_{m'}}{2}\rceil+1,\dots,\lfloor\frac{u_{m'}}{2}\rfloor\right\}$ modulo a good hashing prime $u_{m'}$. Since f and $f+\eta$ are bandlimited, the second summand in (3.13) is 0. We find the following

estimate

$$\begin{aligned}
\delta &\leq \sum_{\nu=-\left\lceil \frac{u_{m'}}{2} \right\rceil+1}^{\left\lfloor \frac{u_{m'}}{2} \right\rfloor} \delta^{(m',\nu)} \\
&= \sum_{\nu=-\left\lceil \frac{u_{m'}}{2} \right\rceil+1}^{\left\lfloor \frac{u_{m'}}{2} \right\rfloor} \frac{1}{2dn} \left\| \mathbf{c}(N, u_{m'}, \nu) - \mathbf{c}_{2dn}^{\mathrm{opt}}(N, u_{m'}, \nu) \right\|_1 \\
&= \sum_{\nu=-\left\lceil \frac{u_{m'}}{2} \right\rceil+1}^{\left\lfloor \frac{u_{m'}}{2} \right\rfloor} \frac{1}{2dn} \sum_{\substack{\omega=-\left\lceil \frac{N}{2} \right\rceil+1 \\ \omega \equiv \nu \bmod u_{m'}}}^{\left\lfloor \frac{N}{2} \right\rfloor} \underbrace{\left| c_\omega - \left(\mathbf{c}_{2dn}^{\mathrm{opt}}(N, u_{m'}, \nu) \right)_\omega \right|}_{= \begin{cases} |c_\omega| & \text{if } \omega \notin R_{2dn}^{(m',\nu),\,\mathrm{opt}}, \\ 0 & \text{otherwise.} \end{cases}} \\
&= \sum_{\substack{\omega=-\left\lceil \frac{N}{2} \right\rceil+1 \\ \omega \notin \bigcup_{\nu=-\left\lceil \frac{u_{m'}}{2} \right\rceil+1}^{\left\lfloor \frac{u_{m'}}{2} \right\rfloor} R_{2dn}^{(m',\nu),\,\mathrm{opt}}}}^{\left\lfloor \frac{N}{2} \right\rfloor} \frac{1}{2dn} |c_\omega| \\
&\leq \sum_{\omega=-\left\lceil \frac{N}{2} \right\rceil+1}^{\left\lfloor \frac{N}{2} \right\rfloor} \frac{1}{2dn} \underbrace{\left| c_\omega - \left(\mathbf{c}_{2Bn}^{\mathrm{opt}}(N) \right)_\omega \right|}_{= \begin{cases} |c_\omega| & \text{if } \omega \notin R_{2Bn}^{\mathrm{opt}}, \\ 0 & \text{otherwise.} \end{cases}} \\
&= \frac{1}{2dn} \left\| \mathbf{c}(N) - \mathbf{c}_{2Bn}^{\mathrm{opt}}(N) \right\|_1,
\end{aligned}$$

since

$$\left| \bigcup_{\nu=-\left\lceil \frac{u_{m'}}{2} \right\rceil+1}^{\left\lfloor \frac{u_{m'}}{2} \right\rfloor} R_{2dn}^{(m',\nu),\,\mathrm{opt}} \right| = 2dn \cdot u_{m'} \geq 2Bn = \left| R_{2Bn}^{\mathrm{opt}} \right|.$$

Consequently, the error bound from Theorem 3.28 reduces to

$$\begin{aligned}
\left\| \mathbf{c}(N) - \mathbf{x}_R \right\|_2 &\leq \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right\|_2 + \sqrt{Bn} \cdot (\varepsilon + 6\delta) \\
&\leq \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right\|_2 + \sqrt{Bn} \cdot \left(\varepsilon + \frac{3}{dn} \left\| \mathbf{c}(N) - \mathbf{c}_{2Bn}^{\mathrm{opt}}(N) \right\|_1 \right).
\end{aligned}$$

□

3.4 Algorithm for Functions with Simplified Fourier Structure

Algorithm 5, which we introduced in Section 3.3.2, always requires M hashing primes of which more than $\frac{M}{2}$ have to be good, since in general it is not possible to guarantee that any given prime is a good hashing prime. However, in certain special cases, the frequency

structure already implies that primes satisfying some additional, easily to check property are good hashing primes. A very important example for this are block sparse functions, where each of the n support sets S_j is an interval of length B,

$$S_j \coloneqq \{\omega_j, \omega_j + 1, \dots, \omega_j + B - 1\},$$

so the support consists of n blocks of length B. For the function classes for which one good hashing prime suffices, we will introduce a simplified, faster version of Algorithm 5 in the following section.

3.4.1 Structured Sparse Functions Requiring Only One Hashing Prime

If certain additional information about the polynomials generating the support sets $S_1, \dots, S_n$ is known, the number M of required hashing primes can be reduced to one. We know by Lemma 3.6 that a prime u does not hash a support set S_j well if and only if u divides all coefficients corresponding to the non-constant terms. Thus, we can make the following observation.

Theorem 3.30 (Theorem 4.1 in [BZI19]) *Let $f \in C_{2\pi}$ be $P(n, d, B)$-structured sparse with bandwidth N and noise threshold $\varepsilon > 0$. Let $\eta \in C_{2\pi}$ such that $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. Let the support set $S = \bigcup_{j=1}^n S_j$ of f be generated by the non-constant polynomials $P_j(x) = \sum_{k=0}^d a_{jk} x^k$ for $j \in \{1, \dots, n\}$. Let $u > B$ be a prime such that for all $j \in \{1, \dots, n\}$ there exists a $k_j \in \{1, \dots, d\}$ with $p \nmid a_{jk_j}$. Then u hashes all support sets well. Set $M = 1$ and $s_1, \dots, s_K$ and $t_1, \dots, t_L$ as in Theorem 3.28. If $B > s_K$, the runtime of Algorithm 5 reduces to*

$$\mathcal{O}\left(\frac{u \log u \cdot (dn)^2 \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log\left(2dn \log \frac{N}{B}\right)}{\log^2(2dn) \log\log \frac{N}{2Bdn}}\right),$$

while only

$$\mathcal{O}\left(\frac{u \cdot (dn)^2 \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log\left(2dn \log \frac{N}{B}\right)}{\log^2(2dn) \log\log \frac{N}{2Bdn}}\right)$$

samples of $f + \eta$ are being used. If $B \leq s_K$, we obtain a runtime of

$$\mathcal{O}\left(\frac{u \cdot (dn)^2 \cdot \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log^2\left(2dn \log \frac{N}{B}\right)}{\log^2(2dn) \log\log \frac{N}{2Bdn}}\right)$$

and a sampling complexity of

$$\mathcal{O}\left(\frac{u \cdot (dn)^2 \cdot \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log\left(2dn \log \frac{N}{B}\right)}{\log^2(2dn) \log\log \frac{N}{2Bdn}}\right).$$

Proof. Lemma 3.6 implies that u hashes all n support sets well, so the restriction of f to the frequencies congruent to ν modulo u is at most dn-sparse for all residues $\nu \in \left\{-\left\lceil \frac{u_m}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u_m}{2} \right\rfloor\right\}$. Hence, we can apply Algorithm 4 with sparsity dn to $\mathcal{G}_{L,K} \cdot (\mathcal{H}_{M,L,K} \circledast (\widehat{\mathbf{a}^q})^T)^T_{\boldsymbol{\rho}^T_{m,\nu}}$ for every residue ν modulo u, always obtaining a good reconstruction of the restriction by Lemma 3.18. As there are no residues modulo u for which more than dn energetic frequencies can collide, there is no need for us to use any further hashing primes and employ median arguments. Hence, it suffices to set $u_1 = u$

and $M = 1$ in Algorithm 5. Lines 12 to 15 of Algorithm 5 do not have to be executed in this setting, because every energetic frequency will be recovered for exactly one residue in line 9.

If $u > s_K$ for the prime s_K used by Algorithm 5, we use the primes $t_1, \dots, t_L$ and $s_1, \dots, s_K$ from Definition 3.20. If $u \leq s_K$, then u might collide with one of the s_k or t_l. In that case we shift the t_l and s_k that are greater than or equal to u to the next greatest prime, so the new t_l and s_k are at most the first prime greater than the original t_l and s_k for Algorithm 5. This small shift does not change the estimates in the proof of Theorem 3.28.

Let us first consider the case that $u > s_K$. We obtain that the computation of the CDFTs in line 5, which, as we have seen in the proof of Theorem 3.28, determines the runtime complexity of Algorithm 5, requires

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u \log\left(s_k t_l u\right)\right) \\
&= \mathcal{O}\left(u \log u \cdot \frac{t_L^2}{\log t_L} \cdot \frac{s_K^2}{\log s_K}\right) \\
&= \mathcal{O}\left(\frac{u \log u \cdot (dn)^2 \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log\left(2dn \log \frac{N}{B}\right)}{\log^2(2dn) \log\log \frac{N}{2Bdn}}\right)
\end{aligned}$$

arithmetical operations. The sampling complexity of the method is

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u\right) \\
&= \mathcal{O}\left(u \cdot \frac{t_L^2}{\log t_L} \cdot \frac{s_K^2}{\log s_K}\right) \\
&= \mathcal{O}\left(\frac{u \cdot (dn)^2 \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log\left(2dn \log \frac{N}{B}\right)}{\log^2(2dn) \log\log \frac{N}{2Bdn}}\right).
\end{aligned}$$

If $u \leq s_K$, we obtain a runtime of

$$\begin{aligned}
&\mathcal{O}\left(u \cdot \sum_{l=0}^{L} t_l \sum_{k=1}^{K} s_k \log s_k\right) \\
&= \mathcal{O}\left(u \cdot \frac{t_L^2}{\log t_L} \cdot s_K^2\right) \\
&= \mathcal{O}\left(\frac{u \cdot (dn)^2 \cdot \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log^2\left(2dn \log \frac{N}{B}\right)}{\log^2(2dn) \log\log \frac{N}{2Bdn}}\right)
\end{aligned}$$

and a sampling complexity of

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u\right) = \mathcal{O}\left(u \cdot \frac{t_L^2}{\log t_L} \cdot \frac{s_K^2}{\log s_K}\right) \\
&= \mathcal{O}\left(\frac{u \cdot (dn)^2 \cdot \log^2 \frac{N}{2Bdn} \log^2 \frac{N}{B} \log\left(2dn \log \frac{N}{B}\right)}{\log^2(2dn) \log\log \frac{N}{2Bdn}}\right).
\end{aligned}$$

□

We now give some conditions on the coefficients of the polynomials $P_1, \dots, P_n$ generating the support sets $S_1, \dots, S_n$ which guarantee that all S_j are hashed well. The conditions arise by tightening the necessary and sufficient requirement of the existence of a coefficient corresponding to a term of degree at least one that is not divisible by u in Theorem 3.30. Hence, all of the conditions are sufficient, but they may not be necessary anymore, which causes them to be easier to prove in practice.

Lemma 3.31 (Lemma 4.2 in [BZI19]) Let $f \in C_{2\pi}$ be $P(n, d, B)$-structured sparse with generating polynomials $P_j(x) = \sum_{k=0}^{d} a_{jk} x^k$ for $j \in \{1, \dots, n\}$. In the following cases any prime $u > B$ is guaranteed to hash all frequency subsets well.

(i) $\forall\, j \in \{1, \dots, n\}\colon \gcd(a_{j1}, \dots, a_{jd}) < B$, which includes $\gcd(a_{j1}, \dots, a_{jd}) = 1$,

(ii) $\forall\, j \in \{1, \dots, n\}\, \exists\, k_j \in \{1, \dots, d\}\colon \left|a_{jk_j}\right| < B$,

(iii) $\forall\, j \in \{1, \dots, n\}\, \exists\, k_j \in \{1, \dots, d\}\colon a_{jk_j} = 1$, which includes monic polynomials,

(iv) $\forall\, j \in \{1, \dots, n\}\colon \deg(P_j) = 1$ and $a_{j1} = 1$, which is the block sparse case.

Example 3.32 We illustrate the conditions from Lemma 3.31 by some examples for $P(2, 2, 9)$-structured sparse functions which can be obtained by slightly modifying the polynomials occurring in Example 3.2.

(i) $P_1(x) = 11x^2 - 21x - 200$ and $P_2(x) = -13x^2 + 27x + 350$

P_1 and P_2 satisfy

$$\gcd(a_{11}, a_{12}) = \gcd(-21, 11) = 1 \quad \text{and} \quad \gcd(a_{21}, a_{22}) = \gcd(27, -13) = 1.$$

(ii) $P_1(x) = 8x^2 - 22x - 200$ and $P_2(x) = -3x^2 + 26x + 350$

Here, we have that $|a_{12}| = 8 < 9 = B$ and $|a_{22}| = 3 < 9 = B$.

(iii) $P_1(x) = 11x^2 + x - 400$ and $P_2(x) = x^2 + 26x - 400$

Here, we have that $a_{11} = 1$ and $a_{22} = 1$.

(iv) $P_1(x) = x - 200$ and $P_2(x) = x + 350$.

Both polynomials are monic and of degree 1, so they each generate a B-length block of frequencies.

For all these pairs of polynomials any prime $u \geq 11 > 9 = B$ hashes the generated support sets S_1 and S_2 well. ◊

If one has already fixed a prime $u > B$ that is supposed to be used as the hashing prime, the following conditions imply that u indeed hashes all support sets well.

Lemma 3.33 (Lemma 4.4 in [BZI19]) Let f be $P(n, d, B)$-structured sparse and generated by the polynomials $P_j(x) = \sum_{k=0}^{d} a_{jk} x^k$ for $j \in \{1, \dots, n\}$. In the following cases a fixed prime $u > B$ is guaranteed to hash all support sets well.

(v) $\forall\, j \in \{1, \dots, n\}\colon u \nmid \sum_{k=1}^{d} a_{jk}$,

(vi) $\forall\, j \in \{1, \dots, n\}\, \exists\, \varepsilon^j = \left(\varepsilon_k^j\right)_{k=1}^{d} \in \{0, 1\}^d\colon u \nmid \sum_{k=1}^{d} (-1)^{\varepsilon_k^j} a_{jk}$,

(vii) One of the conditions (i)–(iv) from Lemma 3.31 is satisfied, where B can be replaced by u in (i) and (ii).

Example 3.34 (Example 3.32 continued) We also illustrate the additional conditions by examples for $P(2,2,9)$-structured sparse functions.

(v) $P_1(x) = 11x^2 - 22x - 200$ and $P_2(x) = -13x^2 + 26x + 350$

For P_1 and P_2 as in Example 3.2 we have that

$$\sum_{k=1}^{2} a_{1k} = -11 \quad \text{and} \quad \sum_{k=1}^{2} a_{2k} = 13,$$

so any fixed prime $u \geq 17$ satisfies (v).

(vi) $P_1(x) = 11x^2 - 22x - 200$ and $P_2(x) = -13x^2 + 26x + 350$

For P_1 and P_2 as in Example 3.2 we have that

$$\sum_{k=1}^{2} (-1)^{\varepsilon_k^1} a_{1k} \in \{-33, -11, 11, 33\} \quad \text{and} \quad \sum_{k=1}^{2} (-1)^{\varepsilon_k^2} a_{2k} \in \{-39, -13, 13, 39\}$$

for all $\varepsilon^j \in \{0,1\}^2$, so any fixed prime $u \geq 17$ satisfies (vi).

$\Diamond$

3.4.2 Block Frequency Sparse Functions

The probably most practically useful condition is condition (iv) in Lemma 3.31, which characterizes functions with frequency support consisting of n blocks of length B. This means that the support sets $S_1, \ldots, S_n$ are of the form

$$S_j = \{\omega_j, \omega_j + 1, \ldots, \omega_j + B - 1\}, \qquad \forall\, j \in \{1, \ldots, n\},$$

where $P_j(x) = x + \omega_j - 1$. In this section we will investigate this special case in more detail and develop a specially adapted version of Algorithm 5 for it, which will have a further improved runtime. Let us first formally define the concept of block sparsity.

Definition 3.35 ((n,B)-block Sparsity (Definition 4.6 in [BZI19])) A $P(n,1,B)$-structured sparse function f is called (n,B)-*block sparse* if the support sets $S_1, \ldots, S_n$ are generated by monic linear polynomials

$$P_j(x) \coloneqq x + a_j, \quad j \in \{1, \ldots, n\}.$$

Example 3.36 Let $N = 1{,}024$, $n = 2$, $B = 9$ and set

$$P_1(x) = x - 200 \qquad \text{and} \qquad P_2(x) = x + 350.$$

Then the support sets generated by P_1 and P_2 are

$$S_1 = \{-199, -198, \ldots, -191\} \qquad \text{and} \qquad S_2 = \{351, 352, \ldots, 359\},$$

and, with $S = S_1 \cup S_2$, the function

$$f(x) \coloneqq \sum_{\omega \in S} e^{i\omega x}$$

is $(2, 9)$-block sparse. ◊

For block sparse functions we can extend the definition of good hashing primes to good hashing integers, because no monic linear polynomial P can satisfy $P(x) \equiv P(y) \mod p$ for some $x \neq y$, $x, y \in \{1, \dots, B\}$ if $p > B$. Hence, we do not require to consider the field $\mathbb{Z}/p\mathbb{Z}$ anymore, which was necessary in order to be able to apply Theorem 3.5 in the proofs of Lemmas 3.6, 3.7, and 3.11.

Definition 3.37 (Definition 4.8 in [BZI19]) Let f be (n, B)-block sparse with support $S = \bigcup_{j=1}^{n} S_j$ generated by the polynomials $P_1, \dots, P_n$. An integer $u > B$ *hashes* a support set S_j *well* if the cardinality of the set of residues of elements of S_j modulo u is B, i.e.,

$$|\{\omega \bmod u : \omega \in S_j\}| = B \quad \forall j \in \{1, \dots, n\}.$$

Example 3.38 (Example 3.36 continued) We consider the same polynomials and support sets as in Example 3.36. Since

$$\begin{aligned} (\omega \bmod 16)_{\omega \in S_1} &= (9, 10, 11, 12, 13, 14, 15, 0, 1)^T \qquad \text{and} \\ (\omega \bmod 16)_{\omega \in S_2} &= (15, 0, 1, 2, 3, 4, 5, 6, 7)^T, \end{aligned}$$

we obtain that

$$|\{\omega \bmod 16 : \omega \in S_1\}| = 9 \quad \text{and} \quad |\{\omega \bmod 16 : \omega \in S_1\}| = 9.$$

Consequently, we find that 16 hashes both S_1 and S_2 well. Further, for each residue modulo 16 there are at most two elements in $S = S_1 \cup S_2$ that are congruent to it,

$$|\{\omega \equiv \nu \bmod 16 : \omega \in S\}| = \begin{cases} 0 & \nu = 8, \\ 1 & \nu \in \{2, \dots, 7, 9, \dots, 14\}, \\ 2 & \nu \in \{0, 1, 15\}. \end{cases}$$

◊

Remark 3.39 For an (n, B)-block sparse function f any integer $u > B$ hashes every support set S_j well, since S_j consists of B consecutive frequencies. Thus, for every residue $\nu \in \left\{-\left\lceil \frac{u}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u}{2} \right\rfloor\right\}$ modulo u the restriction of S to the frequencies congruent to ν is at most n-sparse,

$$|\{\omega \equiv \nu \bmod u : \omega \in S\}| \leq n \quad \forall \nu \in \left\{-\left\lceil \frac{u}{2} \right\rceil + 1, \dots, \left\lfloor \frac{u}{2} \right\rfloor\right\}.$$

We choose the hashing integer u to be the smallest power of 2 greater than the block length B, i.e.,

$$u \coloneqq 2^{\lfloor \log_2 B \rfloor + 1}.$$

Then $u = \mathcal{O}(B)$, which allows us to simplify the runtime estimates. Additionally, computing CDFTs of length stu, where s and t are small primes and u is a power of 2, is faster than if u were a prime of the same size.

Setting u as a power of 2 implies that we now have to slightly modify the primes t_l and s_k. Similarly to the choice of the primes in Definition 3.20 for Algorithm 5, we use the smallest L odd primes such that their product is greater than or equal to $\frac{N}{un}$,

$$\prod_{l=1}^{L-1} t_l < \frac{N}{un} \leq \prod_{l=1}^{L} t_l, \quad t_1 \coloneqq 3.$$

Let s_1 be the smallest prime greater than n and t_L. In this setting we can use the minimal K from Algorithm 4 for $\epsilon^{-1} = 2$, since u is already fixed,

$$K = 8n \left\lfloor \log_{s_1} \frac{N}{u} \right\rfloor + 1.$$

The remaining s_k can be set as the $K-1$ smallest primes greater than s_1. Then $u > B$, the integers $s_1, \ldots, s_K, t_1, \ldots, t_L, u$ are pairwise relatively prime, and

$$\prod_{l=1}^{L} t_l \geq \frac{N}{s_1 u},$$

so the CRT reconstruction method from Algorithm 1 can be applied. Since we chose $t_1 = 3$, the prime t_L in this case is at most the smallest prime greater than the t_L from Definition 3.20 for $d = 1$ and $M = 1$. ◊

Example 3.40 (Example 3.36 continued) For the $(2, B)$-block sparse function in Example 3.36 choosing the required integers as in Remark 3.39 yields

$$u \coloneqq 2^{\lfloor \log_2 B \rfloor + 1} = 16.$$

Since

$$3 \cdot 5 < \frac{N}{un} \leq 3 \cdot 5 \cdot 7,$$

we set $L \coloneqq 3$, $t_1 \coloneqq 3$, $t_2 \coloneqq 5$ and $t_3 \coloneqq 7$. Selecting s_1 as the smallest prime greater than both n and t_L, we obtain

$$s_1 \coloneqq 11 > \max\{2, 7\} = \max\{n, t_L\}.$$

Then $K = 8n \left\lfloor \log_{s_1} \frac{N}{u} \right\rfloor + 1 = 17$ and we find

$$\{s_1, \ldots, s_K\} = \{11, 13, 17, \ldots, p_{21} = 73\}.$$

If we chose the primes as in Definition 3.20 for $d = 1$ and $M = 1$, we would obtain $t_1 = 2$, $t_2 = 3$, $t_3 = 5$ and $t_4 = 7$, since

$$2 \cdot 3 \cdot 5 < \frac{N}{Bn} \leq 2 \cdot 3 \cdot 5 \cdot 7.$$

Then we would find that

$$s_1 \geq 11 > \max\{2, 7\} = \max\{n, t_L\}.$$

Hence, defining the primes as in Remark 3.39 results in one fewer prime t_l. As we would

also have that $K = 8n\left\lfloor \log_{s_1} \frac{N}{B} \right\rfloor + 1 = 17$, we would obtain

$$\{s_1, \dots, s_K\} = \{11, 13, 17, \dots, p_{21} = 73\} \qquad \text{and}$$
$$u = p_{22} = 79 > \max\{9, 73\} = \max\{B, s_K\}.$$

This already illustrates that by setting u to be the smallest power of 2 greater than B, we can reduce the runtime of the method. ◊

In Algorithm 6 we provide the explicit pseudocode for Algorithm 5 in the special case of (n, B)-block sparse functions, including the CRT reconstruction procedure from Algorithm 1 and the required steps from Algorithm 4, as the latter method is applied in a slightly different way here due to the fact that we only use one hashing integer u. We will investigate the performance of Algorithm 6 with respect to runtime and noisy input data in numerical experiments in Section 3.5, also comparing it to other methods.

As in Section 2.2.1, the function `extended_gcd` in line 12 denotes the extended Euclidean algorithm, which finds the greatest common divisor g of two integers a and b, as well as two integers v and w such that Bézout's identity

$$g = \gcd(a, b) = v \cdot a + w \cdot b,$$

see Theorem 2.7, is satisfied. By definition of u and $s_1, \dots, s_K$, we always have $g = 1$ in line 12.

Corollary 3.41 (Corollary 4.12 in [BZI19]) Let $f \in C_{2\pi}$ be (n, B)-block sparse with bandwidth N and noise threshold $\varepsilon > 0$. Let $\eta \in C_{2\pi}$ such that $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. Set u, $s_1, \dots, s_K$ and $t_1, \dots, t_L$ as in Remark 3.39. If $u > s_K$, the runtime of Algorithm 6 is given by

$$\mathcal{O}\left(\frac{B \log B \cdot n^2 \log^2 \frac{N}{2Bn} \log^2 \frac{N}{B} \log\left(2n \log \frac{N}{B}\right)}{\log^2(2n) \log\log \frac{N}{2Bn}} \right),$$

and otherwise, if $u < s_K$, by

$$\mathcal{O}\left(\frac{Bn^2 \cdot \log^2 \frac{N}{2Bn} \log^2 \frac{N}{B} \log^2\left(2n \log \frac{N}{B}\right)}{\log^2(2n) \log\log \frac{N}{2Bn}} \right).$$

In both cases the algorithm has a sampling complexity of

$$\mathcal{O}\left(\frac{Bn^2 \cdot \log^2 \frac{N}{2Bn} \log^2 \frac{N}{B} \log\left(2n \log \frac{N}{B}\right)}{\log^2(2n) \log\log \frac{N}{2Bn}} \right).$$

For Algorithm 6 the error bounds from Theorem 3.28 are still satisfied.

Proof. Since we set $t_1 = 3$ in Remark 3.39, the prime t_L for Algorithm 6 is at most the smallest prime greater than the t_L from Definition 3.20 for $d = 1$ and $M = 1$. Thus, the estimates

$$t_L = \mathcal{O}\left(\log \frac{N}{2un}\right) \quad \text{and} \quad s_K = \mathcal{O}\left(n \log_{2n} \frac{N}{u} \log\left(2n \log \frac{N}{u}\right)\right)$$

from the proof of Theorem 3.28 are still satisfied. Using additionally that $u = \mathcal{O}(B)$, the

Algorithm 6 Fourier Approximation for (n, B)-block Sparse Functions (Algorithm 2 in [BZI19])

Input: Function $f + \eta$, $n, B, N \in \mathbb{N}$ with $B < N$, where $f \in C_{2\pi}$ has bandwidth N and is (n, B)-block sparse.

Output: $R, \mathbf{x}_R$, where R contains the nB frequencies ω with greatest magnitude coefficient estimates x_ω.

Set $u = 2^\alpha$ with $\alpha = \lfloor \log_2 B \rfloor + 1$ and find the L smallest odd primes $t_1, \ldots, t_L$ such that $\prod_{l=1}^{L-1} t_l < \frac{N}{un} \leq \prod_{l=1}^{L} t_l$. Set $t_0 = 1$.

Let $s_1 > \max\{n, t_L\}$ be the smallest prime, $K = 2n \left\lfloor \log_{s_1} \frac{N}{u} \right\rfloor + 1$ and $s_2, \ldots, s_K$ be the smallest primes greater than s_1.

Initialize $R = \emptyset$, $\mathbf{x}_R = \mathbf{0}_N$.

for k from 1 to K **do** ▷ computation of $\mathcal{G}_{L,K}$ and $\mathcal{E}_K$

  for l from 0 to L **do**

    $\mathbf{a}^{s_k t_l u} \leftarrow \left(f\left(\frac{2\pi j}{s_k t_l u}\right) \right)_{j=0}^{s_k t_l u - 1}$

    $\widehat{\mathbf{a}^{s_k t_l u}} \leftarrow \mathbf{CDFT}[\mathbf{a}^{s_k t_l u}]$

  end for

end for

Identification of the Energetic Frequencies

for ν from $-\frac{u}{2} + 1$ to $\frac{u}{2}$ **do**

  for k from 1 to K **do**

    $(1, v, w) \leftarrow \texttt{extended\_gcd}(s_k, u)$ ▷ i.e., $1 = v \cdot s_k + w \cdot u$

    for h from $-\left\lceil \frac{s_k}{2} \right\rceil + 1$ to $\left\lfloor \frac{s_k}{2} \right\rfloor$ **do**

      $r_0^{k,h} \leftarrow ((h - \nu) w \mod s_k) \cdot u + \nu$ ▷ residue modulo $s_k u$

      for l from 1 to L **do**

        $b_{\min} \leftarrow \underset{b \in \left\{ -\left\lceil \frac{t_l}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{t_l}{2} \right\rfloor \right\}}{\operatorname{argmin}} \left| \widehat{a^{s_k u}}_{r_0^{k,h}} - \widehat{a^{s_k t_l u}}_{r_0^{k,h} + b \cdot s_k u} \right|$

        $r_l^{k,h} \leftarrow \left(r_0^{k,h} + b_{\min} \cdot s_k u \right) \mod t_l$

      end for

      Recover $\omega^{k,h}$ by $\omega^{k,h} \equiv r_0^{k,h} \mod s_k u$, $\omega^{k,h} \equiv r_l^{k,h} \mod t_l$, $l = 1, \ldots, L$.

    end for

  end for

Fourier Coefficient Estimation

  for each $\omega^{k,h} \equiv \nu \mod u$ value reconstructed more than $\frac{K}{2}$ times **do**

    $\operatorname{Re}(x_\omega) \leftarrow \text{median} \left\{ \operatorname{Re}\left(\widehat{a^{s_k t_L u}}_{\omega \bmod s_k t_L u} \right) : k \in \{1, \ldots, K\} \right\}$

    $\operatorname{Im}(x_\omega) \leftarrow \text{median} \left\{ \operatorname{Im}\left(\widehat{a^{s_k t_L u}}_{\omega \bmod s_k t_L u} \right) : k \in \{1, \ldots, K\} \right\}$

  end for

  Sort the coefficients by magnitude s.t. $|x_{\omega_1}| \geq |x_{\omega_2}| \geq \cdots$.

  $R^{(1,\nu)} = \{\omega_1, \omega_2, \ldots, \omega_{2n}\}$

end for

Sort the coefficients in $\bigcup_{\nu=0}^{u-1} R^{(1,\nu)}$ by magnitude s.t. $|x_{\omega_1}| \geq |x_{\omega_2}| \geq \cdots$.

Output: : $R = \{\omega_1, \omega_2, \ldots, \omega_{nB}\}, \mathbf{x}_R$.

runtime of Algorithm 6 for $u > s_K$ is

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u \log\left(s_k t_l u\right)\right) \\
&= \mathcal{O}\left(u \log u \cdot \frac{s_K^2}{\log s_K} \cdot \frac{t_L^2}{\log t_L}\right) \\
&= \mathcal{O}\left(\frac{B \log B \cdot n^2 \log^2 \frac{N}{2Bn} \log^2 \frac{N}{B} \log\left(2n \log \frac{N}{B}\right)}{\log^2(2n) \log\log \frac{N}{2Bn}}\right).
\end{aligned}$$

If $u < s_K$, we obtain a runtime of

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u \log\left(s_k t_l u\right)\right) \\
&= \mathcal{O}\left(u \cdot s_K^2 \cdot \frac{t_L^2}{\log t_L}\right) \\
&= \mathcal{O}\left(\frac{Bn^2 \cdot \log^2 \frac{N}{2Bn} \log^2 \frac{N}{B} \log^2\left(2n \log \frac{N}{B}\right)}{\log^2(2n) \log\log \frac{N}{2Bn}}\right).
\end{aligned}$$

In both cases the number of required samples of $f + \eta$ is

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u\right) \\
&= \mathcal{O}\left(u \cdot \frac{s_K^2}{\log s_K} \cdot \frac{t_L^2}{\log t_L}\right) \\
&= \mathcal{O}\left(\frac{Bn^2 \cdot \log^2 \frac{N}{2Bn} \log^2 \frac{N}{B} \log\left(2n \log \frac{N}{B}\right)}{\log^2(2n) \log\log \frac{N}{2Bn}}\right).
\end{aligned}$$

The error bound from Theorem 3.28 is still satisfied, as it is unaffected by the slightly altered choice of the primes. □

If not only f, but also $f + n$ is bandlimited with bandwidth N, and f is (n, B)-block sparse, the error bound in Theorem 3.28 can be simplified, analogously to Corollary 3.29.

Corollary 3.42 (Corollary 4.13 in [BZI19]) Let $f \in C_{2\pi}$ be (n, B)-block sparse with bandwidth N and noise threshold $\varepsilon > 0$. Let $\eta \in C_{2\pi}$ such that $f + \eta$ also has bandwidth N and $\eta \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. Choosing u and the s_k and t_l as in Remark 3.39, the output $(R, \mathbf{x}_R)$ of Algorithm 6 satisfies

$$\|\mathbf{c}(N) - \mathbf{x}_R\|_1 \leq 4 \left\|\mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N)\right\|_1 + 2Bn\varepsilon.$$

Proof. As we do not have to take medians over the estimates obtained for the different

hashing primes, we can consider the following equality,

$$\begin{aligned}
\|\mathbf{c}(N) - \mathbf{x}_R\|_1 &= \sum_{\omega=-\lceil \frac{N}{2} \rceil+1}^{\lfloor \frac{N}{2} \rfloor} |c_\omega - x_\omega| \\
&= \sum_{\nu=-\frac{u}{2}+1}^{\frac{u}{2}} \sum_{\substack{\omega=-\lceil \frac{N}{2} \rceil+1 \\ \omega \equiv \nu \bmod u}}^{\lfloor \frac{N}{2} \rfloor} |c_\omega - x_\omega| \\
&= \sum_{\nu=-\frac{u}{2}+1}^{\frac{u}{2}} \Bigg(\sum_{\omega \in R^{(1,\nu)}} |c_\omega - x_\omega| + \sum_{\substack{\omega \notin R^{(1,\nu)} \\ \omega \equiv \nu \bmod u}} |c_\omega| \Bigg) \\
&= \sum_{\nu=-\frac{u}{2}+1}^{\frac{u}{2}} \Bigg(\sum_{\omega \in R^{(1,\nu)}} |c_\omega - x_\omega| + \sum_{\substack{\omega \notin R_n^{(1,\nu),\,\mathrm{opt}} \\ \omega \equiv \nu \bmod u}} |c_\omega| \\
&\quad + \sum_{\omega \in R_n^{(1,\nu),\,\mathrm{opt}} \setminus R^{(1,\nu)}} |c_\omega| - \sum_{\omega \in R^{(1,\nu)} \setminus R_n^{(1,\nu),\,\mathrm{opt}}} |c_\omega| \Bigg). \qquad (3.14)
\end{aligned}$$

By (3.8) in the proof of Lemma 3.26, all of the $2n$ elements of $R^{(1,\nu)}$ have to satisfy that $|c_\omega - x_\omega| \leq \sqrt{2}\delta^{(1,\nu)}$, since each frequency in $R^{(1,\nu)}$ was reconstructed more than $\frac{K}{2}$ times. Furthermore, note that $\left(\mathbf{c}_n^{\mathrm{opt}}\right)_\omega = 0$ for all $\omega \equiv \nu \bmod u$ with $\omega \notin R^{(1,\nu),\,\mathrm{opt}}$ and $\left(\mathbf{c}(N,u,\nu)\right)_\omega = \left(\mathbf{c}_n^{\mathrm{opt}}\right)_\omega$ for all $\omega \equiv \nu \bmod u$ with $\omega \in R^{(1,\nu),\,\mathrm{opt}}$. It also follows from the proof of Lemma 3.26 that $|c_\omega| \leq \varepsilon + 2\sqrt{2}\delta^{(1,\nu)}$ for all at most n frequencies ω contained in $R_n^{(1,\nu),\,\mathrm{opt}} \setminus R^{(1,\nu)}$. Recall that by definition of $\delta^{(1,\nu)}$, we have that

$$\delta^{(1,\nu)} \coloneqq \frac{1}{2n} \left\| \mathbf{c}(N,u,\nu) - \mathbf{c}_{2n}^{\mathrm{opt}}(N,u,\nu) \right\|_1 + \underbrace{\|\mathbf{c}(N,\mathbb{Z},u,\nu) - \mathbf{c}(u,\nu)\|_1}_{=0},$$

as we assume $f + \eta$ to be bandlimited. Combining these considerations, (3.14) yields

$$\begin{aligned}
&\|\mathbf{c}(N) - \mathbf{x}_R\|_1 \\
&\leq \sum_{\nu=-\frac{u}{2}+1}^{\frac{u}{2}} \left(2n\sqrt{2}\delta^{(1,\nu)} + \left\| \mathbf{c}(N,u,\nu) - \mathbf{c}_n^{\mathrm{opt}}(N,u,\nu) \right\|_1 + n\left(\varepsilon + 2\sqrt{2}\delta^{(1,\nu)}\right) \right) \\
&= \sum_{\nu=-\frac{u}{2}+1}^{\frac{u}{2}} \left(\left\| \mathbf{c}(N,u,\nu) - \mathbf{c}_n^{\mathrm{opt}}(N,u,\nu) \right\|_1 + \frac{4\sqrt{2}n}{2n} \left\| \mathbf{c}(N,u,\nu) - \mathbf{c}_{2n}^{\mathrm{opt}}(N,u,\nu) \right\|_1 \right) + nu\varepsilon.
\end{aligned}$$

Analogously to the proof of Corollary 3.29 we find the following estimate,

$$
\begin{aligned}
&\sum_{\nu=-\frac{u}{2}+1}^{\frac{u}{2}} \left(\left\| \mathbf{c}(N,u,\nu) - \mathbf{c}_n^{\mathrm{opt}}(N,u,\nu) \right\|_1 + 2\sqrt{2} \left\| \mathbf{c}(N,u,\nu) - \mathbf{c}_{2n}^{\mathrm{opt}}(N,u,\nu) \right\|_1 \right) \\
&= \sum_{\nu=-\frac{u}{2}+1}^{\frac{u}{2}} \sum_{\substack{\omega=-\lceil \frac{N}{2} \rceil+1 \\ \omega \equiv \nu \bmod u}}^{\lfloor \frac{N}{2} \rfloor} \Bigg(\underbrace{\left| c_\omega - \left(\mathbf{c}_n^{\mathrm{opt}}(N,u,\nu) \right)_\omega \right|}_{= \begin{cases} |c_\omega| & \text{if } \omega \notin R_n^{(1,\nu),\mathrm{opt}}, \\ 0 & \text{otherwise.} \end{cases}} + 2\sqrt{2} \underbrace{\left| c_\omega - \left(\mathbf{c}_{2n}^{\mathrm{opt}}(N,u,\nu) \right)_\omega \right|}_{= \begin{cases} |c_\omega| & \text{if } \omega \notin R_{2n}^{(1,\nu),\mathrm{opt}}, \\ 0 & \text{otherwise.} \end{cases}} \Bigg) \\
&= \sum_{\substack{\omega \in -\lceil \frac{N}{2} \rceil+1 \\ \omega \notin \bigcup_{\nu=-\frac{u}{2}+1}^{\frac{u}{2}} R_{dn}^{(1,\nu),\mathrm{opt}}}}^{\lfloor \frac{N}{2} \rfloor} |c_\omega| + 2\sqrt{2} \cdot \sum_{\substack{\omega \in -\lceil \frac{N}{2} \rceil+1 \\ \omega \notin \bigcup_{\nu=-\frac{u}{2}+1}^{\frac{u}{2}} R_{2dn}^{(1,\nu),\mathrm{opt}}}}^{\lfloor \frac{N}{2} \rfloor} |c_\omega| \\
&\leq (1+2\sqrt{2}) \sum_{\substack{\omega \in -\lceil \frac{N}{2} \rceil+1 \\ \omega \notin \bigcup_{\nu=-\frac{u}{2}+1}^{\frac{u}{2}} R_{dn}^{(1,\nu),\mathrm{opt}}}}^{\lfloor \frac{N}{2} \rfloor} |c_\omega| \leq (1+2\sqrt{2}) \sum_{\omega=-\lceil \frac{N}{2} \rceil+1}^{\lfloor \frac{N}{2} \rfloor} \underbrace{\left| c_\omega - \left(\mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right)_\omega \right|}_{= \begin{cases} |c_\omega| & \text{if } \omega \notin R_{Bn}^{\mathrm{opt}}, \\ 0 & \text{otherwise.} \end{cases}} \\
&= (1+2\sqrt{2}) \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right\|_1 .
\end{aligned}
$$

Consequently, we find that

$$
\begin{aligned}
\left\| \mathbf{c}(N) - \mathbf{x}_R \right\|_1 &\leq (1+2\sqrt{2}) \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right\|_1 + nu\varepsilon \\
&\leq 4 \cdot \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N) \right\|_1 + 2Bn\varepsilon.
\end{aligned}
$$

□

3.5 Numerical Results for Algorithm 6

In this section we will evaluate the performance of Algorithm 5 with respect to runtime and robustness to noisy data. As the most interesting and practically useful example for polynomially structured sparse functions are block sparse functions, we restrict ourselves to investigating the simplified version of Algorithm 5, namely Algorithm 6. We consider two variants of this method: on the one hand the deterministic algorithm, which we developed for (n, B)-block sparse functions in Section 3.4.2, and on the other hand a randomized implementation of Algorithm 6 which only utilizes a small random subset of the K primes $s_1, \ldots, s_K$ used by Algorithm 6. Instead of K primes we chose the first

$$\widetilde{K} = \frac{3}{2} \log(2n) + R$$

primes $s_1, \ldots, s_{\widetilde{K}} > \max\left\{n, \tilde{t}_{\tilde{L}}\right\}$, where R is an odd natural number controlling the probability of correct recovery. In the numerical experiments below we always set R such that the probability of correct recovery is at least 0.9. The probability of correct recovery increases with increasing R.

Both methods have been implemented in C++ and the code is publicly available in [BZI17b].[1] We also compare these implementations' runtime and robustness characteristics to those of the deterministic Algorithm 4 (Algorithm 3 in [Iwe13]), which we briefly sketched in Section 3.2, using an optimized implementation in C++ based on [SI13]. The code is also publicly available in [BZI17b, SI17]. Additionally, we compare our methods to FFTW 3.3.4 and sFFT 2.0. FFTW 3.3.4 is a highly optimized and publicly available implementation of the traditional FFT algorithm with runtime $\mathcal{O}(N \log N)$ for input vectors of length N, as we have seen in Section 1.1.1. See [FJ17] for more information on the implementation. All the FFTW results below were obtained using FFTW 3.3.4 with its FFTW_MEASURE plan. sFFT 2.0 is a randomized sparse Fourier transform that is robust with respect to noise; see [HIKP12c, HIKP12b] for more detailed information and an implementation. It has a theoretical runtime of $\mathcal{O}\left(\log N \sqrt{NB \log N}\right)$ for a B-sparse input function with bandwidth N.

Note that both the deterministic and the randomized version of Algorithm 6 are designed to approximate functions that are (n, B)-block sparse. Thus, both methods require upper bounds on the number of blocks n and the block length B of the functions they aim to recover as parameters. In contrast, both Algorithm 4 and sFFT 2.0 only need an upper bound on the effective sparsity s of the Fourier coefficients of the function. For an (n, B)-block sparse function the effective sparsity s is nB. Hence, for the remainder of this section, s is always set such that $s = nB$ for Algorithm 4 and sFFT 2.0.

For the numerical experiments investigating the runtime, each test function f was formed by choosing an (n, B)-block sparse set S of frequencies uniformly at random from $\left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$. Each frequency in the set S was then assigned a Fourier coefficient c_ω with magnitude 1 and a phase chosen uniformly at random from $[0, 2\pi)$. The Fourier coefficients of the remaining frequencies of f were all set to zero. The following figures were obtained by computing the average over 100 runs on 100 different test functions as described above. Depending on the choice of the number of blocks n and their length B, the parameters in the two randomized algorithms, i.e., the randomized variant of Algorithm 6 and sFFT 2.0, were chosen such that the probability of correctly recovering an (n, B)-block sparse function was at least 0.9 for each run. For the randomized version of Algorithm 6 with $n = 2$ it suffices to set $R = 1$ if $B \leq 2^{10}$ and $R = 3$ if $B \leq 2^{11}$ to achieve this probability. For $n = 3$ blocks we can set $R = 1$ if $B \leq 2^4$ and $R = 3$ if $B \leq 2^{11}$.

In Figure 3.1 we plot the average runtimes of Algorithm 6 and its randomized variant, Algorithm 4, sFFT 2.0 and FFTW for $n = 2$ blocks, a bandwidth of $N = 2^{26}$ and block lengths B varying between 2^2 and 2^{11}. In Figure 3.2 we perform the same numerical experiments for a block sparse function with $n = 3$ blocks.

As expected, due to the independence of its runtime of the actual sparsity of the function, the runtime of FFTW is constant for increasing block lengths. The theoretical runtimes of all sparse Fourier transform algorithms other than Algorithm 4 are subquadratic in B. Indeed, the plots of their average runtimes for varying B have similar slopes. Figures 3.1 and 3.2 also demonstrate that allowing a small probability of incorrect recovery for the randomized algorithms sFFT 2.0 and the randomized variant of Algorithm 6 lets these methods outperform the deterministic algorithms with respect to runtime for all considered block lengths B. Among the deterministic algorithms, Algorithm 6 is always faster than Algorithm 4, and only becomes slower than FFTW when the block length B is greater than 256. The runtimes of both the randomized variant of

[1] There is also an implementation of the general Algorithm 5 in MATLAB 2016b publicly available in [BZI17a].

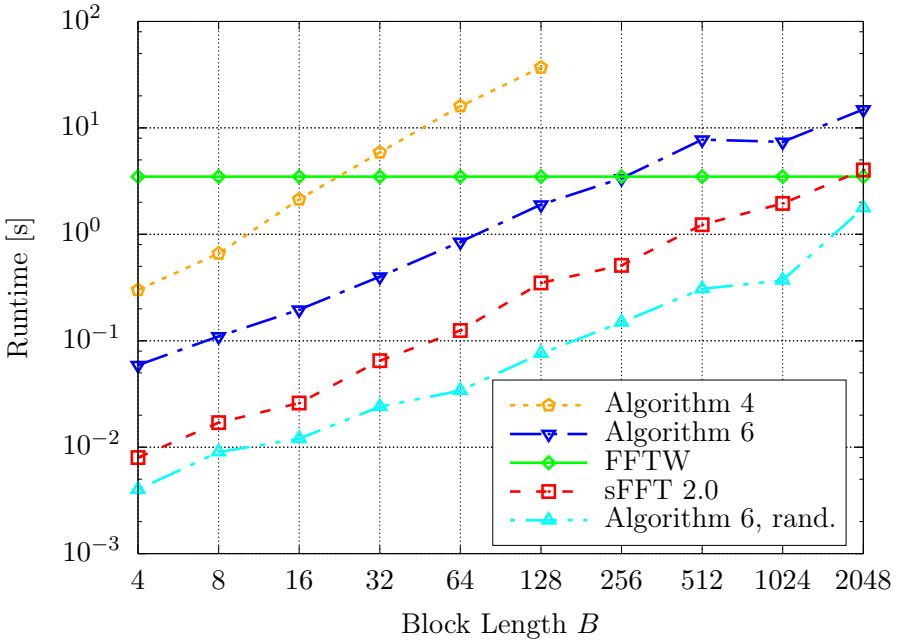

Figure 3.1: Average runtimes of Algorithm 4, Algorithm 6 (deterministic), FFTW, sFFT 2.0 and Algorithm 6 (randomized) for 100 random input functions with $n = 2$ blocks of length B and bandwidth $N = 2^{26}$

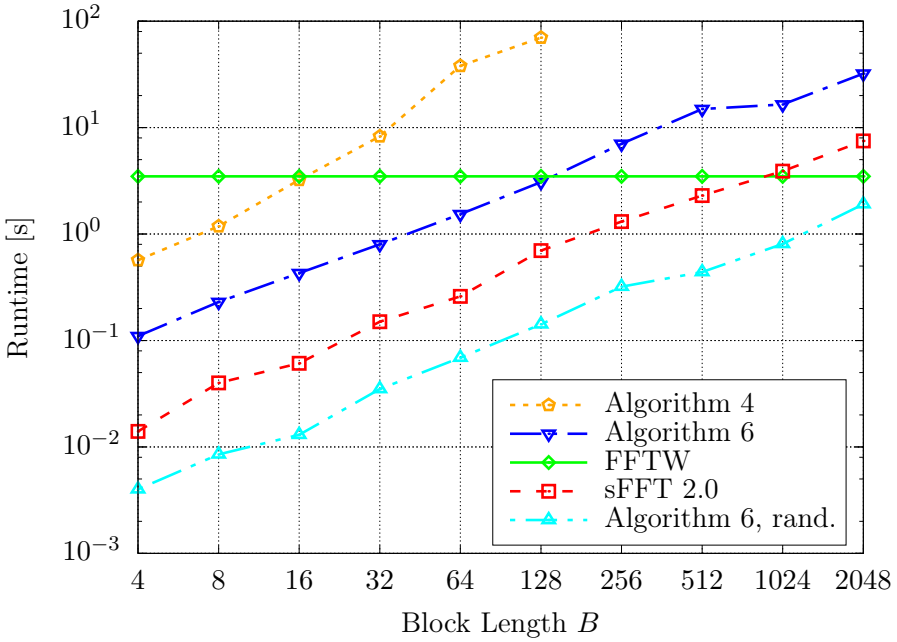

Figure 3.2: Average runtimes of Algorithm 4, Algorithm 6 (deterministic), FFTW, sFFT 2.0 and Algorithm 6 (randomized) for 100 random input functions with $n = 3$ blocks of length B and bandwidth $N = 2^{26}$

Algorithm 6 and sFFT 2.0 are still comparable with the one of FFTW when the block length B is as large as 2,048 for $n = 2$ and 1,024 for $n = 3$. Compared to sFFT 2.0, the randomized variant of Algorithm 6 has a better runtime performance for the considered

parameters. It is also the only algorithm that is still faster than FFTW when $B = 2{,}048$ for both $n = 2$ and $n = 3$.

In Figure 3.3 we fix the bandwidth and block length to be $N = 2^{26}$ and $B = 32$ and vary the number of blocks n from 1 to 10.

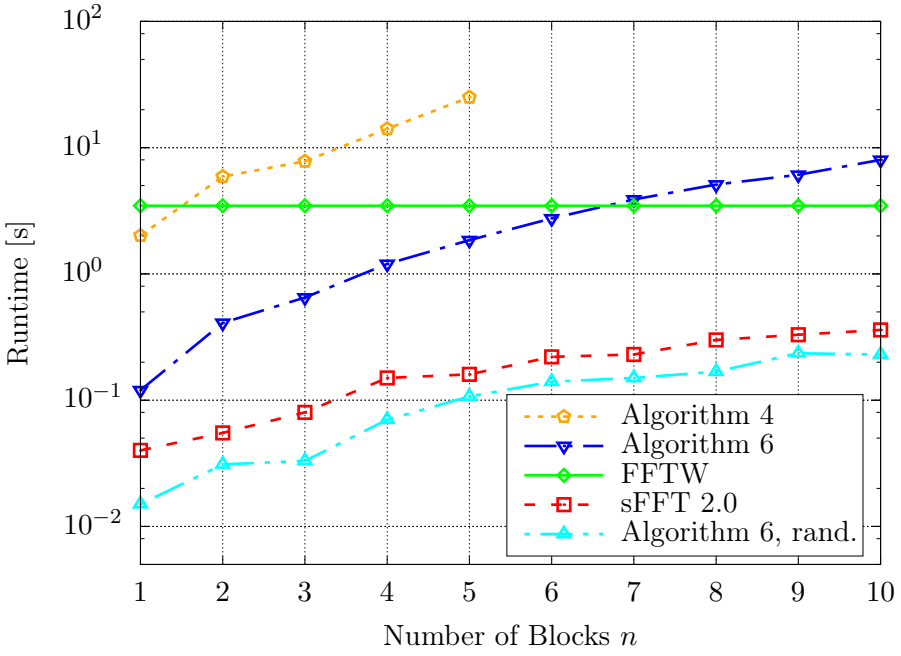

Figure 3.3: Average runtimes of Algorithm 4, Algorithm 6 (deterministic), FFTW, sFFT 2.0 and Algorithm 6 (randomized) for 100 random input functions with n blocks of length $B = 32$ and bandwidth $N = 2^{26}$

Then we can see that the deterministic sparse Fourier methods, Algorithms 4 and 6, both have runtimes that increase more rapidly in n than those of their randomized competitors. Among the three deterministic methods, Algorithm 6 has the best performance when the number of blocks is at most 6. Similar to the previous experiments, FFTW becomes the fastest deterministic algorithm when the sparsity $s = Bn$ is at least 224, as its runtime does not depend on the sparsity of the function. The two randomized algorithms are both faster than FFTW by an order of magnitude even if the number of blocks is 10. As in the experiments where we varied the block length B, the randomized version of Algorithm 6 is always faster than sFFT 2.0 for the examined value of N.

In Figure 3.4 we set the number of blocks and the block length to be $n = 2$ and $B = 64$ and examine the performance of the different algorithms for varying bandwidths N.

It can be seen that FFTW is the fastest deterministic algorithm for small bandwidth values. However, the runtime of FFTW becomes slower than the one of Algorithm 6 when the bandwidth N is greater than 2^{24}. Algorithm 4 is the slowest deterministic algorithm for this fixed sparsity of 128 for all considered bandwidths N. Comparing the randomized sparse Fourier methods, the randomized variant of Algorithm 6 always performs better than sFFT 2.0 when the bandwidth N is greater than 2^{18}.

In order to test the robustness of the methods with respect to noise, we add Gaussian noise to each of the samples of the (n, B)-block sparse function $f \in C_{2\pi}$ utilized in the algorithms, and then measure the average approximation errors of the reconstructed Fourier coefficients. As parameters we choose a bandwidth of $N = 2^{22}$ and $n = 3$ blocks

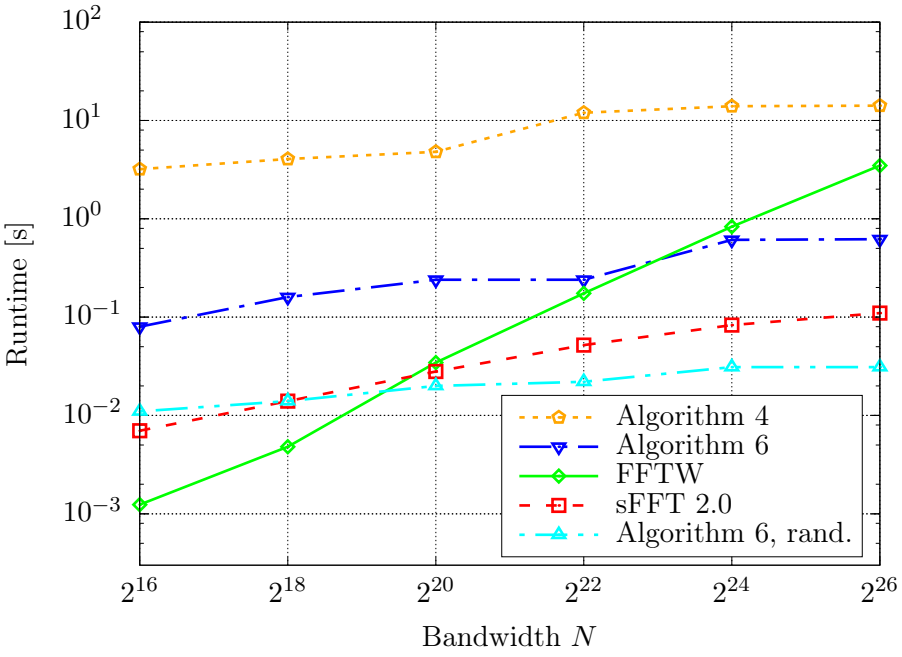

Figure 3.4: Average runtimes of Algorithm 4, Algorithm 6 (deterministic), FFTW, sFFT 2.0 and Algorithm 6 (randomized) for 100 random input functions with $n = 2$ blocks of length $B = 64$ and bandwidth N

of length $B = 2^4$. More specifically, for exact data, each method considered herein utilizes one or several vectors of equidistant samples of f for suitable lengths $M \leq N$. For the robustness experiments we provide each algorithm with noisy samples of the form

$$\left(f\left(\frac{2\pi j}{M}\right) + \eta_j \right)_{j=0}^{M-1},$$

where each $\eta_j \in \mathbb{C}$ is a complex Gaussian random variable with mean 0. The η_j are then rescaled such that the total additive noise $\boldsymbol{\eta} = (\eta_j)_{j=0}^{M-1}$ achieves *signal-to-noise ratios (SNRs)* between 0 and 60, where the SNR is defined as

$$\mathrm{SNR} := 20 \log \left(\frac{\left\| \mathbf{a}^M \right\|_2}{\|\boldsymbol{\eta}\|_2} \right).$$

The resulting reconstruction errors are depicted in Figure 3.5.

Recall that the two randomized algorithms compared herein, sFFT 2.0 and the randomized variant of Algorithm 6, are both tuned to guarantee exact recovery of block sparse functions with probability at least 0.9 in all experiments. For the numerical experiments investigating the robustness with respect to noise, this ensures that the correct support set S is found for at least 90 of the 100 test functions used to generate the points plotted in Figure 3.5. All deterministic methods always find the correct support S for all considered noise levels after sorting their output Fourier coefficient estimates by magnitude. Figure 3.5 depicts the average ℓ^1-error between the true Fourier coefficients for frequencies in the correct frequency support S of each test signal and the corresponding coefficient estimate x_ω, averaged over the at least 90 runs for which the respective sparse

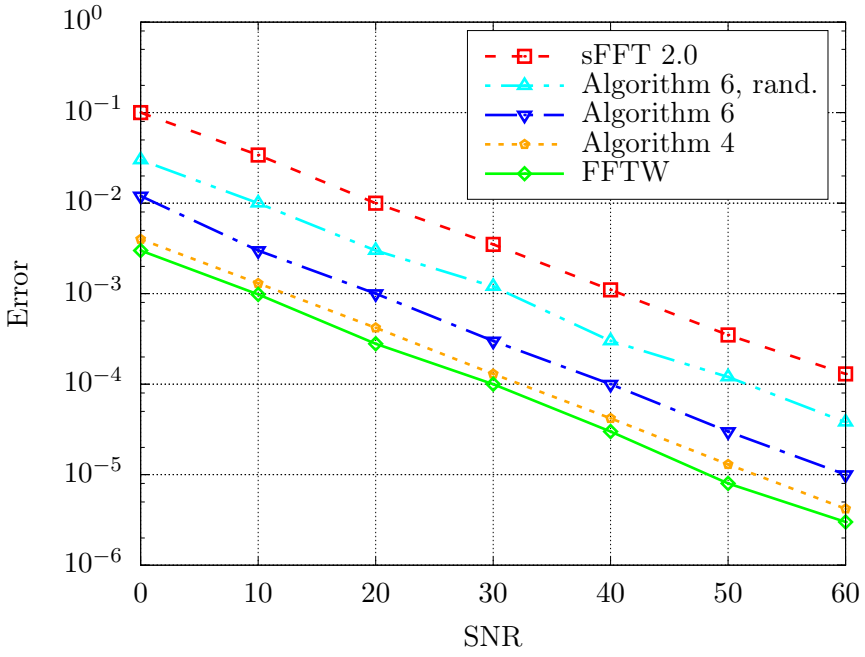

Figure 3.5: Average reconstruction errors of Algorithm 4, Algorithm 6 (deterministic), FFTW, sFFT 2.0 and Algorithm 6 (randomized) for 100 random input functions with $n = 3$ blocks of length $B = 2^4$ and bandwidth $N = 2^{22}$

Fourier transform correctly identified S. More specifically, it plots

$$\frac{1}{Bn} \sum_{\omega \in S} \left| c_\omega - \bar{x}_\omega \right|,$$

where c_ω are the true Fourier coefficients for the frequencies $\omega \in S$, and $\bar{x}_\omega$ are their recovered approximations, averaged over the at least 90 test signals where the respective method correctly identified S.

Looking at Figure 3.5, one can see that all algorithms considered in our experiments are robust with respect to noise. Overall, the deterministic methods Algorithm 6, Algorithm 4 and FFTW are more robust than the randomized methods sFFT 2.0 and the randomized variant of Algorithm 6. As expected, FFTW is the most robust algorithm in this experiment, followed closely by Algorithm 4. Still, the performance of Algorithm 6 for noisy data is comparable to their performance, and Algorithm 6 is also more stable than the randomized methods. For the randomized algorithms, the randomized variant of Algorithm 6 is more robust than sFFT 2.0.

In this chapter we introduced the first deterministic algorithm for reconstructing block sparse functions from samples. The numerical experiments presented above show that Algorithm 6 is faster than all existing general sparse FFT methods, while also being very robust with respect to noisy input data. Furthermore, the investigation of the runtime of a randomized version of Algorithm 6 showed that it is also faster and more robust than the fastest existing randomized general sparse FFT algorithms.

Part III

Sparse Fast Cosine Transform

4 Discrete Cosine Transform

As mentioned in the introduction to this thesis, the discrete cosine and sine transforms (DCT and DST) of types I to VIII can be obtained by applying combinations of Neumann and Dirichlet boundary conditions to the discretized solution of the homogeneous harmonic oscillator equation. Furthermore, the DCT-II approximates the statistically optimal Karhunen-Loéve transform, which decorrelates stationary Markov-1 signals. Since in many practical applications the occurring signals can be approximated by stationary Markov-1 signals, the DCT-II is an extremely useful tool for solving the arising problems.

Similarly as for the DFT, only the development of fast algorithms with a runtime of $\mathcal{O}(N \log N)$ instead of $\mathcal{O}(N^2)$ allowed the widespread practical use of different types of the DCT and the DST. Some of these methods employ existing fast DFT algorithms, see, e.g., [PPST19], Section 6.3.1, whereas others only use real arithmetic. See, e.g, [BYR06], Section 4.4 and [PPST19], Section 6.3.2, for an overview of such real methods.

There is a lower limit for the runtime of fast DCT and DST algorithms for arbitrary input vectors of length N, though, since the order $\mathcal{O}(N \log N)$ of the runtime, like the one of the DFT, can be proven to be optimal. Any further speeding up of the methods therefore requires additional a priori knowledge about the vector we aim to recover; in practice this is usually information about its sparsity. After providing the required theoretical background about the DCT in this chapter, we will investigate sparse DCT-II problems in more detail in Chapters 5 and 6.

4.1 Discrete Cosine Transform

Let us begin by formally defining the first four types of the discrete cosine transform as matrix-vector multiplications for real vectors. The DCTs of types V to VIII are not of interest for this thesis. The following definitions are based on [PPST19], Section 3.5.

Definition 4.1 (Discrete Cosine Transform (DCT)) Let $N \in \mathbb{N}$, $\mathbf{x} = (x_k)_{k=0}^{N-1} \in \mathbb{R}^N$ and $\tilde{\mathbf{x}} \in \mathbb{R}^{N+1}$. The *cosine matrices of types I-IV* are defined as

$$\mathbf{C}_{N+1}^{\mathrm{I}} \coloneqq \sqrt{\frac{2}{N}} \left(\varepsilon_N(k)\varepsilon_N(l) \cos\left(\frac{kl\pi}{N}\right) \right)_{k,l=0}^{N} \in \mathbb{R}^{(N+1)\times(N+1)},$$

$$\mathbf{C}_{N}^{\mathrm{II}} \coloneqq \sqrt{\frac{2}{N}} \left(\varepsilon_N(k) \cos\left(\frac{k(2l+1)\pi}{2N}\right) \right)_{k,l=0}^{N-1} \in \mathbb{R}^{N\times N},$$

$$\mathbf{C}_{N}^{\mathrm{III}} \coloneqq \sqrt{\frac{2}{N}} \left(\varepsilon_N(l) \cos\left(\frac{(2k+1)l\pi}{2N}\right) \right)_{k,l=0}^{N-1} \in \mathbb{R}^{N\times N},$$

$$\mathbf{C}_{N}^{\mathrm{IV}} \coloneqq \sqrt{\frac{2}{N}} \left(\cos\left(\frac{(2k+1)(2l+1)\pi}{4N}\right) \right)_{k,l=0}^{N-1} \in \mathbb{R}^{N\times N},$$

where

$$\varepsilon_N(k) \coloneqq \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k \equiv 0 \mod N, \\ 1 & \text{otherwise.} \end{cases}$$

Then the *discrete cosine transforms of types I-IV* of the vectors $\tilde{\mathbf{x}}$ and $\mathbf{x}$, respectively, are given by

$$\begin{aligned}\tilde{\mathbf{x}}^{\widehat{\mathrm{I}}} &:= \mathbf{C}_{N+1}^{\mathrm{I}}\tilde{\mathbf{x}},\\ \mathbf{x}^{\widehat{\mathrm{II}}} &:= \mathbf{C}_{N}^{\mathrm{II}}\mathbf{x},\\ \mathbf{x}^{\widehat{\mathrm{III}}} &:= \mathbf{C}_{N}^{\mathrm{III}}\mathbf{x},\\ \mathbf{x}^{\widehat{\mathrm{IV}}} &:= \mathbf{C}_{N}^{\mathrm{IV}}\mathbf{x}.\end{aligned}$$

The cosine matrices of types I-IV can in fact be shown to be orthogonal, analogously to the almost unitary Fourier matrix $\mathbf{F}_N$, see Definition 1.1.

Theorem 4.2 *Let $N \in \mathbb{N}$ and $\mathbf{x} \in \mathbb{R}^N$. Then the cosine matrices of types I-IV are orthogonal with*

(i) $\mathbf{C}_{N+1}^{\mathrm{I}}{}^{-1} = \mathbf{C}_{N+1}^{\mathrm{I}}{}^{T} = \mathbf{C}_{N+1}^{\mathrm{I}}$,

(ii) $\mathbf{C}_{N}^{\mathrm{II}}{}^{-1} = \mathbf{C}_{N}^{\mathrm{II}}{}^{T} = \mathbf{C}_{N}^{\mathrm{III}}$,

(iii) $\mathbf{C}_{N}^{\mathrm{IV}}{}^{-1} = \mathbf{C}_{N}^{\mathrm{IV}}{}^{T} = \mathbf{C}_{N}^{\mathrm{IV}}$.

For a proof see, e.g., [PPST19], Section 3.5, Lemmas 3.46 to 3.48. Apart from the discrete cosine transforms, there exists another, closely related, set of linear trigonometric transforms, the discrete sine transforms of types I to VIII. For this thesis only the sine matrix of type IV, which was introduced in [Jai79], is of interest. Analogously to the DCT-IV, the sine matrix of type IV defines the discrete sine transform of type IV.

Definition 4.3 (Sine Matrix of Type IV) Let $N \in \mathbb{N}$ and $\mathbf{x} = (x_k)_{k=0}^{N-1} \in \mathbb{R}^N$. The *sine matrix of type IV* is defined as

$$\mathbf{S}_N^{\mathrm{IV}} := \sqrt{\frac{2}{N}}\left(\sin\left(\frac{(2k+1)(2l+1)\pi}{4N}\right)\right)_{k,\,l=0}^{N-1} \in \mathbb{R}^{N\times N}.$$

The following theorem proves the orthogonality of the sine matrix of type IV and provides us with its connection to the cosine matrix of type IV.

Theorem 4.4 *Let $N \in \mathbb{N}$. We define the* counter identity matrix of size $N \times N$ *as*

$$\mathbf{J}_N := (\delta_{k,\,N-1-l})_{k,\,l=0}^{N-1} = \begin{pmatrix} 0 & \dots & 0 & 1\\ 0 & & 1 & 0\\ \vdots & \iddots & & \vdots\\ 1 & \dots & 0 & 0\end{pmatrix} \in \mathbb{R}^{N\times N},$$

and set

$$\mathbf{D}_N := \operatorname{diag}\left(\left((-1)^k\right)_{k=0}^{N-1}\right) \in \mathbb{R}^{N\times N}.$$

Then the following statements are true.

(i) The sine matrix of type IV is orthogonal with

$$\mathbf{S}_N^{\mathrm{IV}}{}^{-1} = \mathbf{S}_N^{\mathrm{IV}}{}^{T} = \mathbf{S}_N^{\mathrm{IV}}.$$

(ii) The cosine and sine matrices of type IV satisfy

$$\mathbf{S}_N^{\mathrm{IV}} = \mathbf{D}_N \mathbf{C}_N^{\mathrm{IV}} \mathbf{J}_N \qquad \text{and} \qquad \mathbf{S}_N^{\mathrm{IV}} = \mathbf{J}_N \mathbf{C}_N^{\mathrm{IV}} \mathbf{D}_N.$$

A proof of (i) can be found in [Jai79], Section III A. For a proof of (ii) see [Wan84], Section IV, equation (56).

4.2 Fast DCT-II Algorithms

Computing the DCT of a vector $\mathbf{x} \in \mathbb{R}^N$ via the matrix-vector multiplications from Definition 4.1 has a runtime of $\mathcal{O}(N^2)$. Fortunately, as for the DFT, there exist more efficient techniques for computing DCTs, which can achieve runtimes of $\mathcal{O}(N \log N)$. Some of these methods use a divide-and-conquer approach like the one we explained in Section 1.1.1, some only employ real arithmetic, whereas others are based on existing FFT algorithms. For more details on an efficient algorithm for computing the DCT-II via FFTs see Section 5.2 of this thesis; for FFT-based algorithms for the other types of the DCT see, e.g., [PPST19], Section 6.3.1. It can also be shown that the order $\mathcal{O}(N \log N)$ for the runtime of the fast DCT is optimal for arbitrary input vectors of length N.

As the existence of fast algorithms for the DCT-II is integral for the methods we will present in Chapter 6, we now briefly sketch a fast algorithm for the DCT-II. This section, in which we present a method based on orthogonal matrix factorizations that only employs real arithmetic, is based on [PPST19], Section 6.3.2.

Let $N \in \mathbb{N}$ be even with $N \geq 4$. Our aim is to factorize the matrix $\mathbf{C}_N^{\mathrm{II}}$ into a product of sparse orthogonal matrices that allow for divide-and-conquer steps. For the DFT the factorizations of $\mathbf{F}_N$ corresponding to the radix-2 algorithms which we sketched in Section 1.1.1 mainly depend on $\mathbf{F}_{\frac{N}{2}}$, a permutation matrix and a simple diagonal matrix, see, e.g., [PPST19], Section 5.2.3. However, the factorization of $\mathbf{C}_N^{\mathrm{II}}$ we want to employ does not only depend on $\mathbf{C}_{\frac{N}{2}}^{\mathrm{II}}$, but also on $\mathbf{C}_{\frac{N}{2}}^{\mathrm{IV}}$. Hence, we also require a factorization of $\mathbf{C}_{\frac{N}{2}}^{\mathrm{IV}}$ into orthogonal matrices.

The following factorization of $\mathbf{C}_N^{\mathrm{II}}$ was proved in [PT05], Lemma 2.2 (i). See also [PPST19], Theorem 6.32 (i), for more details.

Lemma 4.5 Let $N \in \mathbb{N}$ be even, $N \geq 4$, and let

$$\mathbf{P}_N \coloneqq \begin{pmatrix} (\delta_{2k,l})_{k,l=0}^{\frac{N}{2}-1,N-1} \\ (\delta_{2k+1,l})_{k,l=0}^{\frac{N}{2}-1,N-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ \vdots & & & \ddots & & \ddots & & \vdots \\ 0 & & & \dots & & \dots & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & & & & \ddots & & & \vdots \\ 0 & \dots & & \dots & & \dots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{N \times N}$$

be the *even-odd permutation matrix*. Further, define

$$\mathbf{T}_N \coloneqq \frac{1}{\sqrt{2}} \left(\begin{array}{c|c} \mathbf{I}_{\frac{N}{2}} & \mathbf{J}_{\frac{N}{2}} \\ \hline \mathbf{I}_{\frac{N}{2}} & -\mathbf{J}_{\frac{N}{2}} \end{array} \right) \in \mathbb{R}^{N \times N},$$

where $\mathbf{I}_{\frac{N}{2}}$ denotes the *identity matrix* of size $\frac{N}{2} \times \frac{N}{2}$ and $\mathbf{J}_{\frac{N}{2}}$ the counter identity matrix of size $\frac{N}{2} \times \frac{N}{2}$ from Theorem 4.4. Then $\mathbf{C}_N^{\mathrm{II}}$ satisfies the following factorization,

$$\mathbf{C}_N^{\mathrm{II}} = \mathbf{P}_N{}^T \left(\begin{array}{c|c} \mathbf{C}_{\frac{N}{2}}^{\mathrm{II}} & \mathbf{0}_{\frac{N}{2}} \\ \hline \mathbf{0}_{\frac{N}{2}} & \mathbf{C}_{\frac{N}{2}}^{\mathrm{IV}} \end{array} \right) \mathbf{T}_N. \tag{4.1}$$

Remark 4.6 Note that for $\mathbf{x} = (x_k)_{k=0}^{N-1} \in \mathbb{R}^N$, N even, we have that

$$\mathbf{P}_N \mathbf{x} = \begin{pmatrix} (x_{2k})_{k=0}^{\frac{N}{2}-1} \\ (x_{2k+1})_{k=0}^{\frac{N}{2}-1} \end{pmatrix}, \tag{4.2}$$

i.e., multiplying $\mathbf{P}_N$ by a vector $\mathbf{x}$ returns the evenly indexed entries of $\mathbf{x}$ in the first half and the oddly indexed ones in the second half. ◊

Proof of Lemma 4.5. Let us consider the matrix $\mathbf{P}_N \mathbf{C}_N^{\mathrm{II}}$. By Remark 4.6 we obtain that

$$\mathbf{P}_N \mathbf{C}_N^{\mathrm{II}} = \sqrt{\frac{2}{N}} \begin{pmatrix} \left(\varepsilon_N(2k) \cos\left(\frac{2k(2l+1)\pi}{2N} \right) \right)_{k,\,l=0}^{\frac{N}{2}-1,\,N-1} \\ \left(\cos\left(\frac{(2k+1)(2l+1)\pi}{2N} \right) \right)_{k,\,l=0}^{\frac{N}{2}-1,\,N-1} \end{pmatrix}, \tag{4.3}$$

since $2k+1 \not\equiv 0 \mod N$ for all $k \in \{0, \ldots, \frac{N}{2}-1\}$. Writing the right-hand side of (4.3) as a block matrix consisting of four submatrices indexed from 0 to $\frac{N}{2}-1$ yields

$$\begin{aligned} &\mathbf{P}_N \mathbf{C}_N^{\mathrm{II}} \\ &= \sqrt{\frac{2}{N}} \left(\begin{array}{c|c} \left(\varepsilon_{\frac{N}{2}}(k) \cos\left(\frac{2k(2l+1)\pi}{2N} \right) \right)_{k,\,l=0}^{\frac{N}{2}-1} & \left(\varepsilon_{\frac{N}{2}}(k) \cos\left(\frac{2k\left(2\left(\frac{N}{2}+l\right)+1\right)\pi}{2N} \right) \right)_{k,\,l=0}^{\frac{N}{2}-1} \\ \hline \left(\cos\left(\frac{(2k+1)(2l+1)\pi}{2N} \right) \right)_{k,\,l=0}^{\frac{N}{2}-1} & \left(\cos\left(\frac{(2k+1)\left(2\left(\frac{N}{2}+l\right)+1\right)\pi}{2N} \right) \right)_{k,\,l=0}^{\frac{N}{2}-1} \end{array} \right) \\ &= \sqrt{\frac{2}{N}} \left(\begin{array}{c|c} \sqrt{\frac{N}{4}} \mathbf{C}_{\frac{N}{2}}^{\mathrm{II}} & \left(\varepsilon_{\frac{N}{2}}(k) \cos\left(\frac{k(N+2l+1)\pi}{N} \right) \right)_{k,\,l=0}^{\frac{N}{2}-1} \\ \hline \sqrt{\frac{N}{4}} \mathbf{C}_{\frac{N}{2}}^{\mathrm{IV}} & \left(\cos\left(\frac{(2k+1)(N+2l+1)\pi}{2N} \right) \right)_{k,\,l=0}^{\frac{N}{2}-1} \end{array} \right). \end{aligned} \tag{4.4}$$

Since

$$\begin{aligned} &\cos\left(\frac{k(N+2l+1)\pi}{N} \right) \\ &= \cos(k\pi) \cos\left(\frac{k(2l+1)\pi}{N} \right) \underbrace{- \sin(k\pi) \sin\left(\frac{k(2l+1)\pi}{N} \right)}_{=0} \\ &= \cos(k\pi) \cos\left(\frac{k(2l+1)\pi}{N} \right) \underbrace{+ \sin(k\pi) \sin\left(\frac{k(2l+1)\pi}{N} \right)}_{=0} \end{aligned}$$

$$
\begin{aligned}
&= \cos\left(\frac{k(N-2l-1)\pi}{N}\right) \\
&= \cos\left(\frac{k\left(2\left(\frac{N}{2}-l-1\right)+1\right)\pi}{N}\right),
\end{aligned}
$$

we find for the top-right quadrant of the matrix in (4.4) that

$$
\begin{aligned}
&\left(\varepsilon_{\frac{N}{2}}(k)\cos\left(\frac{k\left(N+2l+1\right)\pi}{N}\right)\right)_{k,\,l=0}^{\frac{N}{2}-1} \\
&= \left(\varepsilon_{\frac{N}{2}}(k)\cos\left(\frac{k\left(2\left(\frac{N}{2}-l-1\right)+1\right)\pi}{N}\right)\right)_{k,\,l=0}^{\frac{N}{2}-1} \\
&= \sqrt{\frac{N}{4}}\mathbf{C}^{\mathrm{II}}_{\frac{N}{2}}\mathbf{J}_{\frac{N}{2}}. \qquad (4.5)
\end{aligned}
$$

Similarly, using

$$
\begin{aligned}
&\cos\left(\frac{(2k+1)(N+2l+1)\pi}{2N}\right) \\
&= \underbrace{\cos\left(\frac{(2k+1)\pi}{2}\right)\cos\left(\frac{(2k+1)(2l+1)\pi}{2N}\right)}_{=0} \\
&\quad - \sin\left(\frac{(2k+1)\pi}{2}\right)\sin\left(\frac{(2k+1)(2l+1)\pi}{2N}\right) \\
&= \underbrace{-\cos\left(\frac{(2k+1)\pi}{2}\right)\cos\left(\frac{(2k+1)(2l+1)\pi}{2N}\right)}_{=0} \\
&\quad - \sin\left(\frac{(2k+1)\pi}{2}\right)\sin\left(\frac{(2k+1)(2l+1)\pi}{2N}\right) \\
&= -\cos\left(\frac{(2k+1)(N-2l-1)\pi}{2N}\right) \\
&= -\cos\left(\frac{(2k+1)\left(2\left(\frac{N}{2}-l-1\right)+1\right)\pi}{2N}\right),
\end{aligned}
$$

the bottom-right quadrant of the matrix in (4.4) can be written as

$$
\begin{aligned}
&\left(\cos\left(\frac{(2k+1)\left(N+2l+1\right)\pi}{2N}\right)\right)_{k,\,l=0}^{\frac{N}{2}-1} \\
&= -\left(\cos\left(\frac{(2k+1)\left(2\left(\frac{N}{2}-l-1\right)+1\right)\pi}{2N}\right)\right)_{k,\,l=0}^{\frac{N}{2}-1} \\
&= -\sqrt{\frac{N}{4}}\mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}}\mathbf{J}_{\frac{N}{2}}. \qquad (4.6)
\end{aligned}
$$

Combining (4.5) and (4.6) with (4.4), we obtain that

$$\begin{aligned}
\mathbf{P}_N \mathbf{C}_N^{\mathrm{II}} &= \sqrt{\frac{2}{N}} \left(\begin{array}{c|c} \sqrt{\frac{N}{4}}\mathbf{C}^{\mathrm{II}}_{\frac{N}{2}} & \sqrt{\frac{N}{4}}\mathbf{C}^{\mathrm{II}}_{\frac{N}{4}}\mathbf{J}_{\frac{N}{2}} \\ \hline \sqrt{\frac{N}{4}}\mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} & -\sqrt{\frac{N}{4}}\mathbf{C}^{\mathrm{IV}}_{\frac{N}{4}}\mathbf{J}_{\frac{N}{2}} \end{array}\right) \\
&= \frac{1}{\sqrt{2}} \left(\begin{array}{c|c} \mathbf{C}^{\mathrm{II}}_{\frac{N}{2}} & \mathbf{C}^{\mathrm{II}}_{\frac{N}{2}}\mathbf{J}_{\frac{N}{2}} \\ \hline \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} & -\mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}}\mathbf{J}_{\frac{N}{2}} \end{array}\right) \\
&= \left(\begin{array}{c|c} \mathbf{C}^{\mathrm{II}}_{\frac{N}{2}} & \mathbf{0}_{\frac{N}{2}} \\ \hline \mathbf{0}_{\frac{N}{2}} & \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \end{array}\right) \frac{1}{\sqrt{2}} \left(\begin{array}{c|c} \mathbf{I}_{\frac{N}{2}} & \mathbf{J}_{\frac{N}{2}} \\ \hline \mathbf{I}_{\frac{N}{2}} & -\mathbf{J}_{\frac{N}{2}} \end{array}\right) \\
&= \left(\begin{array}{c|c} \mathbf{C}^{\mathrm{II}}_{\frac{N}{2}} & \mathbf{0}_{\frac{N}{2}} \\ \hline \mathbf{0}_{\frac{N}{2}} & \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \end{array}\right) \mathbf{T}_N,
\end{aligned}$$

which proves the claim. □

Note that both $\mathbf{P}_N$, as a permutation matrix, and $\mathbf{T}_N$ are orthogonal, which implies that the factorization from Lemma 4.5 is indeed one into real orthogonal sparse matrices. This factorization now provides us with the necessary tools for a first divide-and-conquer step. Let $\mathbf{x} \in \mathbb{R}^N$, where $N \geq 4$ is even. Let us denote by $\mathbf{x}_{(0)}$ and $\mathbf{x}_{(1)}$ the *first and second half* of $\mathbf{x}$, respectively, i.e.,

$$\mathbf{x}_{(0)} \coloneqq (x_k)_{k=0}^{\frac{N}{2}-1} \in \mathbb{R}^{\frac{N}{2}} \qquad \text{and} \qquad \mathbf{x}_{(1)} \coloneqq (x_k)_{k=\frac{N}{2}}^{N-1} \in \mathbb{R}^{\frac{N}{2}}.$$

Then we find that

$$\begin{aligned}
\mathbf{x}^{\widehat{\mathrm{II}}} = \mathbf{C}_N^{\mathrm{II}}\mathbf{x} &= {\mathbf{P}_N}^T \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{\frac{N}{2}} & \mathbf{0}_{\frac{N}{2}} \\ \mathbf{0}_{\frac{N}{2}} & \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I}_{\frac{N}{2}} & \mathbf{J}_{\frac{N}{2}} \\ \mathbf{I}_{\frac{N}{2}} & -\mathbf{J}_{\frac{N}{2}} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}_{(0)} \\ \mathbf{x}_{(1)} \end{pmatrix} \\
&= {\mathbf{P}_N}^T \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{\frac{N}{2}} \left(\mathbf{x}_{(0)} + \mathbf{J}_{\frac{N}{2}}\mathbf{x}_{(1)}\right) \\ \mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}} \left(\mathbf{x}_{(0)} - \mathbf{J}_{\frac{N}{2}}\mathbf{x}_{(1)}\right) \end{pmatrix}.
\end{aligned}$$

Consequently, the first subproblem of half size is to compute the DCT-II of the vector $\mathbf{x}_{(0)} + \mathbf{J}_{\frac{N}{2}}\mathbf{x}_{(1)} \in \mathbb{R}^{\frac{N}{2}}$, and the second subproblem of half size is to compute the DCT-IV of the vector $\mathbf{x}_{(0)} - \mathbf{J}_{\frac{N}{2}}\mathbf{x}_{(1)} \in \mathbb{R}^{\frac{N}{2}}$. Thus, we also require a factorization of the matrix $\mathbf{C}^{\mathrm{IV}}_{\frac{N}{2}}$ into real orthogonal sparse matrices such that we can apply the divide-and-conquer paradigm.

The following factorization of $\mathbf{C}_N^{\mathrm{IV}}$ was shown in [PT05], Lemma 2.4. See [PPST19], Section 6.3.2, Theorem 6.33, for more details.

Lemma 4.7 Let $N \in \mathbb{N}$ be even, $N \geq 4$. Define

$$\mathbf{A}_N := \frac{1}{\sqrt{2}} \left(\begin{array}{c|cc|c} \sqrt{2} & & & \\ \hline & \mathbf{I}_{\frac{N}{2}-1} & \mathbf{I}_{\frac{N}{2}-1} & \\ & \mathbf{I}_{\frac{N}{2}-1} & -\mathbf{I}_{\frac{N}{2}-1} & \\ \hline & & & -\sqrt{2} \end{array}\right) \cdot \left(\begin{array}{c|c} \mathbf{I}_{\frac{N}{2}} & \mathbf{0}_{\frac{N}{2}} \\ \hline \mathbf{0}_{\frac{N}{2}} & \mathbf{D}_{\frac{N}{2}} \mathbf{J}_{\frac{N}{2}} \end{array}\right) \in \mathbb{R}^{N \times N}.$$

Further, let

$$\mathbf{c}_{\frac{N}{2}} := \left(\cos\left(\frac{(2k+1)\pi}{4N}\right)\right)_{k=0}^{\frac{N}{2}-1} \quad \text{and} \quad \mathbf{s}_{\frac{N}{2}} := \left(\sin\left(\frac{(2k+1)\pi}{4N}\right)\right)_{k=0}^{\frac{N}{2}-1},$$

and set

$$\mathbf{T}_N(1) := \left(\begin{array}{c|c} \mathbf{I}_{\frac{N}{2}} & \\ \hline & \mathbf{D}_{\frac{N}{2}} \end{array}\right) \cdot \left(\begin{array}{c|c} \operatorname{diag}\left(\mathbf{c}_{\frac{N}{2}}\right) & \operatorname{diag}\left(\mathbf{s}_{\frac{N}{2}}\right) \mathbf{J}_{\frac{N}{2}} \\ \hline -\mathbf{J}_{\frac{N}{2}} \operatorname{diag}\left(\mathbf{s}_{\frac{N}{2}}\right) & \operatorname{diag}\left(\mathbf{J}_{\frac{N}{2}} \mathbf{c}_{\frac{N}{2}}\right) \end{array}\right) \in \mathbb{R}^{N \times N}.$$

Then $\mathbf{C}_N^{\mathrm{IV}}$ satisfies the following factorization,

$$\mathbf{C}_N^{\mathrm{IV}} = \mathbf{P}_N{}^T \mathbf{A}_N \left(\begin{array}{c|c} \mathbf{C}_{\frac{N}{2}}^{\mathrm{II}} & \mathbf{0}_{\frac{N}{2}} \\ \hline \mathbf{0}_{\frac{N}{2}} & \mathbf{C}_{\frac{N}{2}}^{\mathrm{II}} \end{array}\right) \mathbf{T}_N(1).$$

Note that $\mathbf{A}_N$ and $\mathbf{T}_N(1)$ are indeed orthogonal matrices. With the factorization from Lemma 4.7 the subproblem of size $\frac{N}{2}$ of computing the DCT-IV of the vector $\mathbf{x}_{(0)} - \mathbf{J}_{\frac{N}{2}} \mathbf{x}_{(1)}$ can be reduced to two subproblems of size $\frac{N}{4}$ of essentially computing DCT-IIs of length $\frac{N}{4}$, since

$$\begin{aligned} & \mathbf{C}_N^{\mathrm{IV}} \cdot \left(\mathbf{x}_{(0)} - \mathbf{J}_{\frac{N}{2}} \mathbf{x}_{(1)}\right) \\ &= \mathbf{P}_{\frac{N}{2}}{}^T \mathbf{A}_{\frac{N}{2}} \left(\begin{array}{c|c} \mathbf{C}_{\frac{N}{4}}^{\mathrm{II}} & \mathbf{0}_{\frac{N}{4}} \\ \hline \mathbf{0}_{\frac{N}{4}} & \mathbf{C}_{\frac{N}{4}}^{\mathrm{II}} \end{array}\right) \mathbf{T}_{\frac{N}{2}}(1) \cdot \left(\mathbf{x}_{(0)} - \mathbf{J}_{\frac{N}{2}} \mathbf{x}_{(1)}\right) \\ &= \mathbf{P}_N{}^T \mathbf{A}_N \begin{pmatrix} \mathbf{C}_{\frac{N}{4}}^{\mathrm{II}} \left(\mathbf{T}_{\frac{N}{2}}(1) \left(\mathbf{x}_{(0)} - \mathbf{J}_{\frac{N}{2}} \mathbf{x}_{(1)}\right)\right)_{(0)} \\ \mathbf{C}_{\frac{N}{4}}^{\mathrm{II}} \left(\mathbf{T}_{\frac{N}{2}}(1) \left(\mathbf{x}_{(0)} - \mathbf{J}_{\frac{N}{2}} \mathbf{x}_{(1)}\right)\right)_{(1)} \end{pmatrix}. \end{aligned}$$

These subproblems of size $\frac{N}{4}$ can again be split into a DCT-II and a DCT-IV computation of length $\frac{N}{8}$ if 8 divides N. Continuing these splitting steps until the vectors in the subproblems have length 2, for which we compute the DCT-II and DCT-IV directly, yields fast algorithms for the DCT-II and DCT-IV. Careful consideration of the matrix factorizations given by Lemmas 4.5 and 4.7 shows that, for a vector $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$,

the fast DCT-II algorithm performs

$$\frac{4}{3}NJ - \frac{8}{9}N - \frac{1}{9}(-1)^J + 1 = \mathcal{O}(NJ) = \mathcal{O}\left(N \log_2 N\right)$$

complex additions and

$$NJ - \frac{4}{3}N + \frac{1}{3}(-1)^J + 1 = \mathcal{O}\left(N \log_2 N\right)$$

complex multiplications. The fast DCT-IV algorithm also has a runtime of $\mathcal{O}\left(N \log_2 N\right)$, with similarly small constants. See [PPST19], Section 6.3.2, Theorem 6.39, for a proof of these runtime complexities.

Remark 4.8 Since $\mathbf{C}_N^{\mathrm{III}} = \mathbf{C}_N^{\mathrm{II}\,T} = \mathbf{C}_N^{\mathrm{II}\,-1}$, Lemma 4.5 also provides us with an orthogonal factorization of the cosine matrix of type III. Thus, we directly obtain a fast algorithm with runtime $\mathcal{O}\left(N \log_2 N\right)$ for the DCT-III, which is the same as the IDCT-II. It can be shown that for the DCT-I there also exist fast algorithms with a runtime of $\mathcal{O}(N \log_2 N)$. For an overview of several fast methods for the different DCT and DST types see, e.g., [BYR06], Section 4.4.

Besides the already mentioned possibility of computing the DCT of a vector via the DFT, which will be explained for the DCT-II in detail in Section 5.2, there also exist fast DCT algorithms based on Chebyshev polynomials. These methods use factorizations of the cosine matrices which are not orthogonal, thus leading to less stable algorithms, but also only require real arithmetic, see, e.g., [Fei90, FW92, PM03, Ste92, ST91]. Further, the DCT-II can be computed via the Walsh-Hadamard transform, see, e.g., [AR75] and [BYR06], Section 4.4.3.3. There also exist split-radix methods for the DCT-II, see, e.g., [BYR06], Section 4.4.3.4. Other algorithms include, for example, [SH86, Wan84, Wan83, CSF77]. All of these methods have a runtime of $\mathcal{O}(N \log N)$ for arbitrary vectors of length N. ◊

4.3 2-Dimensional Discrete Cosine Transform

Some of the main areas of application for discrete cosine and sine transforms are digital image or video processing and compression, and transform-based coding applications. All of these problems are at least 2-dimensional, so there has also been extensive research regarding the development of fast 2-dimensional DCT and DST algorithms, with particular focus on the DCT-II.

We now define the 2-dimensional discrete cosine transforms of types II and IV, see, e.g., [RY90], Chapter 5 and [BYR06], Section 4.5.

Definition 4.9 (2-Dimensional DCT-II and DCT-IV) Let $\mathbf{A} \in \mathbb{R}^{M \times N}$. Then the *2-dimensional discrete cosine transforms of types II and IV* of $\mathbf{A}$ are defined as

$$\mathbf{A}^{\widehat{\mathrm{II}}} \coloneqq \mathbf{C}_M^{\mathrm{II}} \mathbf{A} \mathbf{C}_N^{\mathrm{II}\,T} \qquad \text{and} \qquad \mathbf{A}^{\widehat{\mathrm{IV}}} \coloneqq \mathbf{C}_M^{\mathrm{IV}} \mathbf{A} \mathbf{C}_N^{\mathrm{IV}}.$$

The other 2-dimensional discrete trigonometric transforms are defined analogously. As for the 1-dimensional DCT, the 2-dimensional DCT of a real $M \times N$ matrix can be calculated by applying a 2-dimensional DFT, see, e.g., [RY90], Section 5.4. However, there also exist direct approaches for computing 2-dimensional discrete trigonometric transforms. The first method, the so-called *row-column method*, is based on the ability

to rapidly calculate 1-dimensional DCTs or DSTs. It first applies a fast 1-dimensional DCT or DST algorithm of length N to all row vectors of the input matrix $\mathbf{A}$. Then another fast 1-dimensional M-length DCT or DST algorithm is applied to all column vectors of the resulting matrix, which yields an overall runtime of

$$\mathcal{O}\left(MN \log_2(MN)\right),$$

while using MN samples of $\mathbf{A}^{\widehat{\Pi}}$.

The second approach is a 2-dimensional vector-radix method, which uses 2-dimensional decomposition ideas, see, e.g., [CH91, WP89b, WP89a, WP91, BR00]. Such methods can be directly applied to 2-dimensional data, as they decompose the $M \times N$ DCT or DST into sums of four DCTs or DSTs of size $\frac{M}{2} \times \frac{N}{2}$. Applying this idea recursively until DCTs or DSTs of size 2×2 are achieved, these methods are even faster than row-column algorithms, having to perform

$$\frac{3}{4} N^2 \log_2 N$$

instead of

$$N^2 \log_2 N$$

multiplications for a matrix of size $N \times N$. Other approaches include algorithms based on polynomial transforms, see [DG90, PD96].

4.4 Vandermonde Matrices and Chebyshev Polynomials

Even though the connection between discrete cosine transforms on the one hand and polynomial interpolation on the other hand may not seem evident at first, we will now briefly recall some basic results in the second topic. They will lead us to the concept of (odd) Vandermonde matrices and Chebyshev zero nodes, which we will employ in the reconstruction procedures in Chapter 6.

Polynomial interpolation is the problem of, given a set of $n+1$ data tuples $(x_0, y_0), \ldots, (x_n, y_n) \in \mathbb{R}^2$, finding a polynomial P of degree at most n that satisfies

$$P(x_k) = y_k \qquad \forall\, k \in \{0, \ldots, n\}. \tag{4.7}$$

It can be easily shown that the interpolation problem (4.7) has a unique solution if the interpolation points x_k, $k \in \{0, \ldots, n\}$, are distinct, see, e.g., [Atk89], Chapter 3, Theorem 3.1. Let us denote the *vector space of polynomials with real coefficients of degree at most* n by Π_n, i.e.,

$$\Pi_n \coloneqq \left\{ P(x) = \sum_{l=0}^{n} a_l x^l : a_l \in \mathbb{R}\ \forall\, l \in \{0, \ldots, n\} \right\}.$$

Assume that $P \in \Pi_n$ with

$$P(x) = \sum_{l=0}^{n} a_l x^l$$

satisfies (4.7), i.e.,

$$P(x_k) = \sum_{l=0}^{n} a_l {x_k}^l = y_k \qquad \forall\, k \in \{0, \ldots, n\}.$$

Writing this as a linear equation system, we obtain

$$\begin{pmatrix} 1 & x_0 & {x_0}^2 & \dots & {x_0}^n \\ 1 & x_1 & {x_1}^2 & \dots & {x_1}^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & {x_n}^2 & \dots & {x_n}^n \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}. \tag{4.8}$$

The matrix in (4.8) is known as the Vandermonde matrix.

Definition 4.10 (Vandermonde Matrix) Let $n \in \mathbb{N}$ and $\mathbf{x} = (x_k)_{k=0}^n \in \mathbb{R}^{n+1}$. The matrix

$$\mathbf{V}(\mathbf{x}) \coloneqq \left({x_k}^l\right)_{k,l=0}^n = \begin{pmatrix} 1 & x_0 & {x_0}^2 & \dots & {x_0}^n \\ 1 & x_1 & {x_1}^2 & \dots & {x_1}^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & {x_n}^2 & \dots & {x_n}^n \end{pmatrix}$$

is called *Vandermonde matrix*.

The following property of Vandermonde matrices is very well known. For a proof see, e.g., [Sch02], Section 3.1.2, Lemma 3.1.2.

Lemma 4.11 Let $n \in \mathbb{N}$ and $\mathbf{x} = (x_k)_{k=0}^n \in \mathbb{R}^{n+1}$. Then we have that

$$\det(\mathbf{V}(\mathbf{x})) = \prod_{0 \le k < l \le n} (x_l - x_k),$$

so $\mathbf{V}(\mathbf{x})$ is invertible if and only if $x_0, \dots, x_n$ are pairwise distinct.

Consequently, since the monomials $1, x, \dots, x^n$ form a basis of Π_n, the interpolation problem (4.7) has a unique solution if the interpolation points $x_0, \dots, x_n$ are all distinct. However, Vandermonde matrices are, in general, ill-conditioned, so interpolating polynomials are not found by inverting (4.8) in practice. Instead of the monomial basis, which corresponds to the Vandermonde matrix, bases consisting of, e.g., Lagrange, Newton or Chebyshev polynomials are used.

If the interpolation points are generated by evaluating a function, i.e., if

$$y_k = f(x_k) \qquad \forall\, k \in \{0, \dots, n\}$$

for a function $f\colon [x_0, x_n] \to \mathbb{R}$, then it is also of interest whether the interpolating polynomial P approximates the function f well in some sense. To be more precise, the goal is to find a polynomial $P \in \Pi_n$ such that the *interpolation error*

$$r_n(x) \coloneqq f(x) - P(x) \tag{4.9}$$

is sufficiently small. The following result can be found, e.g., in [Atk89], Section 3.2, equations (3.2.11) and (3.2.12).

Theorem 4.12 *Let $n \in \mathbb{N}$ and $x_0, \dots, x_n \in \mathbb{R}$ be pairwise distinct. Let $x \in \mathbb{R}$ and $f\colon I \to \mathbb{R}$ be $(n+1)$-times continuously differentiable, where I is an interval containing x and $x_0, \dots, x_n$. Let $P \in \Pi_n$ be the interpolating polynomial for the interpolation problem*

$$P(x_k) = f(x_k) \qquad \forall\, k \in \{0, \dots, n\}.$$

Then we have that

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{k=0}^{n} (x - x_k)$$

for some $\xi \in I$ depending on x. Further, it follows that

$$|f(x) - P(x)| \leq \frac{\left\| f^{(n+1)} \right\|_\infty}{(n+1)!} \cdot \left| \prod_{k=0}^{n} (x - x_k) \right|. \tag{4.10}$$

The first factor in the error estimate only depends on f and the degree of the interpolating polynomial, so we cannot influence it, since for a fixed degree the only parameters that can be varied are the interpolation points. These considerations give rise to the problem of finding interpolation points which minimize

$$\left| \prod_{k=0}^{n} (x - x_k) \right|$$

for x in a given interval I. Such optimal interpolation points do indeed exist; they are called Chebyshev (zero) nodes, as they are the zeros of the Chebyshev polynomial of the first kind.

Definition 4.13 (Chebyshev Polynomials of the First Kind) Let $n \in \mathbb{N}_0$ and $x \in \mathbb{R}$. Then the *Chebyshev polynomial of the first kind of degree $n+1$* is defined as

$$T_{n+1}(x) \coloneqq 2^n \prod_{k=0}^{n} \left(x - \cos\left(\frac{(2k+1)\pi}{2(n+1)} \right) \right) =: \sum_{l=0}^{n+1} \alpha_{n+1,\,l}\, x^l,$$

and $T_0(x) \coloneqq 1$.

Some of the most important properties of the Chebyshev polynomials of the first kind are summarized in the next lemma.

Lemma 4.14 Let $n \in \mathbb{N}_0$ and $x \in \mathbb{R}$.

(i) T_n is a polynomial of degree n.

(ii) The leading coefficient of T_n satisfies

$$\alpha_{n,\,n} = \begin{cases} 1 & \text{if } n = 0, \\ 2^{n-1} & \text{if } n \geq 1. \end{cases} \tag{4.11}$$

(iii) T_n is odd if n is odd, and T_n is even if n is even.

(iv) The n zeros of T_n are

$$t_{n,\,l} \coloneqq \cos\left(\frac{(2l+1)\pi}{2n} \right), \qquad l \in \{0, \ldots, n-1\},$$

and they are called *Chebyshev zero nodes*.

(v) If $|x| \leq 1$, then T_n can be written as

$$T_n(x) = \cos(n \arccos x).$$

(vi) Evaluating T_k at the Chebyshev zero nodes $t_{n,l}$ for $l \in \{0, \dots, n-1\}$, $n \in \mathbb{N}$ and $k \in \mathbb{N}_0$ yields

$$T_k\left(t_{n,l}\right) = \cos\left(\frac{k(2l+1)\pi}{2n}\right).$$

Claims (i), (ii) and (iv) follow directly from Definition 4.13. Proofs of (iii) and (v) can be found in [PPST19], Section 6.1. The claim in (vi) follows directly from (v).

Using the properties in Lemma 4.14, one can prove that the maximal absolute value of the polynomial $2^{-n}T_{n+1}$ in the interval $[-1,1]$ is indeed not greater than the maximal absolute value of any other polynomial of degree $n+1$ with leading coefficient 1 in $[-1,1]$. Consequently, the Chebyshev zero nodes $x_k = t_{n+1,\,k}$ minimize the value

$$\max_{x\in[-1,1]} \left| \prod_{k=0}^{n} (x - x_k) \right|$$

in (4.10) in Theorem 4.12.

The Chebyshev polynomials, which form a complete orthogonal system, are used in many areas of numerical mathematics besides polynomial interpolation, e.g., for spectral methods and for the so-called Chebyshev filters. We will utilize them in Chapter 6 in order to obtain an invertible factorization of a special submatrix of the cosine matrix of type IV. Apart from Chebyshev polynomials and Chebyshev zero nodes this will require the notion of odd Vandermonde matrices. We saw at the beginning of this section how Vandermonde matrices are related to polynomial interpolation in the monomial basis. If it is known a priori that an odd function f is being interpolated, then the interpolating polynomial will be odd as well. Hence, it suffices to restrict the linear equation system (4.8) to the columns corresponding to odd exponents, as the polynomial coefficients a_{2k}, $k \in \left\{0, \dots, \left\lfloor \frac{n}{2} \right\rfloor\right\}$, have to be zero. This motivates the notion of odd Vandermonde matrices introduced in [BP18a].

Definition 4.15 (Odd Vandermonde Matrix) Let $n \in \mathbb{N}$ and $\mathbf{x} = (x_k)_{k=0}^{n} \in \mathbb{R}^{n+1}$. The matrix

$$\mathbf{V}^{\mathrm{odd}}(\mathbf{x}) \coloneqq \left({x_k}^{2l+1}\right)_{k,\,l=0}^{n} = \begin{pmatrix} x_0 & {x_0}^3 & {x_0}^5 & \dots & {x_0}^{2n+1} \\ x_1 & {x_1}^3 & {x_1}^5 & \dots & {x_1}^{2n+1} \\ \vdots & \vdots & \vdots & & \vdots \\ x_n & {x_n}^3 & {x_n}^5 & \dots & {x_n}^{2n+1} \end{pmatrix}$$

is called *odd Vandermonde matrix*.

The determinant of an odd Vandermonde matrix is related to the determinant of a standard Vandermonde matrix as given by Definition 4.10.

Lemma 4.16 (Lemma 3.1 in [BP18a]) Let $n \in \mathbb{N}$ and $\mathbf{x} = (x_k)_{k=0}^{n} \in \mathbb{R}^{n+1}$ such that $x_0, \dots, x_n \neq 0$ and $|x_k| \neq |x_l|$ for all $k \neq l$, where $k, l \in \{0, \dots, n\}$. Then the odd Vandermonde matrix $\mathbf{V}^{\mathrm{odd}}(\mathbf{x})$ is invertible with

$$\det\left(\mathbf{V}^{\mathrm{odd}}(\mathbf{x})\right) = \prod_{j=0}^{n} x_j \cdot \det\left(\mathbf{V}\left({x_0}^2, \dots, {x_n}^2\right)\right) = \prod_{j=0}^{n} x_j \prod_{0\le k<l\le n} \left({x_l}^2 - {x_k}^2\right).$$

Proof. Using the multilinearity of the determinant and Lemma 4.11, we obtain that

$$\begin{aligned}
\det\left(\mathbf{V}^{\mathrm{odd}}(\mathbf{x})\right) &= \det\begin{pmatrix} x_0 & {x_0}^3 & {x_0}^5 & \dots & {x_0}^{2n+1} \\ x_1 & {x_1}^3 & {x_1}^5 & \dots & {x_1}^{2n+1} \\ \vdots & \vdots & \vdots & & \vdots \\ x_n & {x_n}^3 & {x_n}^5 & \dots & {x_n}^{2n+1} \end{pmatrix} \\
&= \prod_{j=0}^{n} x_j \cdot \det\begin{pmatrix} 1 & {x_0}^2 & {x_0}^4 & \dots & {x_0}^{2n} \\ 1 & {x_1}^2 & {x_1}^4 & \dots & {x_1}^{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & {x_n}^2 & {x_n}^4 & \dots & {x_n}^{2n} \end{pmatrix} \\
&= \prod_{j=0}^{n} x_j \cdot \det\left(\mathbf{V}\left({x_0}^2, \dots, {x_n}^2\right)\right) \\
&= \prod_{j=0}^{n} x_j \prod_{0 \le k < l \le n} \left({x_l}^2 - {x_k}^2\right).
\end{aligned}$$

As $x_k \neq 0$ and $|x_k| \neq |x_l|$ for $k \neq l$, $k, l \in \{0, \dots, n\}$, it follows that $\mathbf{V}^{\mathrm{odd}}(x_0, \dots, x_n)$ is invertible. □

Recall that any odd function f is rotational symmetric with respect to the origin and always satisfies that $f(0) = 0$. Consequently, if it is known a priori that we interpolate an odd function f with an odd polynomial, the interpolation point 0 cannot yield additional knowledge about the function and must not be used. Further, since $f(x) = -f(-x)$, two nodes x_k, x_l with $|x_k| = |x_l|$ provide the same information. Thus, it makes sense that $\mathbf{V}^{\mathrm{odd}}(\mathbf{x})$ is invertible if and only if none of the interpolation points is 0 and their absolute values are pairwise distinct.

5 Sparse Fast IDCT-II for Vectors with One-Block Support Based on IFFT

In the second part of this thesis we are interested in deterministically reconstructing real vectors from their DCT-II. For the closely related case of recovering a vector from its DFT under the assumption of sparsity there has been extensive research in recent years. We listed several sparse IFFT and FFT methods at the beginning of Chapter 2, including, e.g., the deterministic IFFT methods for vectors [PW16a, PW17a, PWCW18], which we will explain in more detail in Sections 5.3 and 6.5.1. The investigation of sparse discrete cosine and sine transforms has not yet been that thorough, even though, as mentioned in Section 4, there exists a variety of fast DCT and DST algorithms for arbitrary input vectors. They have a runtime of $\mathcal{O}(N \log N)$, which is optimal for arbitrary input vectors of length N. As for the 2π-periodic functions considered in the first part of this thesis, we can only hope to achieve lower runtimes if we additionally assume sparsity of the output vector.

Being one of the most widely used algorithms in applied mathematics, engineering and signal processing due to the fact that it approximates the statistically optimal KLT, there indeed exist various applications not only for general DCTs but also for sparse DCTs. For example, sparse DCTs can be employed to rapidly evaluate polynomials in monomial form from sparse expansions of Chebyshev polynomials, see, e.g., [PPST19], Section 6.2. Since DCTs also play an important part in data compression, e.g., for images and videos, the ability to incorporate sparsity into such methods can also lead to faster data compression algorithms.

Of course, given a vector $\mathbf{x} = (x_k)_{k=0}^{N-1} \in \mathbb{R}^N$, it is always possible to obtain $\mathbf{x}^{\widehat{\mathrm{II}}}$ by applying a DFT to the vector

$$\mathbf{y} = (x_0, x_1, \ldots, x_{N-1}, x_{N-1}, x_{N-2}, \ldots, x_0) \in \mathbb{R}^{2N},$$

since

$$x^{\widehat{\mathrm{II}}} = \frac{\varepsilon_N(k)}{\sqrt{2N}} {\omega_{4N}}^k \cdot \widehat{y}_k \qquad \forall\, k \in \{0, \ldots, N-1\},$$

as we will explain in more detail in Section 5.2. Similar FFT-based approaches also exist for the computation of DCTs of types I, III and IV, see, e.g., [PPST19], Section 6.3.1. If $\mathbf{x}$ is m-sparse, then the auxiliary vector $\mathbf{y}$ is $2m$-sparse. Thus, by applying a general $2m$-sparse FFT algorithm to $\mathbf{y}$, one can obtain the DCT-II of $\mathbf{x}$ faster than by performing a fast full-length DCT-II like the one described in Section 4.2. However, $\mathbf{y}$ has twice as many nonzero entries as $\mathbf{x}$, so applying a $2m$-sparse FFT will not be the most efficient method, especially since $\mathbf{y}$ is symmetric.

If the input vector $\mathbf{x}$ is not only sparse but satisfies some additional structural properties, for example having a short support, then the structure of the support of $\mathbf{y}$ is closely related to that of $\mathbf{x}$. This special structure of $\mathbf{y}$ can be employed to find a faster sparse FFT algorithm for $\mathbf{y}$ and thus a faster DCT-II algorithm for $\mathbf{x}$. Another way to obtain a fast sparse DCT-II method is to utilize the special structure of $\mathbf{x}$ directly for the DCT-II computations without employing DFTs.

Our topic of interest for this and the following chapter is the deterministic recovery of a vector $\mathbf{x} \in \mathbb{R}^N$ with short support of length m from its DCT-II transformed vector $\mathbf{x}^{\widehat{\mathrm{II}}}$, so we will focus on inverse DCT-IIs. For vectors having a short support means that the indices corresponding to its significantly large entries are contained in an interval $S^{\mathbf{x}}$ of length m. In this chapter we will develop a fast IDCT-II algorithm for vectors with short support that is based on IFFTs, whereas in Chapter 6 we will introduce an IDCT-II algorithm for vectors with short support that only requires real arithmetic. We will compare the performances of both methods numerically in Section 6.5.

To the best of of our knowledge these are the first IDCT-II methods that are specifically optimized for the DCT-II setting and utilize the special support structures of $\mathbf{x}$ and the auxiliary vector $\mathbf{y}$ that arise if $\mathbf{x}$ has a short support. We are not aware of other sparse algorithms for any of the DCT and DST types that are specifically tailored to the respective cosine and sine bases or the occurring sparsity structures.

The following chapter is based on our paper [BP18c] and is in parts identical with the representations therein.

5.1 One-Block Support

Throughout this chapter we will always consider a real vector $\mathbf{x} = (x_k)_{k=0}^{N-1} \in \mathbb{R}^N$, $N \coloneqq 2^{J-1}$, that has a short or one-block support. Unlike in Chapters 2 and 3, we use the representative system $\{0, 1, \ldots, N-1\}$ for the residues modulo N, since both $\mathbf{x}$ and $\mathbf{x}^{\widehat{\mathrm{II}}}$ are indexed from 0 to $N-1$ by definition. In order to formally define the concept of a one-block support, we first have to introduce periodized intervals.

Definition 5.1 Let $n = 2^j$ with $j \in \mathbb{N}_0$ and $a, b \in \mathbb{N}_0$. Then we denote by $I_{a,b}^{(j)}$ the *periodized interval*

$$I_{a,b}^{(j)} \coloneqq \{a \bmod n, (a+1) \bmod n, \ldots, b \bmod n\} \subseteq \{0, \ldots, n-1\}.$$

Periodized intervals allow us to consider an index set, corresponding to vector entries, that is wrapped periodically around the boundary of the vector as a single set instead of as two separated sets. With their help we can now define the notion of one-block supports.

Definition 5.2 Let $N = 2^{J-1}$ with $J \geq 2$ and $\mathbf{x} = (x_k)_{k=0}^{N-1} \in \mathbb{R}^N$. Then $\mathbf{x}$ has a *one-block support of length* m if m is the minimal integer such that

$$x_k = 0 \qquad \forall\, k \notin I_{\mu^{\mathbf{x}},\nu^{\mathbf{x}}}^{(J-1)} = \{\mu^{\mathbf{x}}, (\mu^{\mathbf{x}}+1) \bmod N, \ldots, \nu^{\mathbf{x}}\},$$

for some $\mu^{\mathbf{x}} \in \{0, \ldots, N-1\}$ and $\nu^{\mathbf{x}} \coloneqq (\mu^{\mathbf{x}} + m - 1) \bmod N$.

The interval $S^{\mathbf{x}} \coloneqq I_{\mu^{\mathbf{x}},\nu^{\mathbf{x}}}^{(J-1)}$ is called the *support interval*, $\mu^{\mathbf{x}}$ the *first support index* and $\nu^{\mathbf{x}}$ the *last support index* of $\mathbf{x}$.

Remark 5.3 Recall that in Definition 2.23 we already defined the notion of a first support index. However, in Section 2.4, we knew that $\widehat{\mathbf{a}^M}$ had a short support of length at most B if $M > 2B$, so we used B in the definition of the first support index. In this chapter, though, we require a definition for which neither the support length m nor the first support index $\mu^{\mathbf{x}}$ are known a priori; hence, Definition 5.2 is slightly different from Definition 2.23. In both cases we allow the support to be periodically wrapped around the boundary of the vector, i.e., around $N-1$ and 0 in this chapter, and around $\lfloor \frac{M}{2} \rfloor$

and $-\left\lceil \frac{M}{2} \right\rceil + 1$ in Section 2.4. Note that in the definition of a short support which we will give in Section 6.1 and which we will use throughout Chapter 6, the support will not be considered periodically. To be able to differentiate between these two cases we will call the support *one-block support* if we allow it to be wrapped periodically around the boundary of the vector, and *short support* if this is not the case in Chapters 5 and 6. According to this nomenclature, the vectors $\mathbf{a}^M$ in Section 2.4 would actually have a one-block support. However, as short support is the commonly used term and we only introduced the name *one-block support* to avoid confusion with respect to the closely related concepts used in Chapters 5 and 6, we decided to use the word *short support* in Chapters 2 and 3, since there are no ambiguities there.

The support interval $S^{\mathbf{x}}$ contains all indices at which $\mathbf{x}$ has nonzero entries. Since for some of the theoretical concepts used hereafter we require the support of $\mathbf{x}$ to be a periodized interval in $\mathbb{N}_0$, some of the indices in $S^{\mathbf{x}}$ may correspond to zero entries of $\mathbf{x}$.

By definition, the support length m of $\mathbf{x} \in \mathbb{R}^N$ is uniquely determined, but the first support index is not unique if $m > N/2$. Consider for example the vector $\mathbf{x} \in \mathbb{R}^8$ with

$$\mathbf{x} = (1, 0, 0, 0, 1, 0, 0, 0)^T.$$

Then $\mathbf{x}$ has a short support of length $m = 5$, but, according to Definition 5.2, we can choose the first support index $\mu^{\mathbf{x}}$ to be either 0 or 4, resulting in the support intervals $S^{\mathbf{x}} = I_{0,4}^{(3)}$ or $S^{\mathbf{x}} = I_{4,0}^{(3)}$, respectively. For $m \leq \frac{N}{2}$, though, the first support index is always uniquely determined. ◊

5.2 DCT-II via FFT

Our aim in this chapter is to develop a fast algorithm that deterministically recovers a vector $\mathbf{x} \in \mathbb{R}^N$, $N = 2^{J-1}$, from its DCT-II transformed vector $\mathbf{x}^{\widehat{\mathrm{II}}} \in \mathbb{R}^N$ if we know a priori that $\mathbf{x}$ has a one-block support of some unknown length. In our newly introduced notation this means that $\mathbf{x}$ has the support set

$$S^{\mathbf{x}} = I_{\mu^{\mathbf{x}},\nu^{\mathbf{x}}}^{(J-1)} = \{\mu^{\mathbf{x}}, (\mu^{\mathbf{x}} + 1) \bmod N, \ldots, \nu^{\mathbf{x}}\}$$

for some $\mu^{\mathbf{x}}, \nu^{\mathbf{x}} \in I_{0,N-1}^{(J-1)}$ and the support length $m = (\nu^{\mathbf{x}} - \mu^{\mathbf{x}} + 1) \bmod N$. We do not require any a priori knowledge of m in this chapter.

Apart from the fast DCT-II algorithm using a factorization of $\mathbf{C}_N^{\mathrm{II}}$ into real invertible matrices and the divide-and-conquer method, which we explained briefly in Section 4.2, one can also obtain a fast DCT-II algorithm using the FFT. More precisely, the DCT-II of any vector $\mathbf{x} \in \mathbb{R}^N$ can be computed from the DFT of the auxiliary vector $\mathbf{y} \in \mathbb{R}^{2N}$ of double length, where

$$\mathbf{y} \coloneqq \left(\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T\right)^T = \big(\underbrace{x_0, x_1, \ldots, x_{N-1}}_{=\mathbf{x}^T}, \underbrace{x_{N-1}, x_{N-2}, \ldots, x_0}_{=(\mathbf{J}_N \mathbf{x})^T}\big)^T,$$

i.e., the first half of the vector $\mathbf{y}$ is $\mathbf{x}$ and the second half of $\mathbf{y}$ is the reflection of $\mathbf{x}$, $\mathbf{J}_N \mathbf{x}$. As in Theorem 4.4, $\mathbf{J}_N$ denotes the counter identity matrix of size $N \times N$. Equivalently, we can write that

$$y_k \coloneqq \begin{cases} x_k & \text{if } k \in \{0, \ldots, N-1\}, \\ x_{N-1-k} & \text{if } k \in \{N, \ldots, 2N-1\}. \end{cases} \tag{5.1}$$

The following lemma shows the close relation between $\mathbf{x}^{\widehat{\mathrm{II}}}$ and $\widehat{\mathbf{y}}$, namely that $\mathbf{x}^{\widehat{\mathrm{II}}}$ can be computed from $\widehat{\mathbf{y}}$ and vice versa, see also [PPST19], Section 6.3.1.

Lemma 5.4 (Lemma 1 in [BP18c]) Let $N \in \mathbb{N}$, $\mathbf{x} \in \mathbb{R}^N$ and $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T \in \mathbb{R}^{2N}$.

(i) $\mathbf{y}$ is symmetric, i.e.,

$$\mathbf{y} = \mathbf{J}_{2N}\mathbf{y}.$$

(ii) $\mathbf{x}^{\widehat{\mathrm{II}}} = \left(x_k^{\widehat{\mathrm{II}}}\right)_{k=0}^{N-1} = \mathbf{C}_N^{\mathrm{II}}\mathbf{x}$ is given by

$$x_k^{\widehat{\mathrm{II}}} = \frac{\varepsilon_N(k)}{\sqrt{2N}}{\omega_{4N}}^k \cdot \widehat{y}_k \qquad \forall\, k \in \{0, \dots, N-1\},$$

where $\widehat{\mathbf{y}} = (\widehat{y}_k)_{k=0}^{2N-1} = \mathbf{F}_{2N}\mathbf{y}$.

(iii) $\widehat{\mathbf{y}}$ is completely determined by $\mathbf{x}^{\widehat{\mathrm{II}}}$ via

$$\widehat{y}_k = \begin{cases} \frac{\sqrt{2N}}{\varepsilon_N(k)}{\omega_{4N}}^{-k} \cdot x_k^{\widehat{\mathrm{II}}} & \text{if } k \in \{0, \dots, N-1\}, \\ 0 & \text{if } k = N, \\ -\frac{\sqrt{2N}}{\varepsilon_N(2N-k)}{\omega_{4N}}^{-k} \cdot x_{2N-k}^{\widehat{\mathrm{II}}} & \text{if } k \in \{N+1, \dots, 2N-1\}. \end{cases}$$

Proof. (i) Note that $\mathbf{y}$ is symmetric by construction, since

$$\mathbf{J}_{2N}\mathbf{y} = \mathbf{J}_{2N}\begin{pmatrix} \mathbf{x} \\ \mathbf{J}_N\mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{J}_N\mathbf{x} \end{pmatrix} = \mathbf{y}.$$

(ii) Let $k \in \{0, \dots, N-1\}$. Using that for all $x \in \mathbb{R}$

$$\cos(x) = \frac{e^{ix} + e^{-ix}}{2},$$

we find that

$$\begin{aligned} x_k^{\widehat{\mathrm{II}}} &= \sqrt{\frac{2}{N}}\varepsilon_N(k) \sum_{l=0}^{N-1} \cos\left(\frac{2 \cdot k(2l+1)\pi}{2 \cdot 2N}\right) x_l \\ &= \frac{\varepsilon_N(k)}{\sqrt{2N}} \sum_{l=0}^{N-1} \left({\omega_{4N}}^{k(2l+1)} + {\omega_{4N}}^{-k(2l+2-1)}\right) x_l \\ &= \frac{\varepsilon_N(k)}{\sqrt{2N}} \sum_{l=0}^{N-1} \left({\omega_{4N}}^{2kl}{\omega_{4N}}^{k} + {\omega_{4N}}^{2k(-l-1)}{\omega_{4N}}^{k}\right) x_l \\ &= \frac{\varepsilon_N(k)}{\sqrt{2N}}{\omega_{4N}}^{k} \sum_{l=0}^{N-1} \left({\omega_{2N}}^{kl} + {\omega_{2N}}^{k(2N-1-l)}\right) x_l \\ &= \frac{\varepsilon_N(k)}{\sqrt{2N}}{\omega_{4N}}^{k} \left(\sum_{l=0}^{N-1} {\omega_{2N}}^{kl} x_l + \sum_{l'=N}^{2N-1} {\omega_{2N}}^{kl'} x_{2N-1-l'}\right) \\ &= \frac{\varepsilon_N(k)}{\sqrt{2N}}{\omega_{4N}}^{k} \cdot \widehat{y}_k, \end{aligned}$$

where we set $l' = 2N - 1 - l$ for $l \in \{0, \dots, N-1\}$.

(iii) If $k \in \{0, \dots, N-1\}$, the claim in (iii) follows directly from (ii). For any index $k \in \{N, \dots, 2N-1\}$ the symmetry of $\mathbf{y}$ guaranteed by (i) implies that

$$\begin{aligned}
\widehat{y}_k = \left(\widehat{\mathbf{J}_{2N}\mathbf{y}}\right)_k &= \sum_{l'=0}^{2N-1} {\omega_{2N}}^{kl'} y_{2N-1-l'} \\
&= \sum_{l=0}^{2N-1} {\omega_{2N}}^{k(2N-1-l)} y_l \\
&= \sum_{l=0}^{2N-1} {\omega_{2N}}^{-k(l+1)} y_l \\
&= {\omega_{2N}}^{-k} \sum_{l=0}^{2N-1} {\omega_{2N}}^{-kl} y_l, \qquad (5.2)
\end{aligned}$$

where we set $l \coloneqq 2N - 1 - l'$ for $l' \in \{0, \dots, 2N-1\}$. If $k = N$, (5.2) yields

$$\begin{aligned}
\widehat{y}_N &= {\omega_{2N}}^{-N} \sum_{l=0}^{2N-1} {\omega_{2N}}^{-Nl} y_l \\
&= -\sum_{l=0}^{2N-1} {\omega_{2N}}^{Nl} y_l \\
&= -\widehat{y}_N,
\end{aligned}$$

so it follows that $\widehat{y}_N = 0$. If $k \in \{N+1, \dots, 2N-1\}$, then $2N - k \in \{1, \dots, N-1\}$, and we obtain from (5.2) that

$$\begin{aligned}
\widehat{y}_k &= {\omega_{2N}}^{-k} \sum_{l=0}^{2N-1} {\omega_{2N}}^{-kl} y_l \\
&= {\omega_{2N}}^{-k} \sum_{l=0}^{2N-1} {\omega_{2N}}^{(2N-k)l} y_l \\
&= {\omega_{2N}}^{-k} \cdot \widehat{y}_{2N-k} \\
&= -\frac{\sqrt{2N}}{\varepsilon_N(2N-k)} {\omega_{4N}}^{-k} \cdot x^{\widehat{\mathrm{II}}}_{2N-k}.
\end{aligned}$$

□

Lemma 5.4 implies that we can compute $\widehat{\mathbf{y}}$ from $\mathbf{x}^{\widehat{\mathrm{II}}}$ in $\mathcal{O}(N)$ time, and that each entry of $\widehat{\mathbf{y}}$ depends on only one entry of $\mathbf{x}^{\widehat{\mathrm{II}}}$. Thus, the problem of reconstructing $\mathbf{x} \in \mathbb{R}^N$ from $\mathbf{x}^{\widehat{\mathrm{II}}}$ can be transferred to recovering the vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T \in \mathbb{R}^{2N}$ from $\widehat{\mathbf{y}}$. If $\mathbf{x}$ has a one-block support of length m, we can apply any deterministic $2m$-sparse IDFT algorithm for recovering $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T \in \mathbb{R}^{2N}$, which also gives us $\mathbf{x}$. However, if such an algorithm required every entry of $\widehat{\mathbf{y}}$, we would not obtain an algorithm for recovering $\mathbf{x}$ from $\mathbf{x}^{\widehat{\mathrm{II}}}$ via IDFTs whose runtime is sublinear in the vector length N. As each entry of $\widehat{\mathbf{y}}$ depends on only one entry of $\mathbf{x}^{\widehat{\mathrm{II}}}$, we can achieve a sublinear runtime if the sampling complexity of the employed IDFT algorithm is sublinear in N.

As already mentioned in Chapter 2, there exist fast algorithms for recovering a sparse vector $\mathbf{y}$ from $\widehat{\mathbf{y}}$. However, to the best of our knowledge, all algorithms for arbitrary sparsity have a runtime that is at least quadratic in the sparsity. We provided an incomplete list of such methods at the beginning of Chapter 2. Actually, with $\mathbf{x}$ having a one-block support of length m, the associated vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T \in \mathbb{R}^{2N}$ is $2m$-sparse with a very special structure, so we can employ this knowledge to achieve a sublinear runtime for our algorithm.

Definition 5.5 (Reflected Block Support) Let $N = 2^{J-1}$ with $J \geq 2$ and $\mathbf{x} \in \mathbb{R}^N$ have a one-block support of length $m < N$. Then we say that the vector

$$\mathbf{y} \coloneqq \left(\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T\right)^T \in \mathbb{R}^{2N}$$

has a *reflected block support*.

Remark 5.6 Note that the vector $\mathbf{y}$ does not necessarily have a support consisting of two blocks of length m. Under certain conditions it is also possible that $\mathbf{y}$ possesses two support blocks of different lengths or just a single support block of length $2m$. We will discuss the support structure of $\mathbf{y}$ in more detail in Section 5.4.1. If $\mathbf{y}$ has only a single support block, we will denote its length by $m^{(J)}$ for algorithmic purposes. If $\mathbf{y}$ has two support blocks, we will refer to their length as $n^{(J)}$ if both blocks have the same length, and as $n_{(0)}^{(J)}$ and $n_{(1)}^{(J)}$ otherwise. The support set of $\mathbf{y}$ will be denoted by $S^{(J)} \subseteq I_{0,2^J-1}^{(J)}$. See Remark 5.13 for additional information about the support of $\mathbf{y}$. ♢

By utilizing the a priori known information about the support structure of $\mathbf{y}$, we hope to obtain an algorithm for reconstructing $\mathbf{y}$ from $\widehat{\mathbf{y}}$ that is faster than all previously existing deterministic methods for general $2m$-sparse vectors. Note that even though $\mathbf{y}$ has a sparsity of $2m$, it is, due to its symmetry guaranteed by Lemma 5.4 (i), already completely determined by the m nonzero entries in its first half.

Consequently, our aim in this chapter is to first develop a deterministic IDFT algorithm for recovering a vector with reflected block support from its DFT transformed vector. If we can then show that this algorithm has runtime and sampling complexities that are subquadratic in the block length and sublinear in the vector length, this will also give us a deterministic IDCT-II algorithm for recovering a vector with one-block support from its DCT-II transformed vector with the same runtime and sampling requirements.

5.3 IDFT Methods for Vectors with One-Block Support

The IDFT algorithm for vectors with reflected block support which we will introduce hereafter is obtained by extending recent methods in [PW16a, PW17a] for the reconstruction of vectors with one-block support from their DFT to our setting. Therefore, we will give a brief overview of the techniques used in said papers which we will also employ.

Both Algorithm 2 in [PW16a] and Algorithm 2.1 in [PW17a] rely on an approach that iteratively recovers certain shorter periodizations of $\mathbf{y}$. The following definition lies at the heart of the two papers.

Definition 5.7 ((2.1) in [PW16a], (1.1) in [PW17a]) For any vector $\mathbf{x} \in \mathbb{R}^{2^j}$ we denote by

$$\mathbf{x}_{(0)} \coloneqq (x_k)_{k=0}^{2^{j-1}-1} \in \mathbb{R}^{2^{j-1}} \quad \text{and} \quad \mathbf{x}_{(1)} \coloneqq (x_k)_{k=2^{j-1}}^{2^j-1} \in \mathbb{R}^{2^{j-1}}$$

the *first and second half* of $\mathbf{x}$, respectively, i.e., $\mathbf{x} = \left(\mathbf{x}_{(0)}^T, \mathbf{x}_{(1)}^T\right)^T$.

Let $N = 2^{J-1}$ with $J \geq 2$ and $\mathbf{y} \in \mathbb{R}^{2N}$. We set $\mathbf{y}^{(J)} \coloneqq \mathbf{y}$. For $j \in \{0, \ldots, J-1\}$ we iteratively define the *periodization* $\mathbf{y}^{(j)} \in \mathbb{R}^{2^j}$ of $\mathbf{y}$ as

$$\mathbf{y}^{(j)} \coloneqq \mathbf{y}_{(0)}^{(j+1)} + \mathbf{y}_{(1)}^{(j+1)} = \left(y_k^{(j+1)} + y_{k+2^j}^{(j+1)}\right)_{k=0}^{2^j-1}.$$

By definition, for any $j \in \{0, \ldots, J-1\}$ the periodization $\mathbf{y}^{(j)} \in \mathbb{R}^{2^j}$ is given by adding the first and the second half of the periodization $\mathbf{y}^{(j+1)} \in \mathbb{R}^{2^{j+1}}$.

Example 5.8 Let $\mathbf{y} \in \mathbb{R}^{16}$ with nonzero entries y_1, y_2. Then its periodizations are

$$\begin{aligned}
\mathbf{y} = \mathbf{y}^{(4)} &= (0, y_1, y_2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T, \\
\mathbf{y}^{(3)} &= (0, y_1, y_2, 0, 0, 0, 0, 0)^T, \\
\mathbf{y}^{(2)} &= (0, y_1, y_2, 0)^T, \\
\mathbf{y}^{(1)} &= (y_2, y_1)^T, \\
\mathbf{y}^{(0)} &= (y_1 + y_2)^T.
\end{aligned}$$

◊

The periodization has several useful properties, some of which are summarized in the following lemma. Claim (iii) was shown in Lemma 2 in [BP18c].

Lemma 5.9 Let $N = 2^{J-1}$ with $J \geq 2$ and $j \in \{0, \ldots, J-1\}$. Let $\mathbf{y} \in \mathbb{R}^{2N}$. Then the following statements are true:

(i) $\mathbf{y}^{(0)} = \sum_{k=0}^{2N-1} y_k$

(ii) $\mathbf{y}^{(j)} = \left(\sum_{l=0}^{2^{J-j}-1} y_{k+2^j l}\right)_{k=0}^{2^j-1}$

(iii) If $\mathbf{y}$ is symmetric, then $\mathbf{y}^{(j)}$ is symmetric as well, i.e., $\mathbf{y}^{(j)} = \mathbf{J}_{2^j}\mathbf{y}^{(j)}$.

Proof. For a proof of (i) and (ii) see, e.g., [PW16a], Section 2, and [PW17a], Section 2.

(iii) Since $\mathbf{y}$ is symmetric by Lemma 5.4 (i), we have that $y_k = y_{2^J-1-k}$ for all indices $k \in \{0, \ldots, 2^J - 1\}$. For the entries of $\mathbf{y}^{(J-1)}$ it follows that

$$\begin{aligned}
y_k^{(J-1)} &= y_k + y_{k+2^{J-1}} \\
&= y_{2^J-1-k} + y_{2^J-1-(k+2^{J-1})} \\
&= y_{2^{J-1}-1-k+2^{J-1}} + y_{2^{J-1}-1-k} \\
&= y_{2^{J-1}-1-k}^{(J-1)}
\end{aligned}$$

for all $k \in \{0, \ldots, 2^{J-1} - 1\}$; thus, $\mathbf{y}^{(J-1)} = \mathbf{J}_{2^{J-1}}\mathbf{y}^{(J-1)}$. Now we assume that $\mathbf{y}^{(j+1)}$ is symmetric for $j \in \{0, \ldots, J-1\}$ and show that $\mathbf{y}^{(j)}$ is symmetric as well. Since $\mathbf{y}^{(j+1)}$

is symmetric, it follows that $y_k^{(j+1)} = y_{2^{j+1}-1-k}^{(j+1)}$ for all $k \in \{0, \dots, 2^j - 1\}$, and hence

$$\begin{aligned} y_k^{(j)} &= y_k^{(j+1)} + y_{k+2^j}^{(j+1)} \\ &= y_{2^{j+1}-1-k}^{(j+1)} + y_{2^{j+1}-1-(k+2^j)}^{(j+1)} \\ &= y_{2^j-1-k}^{(j)}, \end{aligned}$$

which proves the claim. □

The reason why these periodizations are so interesting for the development of fast IDFT algorithms is that for $j \in \{0, \dots, J-1\}$ the DFT of the periodization $\mathbf{y}^{(j)}$ of any $\mathbf{y} \in \mathbb{R}^{2N}$ is already completely determined by the DFT of $\mathbf{y}$ itself, as the following lemma, which was proved in [PW16a], shows.

Lemma 5.10 (Lemma 2.1 in [PW16a]) Let $N = 2^{J-1}$ with $J \geq 2$ and $j \in \{0, \dots, J\}$. Let $\mathbf{y} \in \mathbb{R}^{2N}$. Then $\widehat{\mathbf{y}^{(j)}}$ satisfies

$$\widehat{\mathbf{y}^{(j)}} = (\widehat{y}_{2^{J-j}k})_{k=0}^{2^j-1}.$$

There exist several sparse IFFT methods based on the notion of periodizations from Definition 5.7, e.g., [PW17a, PW16a, PWCW18]. We want to explain these algorithms in more detail, as some of the concepts used for them are also utilized for the sparse IDCT-II algorithms we will present in this chapter and in Chapter 6.

Algorithm 2 in [PW16a]

Let $\mathbf{y} \in \mathbb{C}^N$, $N = 2^J$, have a one-block support of length m. For Algorithm 2 in [PW16a] the authors additionally suppose that an upper bound $M \geq m$ on the support length of $\mathbf{y}$ is known. Then one can show that for any $j \in \{L+1, \dots, J\}$, where L is chosen such that $2^{L-1} < m \leq 2^L$, the periodization $\mathbf{y}^{(j)}$ of $\mathbf{y}$ has already a one-block support of length m. Further, each nonzero entry of $\mathbf{y}$ corresponds to exactly one nonzero entry of $\mathbf{y}^{(j)}$ with the same value, so the nonzero entries of $\mathbf{y}$ and $\mathbf{y}^{(j)}$ are the same. The only remaining unknown parameter is the first support index $\mu^{(j)}$ of $\mathbf{y}^{(j)}$, which can in fact be obtained in a fast way from the periodization $\mathbf{y}^{(j-1)}$ of half length and $\widehat{\mathbf{y}}$.

The main idea of the algorithm is thus to iteratively recover $\mathbf{y}$ from $\widehat{\mathbf{y}}$ and the periodizations $\mathbf{y}^{(L+1)}, \mathbf{y}^{(L+2)}, \dots, \mathbf{y}^{(J)} = \mathbf{y}$, using for the reconstruction of $\mathbf{y}^{(j+1)}$ that $\mathbf{y}^{(j)}$ is already known from the previous iteration step. Since

$$\widehat{\mathbf{y}^{(L+1)}} = (\widehat{y}_{2^{J-L-1}k})_{k=0}^{2^j-1}$$

by Lemma 5.10, the initial vector $\mathbf{y}^{(L+1)}$ can be computed directly from $\widehat{\mathbf{y}}$ via an IFFT of length 2^{L+1}. For $j \in \{L+1, \dots, J-1\}$ it was shown in [PW16a], Section 4, that if $\mathbf{y}^{(j)}$ has the one-block support $S^{(j)} = I_{\mu^{(j)},\,\nu^{(j)}}^{(j)}$ of length $m^{(j)}$, there are only two possibilities for $\mathbf{y}^{(j+1)}$. More precisely, $\mathbf{y}^{(j+1)}$ has either the support interval $S^{(j+1)} = S^{(j)} = I_{\mu^{(j)},\,\nu^{(j)}}^{(j+1)}$ or $S^{(j+1)} = I_{\mu^{(j)}+2^j,\,\nu^{(j)}+2^j}^{(j+1)}$. Denoting the first possible vector by $\mathbf{u}^0$ and the second possible vector by $\mathbf{u}^1$, we find that $\mathbf{u}^1$ is the 2^j-shift of $\mathbf{u}^0$.

Theorem 4.2 in [PW16a] proves that $\mathbf{y}^{(j+1)}$ can be uniquely recovered from $\mathbf{y}^{(j)}$ and

one oddly indexed nonzero entry of $\widehat{\mathbf{y}^{(j+1)}}$, using that

$$\widehat{u^0}_{2k+1} = -\widehat{u^1}_{2k+1}$$

for all $k \in \{0, \dots, 2^j - 1\}$. Comparing $\widehat{u^0}_{2k_0+1}$ and $\widehat{u^1}_{2k_0+1}$ at an index $2k_0 + 1$ corresponding to an oddly indexed nonzero entry $\widehat{y^{(j+1)}}_{2k_0+1} \neq 0$ then allows for the detection of $\mu^{(j+1)}$, since $\mathbf{y}^{(j+1)} = \mathbf{u}^0$ if $\widehat{u^0}_{2k_0+1} = \widehat{y^{(j+1)}}_{2k_0+1}$, and $\mathbf{y}^{(j+1)} = \mathbf{u}^1$ otherwise.

Employing some further stabilization techniques for noisy data and an efficient method for determining the required oddly indexed nonzero entry of $\widehat{\mathbf{y}^{(j+1)}}$ yields a deterministic IFFT algorithm for recovering a vector $\mathbf{y} \in \mathbb{C}^N$, $N = 2^J$, with one-block support of length m from $\widehat{\mathbf{y}}$ and an upper bound M on m in

$$\mathcal{O}(M \log N)$$

time and using

$$\mathcal{O}(M \log N)$$

samples of $\widehat{\mathbf{y}}$. Note that the runtime and sampling complexities are sublinear in the vector length N and even subquadratic in the bound M on the support length.

Algorithm 2.1 in [PW17a]

As seen above, Algorithm 2 in [PW16a] requires a priori knowledge of an upper bound M on the support length m of the sought-after vector $\mathbf{y} \in \mathbb{C}^N$ with one-block support. In [PW17a], on the other hand, an adaptive algorithm which can detect the support length on the fly is presented. However, this approach now requires the entries of the vector $\mathbf{y}$ to be real and nonnegative, as otherwise there might be cancellations of entries corresponding to nonzero entries of $\mathbf{y}$ in some of the periodizations. Again, the algorithm will recover $\mathbf{y}$ iteratively from $\widehat{\mathbf{y}}$ and the periodizations $\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(J)} = \mathbf{y}$, using that for all $j \in \{0, \dots, J\}$ the periodization $\mathbf{y}^{(j)} \in \mathbb{R}^{2^j}$ has a one-block support of length $m^{(j)} \leq m$.

For the initial vector we have that $\mathbf{y}^{(0)} = \widehat{y}_0$. In each step the algorithm computes $\mathbf{y}^{(j+1)}$ via an IFFT of a vector of length $\mathcal{O}\left(m^{(j)}\right)$ instead of 2^j, restricting $\mathbf{y}^{(j+1)}$ to its $\mathcal{O}(m)$ possibly nonzero entries. It was shown in [PW17a], Section 2, that

$$\left(\widehat{y^{(j+1)}}_{2k+1}\right)_{k=0}^{2^j-1} = \mathbf{F}_{2^j} \cdot \operatorname{diag}\left(\left(\omega_{2^{j+1}}{}^l\right)_{l=0}^{2^j-1}\right) \cdot \left(2\mathbf{y}_{(0)}^{(j+1)} - \mathbf{y}^{(j)}\right). \tag{5.3}$$

As $\mathbf{y}^{(j)} = \mathbf{y}_{(0)}^{(j+1)} + \mathbf{y}_{(1)}^{(j+1)}$, (5.3) yields that $\mathbf{y}^{(j+1)}$ can be completely recovered essentially via an IFFT of length 2^j and using Lemma 5.10, since $\mathbf{y}^{(j)}$ is known from the previous iteration step. If $m^{(j)} > 2^{j-1}$, the one-block support of $\mathbf{y}^{(j)}$ and $\mathbf{y}^{(j+1)}$ cannot be used to speed up the computation. However, if $m^{(j)} \leq 2^{j-1}$, one can restrict $\mathbf{y}^{(j)}$ and $\mathbf{y}_{(0)}^{(j+1)}$ to vectors $\mathbf{z}^{(j)}, \mathbf{z}_{(0)}^{(j+1)}$ of length $2^{L^{(j)}}$, where $2^{L^{(j)}-1} < m^{(j)} \leq 2^{L^{(j)}}$, such that all possibly nonzero entries of $\mathbf{y}^{(j)}$ and $\mathbf{y}_{(0)}^{(j+1)}$ are taken into account. Then it suffices to recover these

shorter vectors. Restricting (5.3) to suitable rows, one obtains

$$\begin{aligned}&\left(\widehat{y^{(j+1)}}_{2k+1}\right)_{k=0}^{2^{L^{(j)}}-1}\\ &= \operatorname{diag}\left(\left(\omega_{2^{L^{(j)}}}{}^{\mu^{(j)}p}\right)_{p=0}^{2^{L^{(j)}-1}}\right)\cdot \mathbf{F}_{2^{L^{(j)}}}\cdot \operatorname{diag}\left(\left(\omega_{2^{j+1}}{}^{\left(\mu^{(j)}+r\right) \bmod 2^j}\right)_{r=0}^{2^{L^{(j)}}-1}\right)\\ &\cdot\left(\left(2\mathbf{y}_{(0)}^{(j+1)}-\mathbf{y}^{(j)}\right)_{\left(\mu^{(j)}+r\right) \bmod 2^j}\right)_{r=0}^{2^{L^{(j)}}-1},\end{aligned}$$

implying that periodization $\mathbf{y}^{(j+1)}$ can essentially be reconstructed via an IFFT of length $2^{L^{(j)}} = \mathcal{O}\left(m^{(j)}\right)$. Applying a stable technique for finding the first support index and the support length of $\mathbf{y}^{(j+1)}$ yields a deterministic IFFT algorithm for recovering a vector $\mathbf{y} \in \mathbb{R}^N_{\geq 0}$, $N = 2^J$, with one-block support of length m from $\widehat{\mathbf{y}}$ without a priori knowledge of m in

$$\mathcal{O}\left(m \log m \log \frac{N}{m}\right) \qquad \text{time and using} \qquad \mathcal{O}\left(m \log \frac{N}{m}\right)$$

samples of $\widehat{\mathbf{y}}$. Thus, for Algorithm 2.1 in [PW17a] both the runtime and the number of required samples are sublinear in the vector length N and subquadratic in the support length m, and similar to those of Algorithm 2 in [PW16a].

5.4 Support Structures of Periodizations

Our goal is to reconstruct a vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T \in \mathbb{R}^{2N}$, $N = 2^{J-1}$, with reflected block support from $\widehat{\mathbf{y}}$ by successively computing its periodizations $\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(J)} = \mathbf{y}$ without any a priori knowledge on the block length m. In the jth iteration step of the procedure we thus have to determine $\mathbf{y}^{(j+1)}$ from $\widehat{\mathbf{y}}$ efficiently, which can be done by employing the vector $\mathbf{y}^{(j)}$ known from the previous step. In order for this approach to work, we need to investigate how the support blocks of $\mathbf{y}^{(j+1)}$ can look like if the support of $\mathbf{y}^{(j)}$ is given.

For an iterative reconstruction procedure we need to guarantee that relevant information about the support of $\mathbf{y}$ is not canceled out in any of the periodized vectors $\mathbf{y}^{(j)}$ for $j \in \{0, \ldots, J-1\}$. More precisely, we require that for any nonzero entry $y_k \neq 0$ of $\mathbf{y}$ the entries of $\mathbf{y}^{(j)}$ which depend on y_k do not vanish. Otherwise we cannot hope to be able to correctly recover the support of $\mathbf{y}$. Formally, the periodizations have to satisfy

$$y^{(j)}_{k \bmod 2^j} = \sum_{l=0}^{2^j-1} y_{k+2^j l} \neq 0 \qquad \forall\, j \in \{0, \ldots, J\} \tag{5.4}$$

for any $y_k \neq 0$ with $k \in \{0, \ldots, 2N-1\}$. This assumption holds for example if all nonzero entries of $\mathbf{x}$, and thus of $\mathbf{y}$, are positive or if all nonzero entries are negative, i.e., if $\mathbf{x} \in \mathbb{R}^N_{\geq 0}$ or $\mathbf{x} \in \mathbb{R}^N_{\leq 0}$.

In practical applications for our algorithm the given data will usually not be exact. Hence, for noisy data and given a threshold $\varepsilon > 0$ depending on the noise level, we have to ensure that

$$\left|y^{(j)}_{k \bmod 2^j}\right| > \varepsilon \qquad \forall\, j \in \{0, \ldots, J\}$$

for all significantly large entries $|y_k| > \varepsilon$ of $\mathbf{y}$ with $k \in \{0, \ldots, 2N-1\}$.

5.4.1 Support Structure of $\mathbf{y} = \mathbf{y}^{(J)}$ for Given $\mathbf{x}$

We are interested in vectors $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T \in \mathbb{R}^{2N}$ arising from a vector $\mathbf{x} \in \mathbb{R}^N$ with one-block support of length m. Consequently, the structure of $\mathbf{y}$ also has important characteristics, which we want to inspect in this section. For better illustration we begin by motivating the main possible support structures by three different examples. These examples will be utilized for demonstrating the support structures of the periodized vectors $\mathbf{y}^{(j)}$, $j \in \{0, \dots, J-1\}$, as well.

Example 5.11

1. Let $\mathbf{x} = (0, x_1, x_2, 0, 0, 0, 0, 0)^T \in \mathbb{R}^8$ with nonzero entries x_1, x_2, i.e., with one-block support $S^{\mathbf{x}} = I^{(3)}_{1,2}$ of length $m = 2$. Then we find that

$$\mathbf{y} = \mathbf{y}^{(4)} = (0, x_1, x_2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x_2, x_1, 0)^T .$$

Consequently, $\mathbf{y}$ has the reflected block support $S^{(4)} = I^{(4)}_{1,2} \cup I^{(4)}_{13,14}$ with two blocks of length $n^{(4)} \coloneqq 2 = m$ each.

2. Let $\mathbf{x} = (x_0, 0, 0, 0, 0, 0, 0, 0)^T \in \mathbb{R}^8$ with nonzero entry x_0, i.e., with one-block support $S^{\mathbf{x}} = I^{(3)}_{0,0}$ of length $m = 1$. Then we obtain that

$$\mathbf{y} = \mathbf{y}^{(4)} = (x_0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x_0)^T ,$$

so $\mathbf{y}$ has the reflected block support $S^{(4)} = I^{(4)}_{15,0}$ with one block of length $m^{(4)} \coloneqq 2$.

3. Let $\mathbf{x} = (x_0, x_1, 0, 0, 0, 0, 0, x_7)^T \in \mathbb{R}^8$ with nonzero entries x_0, x_1 and x_7, i.e., with one-block support $S^{\mathbf{x}} = I^{(3)}_{7,1}$ of length $m = 3$. Then we have that

$$\mathbf{y} = \mathbf{y}^{(4)} = (x_0, x_1, 0, 0, 0, 0, 0, x_7, x_7, 0, 0, 0, 0, 0, x_1, x_0)^T$$

has the reflected block support $S^{(4)} = I^{(4)}_{7,8} \cup I^{(4)}_{14,1}$ with block lengths $n^{(4)}_{(0)} \coloneqq 2$ and $n^{(4)}_{(1)} \coloneqq 4$. ◊

It follows from Example 5.11 that, unless the support of $\mathbf{x}$ includes the first or last entry of the vector, $\mathbf{y}$ has a reflected block support consisting of two blocks of the same length. These findings motivate the following lemma.

Lemma 5.12 (Lemma 4 in [BP18c]) Let $N = 2^{J-1}$ with $J \geq 2$ and $\mathbf{x} \in \mathbb{R}^N$ have the one-block support $S^{\mathbf{x}} = I^{(J-1)}_{\mu^{\mathbf{x}}, \nu^{\mathbf{x}}}$ of length $m < N$. Set $\mathbf{y} = \mathbf{y}^{(J)} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T \in \mathbb{R}^{2N}$ and assume that $\mathbf{y}$ satisfies (5.4).

i) If $\mu^{\mathbf{x}} \leq \nu^{\mathbf{x}}$, then $\mathbf{y}$ possesses the (reflected) two-block support

$$S^{(J)} = I^{(J)}_{\mu^{\mathbf{x}}, \nu^{\mathbf{x}}} \cup I^{(J)}_{2N-1-\nu^{\mathbf{x}}, 2N-1-\mu^{\mathbf{x}}}.$$

In the special cases $\mu^{\mathbf{x}} = 0$ or $\nu^{\mathbf{x}} = N-1$ the two support blocks are adjacent and form a (reflected) one-block support.

ii) If $\mu^{\mathbf{x}} > \nu^{\mathbf{x}}$, then $\mathbf{y}$ has the (reflected) two-block support

$$S^{(J)} = I^{(J)}_{\mu^{\mathbf{x}}, 2N-1-\mu^{\mathbf{x}}} \cup I^{(J)}_{2N-1-\nu^{\mathbf{x}}, \nu^{\mathbf{x}}}.$$

Proof. The proof of Lemma 5.12 follows directly from the definition of $\mathbf{y}$. Figures 5.1 and 5.2 illustrate the two cases. □

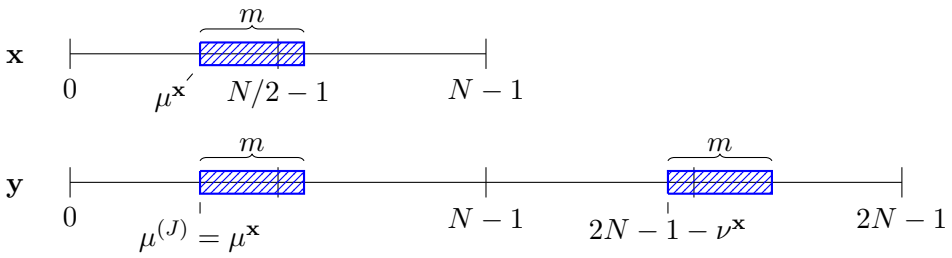

Figure 5.1: Illustration of the support of $\mathbf{y}$ for given $\mathbf{x}$ according to Lemma 5.12 case i)

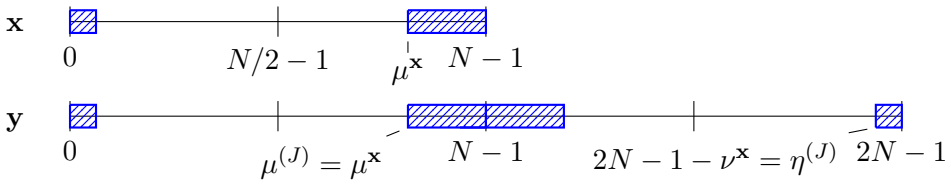

Figure 5.2: Illustration of the support of $\mathbf{y}$ for given $\mathbf{x}$ according to Lemma 5.12 case ii)

Remark 5.13 (Remark 1 in [BP18c])

i) In case i) of Lemma 5.12, if $\mu^{\mathbf{x}} \neq 0$ and $\nu^{\mathbf{x}} \neq N-1$, the two support blocks of $\mathbf{y} = \mathbf{y}^{(J)}$ have the same length $\nu^{\mathbf{x}} - \mu^{\mathbf{x}} + 1 = m$, as in Example 5.11.1. For algorithmic purposes we denote the *length of the support blocks* by $n^{(J)} := m$. As the blocks are separated, we denote by $\mu^{(J)}$ and $\nu^{(J)}$ be the *first and last index of the first support block*. The *support set* of $\mathbf{y}$, which is always the union of two intervals, is then referred to as

$$S^{(J)} = I_{\mu^{(J)},\nu^{(J)}} \cup I_{2^{j+1}-1-\nu^{(J)},2^{j+1}-1-\mu^{(J)}}.$$

If $\mu^{\mathbf{x}} = 0$ or $\nu^{\mathbf{x}} = N-1$, i.e., if the support blocks of $\mathbf{x}$ and $\mathbf{J}_N\mathbf{x}$ are (periodically) adjacent, $\mathbf{y}$ has a one-block support of *length* $m^{(J)} := 2m$. For $\mu^{\mathbf{x}} = 0$ it is of the form

$$S^{(J)} = I^{(J)}_{2N-1-\nu^{\mathbf{x}},\nu^{\mathbf{x}}},$$

see Example 5.11.2, and for $\nu^{\mathbf{x}} = N-1$ it is of the form

$$S^{(J)} = I^{(J)}_{\mu^{\mathbf{x}},2N-1-\mu^{\mathbf{x}}}.$$

Here, we denote the *first and last support indices* of $\mathbf{y}$ by $\mu^{(J)}$ and $\nu^{(J)}$, respectively.

ii) In case ii) of Lemma 5.12, $\mu^{\mathbf{x}}$ cannot be equal to $\nu^{\mathbf{x}}+1$, since $m < N$. This implies that the two support blocks of $\mathbf{y}$ are indeed always separated. For algorithmic purposes we then denote the *first index of the block centered around the middle of the vector* by $\mu^{(J)}$ and the *first index of the block centered around the boundary of the vector* by

$\eta^{(J)}$. The blocks have the possibly different lengths $2(N - \mu^{\mathbf{x}})$ and $2(\nu^{\mathbf{x}} + 1)$, with $2m = 2\left(N - \mu^{\mathbf{x}} + \nu^{\mathbf{x}} + 1\right)$, as in Example 5.11.3. ◊

5.4.2 Support Structure of $\mathbf{y}^{(j)}$ for Given $\mathbf{y}$

We are interested in iteratively reconstructing the vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T \in \mathbb{R}^{2N}$, which we know by Lemma 5.12 to have a reflected block support with two support blocks, except for special cases $\mu^{\mathbf{x}} = 0$ and $\nu^{\mathbf{x}} = N - 1$, from its periodizations $\mathbf{y}^{(j)} \in \mathbb{R}^{2^j}$ for $j \in \{0, \ldots, J-1\}$. Thus, we will now investigate the implications of the above obtained knowledge about the support of $\mathbf{y}$ on the support of $\mathbf{y}^{(j)}$. Again, we will motivate the claims in the next lemma by first looking at exemplary vectors and their periodizations. In fact, we will utilize the same three vectors as in Example 5.11.

Example 5.14 (Example 5.11 continued)

1. Consider again $\mathbf{x} = (0, x_1, x_2, 0, 0, 0, 0, 0)^T \in \mathbb{R}^8$. Then $\mathbf{y}$ and its periodizations are

$$\begin{aligned}
\mathbf{y} = \mathbf{y}^{(4)} &= (0, x_1, x_2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x_2, x_1, 0)^T, \\
\mathbf{y}^{(3)} &= (0, x_1, x_2, 0, 0, x_2, x_1, 0)^T, \\
\mathbf{y}^{(2)} &= (0, x_1 + x_2, x_1 + x_2, 0)^T, \\
\mathbf{y}^{(1)} &= (x_1 + x_2, x_1 + x_2)^T, \\
\mathbf{y}^{(0)} &= (2(x_1 + x_2))^T.
\end{aligned}$$

We assume that $\mathbf{y}$ satisfies (5.4), i.e., that $x_0 + x_2 \neq 0$. If this is the case, then $\mathbf{y}^{(3)}$ has the two-block support $S^{(3)} = I^{(3)}_{1,2} \cup I^{(3)}_{5,6}$ of length $n^{(3)} := 2 = m$ and $\mathbf{y}^{(2)}$ has the one-block support $S^{(2)} = I^{(2)}_{1,2}$ of length $m^{(2)} := 2 < 2m$, centered around the middle of the vector, i.e., around 1 and 2, with first support index $\mu^{(2)} := 1$. The vectors $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(0)}$ both have full support, which can be interpreted as a one-block support centered around the middle with first support indices $\mu^{(1)} := \mu^{(0)} := 0$ and block lengths $m^{(1)} := 2$ and $m^{(0)} := 1$.

2. Now let again $\mathbf{x} = (x_0, 0, 0, 0, 0, 0, 0, 0)^T \in \mathbb{R}^8$. Then $\mathbf{y}$ satisfies (5.4), since $x_0 \neq 0$ by assumption. We have that

$$\begin{aligned}
\mathbf{y} = \mathbf{y}^{(4)} &= (x_0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x_0)^T, \\
\mathbf{y}^{(3)} &= (x_0, 0, 0, 0, 0, 0, 0, x_0)^T, \\
\mathbf{y}^{(2)} &= (x_0, 0, 0, x_0)^T, \\
\mathbf{y}^{(1)} &= (x_0, x_0)^T, \\
\mathbf{y}^{(0)} &= (2x_0)^T.
\end{aligned}$$

Here, $\mathbf{y}^{(3)}$ has the one-block support $S^{(3)} = I^{(3)}_{7,0}$ of length $m^{(3)} := 2$ and $\mathbf{y}^{(2)}$ has the one-block support $S^{(2)} = I^{(2)}_{3,0}$ of length $m^{(2)} := 2$. Both supports are centered around the boundary of the vector and have the first support indices $\mu^{(3)} := 7$ and $\mu^{(2)} := 3$. The vectors $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(0)}$ have full support, which can be interpreted as a one-block support centered around the middle with first support indices $\mu^{(1)} := \mu^{(0)} := 0$ and block lengths $m^{(1)} := 2$ and $m^{(0)} := 1$.

3. Again, we examine $\mathbf{x} = (x_0, x_1, 0, 0, 0, 0, 0, x_7)^T \in \mathbb{R}^8$. Then we obtain

$$\begin{aligned}
\mathbf{y} = \mathbf{y}^{(4)} &= (x_0, x_1, 0, 0, 0, 0, 0, x_7, x_7, 0, 0, 0, 0, 0, 0, x_1, x_0)^T, \\
\mathbf{y}^{(3)} &= (x_0 + x_7, x_1, 0, 0, 0, 0, 0, x_1, x_0 + x_7)^T, \\
\mathbf{y}^{(2)} &= (x_0 + x_7, x_1, x_1, x_0 + x_7)^T, \\
\mathbf{y}^{(1)} &= (x_0 + x_1 + x_7, x_0 + x_1 + x_7)^T, \\
\mathbf{y}^{(0)} &= (2(x_0 + x_1 + x_7))^T .
\end{aligned}$$

Thus, (5.4) is satisfied if $x_0 + x_7 \neq 0$ and $x_0 + x_1 + x_7 \neq 0$. If this is the case, then $\mathbf{y}^{(3)}$ has the one-block support $S^{(3)} = I^{(3)}_{6,1}$ of length $m^{(3)} := 4$ with first support index $\mu^{(3)} := 6$, centered around 7 and 0. All shorter periodizations have a one-block support as well, but their supports have full length and are thus centered around the middle of the respective vectors with first support indices $\mu^{(2)} := \mu^{(1)} := \mu^{(0)} := 0$. ♢

In Example 5.14 the periodizations have either a one-block or a two-block support of length at most $2m$ for all considered vectors $\mathbf{y}$. These observations are generalized in the following lemma.

Lemma 5.15 (Lemma 5 in [BP18c]) Let $N = 2^{J-1}$ with $J \geq 2$ and $j \in \{0, \ldots, J-1\}$. Let $\mathbf{x} \in \mathbb{R}^N$ have a one-block support of length $m < N$. Set $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T$ and assume that $\mathbf{y}$ satisfies (5.4). Then $\mathbf{y}^{(j)}$ possesses either

A) the one-block support

$$S^{(j)} = I^{(j)}_{\mu^{(j)}, 2^j - 1 - \mu^{(j)}}$$

of length $m^{(j)} \leq 2m$, or

B) the two-block support

$$S^{(j)} = I^{(j)}_{\mu^{(j)}, \nu^{(j)}} \cup I^{(j)}_{2^j - 1 - \nu^{(j)}, 2^j - 1 - \mu^{(j)}}$$

with block length $n^{(j)} = m$.

Proof. Note that by Lemma 5.12, $\mathbf{y} = \mathbf{y}^{(J)}$ has a reflected block support with either one or two support blocks. Let us fix a level $j \in \{0, \ldots, J-1\}$. It follows from Definition 5.7 that the number of indices $k \in \{0, \ldots, 2^j - 1\}$ satisfying

$$y^{(j)}_k \neq 0$$

cannot exceed the number of indices $k' \in \{0, \ldots, 2^{j+1} - 1\}$ such that

$$y^{(j+1)}_{k'} \neq 0,$$

i.e., the sparsity of $\mathbf{y}^{(j)}$ cannot exceed the sparsity of $\mathbf{y}^{(j+1)}$.

If $\mathbf{y}^{(j+1)}$ has a two-block support, then $\mathbf{y}^{(j)}$ can have at most two support blocks, as the two blocks of $\mathbf{y}^{(j+1)}$ are either mapped to two separate blocks in $\mathbf{y}^{(j)}$ or to one block consisting of the two partially overlapping blocks of $\mathbf{y}^{(j+1)}$. If $j + 1 = J$ and the two blocks in $\mathbf{y}^{(J)}$ have the possibly different lengths $2(N - \mu^{\mathbf{x}})$ and $2(\nu^{\mathbf{x}} + 1)$, then $\mathbf{y}^{(J-1)}$ has a one-block support of length $m^{(J-1)} \leq 2m$ by Definition 5.7, see also Figure 5.3. If $j \leq J - 1$, the periodization $\mathbf{y}^{(j)}$ cannot have a two-block support with two blocks of different lengths as in Lemma 5.12, case ii).

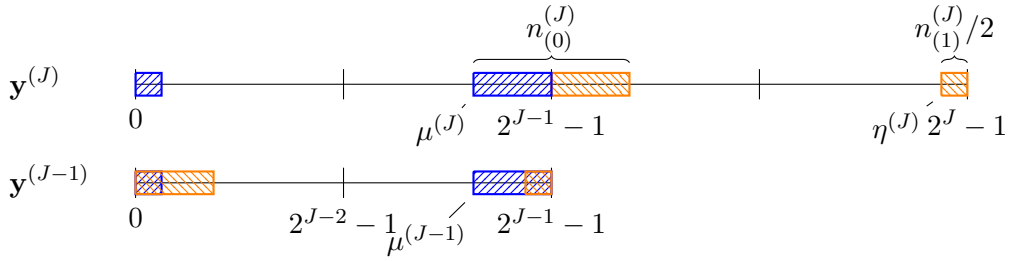

Figure 5.3: Illustration of the support of $\mathbf{y}^{(J-1)}$ if $\mathbf{y}^{(J)}$ has a two-block support with two possibly different block lengths according to Lemma 5.15

Otherwise, i.e., if $j+1 < J$ or if the two blocks in $\mathbf{y}^{(J)}$ both have length m, then $\mathbf{y}^{(j)}$ has either a one-block or a two-block support. If the two blocks are still separated in $\mathbf{y}^{(j)}$, they have the same length as they did in $\mathbf{y}^{(j+1)}$, i.e., $n^{(j)} = n^{(j+1)} = m$. It follows from Lemma 5.9 (iii) that $\mathbf{y}^{(j)}$ is symmetric, so the two support blocks are reflections of each other, which results in a support of the form

$$S^{(j)} = I^{(j)}_{\mu^{(j)},\,\nu^{(j)}} \cup I^{(j)}_{2^j-1-\nu^{(j)},\,2^j-1-\mu^{(j)}},$$

see also Figure 5.4.

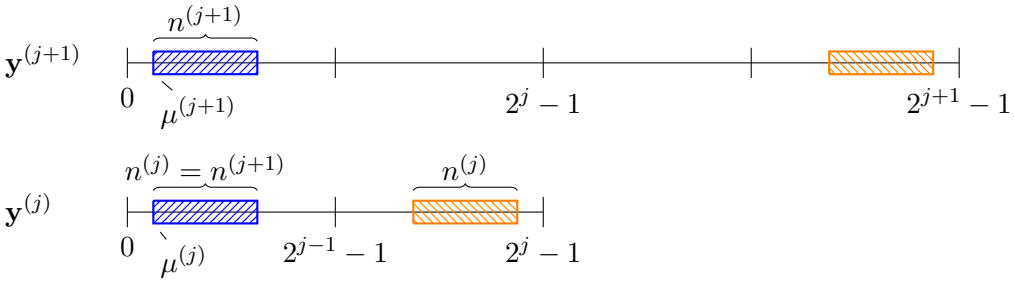

Figure 5.4: Illustration of the support of $\mathbf{y}^{(j)}$ if $\mathbf{y}^{(j+1)}$ and $\mathbf{y}^{(j)}$ have a two-block support according to Lemma 5.15

If $\mathbf{y}^{(j+1)}$ has a two-block support and $\mathbf{y}^{(j)}$ a one-block support, the length $m^{(j)}$ of the support of $\mathbf{y}^{(j)}$ can be at most $2m$ by Definition 5.7. Moreover, due to the vector's symmetry, the single support block has to be centered around the middle of the vector or around its boundary, so it has to be of the form

$$S^{(j)} = I^{(j)}_{\mu^{(j)},\,2^j-1-\mu^{(j)}},$$

see Figure 5.5.

In the case that $\mathbf{y}^{(j+1)}$ already has a one-block support, by the considerations above it must have arisen from a vector $\mathbf{y}^{(k)}$, $k > j+1$, with two-block support or from $\mathbf{y}^{(J)}$ with one-block support, which has to be centered around the middle or the boundary of the vector according to Lemma 5.12. Thus, the support of $\mathbf{y}^{(j+1)}$ is centered around the middle or the boundary of the vector and its length satisfies $m^{(j+1)} \leq 2m$. Periodizing $\mathbf{y}^{(j+1)}$ to obtain $\mathbf{y}^{(j)}$ then yields a vector whose support of length at most $2m$ is centered

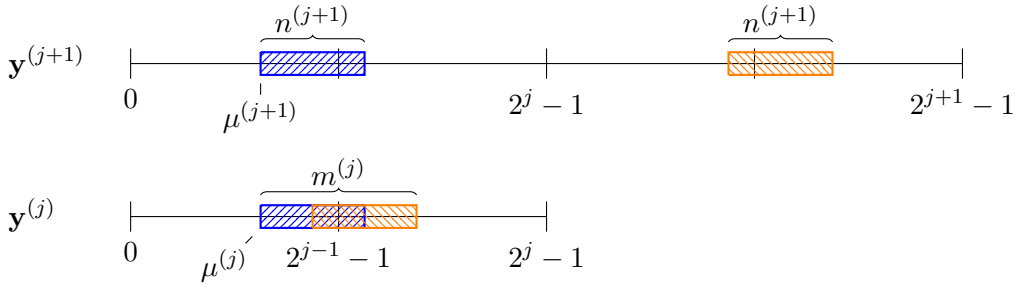

Figure 5.5: Illustration of the support of $\mathbf{y}^{(j)}$ if $\mathbf{y}^{(j+1)}$ has a two-block support and $\mathbf{y}^{(j)}$ a one-block support according to Lemma 5.15

around the boundary of the vector by definition of the periodization, see Figure 5.6. Consequently, the vector $\mathbf{y}^{(j)}$ has a one-block support of length $m^{(j)} \leq 2m$, and thus

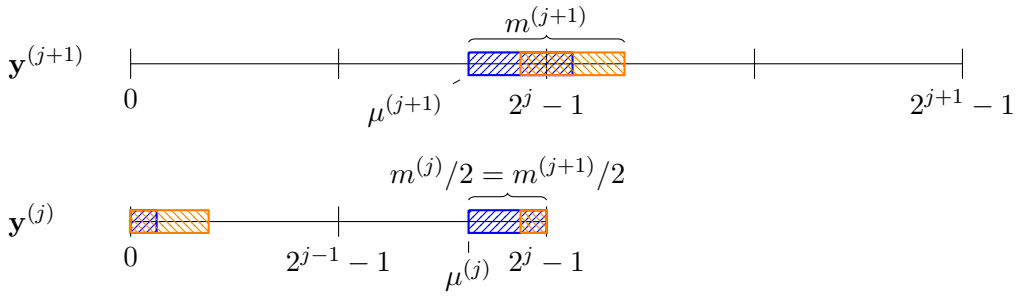

Figure 5.6: Illustration of the support of $\mathbf{y}^{(j)}$ if $\mathbf{y}^{(j+1)}$ has a one-block support according to Lemma 5.15

all shorter periodizations $\mathbf{y}^{(l)}$, $l \in \{0, \ldots, j-1\}$, possess a one-block support of length $m^{(l)} \leq 2m$ as well. □

Remark 5.16 i) Let $j \in \{0, \ldots, J-1\}$. We always denote by $\mu^{(j)}$ and $\nu^{(j)}$ the *first and last support index* of $\mathbf{y}^{(j)}$ if $\mathbf{y}^{(j)}$ has a one-block support, or the *first and last index of the first support block* if $\mathbf{y}^{(j)}$ has a two-block support. In the one-block case, the *support length* of $\mathbf{y}^{(j)}$ is $m^{(j)}$, and in the two-block case, we denote the *length of the two support blocks* by $n^{(j)}$ for algorithmic purposes, even though $n^{(j)} = m$ for exact data. Since $j < J$, the two support blocks must have the same length by Lemma 5.15. The *support set* of $\mathbf{y}^{(j)}$, which is always the union of two intervals, one of which may be empty, is referred to as $S^{(j)} \subseteq I^{(j)}_{0,2^j-1}$. The notation for the support of the periodizations $\mathbf{y}^{(j)}$ is thus analogously to the notation for the support of $\mathbf{y}$ introduced in Remark 5.13.

ii) If $\mathbf{y}^{(j)}$ has a one-block support of length $2^{j-1} < m^{(j)} \leq 2^j$, the first support index $\mu^{(j)}$ may not be uniquely determined. For example, for $\mathbf{y}^{(3)} = (0,1,1,0,0,1,1,0)^T \in \mathbb{R}^8$, which, by definition, can also be considered to have a one-block support of length 6, the support interval can be either $S^{(3)} = I^{(3)}_{1,6}$ or $S^{(3)} = I^{(3)}_{5,2}$. Note that by Lemma 5.15 the support of $\mathbf{y}^{(j)}$ is symmetric, i.e., $S^{(j)} = I^{(j)}_{\mu^{(j)}, 2^j-1-\mu^{(j)}}$, which can be used to exclude some possible first support indices. If the first support index is still not unique, we choose

$0 \leq \mu^{(j)} < 2^{j-1}-1$ such that $S^{(j)}$ is centered around the middle of the vector, i.e., around $2^{j-1}-1$ and 2^{j-1}. In the example above we choose $\mu^{(3)} \coloneqq 1$. If $m = 2^j$, we just fix $\mu^{(j)} \coloneqq 0$, as in Example 5.14, which also results in a support centered around the middle of the vector.

iii) In case A of Lemma 5.15, the first support index $\mu^{(j)}$ of $\mathbf{y}^{(j)}$ is contained in the interval $I^{(j)}_{0,\,2^j-1}$. Due to $\mathbf{y}^{(j)}$ being symmetric, we set $\nu^{(j)} \coloneqq 2^j - 1 - \mu^{(j)}$. If $\nu^{(j)} \geq \mu^{(j)}$, it follows that the support block is centered around the middle of the vector, i.e., $2^{j-1}-1$ and 2^{j-1}, and the support length is $m^{(j)} = \nu^{(j)} - \mu^{(j)} + 1$, since $S^{(j)} = I^{(j)}_{\mu^{(j)},\,\nu^{(j)}}$. If $\nu^{(j)} < \mu^{(j)}$, the support block is centered around the boundary, i.e., $2^j - 1$ and 0, and its length is $m^{(j)} = 2^j - \mu^{(j)} + \nu^{(j)} + 1$.

iv) In case B of Lemma 5.15, the two blocks are always separated and we find that the first support index of the first block, $\mu^{(j)}$, is contained in the interval $I^{(j)}_{0,\,2^{j-1}-1}$, and that, by symmetry, the first support index of the second block, $2^j - 1 - \nu^{(j)}$, is contained in the interval $I^{(j)}_{2^{j-1},\,2^j-1}$. ◊

5.4.3 Support Structure of $\mathbf{y}^{(j+1)}$ for Given $\mathbf{y}^{(j)}$

If we want to iteratively recover the vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T \in \mathbb{R}^{2N}$ from its periodizations $\mathbf{y}^{(j)} \in \mathbb{R}^{2^j}$ for $j \in \{0, \ldots, J-1\}$, using in each step that $\mathbf{y}^{(j)}$ is already known, we have to carefully inspect how much of the support structure of $\mathbf{y}^{(j+1)}$ can already be deduced from $\mathbf{y}^{(j)}$ in order to minimize the computational effort. Before proving a general theorem about the support of $\mathbf{y}^{(j+1)}$ for given $\mathbf{y}^{(j)}$, we will have another look at the different cases that can occur for the periodizations of the vectors considered in Examples 5.11 and 5.14.

Example 5.17 (Examples 5.11 and 5.14 continued) Throughout this example we use that $\mathbf{y}^{(j)}$ is symmetric by Lemma 5.9 (iii), the observations made in Lemma 5.15, as well as the fact that, by definition, $\mathbf{y}^{(j)} = \mathbf{y}^{(j+1)}_{(0)} + \mathbf{y}^{(j+1)}_{(1)}$ for all $j \in \{0, \ldots, J-1\}$. Furthermore, we have to assume that the vector length $16 = 2^J$ is known a priori.

1. Consider $\mathbf{y}^{(1)} = \left(y^{(1)}_0, y^{(1)}_0\right)^T$ with full support $S^{(1)} = I^{(1)}_{0,\,1}$. Similar structures were obtained in Examples 5.14.1, 5.14.2 and 5.14.3. Then it follows that $\mathbf{y}^{(2)}$ has a one-block support of length $m^{(2)} \geq m^{(1)}$ of the general form

$$\mathbf{y}^{(2)} = \left(y^{(2)}_0, y^{(2)}_1, y^{(2)}_1, y^{(2)}_0\right)^T,$$

where $y^{(2)}_0 + y^{(2)}_1 = y^{(1)}_0$. The entries of $\mathbf{y}^{(2)}$ have to be recovered from $\widehat{\mathbf{y}}$ using the methods we will present in Section 5.5. If we find that $y^{(2)}_0 = 0$, the vector $\mathbf{y}^{(2)}$ has the one-block support $S^{(2)} = I^{(2)}_{1,\,2}$ of length $m^{(2)} = 2$ centered around the middle, and if $y^{(2)}_1 = 0$, it has the one-block support $S^{(2)} = I^{(2)}_{3,\,0}$ of length $m^{(2)} = 2$ centered around the boundary. However, if both $y^{(2)}_0$ and $y^{(2)}_1$ are not zero, we obtain the one-block support $S^{(2)} = I^{(2)}_{0,\,3}$ of full length $m^{(2)} = 4$.

2. Let $\mathbf{y}^{(2)} = \left(0, y^{(2)}_1, y^{(2)}_1, 0\right)^T$ have the one-block support $S^{(2)} = I^{(2)}_{1,\,2}$ of length $m^{(2)} = 2$ centered around the middle of the vector, as obtained in Example 5.14.1. Then, by its

symmetry and Definition 5.7, $\mathbf{y}^{(3)}$ has to be of the form

$$\mathbf{y}^{(3)} = \left(0, y_1^{(3)}, y_2^{(3)}, 0, 0, y_2^{(3)}, y_1^{(3)}, 0\right)^T,$$

where $y_1^{(3)} + y_2^{(3)} = y_1^{(2)}$. This means that $\mathbf{y}^{(3)}$ has a two-block support $S^{(3)} \subseteq I_{1,2}^{(3)} \cup I_{5,6}^{(3)}$ with block length $n^{(3)} \leq m^{(2)} = 2$. It is possible that $y_1^{(3)} = 0$ or $y_2^{(3)} = 0$; then, we find that $n^{(3)} = 1$. Otherwise, we have $n^{(3)} = 2$, but we require additional entries of $\widehat{\mathbf{y}}$ in order to be able to recover the possibly nonzero entries $y_1^{(3)}$ and $y_2^{(3)}$ of $\mathbf{y}^{(3)}$. We will show how to compute them in detail in Section 5.5.

3. Let $\mathbf{y}^{(2)} = \left(y_0^{(2)}, 0, 0, y_0^{(2)}\right)^T$ have the one-block support $S^{(2)} = I_{3,0}^{(2)}$ of length $m^{(2)} = 2$ centered around the boundary, as it occurs in Example 5.14.2. Then we can conclude that either

$$\mathbf{y}^{(3)} = \left(y_0^{(3)}, 0, 0, 0, 0, 0, 0, y_0^{(3)}\right)^T \quad \text{or} \quad \mathbf{y}^{(3)} = \left(0, 0, 0, y_3^{(3)}, y_3^{(3)}, 0, 0, 0\right)^T,$$

with $y_0^{(3)} = y_3^{(3)} = y_0^{(2)}$, since no other symmetric vector in $\mathbb{R}^8$ with periodization $\mathbf{y}^{(2)}$ can arise from periodizing a vector $\mathbf{y} \in \mathbb{R}^{16}$ with reflected block support. Therefore, $\mathbf{y}^{(3)}$ must also possess a one-block support of length $m^{(3)} = m^{(2)} = 2$. In particular, since $\mathbf{y}^{(3)}$ is still an "intermediate" vector, i.e., $3 < J$, a two-block support of the form

$$\mathbf{y}^{(3)} = (*, 0, 0, *, *, 0, 0, *)^T,$$

where $*$ denotes the nonzero entries, cannot occur, as the corresponding vector $\mathbf{y}$ would either not be symmetric or of the form

$$\mathbf{y} = (*, 0, 0, *, *, 0, 0, *, *, 0, 0, *, *, 0, 0, *)^T,$$

which violates Lemma 5.12. Thus, $\mathbf{y}^{(3)}$ has either the support $S^{(3)} = I_{7,0}^{(3)}$ or $S^{(3)} = I_{3,4}^{(3)}$. We will show in Section 5.5 how to determine which of the possibilities is the correct one.

4. Consider $\mathbf{y}^{(3)} = \left(y_0^{(3)}, 0, 0, 0, 0, 0, 0, y_0^{(3)}\right)^T$ with one-block support $S^{(3)} = I_{7,0}^{(3)}$ of length $m^{(3)} = 2$ centered around the boundary. Here, $3 = J - 1$, i.e., the periodization $\mathbf{y}^{(4)}$ of double length is the final vector $\mathbf{y}$, see Examples 5.14.2 and 5.14.3. We find that

$$\mathbf{y} = \mathbf{y}^{(4)} = (y_0, 0, 0, 0, 0, 0, 0, y_7, y_7, 0, 0, 0, 0, 0, 0, y_0)^T,$$

where $y_0 + y_7 = y_0^{(3)}$. Thus, $\mathbf{y}$ has a two-block support $S^{(4)} \subseteq I_{(7,8)}^{(4)} \cup I_{(15,0)}^{(4)}$ of length $n^{(4)} := 2$ if y_0 and y_7 are both not zero. If $y_0 = 0$ or $y_7 = 0$, one of the blocks is empty, resulting in a one-block support of length $m^{(4)} = 1$. Since $J = 4$, all these structures are feasible. Using additional information about $\widehat{\mathbf{y}}$, one can compute y_0 and y_7, as we will show in Section 5.5.

5. Let $\mathbf{y}^{(3)} = \left(0, y_1^{(3)}, y_2^{(3)}, 0, 0, y_2^{(3)}, y_1^{(3)}, 0\right)^T$ have the two-block support $S^{(3)} = I_{1,2}^{(3)} \cup I_{5,6}^{(3)}$ of block length $n^{(3)} = 2$, as in Example 5.14.1. Then there are precisely two possibilities

for $\mathbf{y}$, either

$$\mathbf{y} = \mathbf{y}^{(4)} = \left(0, y_1^{(3)}, y_2^{(3)}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, y_2^{(3)}, y_1^{(3)}, 0\right)^T \qquad \text{or}$$
$$\mathbf{y} = \mathbf{y}^{(4)} = \left(0, 0, 0, 0, 0, y_2^{(3)}, y_1^{(3)}, 0, 0, y_1^{(3)}, y_2^{(3)}, 0, 0, 0, 0, 0\right)^T,$$

since $\mathbf{y}$ is symmetric and Lemma 5.12 holds. Thus, $\mathbf{y}$ has either the two-block support $S^{(4)} = I_{1,2}^{(4)} \cup I_{13,14}^{(4)}$ or $S^{(4)} = I_{5,6}^{(4)} \cup I_{9,10}^{(4)}$ with block length $n^{(4)} = n^{(3)} = 2 = m$. Note that the values of the nonzero entries of $\mathbf{y}^{(3)}$ and $\mathbf{y}$ are the same. Which of the two possibilities is the correct one can be determined using the methods we will present in Section 5.5. ◊

Using the considerations from Example 5.17, we now reconstruct the support of the vector from Example 5.11.1 step by step to illustrate this part of our algorithm.

Example 5.18 (Examples 5.11.1 and 5.14.1 continued) We want to recover the vector $\mathbf{x} = (0, x_1, x_2, 0, 0, 0, 0, 0)^T \in \mathbb{R}^8$ with positive entries x_1, x_2 from the periodizations of $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T$. Assume that we know $\widehat{\mathbf{y}} \in \mathbb{C}^{16}$. By Definition 1.1 and Lemma 5.10, we obtain that

$$\mathbf{y}^{(0)} = y_0^{(0)} = \left(\widehat{y^{(0)}}\right)_0 = \widehat{y}_0 \in \mathbb{R}.$$

Thus, $\mathbf{y}^{(0)}$ has full support and, analogously to Example 5.17.1, $\mathbf{y}^{(1)}$ has the one-block support $S^{(1)} = I_{0,1}^{(1)}$ of the form

$$\mathbf{y}^{(1)} = \left(y_0^{(1)}, y_0^{(1)}\right)^T,$$

where $2y_0^{(1)} = y_0^{(0)}$. Similarly, $\mathbf{y}^{(2)}$ satisfies

$$\mathbf{y}^{(2)} = \left(y_0^{(2)}, y_1^{(2)}, y_1^{(2)}, y_0^{(2)}\right)^T,$$

where $y_0^{(2)} + y_1^{(2)} = y_0^{(1)}$. In order to recover $\mathbf{y}^{(2)}$ completely, we require further information, which can be gained from the given Fourier data $\widehat{\mathbf{y}}$, as we will show in Section 5.5. Using these methods we find that $y_0^{(2)} = 0$, so $\mathbf{y}^{(2)}$ has the one-block support $S^{(2)} = I_{1,2}^{(2)}$ of length $m^{(2)} = 2$ centered around the middle. As in Example 5.17.2, we obtain

$$\mathbf{y}^{(3)} = \left(0, y_1^{(3)}, y_2^{(3)}, 0, 0, y_2^{(3)}, y_1^{(3)}, 0\right)^T,$$

where $y_1^{(3)} + y_2^{(3)} = y_1^{(2)}$. Hence, $\mathbf{y}^{(3)}$ has a two-block support of length $n^{(3)} \leq m^{(2)} = 2$. With the help of the given Fourier data and the methods from Section 5.5, we will be able to prove that neither $y_1^{(3)}$ nor $y_2^{(3)}$ are zero, i.e., that $n^{(3)} = 2$ and $S^{(3)} = I_{1,2}^{(3)} \cup I_{5,6}^{(3)}$. Finally, it follows from Example 5.17.5 that either

$$\mathbf{y} = \mathbf{y}^{(4)} = \left(0, y_1^{(3)}, y_2^{(3)}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, y_2^{(3)}, y_1^{(3)}, 0\right)^T \qquad \text{or}$$
$$\mathbf{y} = \mathbf{y}^{(4)} = \left(0, 0, 0, 0, 0, y_2^{(3)}, y_1^{(3)}, 0, 0, y_1^{(3)}, y_2^{(3)}, 0, 0, 0, 0, 0\right)^T.$$

Using the given Fourier data $\widehat{\mathbf{y}}$, we can then determine that the first possibility is the

correct one, so that $S^{(4)} = I^{(4)}_{1,2} \cup I^{(4)}_{13,14}$, which also allows us to recover

$$\mathbf{x} = \left(0, y_1^{(3)}, y_2^{(3)}, 0, 0, 0, 0, 0\right)^T \in \mathbb{R}^8.$$

$\Diamond$

Even the two previous examples, though by no means all-encompassing, illustrate that one already has to consider several cases in order to correctly recover the three simple test vectors iteratively from their periodizations. In the following theorem we give a complete characterization of the supports of $\mathbf{y}^{(j+1)}$ which are possible if $\mathbf{y}^{(j)}$ is given for all cases that can occur for a vector $\mathbf{y}$ with reflected block support.

Theorem 5.19 (Theorem 1 in [BP18c]) *Let $N = 2^{J-1}$, $J \geq 2$, and $j \in \{0, \ldots, J-1\}$. Let $\mathbf{x} \in \mathbb{R}^N$ have a one-block support of length $m < N$. Set $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T$ and assume that $\mathbf{y}$ satisfies (5.4). Then we have to distinguish the following cases.*

A) *$\mathbf{y}^{(j)}$ has the one-block support $S^{(j)} = I^{(j)}_{\mu^{(j)}, 2^j - 1 - \mu^{(j)}}$ of length $m^{(j)} \leq 2m$:*

- A$_1$) *If the support of $\mathbf{y}^{(j)}$ has length $m^{(j)} < 2^j$ and is centered around the middle of the vector, then $\mathbf{y}^{(j+1)}$ possesses the two-block support*

$$S^{(j+1)} = I^{(j+1)}_{\mu^{(j+1)}, \nu^{(j+1)}} \cup I^{(j+1)}_{2^{j+1}-1-\nu^{(j+1)}, 2^{j+1}-1-\mu^{(j+1)}}$$

  with two blocks of length $n^{(j+1)} = m$, see Figure 5.7.

- A$_2$) *If the support of $\mathbf{y}^{(j)}$ has full length $m^{(j)} = 2^j$ and $j < J-1$, then $\mathbf{y}^{(j+1)}$ has a one-block support of length $m^{(j+1)} \geq m^{(j)}$. If $j = J-1$, then $\mathbf{y} = \mathbf{y}^{(J)}$ has a two-block support with two blocks of possibly different lengths or a one-block support of length $m^{(J)} \geq m^{(J-1)}$.*

- A$_3$) *If the support of $\mathbf{y}^{(j)}$ has length $m^{(j)} < 2^j$ and is centered around the boundary of the vector, and $j < J-1$, then $\mathbf{y}^{(j+1)}$ possesses the one-block support*

$$S^{(j+1)} = I^{(j+1)}_{\mu^{(j+1)}, 2^{j+1}-1-\mu^{(j+1)}}$$

  of length $m^{(j+1)} = m^{(j)}$, see Figure 5.8.

- A$_4$) *If the support of $\mathbf{y}^{(J-1)}$ has length $m^{(J-1)} < 2^{J-1}$ and is centered around the boundary of the vector, then $\mathbf{y} = \mathbf{y}^{(J)}$ possesses the two-block support*

$$S^{(J)} = I^{(J)}_{\mu^{(J)}, 2^J-1-\mu^{(J)}} \cup I^{(J)}_{\eta^{(J)}, 2^J-1-\eta^{(J)}} \quad \text{with} \quad \mu^{(J)} < 2^{J-1} \leq \eta^{(J)},$$

  where the two blocks may have different lengths, see Figure 5.9. If $\mu^{\mathbf{x}} = 0$ or $\nu^{\mathbf{x}} = 2^{J-1} - 1$, one of these blocks is empty.

B) *$\mathbf{y}^{(j)}$ has the two-block support $S^{(j)} = I^{(j)}_{\mu^{(j)}, \nu^{(j)}} \cup I^{(j)}_{2^j-1-\nu^{(j)}, 2^j-1-\mu^{(j)}}$ with block length $n^{(j)} = m$:*

Then $\mathbf{y}^{(j+1)}$ has the two-block support

$$S^{(j+1)} = I^{(j+1)}_{\mu^{(j+1)}, \nu^{(j+1)}} \cup I^{(j+1)}_{2^{j+1}-1-\nu^{(j+1)}, 2^{j+1}-1-\mu^{(j+1)}}$$

with block length $n^{(j+1)} = m$, see Figure 5.10.

Proof. Cases A_1 to A_4, henceforth subsumed to case A, summarize the support properties of $\mathbf{y}^{(j+1)}$ if $\mathbf{y}^{(j)}$ possesses a one-block support. Assertion B covers the case that $\mathbf{y}^{(j)}$ has a two-block support. All observations about the possible support blocks of $\mathbf{y}^{(j+1)}$ follow by employing the results from Lemmas 5.12 and 5.15 and utilizing the known support $S^{(j)}$ of $\mathbf{y}^{(j)}$, Definition 5.7 and the symmetry of $\mathbf{y}^{(j+1)}$ given by Lemma 5.9 (iii).

A) One-block support:

A_1) In case A_1 there are, due to the symmetry and the reflected block support of $\mathbf{y}$, exactly two possibilities for the support of the periodization $\mathbf{y}^{(j+1)}$ of double length that have the given periodization $\mathbf{y}^{(j)}$. They are depicted in Figure 5.7. As the support of $\mathbf{y}^{(j)}$ is centered around the middle of the vector, the support blocks have to be separated in $\mathbf{y}^{(j+1)}$, resulting in two blocks of length $n^{(j+1)} = m$ by definition of $\mathbf{y}$. In the first case, since $m \leq m^{(j)}$, the support set $S^{(j+1)}$ of $\mathbf{y}^{(j+1)}$ satisfies

$$\begin{aligned} S^{(j+1)} &= I^{(j+1)}_{\mu^{(j)},\,\mu^{(j)}+m-1} \cup I^{(j+1)}_{2^{j+1}-m-\mu^{(j)},\,2^{j+1}-1-\mu^{(j)}} \\ &\subseteq I^{(j+1)}_{\mu^{(j)},\,\mu^{(j)}+m^{(j)}-1} \cup I^{(j+1)}_{2^j+\mu^{(j)},\,2^j+\mu^{(j)}+m^{(j)}-1} \end{aligned}$$

with $\mu^{(j+1)} = \mu^{(j)}$, and in the second case

$$\begin{aligned} S^{(j+1)} &= I^{(j+1)}_{2^j-m-\mu^{(j)},\,2^j-1-\mu^{(j)}} \cup I^{(j+1)}_{2^j+\mu^{(j)},\,2^j+\mu^{(j)}+m-1} \\ &\subseteq I^{(j+1)}_{\mu^{(j)},\,\mu^{(j)}+m^{(j)}-1} \cup I^{(j+1)}_{2^j+\mu^{(j)},\,2^j+\mu^{(j)}+m^{(j)}-1} \end{aligned}$$

with $2^{j+1} - 1 - \nu^{(j+1)} = \mu^{(j)} + 2^j$.

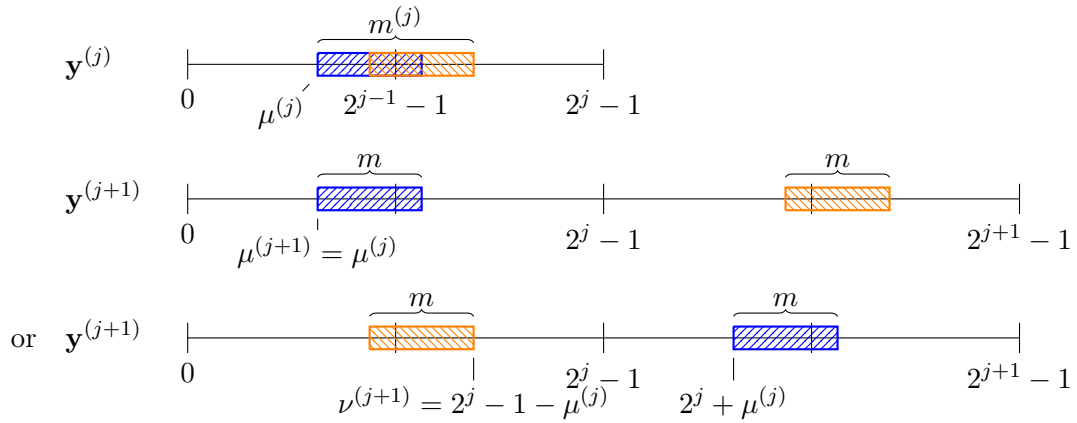

Figure 5.7: Illustration of the two possibilities for the support of $\mathbf{y}^{(j+1)}$ for given $\mathbf{y}^{(j)}$ according to Theorem 5.19, case A_1

A_2) In case A_2 we cannot learn much from the structure of the periodization $\mathbf{y}^{(j)}$. It follows from Lemmas 5.12 and 5.15 that, if $j < J-1$, $\mathbf{y}^{(j+1)}$ has a one-block support of length $m^{(j+1)} \geq m^{(j)}$. If $j = J-1$, then we can only deduce that $\mathbf{y}$ has a reflected block support. However, the inclusion

$$S^{(j+1)} \subseteq I^{(j+1)}_{\mu^{(j)},\,\mu^{(j)}+m^{(j)}-1} \cup I^{(j+1)}_{2^j+\mu^{(j)},\,2^j+\mu^{(j)}+m^{(j)}-1},$$

which already held for case A_1, is still satisfied.

A_3) In case A_3 there are, due to the definition of $\mathbf{y}$, only two possibilities for $\mathbf{y}^{(j+1)}$, which are shown in Figure 5.8. As the support of $\mathbf{y}^{(j)}$ is centered around the boundary of the vector, $\mathbf{y}^{(j+1)}$ must have a one-block support of length $m^{(j+1)} = m^{(j)}$ as well. In the first case we find for the support set $S^{(j+1)}$ of $\mathbf{y}^{(j+1)}$ that

$$\begin{aligned} S^{(j+1)} &= I^{(j+1)}_{\mu^{(j)},\mu^{(j)}+m^{(j)}-1} \\ &\subseteq I^{(j+1)}_{\mu^{(j)},\mu^{(j)}+m^{(j)}-1} \cup I^{(j+1)}_{2^j+\mu^{(j)},2^j+\mu^{(j)}+m^{(j)}-1}, \end{aligned}$$

centered around the middle with $\mu^{(j+1)} = \mu^{(j)}$, and in the second case

$$\begin{aligned} S^{(j+1)} &= I^{(j+1)}_{2^j+\mu^{(j)},2^j+\mu^{(j)}+m^{(j)}-1} \\ &\subseteq I^{(j+1)}_{\mu^{(j)},\mu^{(j)}+m^{(j)}-1} \cup I^{(j+1)}_{2^j+\mu^{(j)},2^j+\mu^{(j)}+m^{(j)}-1}, \end{aligned}$$

centered around the boundary with $\mu^{(j+1)} = 2^j + \mu^{(j)}$.

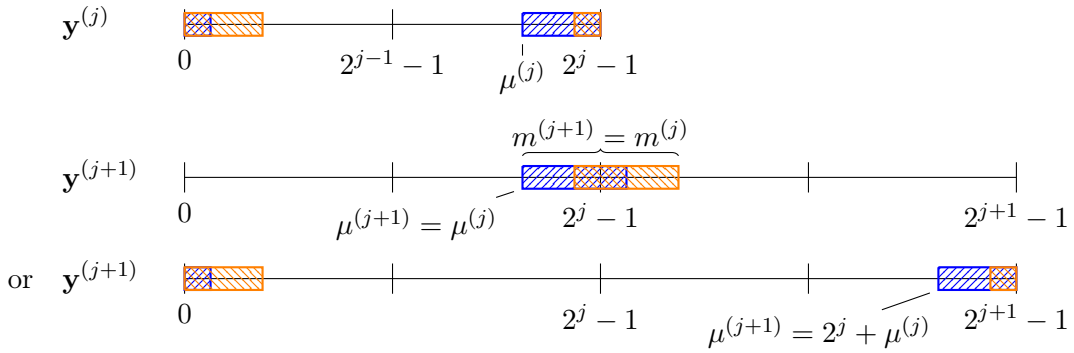

Figure 5.8: Illustration of the two possibilities for the support of $\mathbf{y}^{(j+1)}$ for given $\mathbf{y}^{(j)}$ according to Theorem 5.19, case A_3

A_4) In case A_4 the collided blocks almost always have to separate when we recover $\mathbf{y}$ from $\mathbf{y}^{(J-1)}$, since $\mathbf{y} = \mathbf{y}^{(J)}$ has a reflected block support. The vector $\mathbf{y}$ can only have a one-block support of length $2m$ constituted of two adjacent blocks if the first support index of $\mathbf{x}$ is 0 or if its last support index is $2^{J-1} - 1$.

Hence, $\mathbf{y}^{(J-1)}$ can only be of the form given in case A_4 if 0 or $2^{J-1} - 1$ are contained in the support interval $S^{\mathbf{x}}$ of $\mathbf{x}$. Again, there are two possibilities for the support of $\mathbf{y}$, which are depicted in Figure 5.9.

In the first case we find that

$$\begin{aligned} S^{(J)} &= I^{(J)}_{\mu^{(J)},\mu^{(J)}+n^{(J)}_{(0)}-1} \cup I^{(J)}_{\eta^{(J)},\eta^{(J)}+n^{(J)}_{(1)}-1} \\ &\subseteq I^{(J)}_{\mu^{(J-1)},\mu^{(J-1)}+m^{(J-1)}-1} \cup I^{(J)}_{2^{J-1}+\mu^{(J-1)},2^{J-1}+\mu^{(J-1)}+m^{(J-1)}-1}, \end{aligned}$$

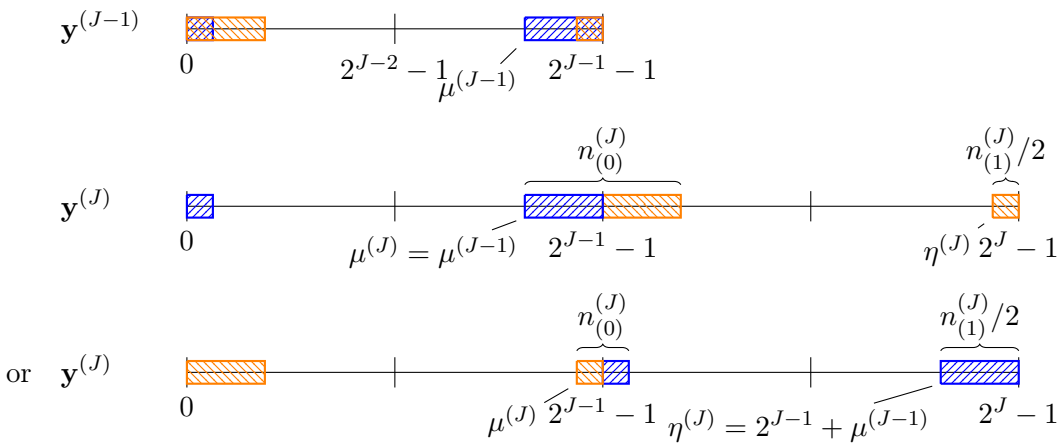

Figure 5.9: Illustration of the possibilities for the support of $\mathbf{y}^{(J)}$ for given $\mathbf{y}^{(J-1)}$ according to Theorem 5.19, case A_4

where $\eta^{(J)}$ is unknown, $\mu^{(J)}=\mu^{(J-1)}$ and $n^{(J)}_{(0)}=m^{(J-1)}$, and in the second case that

$$\begin{aligned} S^{(J)} &= I^{(J)}_{\mu^{(J)},\,\mu^{(J)}+n^{(J)}_{(0)}-1} \cup I^{(J)}_{\eta^{(J)},\,\eta^{(J)}+n^{(J)}_{(1)}-1} \\ &\subseteq I^{(J)}_{\mu^{(J-1)},\,\mu^{(J-1)}+m^{(J-1)}-1} \cup I^{(J)}_{2^{J-1}+\mu^{(J-1)},\,2^{J-1}+\mu^{(J-1)}+m^{(J-1)}-1}, \end{aligned}$$

where $\mu^{(J)}$ is unknown, $\eta^{(J)}=2^{J-1}+\mu^{(J-1)}$ and $n^{(J)}_{(1)}=m^{(J-1)}$. In both cases we have that $n^{(J)}_{(0)}+n^{(J)}_{(1)}=2m$.

If the first support index $\mu^{\mathbf{x}}$ of $\mathbf{x}$ is 0 or its last support index $\nu^{\mathbf{x}}$ is $2^{J-1}-1$, then $\mathbf{y}$ has a one-block support, where the first block in the cases above is empty if $\mu^{\mathbf{x}}=0$ and the second block is empty if $\nu^{\mathbf{x}}=2^{J-1}-1$.

B) Two-block support:

In case B the support blocks are already separated in $\mathbf{y}^{(j)}$, so $\mathbf{y}^{(j+1)}$ has a two-block support of block length $n^{(j+1)}=n^{(j)}=m$ as well. Since there is no collision, the nonzero entries of $\mathbf{y}^{(j+1)}$ and $\mathbf{y}^{(j)}$ are the same, and only the first support indices of the blocks in $\mathbf{y}^{(j+1)}$ are unknown. Again, there are only two possibilities for $\mathbf{y}^{(j+1)}$, which are shown in Figure 5.10. We find for the first case that

$$S^{(j+1)} = I^{(j+1)}_{\mu^{(j)},\,\nu^{(j)}} \cup I^{(j+1)}_{2^{j+1}-1-\nu^{(j)},\,2^{j+1}-1-\mu^{(j)}},$$

where the first blocks of $\mathbf{y}^{(j)}$ and $\mathbf{y}^{(j+1)}$ are identical, with the same support, and the second block of $\mathbf{y}^{(j+1)}$ is the second block of $\mathbf{y}^{(j)}$, shifted by 2^j. Otherwise, $S^{(j+1)}$ satisfies

$$S^{(j+1)} = I^{(j+1)}_{2^j-1-\nu^{(j)},\,2^j-1-\mu^{(j)}} \cup I^{(j+1)}_{2^j+\mu^{(j)},\,2^j+\nu^{(j)}},$$

where the first block of $\mathbf{y}^{(j+1)}$ is the second block of $\mathbf{y}^{(j)}$, with the same support, and the second block of $\mathbf{y}^{(j+1)}$ is the first block of $\mathbf{y}^{(j)}$, shifted by 2^j. Thus, the first support index $\mu^{(j+1)}$ of $\mathbf{y}^{(j+1)}$ is either $\mu^{(j)}$ or $2^j-1-\nu^{(j)}=2^j-m-\mu^{(j)}$.

□

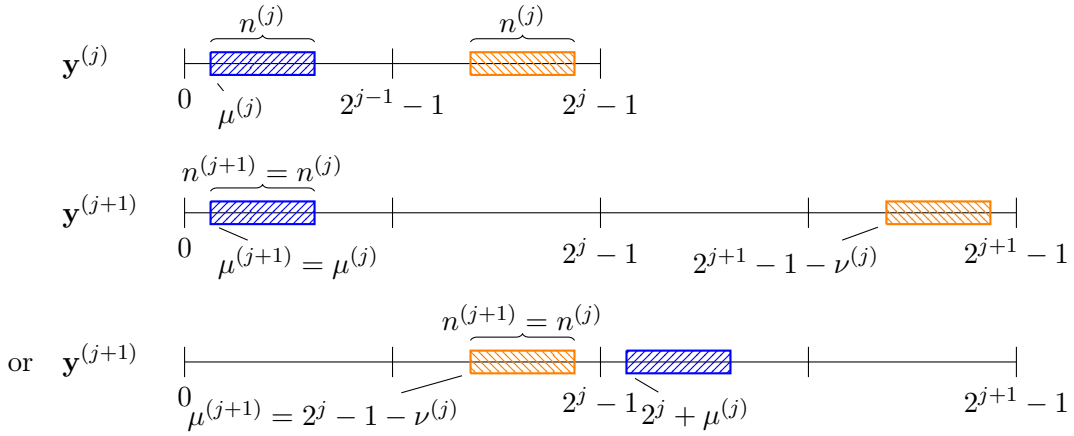

Figure 5.10: Illustration of the two possibilities for the support of $\mathbf{y}^{(j+1)}$ for given $\mathbf{y}^{(j)}$ according to Theorem 5.19, case B

5.5 Iterative Sparse Recovery Procedures

There is an important difference between case A of Theorem 5.19 on the one hand and case B on the other hand. In case A, $\mathbf{y}^{(j)}$ has a one-block support that usually contains overlapping entries of the original vector $\mathbf{y}$, i.e., some entries of $\mathbf{y}^{(j)}$ are obtained as sums of nonzero entries of $\mathbf{y}^{(j+1)}$, and thus also of $\mathbf{y}$. In case B, however, both support blocks of $\mathbf{y}^{(j)}$ are of length $n^{(j)} = m$ and they are separated. This is only possible if no nonzero entries of $\mathbf{y}^{(j+1)}$, and thus of $\mathbf{y}$, have been added in the process of computing $\mathbf{y}^{(j)}$. The nonzero entries of $\mathbf{y}^{(j)}$ and $\mathbf{y}^{(j+1)}$ are the same and we only have to determine the first support indices of the blocks in $\mathbf{y}^{(j+1)}$ in order to recover the vector from $\mathbf{y}^{(j)}$. In this section we will therefore derive two different strategies for computing $\mathbf{y}^{(j+1)}$; the first one has to be employed in case A, and the second one in case B. None of them will require a priori knowledge on the support length m of the vector $\mathbf{x} \in \mathbb{R}^N$ defining $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T \in \mathbb{R}^{2N}$.

5.5.1 Recovery Procedure for Case A: One-Block Support

We begin by deriving a recovery procedure for case A of Theorem 5.19, so let us assume that $j \in \{0, \ldots, J-1\}$ and that $\mathbf{y}^{(j)}$ has the one-block support

$$S^{(j)} = I^{(j)}_{\mu^{(j)}, 2^j - 1 - \mu^{(j)}} = I^{(j)}_{\mu^{(j)}, \mu^{(j)} + m^{(j)} - 1}$$

of length $m^{(j)} \leq 2m$. The definition of the periodization and Theorem 5.19, case A imply that the support set $S^{(j+1)}$ of $\mathbf{y}^{(j+1)}$ satisfies

$$S^{(j+1)} \subseteq I^{(j+1)}_{\mu^{(j)}, \mu^{(j)} + m^{(j)} - 1} \cup I^{(j+1)}_{2^j + \mu^{(j)}, 2^j + \mu^{(j)} + m^{(j)} - 1}.$$

In particular, $\mathbf{y}^{(j+1)}$ can have at most $2m^{(j)}$ nonzero entries.

The procedure developed hereafter utilizes that $\mathbf{y}^{(j+1)}$ is symmetric and thus deter-

mined by its first half. Recall that we denote by

$$\mathbf{y}_{(0)}^{(j+1)} \coloneqq \left(y_k^{(j+1)}\right)_{k=0}^{2^j-1} \quad \text{and} \quad \mathbf{y}_{(1)}^{(j+1)} \coloneqq \left(y_k^{(j+1)}\right)_{k=2^j}^{2^{j+1}-1}$$

the first and second half of $\mathbf{y}^{(j+1)}$, respectively. It also follows from Theorem 5.19 that both halves of $\mathbf{y}^{(j+1)}$ have a one-block support of length at most $m^{(j)}$. Since $\mathbf{y}_{(1)}^{(j+1)}$ is completely determined by $\mathbf{y}^{(j)}$ and $\mathbf{y}_{(0)}^{(j+1)}$ by Definition 5.7, we will only develop a method for recovering $\mathbf{y}_{(0)}^{(j+1)}$.

To efficiently compute the at most $m^{(j)}$ nonzero entries of $\mathbf{y}_{(0)}^{(j+1)}$, we will consider restrictions of $\mathbf{y}^{(j)}$ and $\mathbf{y}_{(0)}^{(j+1)}$ to vectors of length $2^{L^{(j)}}$, where $2^{L^{(j)}-1} < m^{(j)} \leq 2^{L^{(j)}}$, taking into account all nonzero entries. We will then show that $\mathbf{y}_{(0)}^{(j+1)}$ and thus $\mathbf{y}^{(j+1)}$ can be computed using essentially one IFFT of length $2^{L^{(j)}}$ and some further operations of complexity $\mathcal{O}\left(m^{(j)}\right)$. This requires the vector $\mathbf{y}^{(j)}$ known from the previous iteration step and $2^{L^{(j)}}$ suitably chosen oddly indexed entries of $\widehat{\mathbf{y}^{(j+1)}}$.

Theorem 5.20 (Theorem 2 in [BP18c]) *Let $N = 2^{J-1}$, $J \geq 2$, and $j \in \{0, \dots, J-1\}$. Let $\mathbf{x} \in \mathbb{R}^N$ have a one-block support of length $m < N$. Set $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T$ and assume that $\mathbf{y}$ satisfies (5.4). Suppose that $\mathbf{y}^{(j)}$ has a one-block support of length $m^{(j)}$. Assume that we have access to all entries of $\widehat{\mathbf{y}}$. Further, let $L^{(j)} \coloneqq \lceil \log_2 m^{(j)} \rceil \leq j$. Then $\mathbf{y}^{(j+1)}$ can be uniquely recovered from $\left(\widehat{y}_{2^{J-j-1}\left(2^{j+1-L^{(j)}}p+1\right)}\right)_{p=0}^{2^{L^{(j)}}-1}$ and $\mathbf{y}^{(j)}$.*

Proof. It suffices to only consider the oddly indexed entries $\widehat{y^{(j+1)}}_{2k+1}$ of $\widehat{\mathbf{y}^{(j+1)}}$, where $k \in \left\{0, \dots, 2^j - 1\right\}$. By Definition 5.7 we obtain for all $k \in \left\{0, \dots, 2^j - 1\right\}$ that

$$\begin{aligned}
&\widehat{y^{(j+1)}}_{2k+1} \\
&= \left(\left(\omega_{2^{j+1}}{}^{(2k+1)l'}\right)_{l'=0}^{2^{j+1}-1}\right)^T \begin{pmatrix} \mathbf{y}_{(0)}^{(j+1)} \\ \mathbf{y}_{(1)}^{(j+1)} \end{pmatrix} \\
&= \left(\left(\omega_{2^{j+1}}{}^{(2k+1)l'}\right)_{l'=0}^{2^j-1}\right)^T \mathbf{y}_{(0)}^{(j+1)} + \left(\left(\omega_{2^{j+1}}{}^{(2k+1)l'}\right)_{l'=2^j}^{2^{j+1}-1}\right)^T \left(\mathbf{y}^{(j)} - \mathbf{y}_{(0)}^{(j+1)}\right) \\
&= \left(\left(\omega_{2^{j+1}}{}^{(2k+1)l'}\right)_{l'=0}^{2^j-1}\right)^T \mathbf{y}_{(0)}^{(j+1)} + \left(\left(\omega_{2^{j+1}}{}^{(2k+1)l}\, \omega_2{}^{2k+1}\right)_{l=0}^{2^j-1}\right)^T \left(\mathbf{y}^{(j)} - \mathbf{y}_{(0)}^{(j+1)}\right) \\
&= \left(\left(\omega_{2^{j+1}}{}^{(2k+1)l}\right)_{l=0}^{2^j-1}\right)^T \mathbf{y}_{(0)}^{(j+1)} - \left(\left(\omega_{2^{j+1}}{}^{(2k+1)l}\right)_{l=0}^{2^j-1}\right)^T \left(\mathbf{y}^{(j)} - \mathbf{y}_{(0)}^{(j+1)}\right) \\
&= \left(\left(\omega_{2^{j+1}}{}^{(2k+1)l}\right)_{l=0}^{2^j-1}\right)^T \left(2\mathbf{y}_{(0)}^{(j+1)} - \mathbf{y}^{(j)}\right),
\end{aligned}$$

where we set $l \coloneqq l' - 2^j$ in the second summand. Using Lemma 5.10, we find that

$$\begin{aligned}
\left(\widehat{y}_{2^{J-j-1}(2k+1)}\right)_{k=0}^{2^j-1} &= \left(\widehat{y^{(j+1)}}_{2k+1}\right)_{k=0}^{2^j-1} \\
&= \left(\omega_{2^{j+1}}{}^{(2k+1)l}\right)_{k,\,l=0}^{2^j-1} \left(2\mathbf{y}_{(0)}^{(j+1)} - \mathbf{y}^{(j)}\right), \qquad (5.5)
\end{aligned}$$

so $\mathbf{y}_{(0)}^{(j+1)}$ can be computed from $\mathbf{y}^{(j)}$ and the oddly indexed entries of $\widehat{\mathbf{y}^{(j+1)}}$. However, if we had to compute $\mathbf{y}^{(j+1)}$ like this in every step where we have to apply the procedure for case A of Theorem 5.19, our IDFT algorithm could not achieve an overall runtime that is sublinear in the vector length $2N$. Recall that by assumption $\mathbf{y}^{(j)}$ has the support interval

$$S^{(j)} = I^{(j)}_{\mu^{(j)},\, 2^j-1-\mu^{(j)}} = I^{(j)}_{\mu^{(j)},\, \mu^{(j)}+m^{(j)}-1} \tag{5.6}$$

of length $m^{(j)} \leq 2m$ for some $\mu^{(j)}, \nu^{(j)} \in I^{(j)}_{0,\, 2^j-1}$. Then it follows from Definition 5.7 and Theorem 5.19, case A that the support set $S^{(j+1)}$ of $\mathbf{y}^{(j+1)}$ satisfies

$$S^{(j+1)} \subseteq I^{(j+1)}_{\mu^{(j)},\, \mu^{(j)}+m^{(j)}-1} \cup I^{(j+1)}_{2^j+\mu^{(j)},\, 2^j+\mu^{(j)}+m^{(j)}-1}. \tag{5.7}$$

Considering the first half $\mathbf{y}_{(0)}^{(j+1)} \in \mathbb{R}^{2^j}$ of $\mathbf{y}^{(j+1)}$ separately, Theorem 5.19 implies that it also has a one-block support. Analogously, $\mathbf{y}_{(1)}^{(j+1)}$ has a one-block support, which is illustrated by Figures 5.7 to 5.9. From now on we will denote the support of $\mathbf{y}_{(0)}^{(j+1)}$ by $S_{(0)}^{(j+1)}$ and the support of $\mathbf{y}_{(1)}^{(j+1)}$ by $S_{(1)}^{(j+1)}$, respectively. Then the following inclusion holds for $S^{(j+1)}$,

$$\begin{aligned} S^{(j+1)} \subseteq\ & \underbrace{\left\{\left(\mu^{(j)}+r\right) \bmod 2^j : r \in \left\{0,\dots,m^{(j)}-1\right\}\right\}}_{\supseteq S_{(0)}^{(j+1)}} \\ & \cup \underbrace{\left\{2^j + \left(\mu^{(j)}+r\right) \bmod 2^j : r \in \left\{0,\dots,m^{(j)}-1\right\}\right\}}_{\supseteq S_{(1)}^{(j+1)}}. \end{aligned} \tag{5.8}$$

Note that the sets in (5.7) and (5.8) are not the same, as the indices in (5.7) correspond to the support of $\mathbf{y}^{(j+1)} \in \mathbb{R}^{2^{j+1}}$ and are taken modulo 2^{j+1}. The first interval in (5.8) corresponds to the possibly nonzero entries of $\mathbf{y}_{(0)}^{(j+1)} \in \mathbb{R}^{2^j}$ and the second interval to the possibly nonzero entries of $\mathbf{y}_{(1)}^{(j+1)}$, and both are considered modulo 2^j. Thus, the support of $\mathbf{y}_{(0)}^{(j+1)}$ is contained in the first interval in (5.8), and the support of $\mathbf{y}_{(1)}^{(j+1)}$ is contained in the second interval. In particular, $\mathbf{y}^{(j+1)}$ can have at most $2m^{(j)}$ nonzero entries. By Definition 5.7 we have that

$$\mathbf{y}^{(j)} = \mathbf{y}_{(0)}^{(j+1)} + \mathbf{y}_{(1)}^{(j+1)}, \tag{5.9}$$

so, using the symmetry of $\mathbf{y}^{(j+1)}$, the second half $\mathbf{y}_{(1)}^{(j+1)} = \mathbf{J}_{2^j}\mathbf{y}_{(0)}^{(j+1)}$ can be computed via a permutation instead of by solving (5.9). Thus, it suffices to recover $\mathbf{y}_{(0)}^{(j+1)}$.

In order to decrease the runtime of our method, we define the restrictions of $\mathbf{y}^{(j)}$ and $\mathbf{y}_{(0)}^{(j+1)}$ to $2^{L^{(j)}}$-length vectors,

$$\mathbf{z}^{(j)} := \left(y^{(j)}_{\left(\mu^{(j)}+r\right) \bmod 2^j}\right)_{r=0}^{2^{L^{(j)}}-1} \quad \text{and} \quad \mathbf{z}_{(0)}^{(j+1)} := \left(y^{(j+1)}_{\left(\mu^{(j)}+r\right) \bmod 2^j}\right)_{r=0}^{2^{L^{(j)}}-1}.$$

By definition of $L^{(j)}$, we have that $2^{L^{(j)}-1} < m^{(j)} \leq 2^{L^{(j)}}$, and the $m^{(j)}$ nonzero entries of $\mathbf{y}^{(j)}$ are taken into account by the $2^{L^{(j)}}$-length restriction $\mathbf{z}^{(j)}$ of $\mathbf{y}^{(j)}$. Similarly, by

definition of the periodization, $\mathbf{z}_{(0)}^{(j+1)}$ takes into account the at most $m^{(j)}$ nonzero entries of $\mathbf{y}_{(0)}^{(j+1)}$. Hence, it is sufficient to recover the $2^{L^{(j)}}$-length vector $\mathbf{z}_{(0)}^{(j+1)}$ in order to determine $\mathbf{y}^{(j+1)}$. Restricting (5.5) to the vectors $\mathbf{z}^{(j)}$ and $\mathbf{z}_{(0)}^{(j+1)}$ yields

$$\begin{aligned}&\left(\widehat{y}_{2^{J-j-1}(2k+1)}\right)_{k=0}^{2^j-1}\\ &= \left(\omega_{2^{j+1}}{}^{(2k+1)\left(\left(\mu^{(j)}+r\right) \bmod 2^j\right)}\right)_{k,\,r=0}^{2^j-1,\,2^{L^{(j)}}-1} \cdot \left(2\mathbf{z}_{(0)}^{(j+1)} - \mathbf{z}^{(j)}\right). \end{aligned} \tag{5.10}$$

As $\mathbf{z}_{(0)}^{(j+1)}$ and $\mathbf{z}^{(j)}$ have length $2^{L^{(j)}}$, we only have to consider $2^{L^{(j)}}$ equations of (5.10). We choose the ones corresponding to $k_p \coloneqq 2^{j-L^{(j)}}p$ for $p \in \left\{0,\dots,2^{L^{(j)}}-1\right\}$, since $2k_p+1 \in I_{0,\,2^{j+1}-1}^{(j+1)}$ for all p. Then we obtain the factorization

$$\begin{aligned}&\left(\widehat{y}_{2^{J-j-1}\left(2^{j+1-L^{(j)}}p+1\right)}\right)_{p=0}^{2^{L^{(j)}}-1}\\ &= \left(\omega_{2^{j+1}}{}^{\left(2^{j+1-L^{(j)}}p+1\right)\left(\left(\mu^{(j)}+r\right) \bmod 2^j\right)}\right)_{p,\,r=0}^{2^{L^{(j)}}-1} \cdot \left(2\mathbf{z}_{(0)}^{(j+1)} - \mathbf{z}^{(j)}\right)\\ &= \left(\omega_{2^j}{}^{2^{j-L^{(j)}}p\left(\left(\mu^{(j)}+r\right) \bmod 2^j\right)}\,\omega_{2^{j+1}}{}^{\left(\mu^{(j)}+r\right) \bmod 2^j}\right)_{p,\,r=0}^{2^{L^{(j)}}-1} \cdot \left(2\mathbf{z}_{(0)}^{(j+1)} - \mathbf{z}^{(j)}\right)\\ &= \left(\omega_{2^{L^{(j)}}}{}^{p\mu^{(j)}}\,\omega_{2^{L^{(j)}}}{}^{pr}\,\omega_{2^{j+1}}{}^{\left(\mu^{(j)}+r\right) \bmod 2^j}\right)_{p,\,r=0}^{2^{L^{(j)}}-1} \cdot \left(2\mathbf{z}_{(0)}^{(j+1)} - \mathbf{z}^{(j)}\right)\\ &= \operatorname{diag}\left(\omega_{2^{L^{(j)}}}{}^{p\mu^{(j)}}\right)_{p=0}^{2^{L^{(j)}}-1} \cdot \left(\omega_{2^{L^{(j)}}}{}^{pr}\right)_{p,\,r=0}^{2^{L^{(j)}}-1} \cdot \operatorname{diag}\left(\omega_{2^{j+1}}{}^{\left(\mu^{(j)}+r\right) \bmod 2^j}\right)_{r=0}^{2^{L^{(j)}}-1}\\ &\quad \cdot \left(2\mathbf{z}_{(0)}^{(j+1)} - \mathbf{z}^{(j)}\right)\\ &= \mathbf{W}_{(0)}^{(j)} \cdot \mathbf{F}_{2^{L^{(j)}}} \cdot \mathbf{W}_{(1)}^{(j)} \left(2\mathbf{z}_{(0)}^{(j+1)} - \mathbf{z}^{(j)}\right), \end{aligned} \tag{5.11}$$

where

$$\mathbf{W}_{(0)}^{(j)} \coloneqq \operatorname{diag}\left(\omega_{2^{L^{(j)}}}{}^{p\mu^{(j)}}\right)_{p=0}^{2^{L^{(j)}}-1} \quad \text{and} \quad \mathbf{W}_{(1)}^{(j)} \coloneqq \operatorname{diag}\left(\omega_{2^{j+1}}{}^{\left(\mu^{(j)}+r\right) \bmod 2^j}\right)_{r=0}^{2^{L^{(j)}}-1}.$$

Since all matrices occurring in (5.11) are invertible, we find that

$$\mathbf{z}_{(0)}^{(j+1)} = \frac{1}{2}\left(\mathbf{W}_{(1)}^{(j)^{-1}} \cdot \mathbf{F}_{2^{L^{(j)}}}{}^{-1} \cdot \mathbf{W}_{(0)}^{(j)^{-1}} \cdot \left(\widehat{y}_{2^{J-j-1}\left(2^{j+1-L^{(j)}}p+1\right)}\right)_{p=0}^{2^{L^{(j)}}-1} + \mathbf{z}^{(j)}\right).$$

Hence, $\mathbf{y}^{(j+1)}$ is completely determined by its symmetry, guaranteed by Lemma 5.9 (iii), and the definition of $\mathbf{z}_{(0)}^{(j+1)}$, with

$$\left(y_{(0)}^{(j+1)}\right)_{\left(\mu^{(j)}+k\right) \bmod 2^j} = \begin{cases} \left(z_{(0)}^{(j+1)}\right)_k & \text{if } k \in \left\{0,\dots,2^{L^{(j)}}-1\right\},\\ 0 & \text{else,}\end{cases}$$

$$\left(y_{(1)}^{(j+1)}\right)_{2^j-1-\left(\mu^{(j)}+k\right) \bmod 2^j} = \begin{cases} \left(z_{(0)}^{(j+1)}\right)_k & \text{if } k \in \left\{0,\dots,2^{L^{(j)}}-1\right\}, \\ 0 & \text{else.} \end{cases}$$

Note that if $2^{L^{(j)}} = 2^j$, i.e., if $2^{j-1} \leq m^{(j)} \leq 2^j$, then $\mathbf{z}^{(j)} = \mathbf{y}^{(j)}$ and $\mathbf{z}_{(0)}^{(j+1)} = \mathbf{y}_{(0)}^{(j+1)}$. Consequently, (5.5) yields that

$$\begin{aligned} \left(\widehat{y}_{2^{J-j-1}(2k+1)}\right)_{k=0}^{2^j-1} &= \left(\widehat{y^{(j+1)}}_{2k+1}\right)_{k=0}^{2^j-1} \\ &= \left(\omega_{2^{j+1}}{}^{(2k+1)l}\right)_{k,\,l=0}^{2^j-1} \cdot \left(2\mathbf{y}_{(0)}^{(j+1)} - \mathbf{y}^{(j)}\right) \\ &= \left(\omega_{2^j}{}^{kl}\,\omega_{2^{j+1}}{}^{l}\right)_{k,\,l=0}^{2^j-1} \cdot \left(2\mathbf{y}_{(0)}^{(j+1)} - \mathbf{y}^{(j)}\right) \\ &= \mathbf{F}_{2^j} \cdot \operatorname{diag}\left(\left(\omega_{2^{j+1}}{}^{l}\right)_{l=0}^{2^j-1}\right) \cdot \left(2\mathbf{y}_{(0)}^{(j+1)} - \mathbf{y}^{(j)}\right) \end{aligned}$$

and thus

$$\mathbf{y}_{(0)}^{(j+1)} = \frac{1}{2}\left(\operatorname{diag}\left(\left(\omega_{2^{j+1}}{}^{-l}\right)_{l=0}^{2^j-1}\right) \cdot \mathbf{F}_{2^j}{}^{-1} \cdot \left(\widehat{y}_{2^{J-j-1}(2k+1)}\right)_{k=0}^{2^j-1} + \mathbf{y}^{(j)}\right).$$

□

As the recovery of $\mathbf{y}^{(j+1)}$ from $\mathbf{y}^{(j)}$ and $\widehat{\mathbf{y}}$ thus requires essentially an IFFT of length $2^{L^{(j)}} \leq 2m$, we have reason to believe that this procedure for case A sufficiently speeds up the overall runtime of our method. Note that we do not need to know m a priori; it suffices if the support of the periodization $\mathbf{y}^{(j)}$ from the previous iteration step is known.

5.5.2 Recovery Procedure for Case B: Two-Block Support

We still have to devise a procedure for reconstructing $\mathbf{y}^{(j+1)}$ from $\mathbf{y}^{(j)}$ in case B of Theorem 5.19, i.e., if $\mathbf{y}^{(j)}$ has a two-block support of the form

$$S^{(j)} = I^{(j)}_{\mu^{(j)},\,\nu^{(j)}} \cup I^{(j)}_{2^j-1-\nu^{(j)},\,2^j-1-\mu^{(j)}} \tag{5.12}$$

with two blocks of length $n^{(j)} = m$, and $\nu^{(j)} = \mu^{(j)} + m - 1$. We recall that it follows from Theorem 5.19, case B that the support of $\mathbf{y}^{(j+1)}$ satisfies

$$S^{(j+1)} = I^{(j+1)}_{\mu^{(j+1)},\,\nu^{(j+1)}} \cup I^{(j+1)}_{2^{j+1}-1-\nu^{(j+1)},\,2^{j+1}-1-\mu^{(j+1)}}$$

with $\mu^{(j+1)} = \mu^{(j)}$ or $2^{j+1}-1-\nu^{(j+1)} = 2^j + \mu^{(j)}$. Here, the lengths $n^{(j)}$ and $n^{(j+1)}$ of the blocks in $\mathbf{y}^{(j)}$ and $\mathbf{y}^{(j+1)}$, respectively, are the same. They also coincide with the support length m of $\mathbf{x}$. Furthermore, the values of the nonzero entries of $\mathbf{y}^{(j+1)}$ are the same as the values of the nonzero entries of $\mathbf{y}^{(j)}$ and are thus already determined; we just have to find out whether the first support block of $\mathbf{y}^{(j)}$ remains at the same position in $\mathbf{y}^{(j+1)}$ or whether it is shifted by 2^j. The other support block is obtained as the reflection of this block according to Lemma 5.9 (iii), see also Figure 5.10. Which of the two possibilities for $\mathbf{y}^{(j+1)}$ is true can be decided by comparing the DFTs of the two possible vectors at an oddly indexed entry for which the corresponding entry of $\widehat{\mathbf{y}^{(j+1)}}$ has to be nonzero. In a first step we show that such an oddly indexed nonzero entry of $\widehat{\mathbf{y}^{(j+1)}}$ can be found by examining at most $2m$ entries.

Lemma 5.21 (Lemma 6 in [BP18c]) Let $N = 2^{J-1}$ with $J \geq 2$ and $j \in \{0, \dots, J-1\}$. Let $\mathbf{x} \in \mathbb{R}^N$ have a one-block support of length $m < N$. Set $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T$ and assume that $\mathbf{y}$ satisfies (5.4). Suppose that $\mathbf{y}^{(j)}$ has a two-block support. Assume that we have access to all entries of $\widehat{\mathbf{y}}$. Then $\left(\widehat{y^{(j+1)}}_{2k+1}\right)_{k=0}^{2m-1}$ has at least one nonzero entry.

Proof. If $\mathbf{y}^{(j)}$ has the two-block support

$$S^{(j)} = I^{(j)}_{\mu^{(j)},\,\nu^{(j)}} \cup I^{(j)}_{2^j-1-\nu^{(j)},\,2^j-1-\mu^{(j)}}$$

of length $n^{(j)} = \nu^{(j)} - \mu^{(j)} + 1 = m$ for first and last support indices $\mu^{(j)}, \nu^{(j)} \in I^{(j)}_{0,\,2^j-1}$, then by case B of Theorem 5.19 $\mathbf{y}^{(j+1)}$ has the two-block support $S^{(j+1)}$ with two blocks of length $n^{(j+1)} = n^{(j)} = m$ and either

$$S^{(j+1)} = I^{(j+1)}_{\mu^{(j)},\,\mu^{(j)}+m-1} \cup I^{(j+1)}_{2^{j+1}-m-\mu^{(j)},\,2^{j+1}-1-\mu^{(j)}} \qquad \text{or} \tag{5.13}$$

$$S^{(j+1)} = I^{(j+1)}_{2^j-m-\mu^{(j)},\,2^j-1-\mu^{(j)}} \cup I^{(j+1)}_{2^j+\mu^{(j)},\,2^j+\mu^{(j)}+m-1}. \tag{5.14}$$

We want to guarantee the existence of an oddly indexed nonzero entry of $\widehat{\mathbf{y}^{(j+1)}}$. Considering the first $2m$ oddly indexed entries of $\widehat{\mathbf{y}}^{(j+1)}$, we find

$$\begin{aligned}
&\left(\widehat{y^{(j+1)}}_{2k+1}\right)_{k=0}^{2m-1} \\
&= \left(\sum_{l \in S^{(j+1)}} {\omega_{2^{j+1}}}^{(2k+1)l} y_l^{(j+1)}\right)_{k=0}^{2m-1} \\
&= \left({\omega_{2^j}}^{kl}\, {\omega_{2^{j+1}}}^{l}\right)_{k=0,\,l\in S^{(j+1)}}^{2m-1} \cdot \left(y_l^{(j+1)}\right)_{l\in S^{(j+1)}} \\
&= \left(\left({\omega_{2^j}}^{l}\right)^k\right)_{k=0,\,l\in S^{(j+1)}}^{2m-1} \cdot \operatorname{diag}\left(\left({\omega_{2^{j+1}}}^{l}\right)_{l\in S^{(j+1)}}\right) \cdot \left(y_l^{(j+1)}\right)_{l\in S^{(j+1)}}.
\end{aligned} \tag{5.15}$$

Assume that the claim is false, i.e., $\left(\widehat{y^{(j+1)}}_{2k+1}\right)_{k=0}^{2m-1} = \mathbf{0}_{2m}$. Since $\left|S^{(j+1)}\right| = 2m$, the first matrix in (5.15), $\left(\left({\omega_{2^j}}^{l}\right)^k\right)_{k=0,\,l\in S^{(j+1)}}^{2m-1}$, is a square Vandermonde matrix according to Definition 4.10. Thus, by Lemma 4.11, it is invertible if and only if the ${\omega_{2^j}}^{l}$ are pairwise distinct for all $l \in S^{(j+1)}$, or, equivalently, if the residues $l \bmod 2^j$ are pairwise distinct for all $l \in S^{(j+1)}$. It follows from (5.13) and (5.14) that

$$\left\{l \bmod 2^j : l \in S^{(j+1)}\right\} = I^{(j)}_{\mu^{(j)},\,\mu^{(j)}+m-1} \cup I^{(j)}_{2^j-m-\mu^{(j)},\,2^j-1-\mu^{(j)}} = S^{(j)},$$

see also Figure 5.10. Consequently, since $\mathbf{y}^{(j)}$ already has a two-block support with separated blocks, the residues modulo 2^j of all $l \in S^{(j+1)}$ have to be pairwise distinct as well. The second matrix in (5.15) is invertible by construction. Hence, under our assumption, said equation is equivalent to

$$\mathbf{0}_{2m} = \left(y_l^{(j+1)}\right)_{l\in S^{(j+1)}}.$$

However, we have that $\left(y_l^{(j+1)}\right)_{l\in S^{(j+1)}} \neq \mathbf{0}_{2m}$ due to the reflected block support of $\mathbf{y}$

and Definition 5.7. Thus, we obtain a contradiction, implying that there indeed exists an oddly indexed nonzero entry $\widehat{y^{(j+1)}}_{2k_0+1} \neq 0$ with $k_0 \in \{0, \ldots, 2m-1\}$. □

Remark 5.22 In order to obtain an efficient and stable implementation for the procedure for case B, we will later set

$$k_0 \coloneqq \operatorname*{argmax}_{k \in \{0,\ldots,2m-1\}} \left\{ \left| \widehat{y}_{2^{J-j-1}(2k+1)} \right| \right\},$$

employing that the entries of $\widehat{\mathbf{y}^{(j+1)}}$ are given via Lemma 5.10. Then $\widehat{y^{(j+1)}}_{2k_0+1} \neq 0$ and it is likely that this entry is not too close to zero, which is supported empirically by the numerical experiments in Section 6.5. ◊

Now that it is guaranteed that there is at least one nonzero entry among the first $2m$ oddly indexed entries of $\widehat{\mathbf{y}^{(j+1)}}$, we can show how the support of $\mathbf{y}^{(j+1)}$ can be determined from $\mathbf{y}^{(j)}$ and such a nonzero entry.

Theorem 5.23 (Theorem 3 in [BP18c]) *Let $N = 2^{J-1}$, $J \geq 2$, and $j \in \{0, \ldots, J-1\}$. Let $\mathbf{x} \in \mathbb{R}^N$ have a one-block support of length $m < N$. Set $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T$ and assume that $\mathbf{y}$ satisfies (5.4). Suppose that $\mathbf{y}^{(j)}$ has a two-block support. Assume that we have access to all entries of $\widehat{\mathbf{y}} = (\widehat{y}_k)_{k=0}^{2^J-1}$. Then $\mathbf{y}^{(j+1)}$ can be uniquely recovered from $\mathbf{y}^{(j)}$ and one nonzero entry of $\left(\widehat{y}_{2^{J-j-1}(2k+1)}\right)_{k=0}^{2m-1}$.*

Proof. If $\mathbf{y}^{(j)}$ is known and has the two-block support

$$S^{(j)} = I^{(j)}_{\mu^{(j)},\,\nu^{(j)}} \cup I^{(j)}_{2^j-1-\nu^{(j)},\,2^j-1-\mu^{(j)}}$$

of length $n^{(j)} = m$ for some $\mu^{(j)}, \nu^{(j)} \in I^{(j)}_{0,\,2^j-1}$, there are two possibilities for the periodized vector $\mathbf{y}^{(j+1)}$. Theorem 5.19, case B yields that one of the possible vectors is obtained by shifting the other one by 2^j, see also Figure 5.10. We denote these two possible vectors by $\mathbf{u}^0 = \left(u^0_k\right)_{k=0}^{2^{j+1}-1}$ and $\mathbf{u}^1 = \left(u^1_k\right)_{k=0}^{2^{j+1}-1}$, and their support intervals by $S\left(\mathbf{u}^0\right)$ and $S\left(\mathbf{u}^1\right)$. Then $S\left(\mathbf{u}^0\right)$ and $S\left(\mathbf{u}^1\right)$ satisfy that

$$\begin{aligned} S\left(\mathbf{u}^0\right) &= I^{(j+1)}_{\mu^{(j)},\,\nu^{(j)}} \cup I^{(j+1)}_{2^{j+1}-1-\nu^{(j)},\,2^{j+1}-1-\mu^{(j)}} && \text{and} \\ S\left(\mathbf{u}^1\right) &= I^{(j+1)}_{2^j-1-\nu^{(j)},\,2^j-1-\mu^{(j)}} \cup I^{(j+1)}_{2^j+\mu^{(j)},\,2^j+\nu^{(j)}}. \end{aligned}$$

Thus, we obtain

$$u^0_k \coloneqq \begin{cases} y^{(j)}_k & \text{if } k \in \left\{\mu^{(j)}, \ldots, \mu^{(j)}+m-1\right\}, \\ y^{(j)}_{k-2^j} & \text{if } k \in \left\{2^{j+1}-m-\mu^{(j)}, \ldots, 2^{j+1}-1-\mu^{(j)}\right\}, \\ 0 & \text{else} \end{cases} \qquad \text{and}$$

$$u^1_k \coloneqq u^0_{(2^j+k) \bmod 2^{j+1}}, \quad \forall\, k \in \left\{0, \ldots, 2^{j+1}-1\right\}.$$

Lemma 1.5 implies that

$$\widehat{u^1}_{2k+1} = -\widehat{u^0}_{2k+1}, \qquad k \in \left\{0, \ldots, 2^j-1\right\}, \tag{5.16}$$

for all oddly indexed entries of $\widehat{\mathbf{u}^0}$ and $\widehat{\mathbf{u}^1}$, since $\mathbf{u}^1$ is the periodic 2^j-shift of $\mathbf{u}^0$. In order to decide whether $\mathbf{y}^{(j+1)} = \mathbf{u}^0$ or $\mathbf{y}^{(j+1)} = \mathbf{u}^1$, we compare a nonzero entry

$$\widehat{y^{(j+1)}}_{2k_0+1} = \widehat{y}_{2^{J-j-1}(2k_0+1)} \neq 0$$

to the corresponding entry of $\widehat{\mathbf{u}^0}$. It follows from Lemma 5.21 that such an entry can be found by examining the first $2m$ oddly indexed entries of $\widehat{\mathbf{y}^{(j+1)}}$. If $\widehat{u^0}_{2k_0+1} = \widehat{y^{(j+1)}}_{2k_0+1}$, we conclude that $\mathbf{y}^{(j+1)} = \mathbf{u}^0$, and if $\widehat{u^0}_{2k_0+1} = -\widehat{y^{(j+1)}}_{2k_0+1}$, then $\mathbf{y}^{(j+1)} = \mathbf{u}^1$ by (5.16). Numerically, we set $\mathbf{y}^{(j+1)} = \mathbf{u}^0$ if

$$\left|\widehat{u^0}_{2k_0+1} - \widehat{y^{(j+1)}}_{2k_0+1}\right| < \left|\widehat{u^0}_{2k_0+1} + \widehat{y^{(j+1)}}_{2k_0+1}\right|,$$

and $\mathbf{y}^{(j+1)} = \mathbf{u}^1$ otherwise. The required entry of $\widehat{\mathbf{u}^0}$ can be computed from $\mathbf{y}^{(j)}$ using $\mathcal{O}(m)$ operations,

$$\begin{aligned}\widehat{u^0}_{2k_0+1} &= \sum_{l=0}^{2^{j+1}-1} \omega_{2^{j+1}}^{(2k_0+1)l}\, u_l^0 \\ &= \sum_{l=\mu^{(j)}}^{\mu^{(j)}+m-1} \omega_{2^{j+1}}^{(2k_0+1)l}\, y_l^{(j)} + \sum_{l=2^{j+1}-m-\mu^{(j)}}^{2^{j+1}-1-\mu^{(j)}} \omega_{2^{j+1}}^{(2k_0+1)l}\, y_{l-2^j}^{(j)},\end{aligned}$$

since the support of $\mathbf{y}^{(j)}$ and thus of $\mathbf{u}^0$ is known from the previous iteration step. □

The recovery of $\mathbf{y}^{(j+1)}$ from $\mathbf{y}^{(j)}$ and an oddly indexed nonzero entry of $\widehat{\mathbf{y}^{(j+1)}}$ via the procedure for case B requires at most $2m$ samples of $\widehat{\mathbf{y}}$ and $\mathcal{O}(m)$ arithmetical operations, so this approach also contributes sufficiently to obtaining an overall runtime of our sparse IDFT method that is sublinear in the vector length $2N$. Note that, as for the method introduced in Section 5.5.1, a priori knowledge of the block length m of $\mathbf{y}$ is not necessary.

5.6 Sparse Fast IDFT and Sparse Fast IDCT-II

In Section 5.5 we presented the procedures necessary to derive the new sparse fast IDFT for vectors $\mathbf{y} \in \mathbb{R}^{2N}$, $N = 2^{J-1}$, that have a reflected block support and satisfy (5.4). Using Lemma 5.4, we obtain at the same time a new sparse fast IDCT-II algorithm for vectors with one-block support. Note that neither of the procedures for reconstructing $\mathbf{y}^{(j+1)}$ from $\mathbf{y}^{(j)}$ and $\widehat{\mathbf{y}}$ introduced in Section 5.5 requires a priori knowledge of the length of the blocks in $\mathbf{y}$. However, what these procedures do require is the support structure of the vector $\mathbf{y}^{(j)}$ from the previous iteration step. More precisely, we need to detect whether $\mathbf{y}^{(j)}$ has a one-block or a two-block support and determine the corresponding first and last support indices $\mu^{(j)}$ and $\nu^{(j)}$, as well as the support length $m^{(j)}$ or block length $n^{(j)}$.

5.6.1 Detecting the Support Sets

In Sections 5.5.1 and 5.5.2 we showed how to compute $\mathbf{y}^{(j+1)}$ from $\mathbf{y}^{(j)}$ and $\widehat{\mathbf{y}}$. Both reconstruction methods introduced above rely heavily on the fact that the support structure of $\mathbf{y}^{(j)}$ is already known from the previous step. However, if $\mathbf{y}^{(j)}$ has a one-block support, the reconstruction method from Theorem 5.20 does not provide us with both

the block or support length $n^{(j+1)}$ or $m^{(j+1)}$ and the first support index $\mu^{(j+1)}$, even though we need both to recover $\mathbf{y}^{(j+2)}$ from $\mathbf{y}^{(j+1)}$ in the next iteration step. In this section we introduce methods for the stable and efficient detection of the first support index and the block or support length $n^{(j+1)}$ or $m^{(j+1)}$. These methods are designed for noisy data, as input data is usually noisy in practical applications, but can of course also be applied to exact data. Note that for noisy data the found block lengths $n^{(j+1)}$ for $\mathbf{y}^{(j+1)}$ with two-block support might not be the same as the exact block length m of $\mathbf{x}$. For detecting the support sets efficiently, we choose a threshold $\varepsilon > 0$ depending on the noise level of the data.

A) One-block support:

A_1) $\mathbf{y}^{(j)}$ has the one-block support $S^{(j)} = I^{(j)}_{\mu^{(j)},\, 2^j-1-\mu^{(j)}}$ of length $m^{(j)} < 2^j$ centered around the middle of the vector.

Theorem 5.19, case A_1 implies that $\mathbf{y}^{(j+1)}$ has the two-block support

$$S^{(j+1)} = I^{(j+1)}_{\mu^{(j+1)},\,\nu^{(j+1)}} \cup I^{(j+1)}_{2^{j+1}-1-\nu^{(j+1)},\,2^{j+1}-1-\mu^{(j+1)}}$$

of length $n^{(j+1)} \leq m^{(j)}$. Further, we know that the support interval $I^{(j+1)}_{\mu^{(j+1)},\,\nu^{(j+1)}}$ of the first block of $\mathbf{y}^{(j+1)}$ is a subset of $S^{(j)}$. By the proof of Theorem 5.19 and the symmetry guaranteed by Lemma 5.9 (iii), the indices corresponding to the significantly large entries of the first block of $\mathbf{y}^{(j+1)}$ have to be contained in the set

$$\begin{aligned} T^{(j+1)}_{(0)} &:= \left\{ k \in \left\{ \mu^{(j)}, \mu^{(j)}+1, \ldots, 2^j - 1 - \mu^{(j)} \right\} : \left| y^{(j+1)}_k \right| > \varepsilon \right\} \\ &=: \{t_1, \ldots, t_K\}, \end{aligned}$$

where $t_1 < \cdots < t_K$. Thus, we define

$$\mu^{(j+1)} := t_1 \quad \text{and} \quad n^{(j+1)} := t_K - t_1 + 1.$$

Hence, in order to find the first support index and the support length, we have to inspect the $m^{(j)}$ entries of $\mathbf{y}^{(j+1)}$ corresponding to indices in $T^{(j+1)}_{(0)}$, which has a runtime of $\mathcal{O}\left(m^{(j)}\right)$.

A_2) $\mathbf{y}^{(j)}$ has the one-block support $S^{(j)} = I^{(j)}_{0,\,2^j-1}$ of length $m^{(j)} = 2^j$.

If $j < J-1$, Theorem 5.19, case A_2 yields that $\mathbf{y}^{(j+1)}$ has a one-block support whose location and length are unknown. As $\mathbf{y}^{(j+1)}$ is symmetric, the indices of its significantly large entries are

$$\begin{aligned} T^{(j+1)} &:= \left\{ k \in I^{(j+1)}_{0,\,2^{j+1}-1} : \left| y^{(j+1)}_k \right| > \varepsilon \right\} \\ &=: \left\{ t_1, \ldots, t_K, 2^{j+1} - 1 - t_K, \ldots, 2^{j+1} - 1 - t_1 \right\} =: \{t_1, \ldots, t_{2K}\}, \end{aligned}$$

where $t_K \leq 2^j - 1$ and $t_{K+1} \geq 2^j$. If $t_{2K} - t_1 + 1 > 2^j$, we set $\mu^{(j+1)} := 0$ and $m^{(j+1)} := 2^{j+1}$, as we are not able to correctly determine the first support index for support lengths that are greater than half of the vector length. Otherwise, we need to detect whether the support block is centered around the middle or the boundary of $\mathbf{y}^{(j+1)}$. We define

$$d_0 := t_{K+1} - t_K \quad \text{and} \quad d_1 := (t_1 - t_{2K}) \bmod 2^{j+1},$$

since not all entries belonging to the support block have to be significantly large.

If $d_0 < d_1$, i.e., if the distance between the first significantly large entry to the left and right of the middle of the vector is smaller than the periodic distance between the first and last significantly large entry of the vector, then $\mathbf{y}^{(j+1)}$ has a one-block support centered around $2^j - 1$ and 2^j. Analogously, if $d_0 > d_1$, the support block is centered around 0 and $2^{j+1} - 1$. If $d_0 = d_1$, then $\mathbf{y}^{(j+1)}$ must have full support of length $m^{(j+1)} = 2^{j+1}$, so we can conclude that

$$\mu^{(j+1)} := \begin{cases} t_1 & \text{if } d_0 < d_1, \\ t_{K+1} & \text{if } d_0 > d_1, \\ 0 & \text{if } d_0 = d_1, \end{cases}$$

and

$$m^{(j+1)} := \begin{cases} t_{2K} - t_1 + 1 & \text{if } d_0 < d_1, \\ 2^{j+1} - t_{K+1} + t_K + 1 & \text{if } d_0 > d_1, \\ 2^{j+1} & \text{if } d_0 = d_1. \end{cases}$$

By symmetry of $\mathbf{y}^{(j+1)}$ it suffices to find $t_1, \ldots, t_K$; hence, we only have to inspect $2^j = m^{(j)}$ entries of $\mathbf{y}^{(j+1)}$.

In the case that $j = J - 1$, there are some additional possibilities. Due to its symmetry, it suffices to check

$$T_{(0)}^{(J)} := \left\{k \in \left\{0, \ldots, 2^{J-2} - 1\right\} : |y_k| > \varepsilon\right\} =: \{t_1, \ldots, t_K\} \qquad \text{and}$$
$$T_{(1)}^{(J)} := \left\{k \in \left\{2^{J-2}, \ldots, 2^{J-1} - 1\right\} : |y_k| > \varepsilon\right\} =: \{u_1, \ldots, u_L\}.$$

If $T_{(0)}^{(J)} = \emptyset$, then $\mathbf{y}$ has a one-block support centered around the middle with

$$\mu^{(J)} := u_1 \quad \text{and} \quad m^{(J)} := 2\left(2^{J-1} - u_1\right) = 2m.$$

If $T_{(1)}^{(J)} = \emptyset$, then $\mathbf{y}$ has a one-block support centered around the boundary with

$$\mu^{(J)} := 2^J - 1 - t_K \quad \text{and} \quad m^{(J)} := 2\left(t_K + 1\right) = 2m.$$

Otherwise, there are three possibilities for the support of $\mathbf{y}$, where we do not know the correct one a priori:

(i) $S^{(J)} := I_{t_1, 2^J - 1 - t_1}^{(J)}$,

(ii) $S^{(J)} := I_{2^J - 1 - u_L, u_L}^{(J)}$,

(iii) $\mathbf{y}$ has a two-block support with two blocks of possibly different lengths and unknown positions.

However, if case A_2 occurs for $j = J - 1$, then $\mathbf{y}^{(J-1)}$ has a one-block support of full length $m^{(J-1)} = 2^{J-1} \leq 2m$. Consequently, the vector $\mathbf{x}$ is not really sparse and the first support index of $\mathbf{x}$ might not be uniquely determined. Additionally, the iteration stops with $\mathbf{y}$, so not being able to detect its support uniquely does not pose an algorithmic problem. As we have to check $2^{J-1} = m^{(J-1)}$ entries of $\mathbf{y}^{(J)}$, this yields an arithmetical complexity of $\mathcal{O}\left(m^{(J-1)}\right)$.

A$_3$) $\mathbf{y}^{(j)}$ has the one-block support $S^{(j)} = I^{(j)}_{\mu^{(j)},\,2^j-1-\mu^{(j)}}$ of length $m^{(j)} < 2^j$ centered around the boundary of the vector and $j < J-1$.

It follows from Theorem 5.19, case A$_3$ that $\mathbf{y}^{(j+1)}$ has a one-block support of length $m^{(j+1)} := m^{(j)}$ with $\mu^{(j+1)} = \mu^{(j)}$ or $\mu^{(j+1)} = 2^j + \mu^{(j)}$. We compare the entries at the possible locations of the support block and set

$$e_0 := \sum_{k=\mu^{(j)}}^{\mu^{(j)}+m^{(j)}-1} \left|y_k^{(j+1)}\right| \quad \text{and} \quad e_1 := \sum_{k=2^j+\mu^{(j)}}^{\left(2^j+\mu^{(j)}+m^{(j)}-1\right) \bmod 2^{j+1}} \left|y_k^{(j+1)}\right|.$$

Since for exact data one of the sums has only vanishing summands, see Figure 5.8, we choose

$$\mu^{(j+1)} := \begin{cases} \mu^{(j)} & \text{if } e_0 > e_1, \\ 2^j + \mu^{(j)} & \text{if } e_0 < e_1. \end{cases}$$

This detection procedure has a computational effort of $\mathcal{O}\left(m^{(j)}\right)$.

A$_4$) $\mathbf{y}^{(J-1)}$ has the one-block support $S^{(J-1)} = I^{(J-1)}_{\mu^{(J-1)},\,2^{J-1}-1-\mu^{(J-1)}}$ of length $m^{(J-1)} < 2^{J-1}$ centered around the boundary.

Theorem 5.19, case A$_4$ implies that $\mathbf{y}^{(J)} = \mathbf{y}$ has the support

$$S^{(J)} = I^{(J)}_{\mu^{(J)},\,2^J-1-\mu^{(J)}} \cup I^{(J)}_{\eta^{(J)},\,2^J-1-\eta^{(J)}} \quad \text{with} \quad \mu^{(J)} < 2^{J-1} \le \eta^{(J)}. \tag{5.17}$$

This is either a two-block support with two separated blocks of possibly different lengths or, as a boundary case, a one-block support, where one of the two blocks in (5.17) vanishes. In the case of two blocks, one is centered around the middle of the vector, i.e., around $2^{J-1}-1$ and 2^{J-1}, and its support is a subset of $I^{(J)}_{\mu^{(J-1)},\,2^J-1-\mu^{(J-1)}}$, and the other one is centered around the boundary of the vector, i.e., around 2^J-1 and 0, and its support is a subset of $I^{(J)}_{2^{J-1}+\mu^{(J-1)},\,2^{J-1}-1-\mu^{(J-1)}}$. If $\mathbf{y}$ has a one-block support, which can only happen if the first support index $\mu^{\mathbf{x}}$ of $\mathbf{x}$ is 0 or its last support index $\nu^{\mathbf{x}}$ is $2^{J-1}-1$, then one of the two blocks in (5.17) is empty. Since the support blocks have even length by Lemma 5.4 (i), set

$$T^{(J)}_{(0)} := \left\{k \in \left\{\mu^{(J-1)}, \ldots, 2^{J-1}-1\right\} : |y_k| > \varepsilon\right\} =: \{t_1, \ldots, t_K\} \qquad \text{and}$$
$$T^{(J)}_{(1)} := \left\{k \in \left\{2^{J-1}+\mu^{(J-1)}, \ldots, 2^J-1\right\} : |y_k| > \varepsilon\right\} =: \{u_1, \ldots, u_L\},$$

see also Figure 5.9. If $T^{(J)}_{(0)} = \emptyset$, then $\mathbf{y}$ has a one-block support centered around the boundary. In this case we set

$$\mu^{(J)} := u_1 \quad \text{and} \quad m^{(J)} := 2 \cdot \left(2^J - u_1\right) = 2m$$

to obtain the support interval $S^{(J)} := I^{(J)}_{\mu^{(J)},\,2^J-1-\mu^{(J)}}$ of $\mathbf{y}$. If $T^{(J)}_{(1)} = \emptyset$, then $\mathbf{y}$ has a one-block support centered around the middle of the vector, and we set

$$\mu^{(J)} := t_1 \quad \text{and} \quad m^{(J)} := 2 \cdot \left(2^{J-1} - t_1\right) = 2m,$$

implying that the support interval of $\mathbf{y}$ is $S^{(J)} := I^{(J)}_{\mu^{(J)},\,\mu^{(J)}+m^{(J)}-1}$. If neither $T^{(J)}_{(0)}$ nor $T^{(J)}_{(1)}$ is empty, $\mathbf{y}$ has a two-block support with two separated blocks of possibly

different lengths. Recall that we denote the first index of the block centered around the middle by $\mu^{(J)}$ and the first index of the block centered around the boundary by $\eta^{(J)}$ and set

$$\mu^{(J)} \coloneqq t_1 \quad \text{and} \quad \eta^{(J)} \coloneqq u_1.$$

We obtain the support set $S^{(J)} \coloneqq I^{(J)}_{\mu^{(J)},\, 2^J-1-\mu^{(J)}} \cup I^{(J)}_{\eta^{(J)},\, 2^J-1-\eta^{(J)}}$, where the block centered around the middle of the vector has length $n^{(J)}_{(0)} \coloneqq 2\left(2^{J-1}-t_1\right)$ and the block centered around the boundary has length $n^{(J)}_{(1)} \coloneqq 2\left(2^J-u_1\right)$. Thus, the support detection requires us to inspect $2 \cdot m^{(J-1)}/2$ entries of $\mathbf{y}^{(J)}$, which has an arithmetical complexity of $\mathcal{O}\left(m^{(J-1)}\right)$.

B) Two-block support:

$\mathbf{y}^{(j)}$ has the two-block support $S^{(j)} = I^{(j)}_{\mu^{(j)},\,\nu^{(j)}} \cup I^{(j)}_{2^j-1-\nu^{(j)},\,2^j-1-\mu^{(j)}}$ with block length $n^{(j)} = m$.

We know from the proof of Theorem 5.19, case B, the proof of Theorem 5.23 and Figure 5.10 that $\mathbf{y}^{(j+1)}$ has a two-block support of block length $n^{(j+1)} \coloneqq n^{(j)} = m$. Further, the first support index $\mu^{(j+1)}$ of $\mathbf{y}^{(j+1)}$ is either $\mu^{(j)}$ or $2^j + \mu^{(j)}$ with

$$\mu^{(j+1)} \coloneqq \begin{cases} \mu^{(j)} & \text{if } \left|\widehat{u^0}_{2k_0+1} - \widehat{y^{(j+1)}}_{2k_0+1}\right| < \left|\widehat{u^0}_{2k_0+1} + \widehat{y^{(j+1)}}_{2k_0+1}\right|, \\ 2^j + \mu^{(j)} & \text{else,} \end{cases}$$

where $\mathbf{u}^0$ and $\mathbf{u}^1$ denote the two possibilities for $\mathbf{y}^{(j+1)}$. In this case the support detection has a computational effort of $\mathcal{O}(1)$.

The support detection methods described in this section cover all possible cases for the support of $\mathbf{y}^{(j+1)}$ for a given $\mathbf{y}^{(j)}$. Additionally, as we will prove in Section 5.6.3, they are stable and efficient, as they require at most $\mathcal{O}\left(m^{(j)}\right)$ operations.

5.6.2 Sparse Fast IDFT for Vectors with Reflected Block Support

Now that we know how to detect the correct support structure of any periodization $\mathbf{y}^{(j)}$ of $\mathbf{y}$ for $j \in \{0, \ldots, J\}$, we can summarize the insights gained in Section 5.5 into a sparse fast IDFT algorithm. Let us assume that $N = 2^{J-1}$ with $J \geq 2$ and that $\mathbf{y} \in \mathbb{R}^{2N}$ has a reflected block support of unknown block length $m < N$. Further, we suppose that $\mathbf{y}$ satisfies (5.4), i.e., there is no cancellation of nonzero entries in any of the periodization steps, and that we have access to all entries of $\widehat{\mathbf{y}} \in \mathbb{C}^{2N}$.

Using Lemma 5.9 (i), the algorithm starts with the initial vector

$$\mathbf{y}^{(0)} = \sum_{l=0}^{2N-1} y_l = \widehat{y}_0 \in \mathbb{R},$$

which has a one-block support by definition. For $j \in \{0, \ldots, J-1\}$ we perform the following iteration steps.

1. Recovery of $\mathbf{y}^{(j+1)}$:
   a) If $\mathbf{y}^{(j)}$ possesses a one-block support, apply the procedure from Theorem 5.20 to recover $\mathbf{y}^{(j+1)}$.
   b) If $\mathbf{y}^{(j)}$ possesses a two-block support, apply the procedure from Theorem 5.23 to recover $\mathbf{y}^{(j+1)}$.

2. Detect the support set of $\mathbf{y}^{(j+1)}$ with the appropriate method from Section 5.6.1.

Having reconstructed a vector $\mathbf{y}^{(l)}$ with two-block support of block length $n^{(l)}$, it follows from Theorem 5.19, case B that all longer periodizations $\mathbf{y}^{(j)}$, $j \in \{l+1, \ldots, J\}$, also possess a two-block support with the same block length $n^{(j)} = n^{(l)}$, so for $j > l$ we always have to apply step 1b.

If a lower bound $2^{b-1} \leq m$ on the block length of $\mathbf{y}$ is known, we can begin the algorithm with the computation of $\mathbf{y}^{(b)}$ by applying a 2^b-length IFFT algorithm to

$$\widehat{\mathbf{y}^{(b)}} = (\widehat{y}_{2^{J-b}k})_{k=0}^{2^b-1},$$

and detecting its support. Then we only have to execute the above iteration steps for $j \in \{b, \ldots, J-1\}$, thus reducing the runtime slightly. The complete procedure is summarized in Algorithm 7.

5.6.3 Runtime and Sampling Bounds

We show now that the theoretical runtime and sampling complexities of Algorithm 7 are indeed sublinear in the vector length $2N$ and subquadratic in the sparsity $2m$. In Section 6.5.1 we will illustrate the runtime and the stability of Algorithm 7 by numerical examples, also comparing it to other IDFT methods.

Theorem 5.24 (Theorem 4 in [BP18c]) *Let $N = 2^{J-1}$, $J \geq 2$, and $\mathbf{x} \in \mathbb{R}^N$ have a one-block support of length $m < N$. Set $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T$ and assume that $\mathbf{y}$ satisfies (5.4). Further, suppose that there is no a priori knowledge of the support length m of $\mathbf{x}$. Then Algorithm 7 has a runtime of $\mathcal{O}\left(m \log m \log \frac{2N}{m}\right)$ and uses $\mathcal{O}\left(m \log \frac{2N}{m}\right)$ samples of $\widehat{\mathbf{y}}$.*

Proof. (i) Note that the support $S^{(J)}$ of $\mathbf{y} = \mathbf{y}^{(J)}$ has at most cardinality $2m$. Let $2^{L-1} < 2m \leq 2^L \leq 2^J$.

For $j \in \{0, \ldots, L-1\}$ the vector $\mathbf{y}^{(j)}$ has necessarily a one-block support of length $m^{(j)}$ with $2^{j-1} \leq m^{(j)} \leq 2^j$, and we have to apply the procedure from Step 1a. Since $L^{(j)} = 2^j$ in this case, the computation of $\mathbf{y}_{(0)}^{(j+1)}$ requires an IFFT of length 2^j with $\mathcal{O}\left(2^j \log 2^j\right)$ operations and $\mathcal{O}\left(2^j\right)$ additional multiplications, according to line 5 of Algorithm 7. To determine the nonzero entries of $\mathbf{y}^{(j+1)}$ in lines 6 and 7, further $\mathcal{O}\left(2^j\right)$ operations are needed. According to cases A_1 to A_4 in Section 5.6.1, detecting its support structure in line 24 also has a runtime of $\mathcal{O}\left(m^{(j)}\right) = \mathcal{O}\left(2^j\right)$. Thus, the iteration steps for $j \in \{0, \ldots, L-1\}$ have a runtime complexity of

$$\mathcal{O}\left(\sum_{j=0}^{L-1} 2^j \log 2^j\right) = \mathcal{O}\left(2^L (L-2)\right).$$

For $j \in \{L, \ldots, J-1\}$ we have to apply either the recovery step 1a or the recovery step 1b. If $\mathbf{y}^{(j)}$ has a one-block support of length $m^{(j)}$, then $m \leq m^{(j)} \leq 2m$ and $2^{L-1} < m^{(j)} \leq 2^L$, so we have that $L^{(j)} = L$. Computing $\mathbf{z}_{(0)}^{(j+1)}$ in lines 12 and 13 requires an IFFT of length 2^L and further operations of complexity $\mathcal{O}\left(2^L\right)$. In order to detect the support structure in line 24, at most $\mathcal{O}\left(m^{(j)}\right) = \mathcal{O}\left(2^L\right)$ operations are necessary by cases A_1 to A_4 in Section 5.6.1. Altogether, we require $\mathcal{O}\left(2^L \log 2^L\right)$ operations for such an iteration step.

Algorithm 7 Sparse IDFT for Vectors with Reflected Block Support (Algorithm 1 in [BP18c])

Input: $\widehat{\mathbf{y}}$, where the sought-after $\mathbf{y} \in \mathbb{R}^{2N}$ with $N = 2^{J-1}$, $J \geq 2$, has a reflected block support of (unknown) length m and satisfies (5.4), and noise threshold $\varepsilon > 0$. If a lower bound on m is known, let $b \in \mathbb{N}_0$ s.t. $2^{b-1} \leq m$, else $b = 0$.

Output: $\mathbf{y}$.

$\mathbf{y}^{(b)} \leftarrow \mathbf{IFFT}\left[\left(\widehat{y}_{2^{J-b}k}\right)_{k=0}^{2^b-1}\right]$ and, if $b > 0$, detect its support structure.

for j from b to $J-1$ **do**

  if $\mathbf{y}^{(j)}$ has a one-block support of length $m^{(j)}$ **then**

    if $m^{(j)} > 2^{j-1}$ **then**

$$\mathbf{a} \leftarrow \operatorname{diag}\left(\left(\omega_{2^{j+1}}{}^{-l}\right)_{l=0}^{2^j-1}\right)\mathbf{IFFT}\left[\left(\widehat{y}_{2^{J-j-1}(2k+1)}\right)_{k=0}^{2^j-1}\right]$$

$$\left(\mathbf{y}_{(0)}^{(j+1)}\right)_k \leftarrow \begin{cases} \frac{1}{2}\operatorname{Re}\left(\mathbf{y}^{(j)}+\mathbf{a}\right)_k & \text{if } \left|\frac{1}{2}\operatorname{Re}\left(\mathbf{y}^{(j)}+\mathbf{a}\right)_k\right| > \varepsilon, \\ 0 & \text{else,} \end{cases} \qquad k \in I_{0,\,2^j-1}^{(j)}$$

$$\mathbf{y}_{(1)}^{(j+1)} \leftarrow \mathbf{J}_{2^j}\mathbf{y}_{(0)}^{(j+1)}$$

    else if $m^{(j)} \leq 2^{j-1}$ **then**

Set $L^{(j)} = \left\lceil \log_2 m^{(j)} \right\rceil$ and $\mathbf{v} = \left(\widehat{y}_{2^{J-j-1}\left(2^{j+1-L^{(j)}}p+1\right)}\right)_{p=0}^{2^{L^{(j)}}-1}$.

Set $\mathbf{z}^{(j)} = \left(y^{(j)}_{\left(\mu^{(j)}+r\right) \bmod 2^j}\right)_{r=0}^{2^{L^{(j)}}-1}$.

Set $\mathbf{W}_{(1)}^{(j)}{}^{-1} = \operatorname{diag}\left(\omega_{2^{j+1}}{}^{-\left(\mu^{(j)}+r\right) \bmod 2^j}\right)_{r=0}^{2^{L^{(j)}}-1}$.

$$\mathbf{a} \leftarrow \mathbf{W}_{(1)}^{(j)}{}^{-1}\mathbf{IFFT}\left[\operatorname{diag}\left(\omega_{2^{L^{(j)}}}{}^{-p\mu^{(j)}}\right)_{p=0}^{2^{L^{(j)}}-1}\mathbf{v}\right]$$

$$\left(\mathbf{z}_{(0)}^{(j+1)}\right)_k \leftarrow \begin{cases} \frac{1}{2}\operatorname{Re}\left(\mathbf{z}^{(j)}+\mathbf{a}\right)_k & \text{if } \left|\frac{1}{2}\operatorname{Re}\left(\mathbf{z}^{(j)}+\mathbf{a}\right)_k\right| > \varepsilon, \\ 0 & \text{else,} \end{cases} \qquad k \in I_{0,\,2^{L^{(j)}}-1}^{(L^{(j)})}$$

$$\left(\mathbf{y}_{(0)}^{(j+1)}\right)_{\left(\mu^{(j)}+k\right) \bmod 2^j} \leftarrow \begin{cases} \left(\mathbf{z}_{(0)}^{(j+1)}\right)_k & \text{if } k \in I_{0,\,2^{L^{(j)}}-1}^{(L^{(j)})}, \\ 0 & \text{else,} \end{cases} \qquad k \in I_{0,\,2^j-1}^{(j)}$$

$$\left(\mathbf{y}_{(1)}^{(j+1)}\right)_{2^j-1-\left(\mu^{(j)}+k\right) \bmod 2^j} \leftarrow \begin{cases} \left(\mathbf{z}_{(0)}^{(j+1)}\right)_k & \text{if } k \in I_{0,\,2^{L^{(j)}}-1}^{(L^{(j)})}, \\ 0 & \text{else,} \end{cases} \qquad k \in I_{0,\,2^j-1}^{(j)}$$

    end if

  else if $\mathbf{y}^{(j)}$ has a two-block support with block length $n^{(j)}$ **then**

$$k_0 \leftarrow \operatorname*{argmax}_{k \in \{0,\ldots,2n^{(j)}-1\}} \left\{\left|\widehat{y}_{2^{J-j-1}(2k+1)}\right|\right\}$$

$$\alpha \leftarrow \widehat{y}_{2^{J-j-1}(2k_0+1)}.$$

$$\widehat{u^0}_{2k_0+1} \leftarrow \sum_{l=\mu^{(j)}}^{\mu^{(j)}+n^{(j)}-1} \omega_{2^{j+1}}{}^{(2k_0+1)l} y_l^{(j)} + \sum_{l=2^{j+1}-n^{(j)}-\mu^{(j)}}^{2^{j+1}-1-\mu^{(j)}} \omega_{2^{j+1}}{}^{(2k_0+1)l} y_{l-2^j}^{(j)}$$

$$\lambda^{(j+1)} \leftarrow \begin{cases} \mu^{(j)} & \text{if } \left|\widehat{u^0}_{2k_0+1} - \alpha\right| < \left|\widehat{u^0}_{2k_0+1} + \alpha\right|, \\ 2^j + \mu^{(j)} & \text{else} \end{cases}$$

$$y_k^{(j+1)} \leftarrow \begin{cases} y^{(j)}_{k \bmod 2^j} & \text{if } k \in I^{(j+1)}_{\lambda^{(j+1)},\,\lambda^{(j+1)}+n^{(j)}-1}, \\ y^{(j)}_{k \bmod 2^j} & \text{if } k \in I^{(j+1)}_{2^{j+1}-n^{(j)}-\lambda^{(j+1)},\,2^{j+1}-1-\lambda^{(j+1)}}, \\ 0 & \text{else,} \end{cases} \qquad k \in I_{0,\,2^{j+1}-1}^{(j+1)}$$

    end if

    Detect the support structure of $\mathbf{y}^{(j+1)}$ (one-block or two-block) and find $\mu^{(j+1)}$ and $m^{(j+1)}$ or $n^{(j+1)}$.

end for

Output: $\mathbf{y} = \mathbf{y}^{(J)}$.

If $\mathbf{y}^{(j)}$ possesses a two-block support with block length $n^{(j)} = m$, then the execution of lines 18 to 22 requires $\mathcal{O}(m) = \mathcal{O}\left(2^L\right)$ operations, and the support structure of $\mathbf{y}^{(j+1)}$ is already completely determined.

However, in the worst case, we have to apply step 1a, the recovery step for periodizations with one-block support, for every $j \in \{L, \ldots, J-1\}$, which can also be seen in Examples 5.14.2 and 5.14.3. As we cannot tell beforehand how often each method will be used, we can only estimate a theoretical runtime of

$$\mathcal{O}\left(\sum_{j=L}^{J-1} 2^L \log 2^L\right) = \mathcal{O}\left((J-L)2^L \log 2^L\right)$$

for the last $J-L$ iteration steps, even though the algorithm is usually faster in practice. Adding the arithmetical complexities for both cases yields an overall runtime of

$$\begin{aligned} &\mathcal{O}\left(\sum_{j=0}^{L-1} 2^j \log 2^j + \sum_{j=L}^{J-1} 2^L \log 2^L\right) \\ &= \mathcal{O}\left((2^L(L-2) + (J-L)2^L L\right) \\ &= \mathcal{O}\left(m \log m \log \frac{2N}{m}\right), \end{aligned}$$

where we have used that $2m \leq 2^L < 4m$.

(ii) For $j \in \{0, \ldots, L-1\}$ it follows from $2^{L-1} < 2m \leq 2^L$ that the computation of $\mathbf{y}^{(j+1)}$ requires all 2^j samples of $\left(\widehat{y^{(j+1)}}_{2k+1}\right)_{k=0}^{2^j-1}$. With the help of Lemma 5.10, this implies that after performing the first L iteration steps, every entry of the vector $\widehat{\mathbf{y}^{(L)}}$ has been used in one of the L iteration steps necessary to recover $\mathbf{y}^{(L)}$. An iteration step with $j \in \{L, \ldots, J-1\}$ needs $2^{L^{(j)}} = \mathcal{O}\left(m^{(j)}\right)$ samples if $\mathbf{y}^{(j)}$ has a one-block support. For a two-block support it suffices to examine $2m = \mathcal{O}\left(2^L\right)$ entries of $\widehat{\mathbf{y}}$ to find a nonzero one by Lemma 5.21. Hence, Algorithm 7 has an overall sampling complexity of

$$\begin{aligned} &\mathcal{O}\left(\sum_{j=0}^{L-1} 2^j + \sum_{j=L}^{J-1} 2^L\right) \\ &= \mathcal{O}\left(2^L + (J-L)2^L\right) \\ &= \mathcal{O}\left(m \log \frac{2N}{m}\right). \end{aligned}$$

□

Remark 5.25 Note that if the unknown block length m of $\mathbf{y}$ approaches N, then Algorithm 7 has a runtime of $\mathcal{O}(N \log N)$ and a sampling complexity of $\mathcal{O}(N)$, which are the same as the runtime and sampling complexity of a full length IFFT. ◊

5.6.4 Sparse Fast IDCT-II for Vectors with One-Block Support

As already mentioned, the sparse IDFT algorithm for vectors $\mathbf{y} \in \mathbb{R}^{2N}$ with reflected block support presented in Section 5.6.2 can be applied to derive a sparse fast IDCT-II algorithm for vectors $\mathbf{x} \in \mathbb{R}^N$ with one-block support. Recall that by Lemma 5.4 (iii) the

DFT of the vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T$ is completely determined by $\mathbf{x}^{\widehat{\mathrm{II}}}$. Hence, we can compute $\mathbf{y}$ from $\widehat{\mathbf{y}}$ with the help of Algorithm 7 if $\mathbf{x}^{\widehat{\mathrm{II}}}$ is known. By construction, $\mathbf{x}$ is then given as the first half of $\mathbf{y}$. Each entry of $\widehat{\mathbf{y}}$ depends only on one entry of $\mathbf{x}^{\widehat{\mathrm{II}}}$. Thus, our IDCT-II method inherits the sublinearity in N of the runtime and sampling complexities of Algorithm 7. Since Algorithm 7 is adaptive, no a priori knowledge of the support length of $\mathbf{x}$ is required. The resulting sparse fast IDCT-II procedure is summarized in Algorithm 8.

Algorithm 8 Sparse Fast IDCT-II for Vectors with One-Block Support (Algorithm 2 in [BP18c])

Input: $\mathbf{x}^{\widehat{\mathrm{II}}}$, where the sought-after vector $\mathbf{x} \in \mathbb{R}^N$ with $N = 2^{J-1}$, $J \geq 2$, has a one-block support of (unknown) length m and $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T$ satisfies (5.4), and noise threshold $\varepsilon > 0$. If a lower bound on m is known a priori, let $b \in \mathbb{N}_0$ such that $2^{b-1} \leq m$, otherwise $b = 0$.

Output: $\mathbf{x}$.

1: Compute $\widehat{y}_k = \begin{cases} \frac{\sqrt{2N}}{\varepsilon_N(k)} {\omega_{4N}}^{-k} \cdot x_k^{\widehat{\mathrm{II}}} & \text{if } k \in \{0, \dots, N-1\}, \\ 0 & \text{if } k = N, \\ -\frac{\sqrt{2N}}{\varepsilon_N(2N-k)} {\omega_{4N}}^{-k} \cdot x_{2N-k}^{\widehat{\mathrm{II}}} & \text{if } k \in \{N+1, \dots, 2N-1\}, \end{cases}$
if the sample $\widehat{y}_k$ is needed in Algorithm 7.
2: $\mathbf{y} \leftarrow$ Algorithm 7 $[\widehat{\mathbf{y}}, b]$
3: $\mathbf{x} \leftarrow \mathbf{y}_{(0)} = (y_k)_{k=0}^{N-1}$

Output: $\mathbf{x}$.

The following theorem provides us with theoretical estimates for the runtime and sampling complexity of Algorithm 8. We will illustrate them and the numerical stability of the method by numerical examples in Section 6.5.2, also comparing Algorithm 8 to other sparse fast IDCT-II algorithms.

Theorem 5.26 (Theorem 5 in [BP18c]) *Let $N = 2^{J-1}$, $J \geq 2$, and $\mathbf{x} \in \mathbb{R}^N$ have a one-block support of length $m < N$. Set $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T$ and assume that $\mathbf{y}$ satisfies (5.4). Further, suppose that there is no a priori knowledge of the support length m of $\mathbf{x}$. Then Algorithm 8 has a runtime of $\mathcal{O}\left(m \log m \log \frac{2N}{m}\right)$ and uses $\mathcal{O}\left(m \log \frac{2N}{m}\right)$ samples of $\mathbf{x}^{\widehat{\mathrm{II}}}$.*

Proof. As shown in Theorem 5.24, Algorithm 7 requires a total of $\mathcal{O}\left(m \log \frac{2N}{m}\right)$ samples of $\widehat{\mathbf{y}}$, which are used in lines 1, 5, 9, 18 and 19. In an implementation one would compute the respective samples of $\widehat{\mathbf{y}}$ from $\mathbf{x}^{\widehat{\mathrm{II}}}$ directly in the lines of Algorithm 7 where they are needed. In our pseudocode for Algorithm 8, we summarize these calculations in line 1. Since by Lemma 5.4 (iii) each entry of $\widehat{\mathbf{y}}$ depends on only one entry of $\mathbf{x}^{\widehat{\mathrm{II}}}$, the computation of the samples of $\widehat{\mathbf{y}}$ in line 1 has a computational effort of $\mathcal{O}\left(m \log \frac{2N}{m}\right)$ and requires $\mathcal{O}\left(m \log \frac{2N}{m}\right)$ samples of $\mathbf{x}^{\widehat{\mathrm{II}}}$. Hence, Algorithm 8 has the same runtime and sampling complexity as Algorithm 7. □

Remark 5.27 Analogously to Algorithm 7, if the unknown support length m of $\mathbf{x}$ approaches N, the runtime of Algorithm 8 is $\mathcal{O}(N \log N)$ and it requires $\mathcal{O}(N)$ samples of $\mathbf{x}^{\widehat{\mathrm{II}}}$. Thus, it achieves the same runtime and the same sampling complexity as an N-length IDCT-II. ◊

6 Real Sparse Fast IDCT-II for Vectors with Short Support Based on Real Arithmetic

With Algorithm 8 we introduced a deterministic IDCT-II algorithm for reconstructing a vector $\mathbf{x} \in \mathbb{R}^N$ with one-block support from its DCT-II, $\mathbf{x}^{\widehat{\mathrm{II}}}$, in Chapter 5. However, Algorithm 8 effectively recovers the vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T \in \mathbb{R}^{2N}$ via Algorithm 7 from its DFT, $\widehat{\mathbf{y}} \in \mathbb{C}^{2N}$, which can be computed from $\mathbf{x}^{\widehat{\mathrm{II}}}$. Since Algorithm 7 exploits the short support of $\mathbf{x}$ and the resulting symmetric reflected block support of $\mathbf{y}$, it still performs better than general sparse FFT methods. This assertion will be supported by numerical experiments in Section 6.5.1. Nevertheless, despite being an adaptive algorithm which does not need any a priori knowledge of the support length, its assumptions on the sought-after vector $\mathbf{x}$ are quite strict and, without supposing extensive knowledge of $\mathbf{x}$, they can usually only be satisfied if, e.g., $\mathbf{x} \in \mathbb{R}^N_{\geq 0}$. Additionally, the vector $\mathbf{y}$ is $2m$-sparse, whereas $\mathbf{x}$ is only m-sparse, and Algorithm 8 requires complex arithmetic for recovering $\mathbf{x}$, even though the DCT-II is a real transform that can be computed in a fast way using only real arithmetic, as we have seen in Section 4.2.

Thus, investigating fully real sparse fast IDCT-II algorithms that recover $\mathbf{x}$ without using IDFTs and the auxiliary vector $\mathbf{y}$ is the natural next course of action. We will present a deterministic sparse fast algorithm for the inverse DCT-II of vectors with short support that only employs real arithmetic in this section. Unlike in Chapter 5, we now require that an upper bound $M \geq m$ on the support length of $\mathbf{x}$ is known a priori. In Section 6.5.2 we will compare the performance of our new real sparse IDCT-II with respect to runtime and noisy input data to the one of Algorithm 8 by numerical experiments.

The following chapter is based on our preprint [BP18a], and coincides in parts identically with said publication.

In order to derive the algorithm presented hereafter, we transfer some of the concepts introduced in [PW16a,PW17a,PWCW18] and the methods presented in Chapter 5 to the purely real IDCT-II setting. We sketched the main ideas of Algorithm 2 in [PW16a] and Algorithm 2.1 in [PW17a] in Section 5.3. For a summary of Algorithm 2.3 in [PWCW18] see Section 6.5.1. These four methods reconstruct a vector $\mathbf{y} \in \mathbb{R}^N$, $N = 2^J$, with either short support of length M, M-sparse support or reflected block support with block length M from its DFT. In Chapter 5 and [PW16a, PW17a, PWCW18] the sought-after vector $\mathbf{y}$ is recovered iteratively from its 2^j-length periodizations $\mathbf{y}^{(j)}$, where $\mathbf{y}^{(J)} := \mathbf{y}$ and $\mathbf{y}^{(j)}$ is obtained by adding the first and second half of $\mathbf{y}^{(j+1)}$ for all $j \in \{0, \ldots, J-1\}$, see Definition 5.7. However, for recovering a vector $\mathbf{x} \in \mathbb{R}^N$ directly from its DCT-II, the concept of periodizations has to be adapted using an iterative application of both reflections and periodizations, as we will show in Section 6.1. We still set $\mathbf{x}^{[J]} := \mathbf{x}$, but $\mathbf{x}^{[j]} \in \mathbb{R}^{2^j}$ is now defined by adding the first half of $\mathbf{x}^{[j+1]}$ and the reflection of the second

half of $\mathbf{x}^{[j+1]}$, i.e.,

$$\mathbf{x}^{[j]} \coloneqq \left(x_0^{[j+1]} + x_{2^{j+1}-1}^{[j+1]}, x_1^{[j+1]} + x_{2^{j+1}-2}^{[j+1]}, \ldots, x_{2^j-1}^{[j+1]} + x_{2^j}^{[j+1]}\right)^T.$$

Employing this concept for $j \in \{L, \ldots, J-1\}$, where $2^{L-1} \geq M$, our new real IDCT-II algorithm is based on iteratively recovering $\mathbf{x}^{[j+1]}$ from $\mathbf{x}^{\widehat{\mathrm{II}}}$ using that $\mathbf{x}^{[j]}$ is known from the previous step. To the best of our knowledge the algorithm we will present hereafter is the first deterministic sparse IDCT-II algorithm that only uses real arithmetic.

6.1 Short Support and Reflected Periodizations

Throughout this chapter we will always consider a real vector $\mathbf{x} = (x_k)_{k=0}^{N-1} \in \mathbb{R}^N$, with $N = 2^J$, that has a short support of length $m \leq M$, where the upper bound M is known a priori. Unlike in Chapter 5, we do not allow the support of $\mathbf{x}$ to be wrapped periodically around the boundary of the vector anymore. Again, we index both $\mathbf{x}$ and $\mathbf{x}^{\widehat{\mathrm{II}}}$ from 0 to $N-1$, since we do not consider frequencies in this vector setting. In order to formally define the notion of a short support, we introduce notation for non-periodized intervals.

Definition 6.1 Let $a, b \in \mathbb{N}_0$ with $a \leq b$. Then we denote by $I_{a,\,b}$ the interval

$$I_{a,\,b} \coloneqq \{a, a+1, \ldots, b\} \subsetneq \mathbb{N}_0.$$

The above notation for intervals facilitates giving the definition of a vector with short support, which is analogous to the definition of a vector with one-block support from Definition 5.2, where the support could be wrapped periodically around the boundary of the vector.

Definition 6.2 (Short Support) Let $\mathbf{x} = (x_k)_{k=0}^{N-1} \in \mathbb{R}^N$. Then $\mathbf{x}$ has a *short support of length* m if m is the minimal integer such that

$$x_k = 0 \qquad \forall\, k \notin I_{\mu,\,\nu} = \{\mu, \mu+1, \ldots, \nu\}$$

for some $\mu \in \{0, \ldots, N-m\}$ and $\nu \coloneqq \mu + m - 1$ with $x_\mu \neq 0$ and $x_\nu \neq 0$.

The interval $S \coloneqq I_{\mu,\,\nu}$ is called the *support interval*, μ the *first support index* and ν the *last support index* of $\mathbf{x}$.

Remark 6.3 Note that if $\mathbf{x} \in \mathbb{R}^N$ has a short support, then its support is not considered periodically, unlike for vectors with one-block support according to Definition 5.2. Thus, for vectors with short support, the first support index μ is always uniquely determined, even if the support length m is greater than $N/2$, which is not the case for vectors with one-block support. The last support index ν and the support length m are also uniquely determined.

As in Chapter 5, the interval S contains all indices at which the vector $\mathbf{x} \in \mathbb{R}^N$ has nonzero entries, while $\mathbf{x}$ may also be zero at indices in S, since for some of the theoretical concepts employed hereafter we need the support of $\mathbf{x}$ to be an interval in $\mathbb{N}_0$. ◊

Our aim in this chapter is to find a deterministic algorithm for reconstructing a sparse vector $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, from its discrete cosine transform of type II, $\mathbf{x}^{\widehat{\mathrm{II}}}$, using only real arithmetic. More precisely, we assume that $\mathbf{x}$ has a short support of length m and that only an upper bound M on m is known. As in Chapter 5, which is based on [BP18c],

and [PW17a,PW16a,PWCW18], we will use an iterative approach. Thus, we first require an analog to the periodizations $\mathbf{y}^{(j)}$ of $\mathbf{y} \in \mathbb{R}^{2N}$ from Definition 5.7, which were used in the publications cited above. In order to do this in a meaningful way, we inspect one of the possible factorizations of the orthogonal cosine matrix of type II, see Definition 4.1, namely the one given by Lemma 4.5.

Recall that for $N \in \mathbb{N}$ even and $\mathbf{x} \in \mathbb{R}^N$ we have that

$$\mathbf{x}^{\widehat{\mathrm{II}}} = \mathbf{C}_N^{\mathrm{II}}\mathbf{x} = \mathbf{P}_N{}^T \left(\begin{array}{c|c} \mathbf{C}_{\frac{N}{2}}^{\mathrm{II}} & \mathbf{0}_{\frac{N}{2}} \\ \hline \mathbf{0}_{\frac{N}{2}} & \mathbf{C}_{\frac{N}{2}}^{\mathrm{IV}} \end{array}\right) \mathbf{T}_N\mathbf{x},$$

where

$$\mathbf{T}_N\mathbf{x} \coloneqq \frac{1}{\sqrt{2}} \left(\begin{array}{c|c} \mathbf{I}_{\frac{N}{2}} & \mathbf{J}_{\frac{N}{2}} \\ \hline \mathbf{I}_{\frac{N}{2}} & -\mathbf{J}_{\frac{N}{2}} \end{array}\right) \mathbf{x} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{x}_{(0)} + \mathbf{J}_{\frac{N}{2}}\mathbf{x}_{(1)} \\ \mathbf{x}_{(0)} - \mathbf{J}_{\frac{N}{2}}\mathbf{x}_{(1)} \end{pmatrix}, \tag{6.1}$$

$\mathbf{P}_N$ is the even-odd permutation matrix from Lemma 4.5 and $\mathbf{J}_N$ is the counter identity from Theorem 4.4. Recall that according to Definition 5.7 we denote by $\mathbf{x}_{(0)}$ the first half of $\mathbf{x}$ and by $\mathbf{x}_{(1)}$ the second half, respectively. Inspired by (6.1), we define the DCT-II-specific analog to the periodization from Definition 5.7.

Definition 6.4 (Reflected Periodization ((5) in [BP18a])) Let $N = 2^J$ with $J \in \mathbb{N}$ and $\mathbf{x} \in \mathbb{R}^N$. We set $\mathbf{x}^{[J]} \coloneqq \mathbf{x}$. For $j \in \{0, \dots, J-1\}$ we define the *reflected periodization* $\mathbf{x}^{[j]} \in \mathbb{R}^{2^j}$ of $\mathbf{x}$ as

$$\mathbf{x}^{[j]} \coloneqq \mathbf{x}_{(0)}^{[j+1]} + \mathbf{J}_{2^j}\mathbf{x}_{(1)}^{[j+1]} = \left(x_k^{[j+1]} + x_{2^{j+1}-1-k}^{[j+1]}\right)_{k=0}^{2^j-1}.$$

By definition, the reflected periodization $\mathbf{x}^{[j]} \in \mathbb{R}^{2^j}$ is given by adding the first half and the reflection of the second half of the reflected periodization $\mathbf{x}^{[j+1]} \in \mathbb{R}^{2^{j+1}}$ for any $j \in \{0, \dots, J-1\}$.

Example 6.5 Let $\mathbf{x} \in \mathbb{R}^{16}$ with nonzero entries x_{13}, x_{14}. Then $\mathbf{x}$ and its reflected periodizations are

$$\begin{aligned} \mathbf{x} = \mathbf{x}^{[4]} &= (0,0,0,0,0,0,0,0,0,0,0,0,0,0,x_{13},x_{14},0)^T, \\ \mathbf{x}^{[3]} &= (0,x_{14},x_{13},0,0,0,0,0)^T, \\ \mathbf{x}^{[2]} &= (0,x_{14},x_{13},0)^T, \\ \mathbf{x}^{[1]} &= (0,x_{13}+x_{14})^T, \\ \mathbf{x}^{[0]} &= (x_{13}+x_{14})^T. \end{aligned}$$

◊

As the notion of reflected periodizations arises naturally from the factorization of the matrix $\mathbf{C}_N^{\mathrm{II}}$ given in Lemma 4.5, it has many useful properties. Most importantly, similarly as for the periodizations considered in Chapter 5, for $j \in \{0, \dots, J\}$ the DCT-II of the reflected periodization $\mathbf{x}^{[j]}$ of any $\mathbf{x} \in \mathbb{R}^N$ is already completely determined by the DCT-II of $\mathbf{x}$ itself, as we will show in the following lemma.

Lemma 6.6 (Lemma 2.3 in [BP18a]) Let $N = 2^J$ with $J \in \mathbb{N}$ and $j \in \{0, \dots, J\}$. Let $\mathbf{x} \in \mathbb{R}^N$. Then $\left(\mathbf{x}^{[j]}\right)^{\widehat{\mathrm{II}}}$ satisfies

$$\left(\mathbf{x}^{[j]}\right)^{\widehat{\mathrm{II}}} = \sqrt{2}^{J-j} \left(x^{\widehat{\mathrm{II}}}_{2^{J-j}k}\right)_{k=0}^{2^j-1}.$$

Proof. We prove the lemma by induction. For $j = J$ the claim holds, since $\mathbf{x}^{[J]} = \mathbf{x}$ by definition. Now we assume the induction hypothesis for some $j \in \{1, \dots, J\}$ and show that the claim also holds for $j-1$. Lemma 4.5, (6.1) and Definition 6.4 yield

$$\begin{aligned}\mathbf{P}_{2^j}\mathbf{C}^{\mathrm{II}}_{2^j}\mathbf{x}^{[j]} &= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{2^{j-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{j-1}} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{2^{j-1}} & \mathbf{J}_{2^{j-1}} \\ \mathbf{I}_{2^{j-1}} & -\mathbf{J}_{2^{j-1}} \end{pmatrix} \begin{pmatrix} \mathbf{x}^{[j]}_{(0)} \\ \mathbf{x}^{[j]}_{(1)} \end{pmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{2^{j-1}}\mathbf{x}^{[j-1]} \\ \mathbf{C}^{\mathrm{IV}}_{2^{j-1}} \left(\mathbf{x}^{[j]}_{(0)} - \mathbf{J}_{2^{j-1}}\mathbf{x}^{[j]}_{(1)}\right) \end{pmatrix}. \end{aligned} \tag{6.2}$$

It follows from Remark 4.6 and the first 2^{j-1} rows of (6.2) that

$$\begin{aligned}\left(\mathbf{x}^{[j-1]}\right)^{\widehat{\mathrm{II}}} &= \mathbf{C}^{\mathrm{II}}_{2^{j-1}}\mathbf{x}^{[j-1]} \\ &= \sqrt{2}\left(\mathbf{P}_{2^j}\mathbf{C}^{\mathrm{II}}_{2^j}\mathbf{x}^{[j]}\right)_{(0)} \\ &= \sqrt{2}\left(\left(x^{[j]}\right)^{\widehat{\mathrm{II}}}_{2k}\right)_{k=0}^{2^{j-1}-1} \\ &= \sqrt{2}\left(\sqrt{2}^{J-j} x^{\widehat{\mathrm{II}}}_{2^{J-j}2k}\right)_{k=0}^{2^{j-1}-1} \\ &= \sqrt{2}^{J-(j-1)} \left(x^{\widehat{\mathrm{II}}}_{2^{J-(j-1)}k}\right)_{k=0}^{2^{j-1}-1}, \end{aligned}$$

where we used the induction hypothesis in the second to last step. □

6.2 Support Structures of Reflected Periodizations

Our goal is to reconstruct a vector $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, with short support of length $m \le M$ from $\mathbf{x}^{\widehat{\mathrm{II}}}$ by successively computing its reflected periodizations $\mathbf{x}^{[L]}, \mathbf{x}^{[L+1]}, \dots, \mathbf{x}^{[J]} = \mathbf{x}$. Unlike for the DFT case, due to different support constraints, we cannot begin the reconstruction with $\mathbf{x}^{[0]} \in \mathbb{R}$. Instead, we have to start at a level L with $2^{L-1} \ge M$, thus being forced to have a priori knowledge of an upper bound M on the support length m. As in Chapter 5 we have to determine $\mathbf{x}^{[j+1]}$ efficiently from $\mathbf{x}^{\widehat{\mathrm{II}}}$ in the $(j-L)$th iteration step using that $\mathbf{x}^{[j]}$ is already known. Hence, we have to investigate how the support of $\mathbf{x}^{[j+1]}$ can look like if the support of $\mathbf{x}^{[j]}$ is given.

There are three main differences between the method from Chapter 5 and the one we will present hereafter. Firstly, our new real IDCT-II algorithm will only use real arithmetic, whereas Algorithm 8 requires the computation of IFFTs. Secondly, by reconstructing $\mathbf{x}$ directly from its reflected periodizations instead of the periodizations of the auxiliary vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N\mathbf{x})^T)^T$, we have to recover a vector with short support of length at most m in each iteration step instead of a vector with reflected block support, as we will show in the following section. Thus, we do not have to detect whether

the reflected periodization has a one-block or a two-block support, which simplifies the reconstruction. Thirdly, while Algorithm 8 is completely adaptive, our new real IDCT-II method requires a priori knowledge of an upper bound M on the support length m of $\mathbf{x}$.

For our real IDCT-II method recovering $\mathbf{x}$ iteratively from its reflected periodizations can only be possible if we do not lose any relevant information about the support structure in the periodization process. More precisely, we require that, in addition to $x_{\mu^{[J]}} \neq 0$ and $x_{\nu^{[J]}} \neq 0$, $\mathbf{x}$ satisfies the non-cancellation condition

$$x_{\mu^{[J]}} + x_{\nu^{[J]}} \neq 0 \qquad \text{if } m \text{ is even.} \tag{6.3}$$

This condition suffices to guarantee the correct identification of the support of $\mathbf{x}^{[j]}$ for all $j \in \{L, \dots, J\}$, since the real IDCT-II method will not have to detect how many support blocks a given reflected periodization $\mathbf{x}^{[j]}$ has. It is less restrictive than the non-cancellation condition (5.4) for Algorithm 8. Condition (6.3) holds for example if all nonzero entries of $\mathbf{x}$ are positive or if all nonzero entries of $\mathbf{x}$ are negative, i.e., if $\mathbf{x} \in \mathbb{R}^N_{\geq 0}$ or $\mathbf{x} \in \mathbb{R}^N_{\leq 0}$.

Applying our method in practice means that the given data will be noisy. Then we have to guarantee that, for some threshold $\varepsilon > 0$ depending on the noise level, the vector $\mathbf{x}$ satisfies

$$\left|x_{\mu^{[J]}}\right| > \varepsilon, \quad |x_{\nu^{[J]}}| > \varepsilon \qquad \text{and} \qquad \left|x_{\mu^{[J]}} + x_{\nu^{[J]}}\right| > \varepsilon.$$

6.2.1 Support Structure of $\mathbf{x}^{[j]}$ for Given $\mathbf{x}$

We want to iteratively recover the vector $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, from its reflected periodizations $\mathbf{x}^{[j]} \in \mathbb{R}^{2^j}$ for $j \in \{L, \dots, J-1\}$ for a suitable starting index L. Since we learned in Section 5.4 that the DFT-specific periodizations of a vector with reflected block support have a very special structure that can be employed for the iteration procedure, we have reason to expect that similar results also hold for the DCT-II-specific reflected periodizations of a vector with short support. Thus, we will first investigate the support structure of $\mathbf{x}^{[j]}$ for a vector $\mathbf{x} \in \mathbb{R}^N$ with short support.

Remark 6.7 If $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, has a short support of length m, the possibly nonzero entries of $\mathbf{x}$ will definitely collide in the reflected periodizations of length 2^j for $j \in \{0, \dots, 2^{K-2}\}$, where $K \coloneqq \lceil \log_2 m \rceil + 1$, since $2^{K-2} < m \leq 2^{K-1}$, see Example 6.5. Such collisions did not pose much of a problem in the DFT case that was discussed in Chapter 5. However, for the slightly differently structured reflected periodizations, an iterative approach beginning at $\mathbf{x}^{[0]}$ would not be feasible, as we can only undo collisions if the support of $\mathbf{x}^{[j]}$ is contained in the last 2^{K-1} entries. We actually want to reduce collisions of nonzero entries of $\mathbf{x}$ in the reflected periodizations $\mathbf{x}^{[j]}$ as much as possible, which is why we will only consider the reflected periodizations to the level K, similarly to Algorithm 2 in [PW16a]. Thus, in all reflected periodizations $\mathbf{x}^{[j]}$ with $j \in \{K, \dots, J\}$, the support length m of $\mathbf{x}$ is at most half the vector length. Due to this restriction, our method will not be able to detect the support on the fly and require a priori knowledge of an upper bound M on the support length m of $\mathbf{x}$. ◊

Before formulating a lemma about the support structure of the reflected periodization $\mathbf{x}^{[j]}$, we will motivate the claims therein by looking at some exemplary vectors, illustrating the main possible support structures.

Example 6.8

1. Let $\mathbf{x} \in \mathbb{R}^{16}$ satisfy (6.3) with nonzero entries x_{13}, x_{14}, i.e., with the short support $S^{[4]} = I_{13,14}$ of length $m = 2$. Assume that m is known exactly, i.e., that $M = m = 2$. Then $K = 2$, and $\mathbf{x}$ and its reflected periodizations $\mathbf{x}^{[j]}$ for $j \in \{K, \dots, J\}$ are

$$\begin{aligned}\mathbf{x} = \mathbf{x}^{[4]} &= (0,0,0,0,0,0,0,0,0,0,0,0,0,0,x_{13},x_{14},0)^T,\\ \mathbf{x}^{[3]} &= (0,x_{14},x_{13},0,0,0,0,0,0)^T,\\ \mathbf{x}^{[2]} &= (0,x_{14},x_{13},0)^T.\end{aligned}$$

Here, $\mathbf{x}^{[3]}$ and $\mathbf{x}^{[2]}$ have the short support $S^{[3]} = S^{[2]} = I_{1,2}$ of length $m^{[3]} = m^{[2]} = 2 = m$.

2. Let $\mathbf{x} \in \mathbb{R}^{16}$ satisfy (6.3) with nonzero entries x_7, x_8, i.e., with short support $S^{[4]} = I_{7,8}$ of length $m = 2$. Again, we assume that $M = m$, so $K = 2$. Then, for $j \in \{2,3,4\}$, the reflected periodizations of $\mathbf{x}$ are

$$\begin{aligned}\mathbf{x} = \mathbf{x}^{[4]} &= (0,0,0,0,0,0,0,x_7,x_8,0,0,0,0,0,0,0)^T,\\ \mathbf{x}^{[3]} &= (0,0,0,0,0,0,0,x_7+x_8)^T,\\ \mathbf{x}^{[2]} &= (x_7+x_8,0,0,0)^T.\end{aligned}$$

Here, $\mathbf{x}^{[3]}$ has the short support $S^{[3]} = I_{7,7}$ of length $m^{[3]} = 1 < m = 2$ and $\mathbf{x}^{[2]}$ has the short support $S^{[2]} = I_{0,0}$ of length $m^{[2]} = m^{[3]} = 1$. ◊

Note that the second example shows that even though we only computed the reflected periodizations of length 2^j for $j \in \{K, \dots, J\}$, we could not completely avoid the collision of nonzero entries of $\mathbf{x}$, unlike in [PW16a]. For both vectors considered in Example 6.8 all reflected periodizations $\mathbf{x}^{[j]}$ for $j \in \{K, \dots, J\}$ with $K = \lceil \log_2 m \rceil + 1$ have a short support of length at most m. This observation is generalized in the following lemma.

Lemma 6.9 (Lemma 2.4 in [BP18a]) Let $N = 2^J$ with $J \in \mathbb{N}$. Let $\mathbf{x} \in \mathbb{R}^N$ have a short support of length m and assume that $\mathbf{x}$ satisfies (6.3). Set $K := \lceil \log_2 m \rceil + 1$ and let $j \in \{K, \dots, J\}$. Then $\mathbf{x}^{[j]}$ has a short support of length $m^{[j]} \le m$.

Proof. We employ an induction argument. By assumption $\mathbf{x}^{[J]} = \mathbf{x}$ has a short support of length m. Now suppose that for $j \in \{K, \dots, J-1\}$ $\mathbf{x}^{[j+1]}$ has a short support of length $m^{[j+1]} \le m$ with support interval $S^{[j+1]} = I_{\mu^{[j+1]}, \nu^{[j+1]}}$, and first and last support indices $\mu^{[j+1]} \in \{0, \dots, 2^{j+1} - m^{[j+1]}\}$ and $\nu^{[j+1]} := \mu^{[j+1]} + m^{[j+1]} - 1$. We have to distinguish three cases.

(i) $S^{[j+1]} \subseteq I_{0,2^j-1}$, i.e., the nonzero entries are contained in the first half of $\mathbf{x}^{[j+1]}$.

This implies that $\mathbf{x}^{[j+1]}_{(1)} = \mathbf{0}_{2^j}$. Since $\mathbf{x}^{[j]} = \mathbf{x}^{[j+1]}_{(0)} + \mathbf{J}_{2^j}\mathbf{x}^{[j+1]}_{(1)}$ by Definition 6.4, we obtain that $\mathbf{x}^{[j]}$ also has a short support of length $m^{[j]} = m^{[j+1]}$ with

$$\mathbf{x}^{[j]} = \mathbf{x}^{[j+1]}_{(0)} \quad \text{and} \quad S^{[j]} = S^{[j+1]},$$

see Figure 6.1.

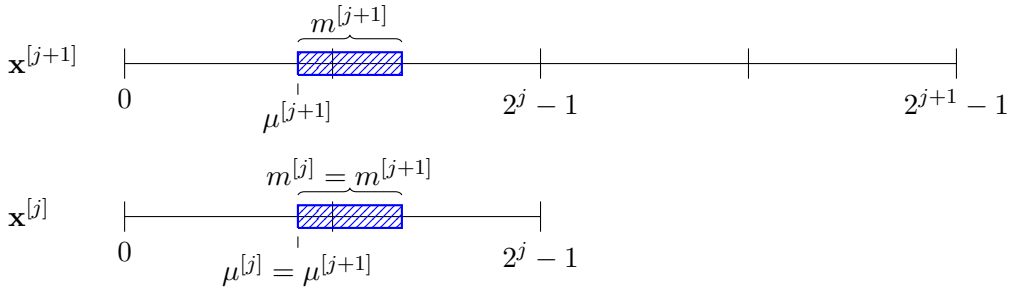

Figure 6.1: Illustration of the support of $\mathbf{x}^{[j]}$ for given $\mathbf{x}^{[j+1]}$ according to Lemma 6.9 if $S^{[j+1]} \subseteq I_{0,\,2^j-1}$

(ii) $S^{[j+1]} \subseteq I_{2^j,\,2^{j+1}-1}$, i.e., the nonzero entries are contained in the second half of $\mathbf{x}^{[j+1]}$.

As $\mathbf{x}^{[j+1]}_{(0)} = \mathbf{0}_{2^j}$, it follows from the definition of the reflected periodization that $\mathbf{x}^{[j]}$ also has a short support of length $m^{[j]} = m^{[j+1]}$, with

$$\mathbf{x}^{[j]} = \mathbf{J}_{2^j}\mathbf{x}^{[j+1]}_{(1)} \quad \text{and} \quad S^{[j]} = I_{2^{j+1}-1-\nu^{[j+1]},\,2^{j+1}-1-\mu^{[j+1]}},$$

see Figure 6.2.

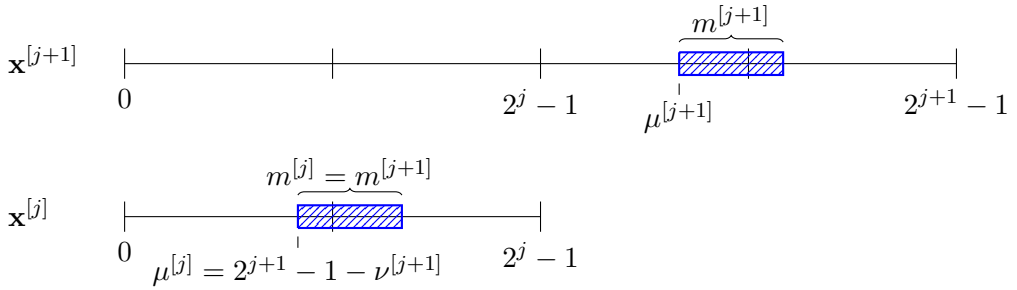

Figure 6.2: Illustration of the support of $\mathbf{x}^{[j]}$ for given $\mathbf{x}^{[j+1]}$ according to Lemma 6.9 if $S^{[j+1]} \subseteq I_{2^j,\,2^{j+1}-1}$

(iii) $I_{2^j-1,\,2^j} \subseteq S^{[j+1]}$

Then at least one possibly nonzero entry from the second half of $\mathbf{x}^{[j+1]}$, $x^{[j+1]}_l$ with $l \in I_{2^j,\,2^{j+1}-1}$, is added to a possibly nonzero entry $x^{[j+1]}_k$ with $k = 2^{j+1} - 1 - l \in I_{0,\,2^j-1}$ from the first half at the reflected index in the computation of $\mathbf{x}^{[j]}$. Thus, it follows from Definition 6.4 that $\mathbf{x}^{[j]}$ has indeed a short support of length $m^{[j]} < m^{[j+1]}$ with support interval

$$\begin{aligned} S^{[j]} &= \left(I_{\mu^{[j+1]},\,\nu^{[j+1]}} \cup I_{2^{j+1}-1-\nu^{[j+1]},\,2^{j+1}-1-\mu^{[j+1]}}\right) \cap I_{0,\,2^j-1} \\ &=: I_{2^j-m^{[j]},\,2^j-1} \subsetneq I_{2^j-m^{[j+1]},\,2^j-1}, \end{aligned}$$

and either $\mu^{[j]} = \mu^{[j+1]}$ or $\mu^{[j]} = 2^{j+1} - 1 - \nu^{[j+1]}$, see Figure 6.3. □

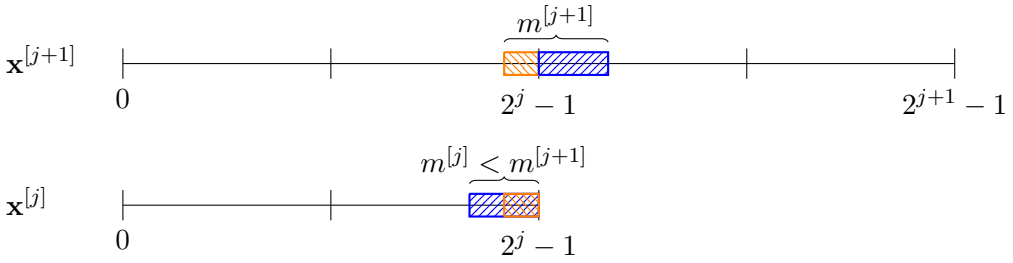

Figure 6.3: Illustration of the support of $\mathbf{x}^{[j]}$ for given $\mathbf{x}^{[j+1]}$ according to Lemma 6.9 if $I_{2^j-1,\,2^j} \subseteq S^{[j+1]}$

Remark 6.10

(i) Note that in cases (i) and (ii) of the proof of Lemma 6.9 the support length does not change, i.e., $m^{[j]} = m^{[j+1]}$, and that the support length $m^{[j]} < m^{[j+1]}$ always decreases in case (iii).

(ii) For any $j \in \{K, \dots, J\}$ we will always denote the *support length*, the *first and last support index* and the *support interval* of the reflected periodization $\mathbf{x}^{[j]} \in \mathbb{R}^{2^j}$ of $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, by $m^{[j]}, \mu^{[j]}, \nu^{[j]}$ and $S^{[j]}$, respectively. Since $\mathbf{x}^{[J]} = \mathbf{x}$, we will usually write m instead of $m^{[J]}$ etc. ◊

6.2.2 Support Structure of $\mathbf{x}^{[j+1]}$ for Given $\mathbf{x}^{[j]}$

The aim of our algorithm is to reconstruct $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, from $\mathbf{x}^{\widehat{\Pi}}$ by successively computing its reflected periodizations $\mathbf{x}^{[L]}, \mathbf{x}^{[L+1]}, \dots, \mathbf{x}^{[J]} = \mathbf{x}$ for a suitable starting index L if only an upper bound M on the support length m of $\mathbf{x}$ is known. Choosing $L := \lceil \log_2 M \rceil + 1$ will guarantee correct recovery of the supports of all these reflected periodizations. Similarly to Chapter 5.4.3 we now investigate how the support of $\mathbf{x}^{[j+1]}$ can look like if the support of $\mathbf{x}^{[j]}$ is already known from the previous iteration step, by which we aim to reduce the runtime of our method. Prior to proving a general theorem about the support of $\mathbf{x}^{[j+1]}$, we will illustrate the main possible cases for the support of $\mathbf{x}^{[j+1]}$ for given $\mathbf{x}^{[j]}$ by the vectors considered in Example 6.8.

Example 6.11 (Example 6.8 continued)
Let us assume again that the support length m of $\mathbf{x}$ is known exactly, i.e., that we have $M = m = 2$. Then it follows that $L = 2$. Further, we assume that $N = 16$ is known.

1. Note that for the vector $\mathbf{x} \in \mathbb{R}^{16}$ with short support $S^{[4]} = I_{13,\,14}$ from Example 6.8.1 we have that $S^{[j]} \not\subseteq I_{2^j-M,\,2^j-1}$ for all $j \in \{2, 3, 4\}$. Consider $\mathbf{x}^{[2]} = \left(0, x_1^{[2]}, x_2^{[2]}, 0\right)^T$ with short support $S^{[2]} = I_{1,\,2}$ of length $m^{[2]} = 2$. It follows from Definition 6.4 that $\mathbf{x}^{[3]}$ has to be either

$$\mathbf{x}^{[3]} = \left(0, x_1^{[2]}, x_2^{[2]}, 0, 0, 0, 0, 0\right)^T \quad \text{or} \quad \mathbf{x}^{[3]} = \left(0, 0, 0, 0, 0, x_2^{[2]}, x_1^{[2]}, 0\right)^T,$$

since no other vector in $\mathbb{R}^8$ with reflected periodization $\mathbf{x}^{[2]}$ can arise from reflectedly periodizing a vector $\mathbf{x} \in \mathbb{R}^{16}$ that has a short support of length at most $M = 2$. Thus,

$\mathbf{x}^{[3]}$ has a short support of length $m^{[3]} = m^{[2]}$ as well, with

$$S^{[3]} = I_{1,2} \qquad \text{or} \qquad S^{[3]} = I_{5,6},$$

where the values of the nonzero entries of $\mathbf{x}^{[2]}$ and $\mathbf{x}^{[3]}$ are the same. We will show in Section 6.3 how we can determine which of the two possibilities is the correct one by using additional entries of $\mathbf{x}^{\widehat{\Pi}}$. Analogously, $\mathbf{x} = \mathbf{x}^{[4]}$ can be recovered from $\mathbf{x}^{[3]}$.

2. Note that for the vector $\mathbf{x} \in \mathbb{R}^{16}$ with short support $S^{[4]} = I_{7,8}$ from Example 6.8.2 we have that $j' = 3$ is the only index contained in $\{2,3,4\}$ such that $S^{[j]} \subseteq I_{2^j-M,\,2^j-1}$. Let us consider now the reflected periodization $\mathbf{x}^{[3]}$, which can be reconstructed from $\mathbf{x}^{[2]}$ as in Example 6.11.1. Then, with the help of the methods from Section 6.3, it follows that $\mathbf{x}^{[3]}$ has the short support $S^{[3]} = I_{7,7}$ of length $m^{[3]}$, i.e.,

$$\mathbf{x}^{[3]} = \left(0,0,0,0,0,0,0,x_7^{[3]}\right)^T.$$

By definition of the reflected periodization and since $M = m = 2$ is known exactly, $\mathbf{x} = \mathbf{x}^{[4]}$ has to be of the form

$$\mathbf{x} = (0,0,0,0,0,0,0,x_7,x_8,0,0,0,0,0,0,0)^T$$

with $x_7 + x_8 = x_7^{[3]}$. The nonzero entries of $\mathbf{x}$, which do not have to be the same as the nonzero entry $x_7^{[3]}$ of $\mathbf{x}^{[3]}$, can be determined from $\mathbf{x}^{\widehat{\Pi}}$ using the methods we will present in Section 6.3. If x_7 and x_8 are both not zero, then $\mathbf{x}$ has the short support $S^{[4]} = I_{7,8}$ of length $m = 2$. Otherwise, it has the short support $S^{[4]} = I_{7,7}$ or $S^{[4]} = I_{8,8}$ of length $m = 1$. ◊

Example 6.11 shows that, at least for the two vectors from Example 6.8, there is at most one index $j' \in \{L,\dots,J\}$ such that the support of $\mathbf{x}^{[j']}$ is contained in the last M entries. Further, if $j \neq j'$, there are precisely two possibilities for the reflected periodization $\mathbf{x}^{[j+1]}$ of double length. These observations are generalized in the following theorem.

Theorem 6.12 (Lemma 2.5 in [BP18a]) *Let $N = 2^J$ with $J \in \mathbb{N}$. Let $\mathbf{x} \in \mathbb{R}^N$ have a short support of length $m \leq M$ and assume that $\mathbf{x}$ satisfies (6.3). Set $L \coloneqq \lceil \log_2 M \rceil + 1$.*

A) *Possible collision:*

There is at most one index $j' \in \{L,\dots,J\}$ such that $S^{[j']} \subseteq I_{2^{j'}-M,\,2^{j'}-1}$, and we have that $S^{[j'+1]} \subsetneq I_{2^{j'}-M,\,2^{j'}+M-1}$ if $j' \leq J-1$.

B) *No collision:*

Let $j \in \{L,\dots,J-1\} \setminus \{j'\}$ with j' as in case A. Let $\mathbf{x}^{[j]}$ have the short support $S^{[j]} = I_{\mu^{[j]},\,\nu^{[j]}}$ of length $m^{[j]}$. Then

(i) $m^{[j]} = m^{[j+1]}$ and

(ii) $\mathbf{x}^{[j+1]}$ is either

$$\mathbf{x}^{[j+1]} = \begin{pmatrix} \mathbf{x}^{[j]} \\ \mathbf{0}_{2^j} \end{pmatrix} \quad \text{or} \quad \mathbf{x}^{[j+1]} = \begin{pmatrix} \mathbf{0}_{2^j} \\ \mathbf{J}_{2^j}\mathbf{x}^{[j]} \end{pmatrix},$$

with $S^{[j+1]} = I_{\mu^{[j]},\,\nu^{[j]}}$ or $S^{[j+1]} = I_{2^{j+1}-1-\nu^{[j]},\,2^{j+1}-1-\mu^{[j]}}$.

Proof. A) Recall that $K = \lceil \log_2 m \rceil + 1 \leq L$, so, by Lemma 6.9, $\mathbf{x}^{[j]}$ has a short support $S^{[j]}$ of length $m^{[j]} \leq m$ for all $j \in \{L, \ldots, J\}$. Set

$$j' := \max \left\{ j \in \{L, \ldots, J\} : S^{[j]} \subseteq I_{2^j - M,\, 2^j - 1} \right\} \tag{6.4}$$

if such an index exists. If there is no such j', claim A is already proven, so let us assume that there exists a $j' \in \{L, \ldots, J\}$ satisfying (6.4). By definition of L we have that $2^{j'} - M \geq 2^{j'-1}$. Thus, if $j' > L$, we obtain

$$\mathbf{x}^{[j'-1]} = \underbrace{\mathbf{x}^{[j']}_{(0)}}_{=\mathbf{0}_{2^{j'-1}}} + \mathbf{J}_{2^{j'-1}} \mathbf{x}^{[j']}_{(1)} = \mathbf{J}_{2^{j'-1}} \mathbf{x}^{[j']}_{(1)},$$

and, consequently,

$$S^{[j'-1]} \subseteq I_{0,\, M-1}.$$

Inductively, for all $j \in \{L, \ldots, j' - 2\}$, we also find that

$$\mathbf{x}^{[j]} = \mathbf{x}^{[j+1]}_{(0)} + \mathbf{J}_{2^j} \underbrace{\mathbf{x}^{[j+1]}_{(1)}}_{=\mathbf{0}_{2^j}} \quad \text{and} \quad S^{[j]} = S^{[j+1]} \subseteq I_{0,\, M-1},$$

since $2^j - M \geq 2^{j-1}$ if $j \geq L$. This implies that j' is the unique index in $\{L, \ldots, J\}$ for which (6.4) holds. Furthermore, we even showed that for $j \in \{L, \ldots, j' - 1\}$ the support of $\mathbf{x}^{[j]}$ is contained in the first $M \leq 2^{j-1}$ entries of the vector. By definition of the reflected periodization, we immediately obtain for the support $S^{[j'+1]}$ of the reflected periodization of length $2^{j'+1}$ that

$$S^{[j'+1]} \subsetneq I_{2^{j'} - M,\, 2^{j'} + M - 1}$$

if $j' \leq J - 1$, as $S^{[j']} \subsetneq I_{2^{j'} - M,\, 2^{j'} - 1}$. Hence, it is possible that $\mathbf{x}^{[j'+1]}$ has a longer support than $\mathbf{x}^{[j']}$. For the special case that $m^{[j']} < m^{[j'+1]}$ the supports of the reflected periodizations are depicted in Figure 6.4.

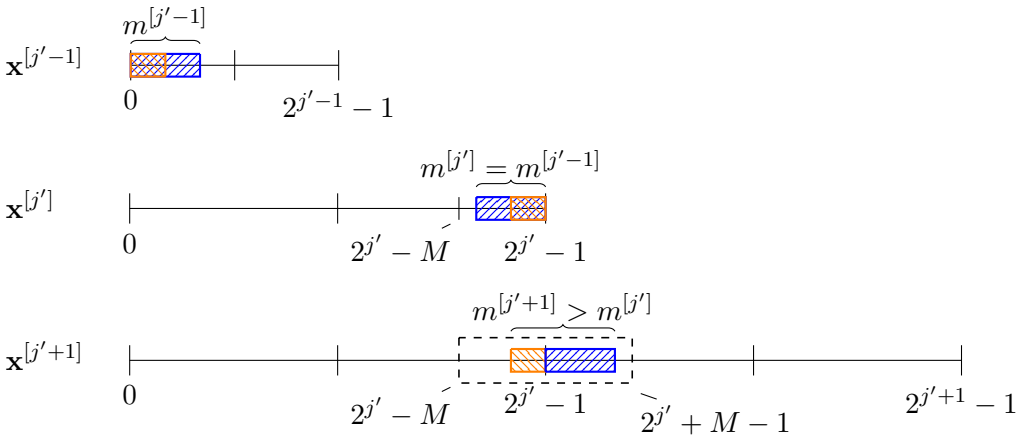

Figure 6.4: Illustration of the support of $\mathbf{x}^{[j'-1]}$, $\mathbf{x}^{[j']}$ and $\mathbf{x}^{[j'+1]}$ if $m^{[j']} < m^{[j'+1]}$ according to Theorem 6.12, case A

B) Let now $j \in \{L, \dots, J-1\} \setminus \{j'\}$ with j' as defined in (6.4).

(i) It follows from the proof of Lemma 6.9 that for decreasing j the support length $m^{[j]}$ of $\mathbf{x}^{[j]}$ cannot increase. Assume that there exists an index $j_1 \in \{L, \dots, J-1\} \setminus \{j'\}$ such that $m^{[j_1]} < m^{[j_1+1]}$. Case (iii) in the proof of Lemma 6.9 yields that $\{2^{j_1} - 1,\, 2^{j_1}\} \subseteq S^{[j_1+1]}$, because otherwise we would have that $m^{[j_1]} = m^{[j_1+1]}$. As $m^{[j_1+1]} \leq m \leq M$, this implies

$$S^{[j_1+1]} \subsetneq I_{2^{j_1}-M,\, 2^{j_1}+M-1},$$

and hence, by Definition 6.4,

$$S^{[j_1]} \subseteq I_{2^{j_1}-M,\, 2^{j_1}-1}.$$

This is a contradiction, since $j_1 \in \{L, \dots, J-1\} \setminus \{j'\}$ and j' is, if it exists, the unique index that satisfies (6.4). Consequently, we obtain that $m^{[j]} = m^{[j+1]}$ for all $j \in \{L, \dots, J-1\} \setminus \{j'\}$.

(ii) For $j \in \{L, \dots, J-1\} \setminus \{j'\}$ we have that $m^{[j]} = m^{[j+1]}$ by (i), which also holds if j' does not exist. Hence, cases (i) and (ii) of the proof of Lemma 6.9 show that either

$$\mathbf{x}^{[j+1]} = \begin{pmatrix} \mathbf{x}^{[j]} \\ \mathbf{0}_{2^j} \end{pmatrix} \qquad \text{or} \qquad \mathbf{x}^{[j+1]} = \begin{pmatrix} \mathbf{0}_{2^j} \\ \mathbf{J}_{2^j}\mathbf{x}^{[j]} \end{pmatrix},$$

as these are the only two 2^{j+1}-length vectors arising from repeatedly applying the reflected periodization to a vector $\mathbf{x}$ with short support of length at most M that have the reflected periodization $\mathbf{x}^{[j]}$. In Figures 6.5 and 6.6 these two possibilities are depicted for the two different cases $j < j'$ and $j > j'$.

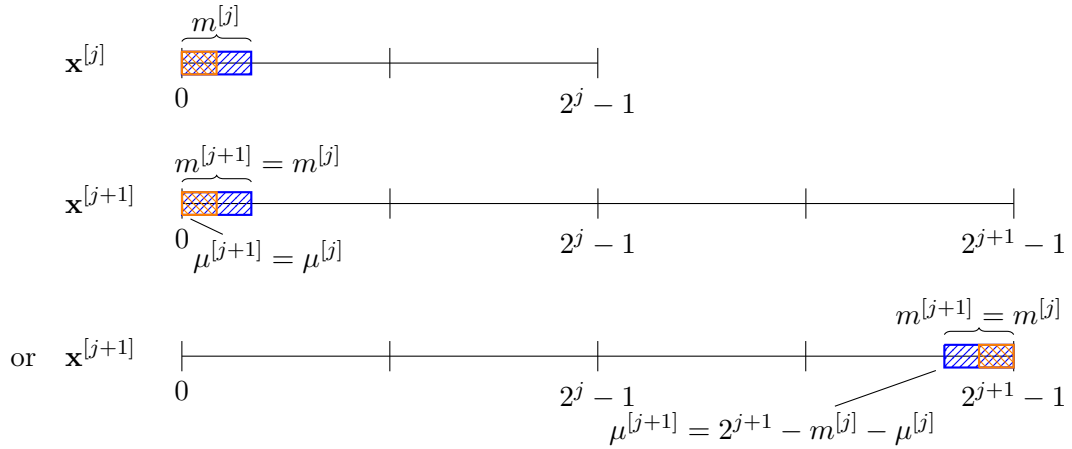

Figure 6.5: Illustration of the two possibilities for the support of $\mathbf{x}^{[j+1]}$ for given $\mathbf{x}^{[j]}$ according to Theorem 6.12, case B for $j \in \{L, \dots, j'-1\}$ with $m^{[j']} < m^{[j'+1]}$

□

Theorem 6.12 shows that even if we only know an upper bound M on m, there is at most one index j' such that the support of $\mathbf{x}^{[j']}$ is contained in its last M entries. This is also the only case for which the support length of the reflected periodization of double length can increase and for which one might have to undo collisions of nonzero entries in order to compute $\mathbf{x}^{[j'+1]}$ from $\mathbf{x}^{[j']}$. For all other indices the values of the nonzero entries of $\mathbf{x}^{[j]}$ and $\mathbf{x}^{[j+1]}$ are the same and there are only two possibilities for $\mathbf{x}^{[j+1]}$.

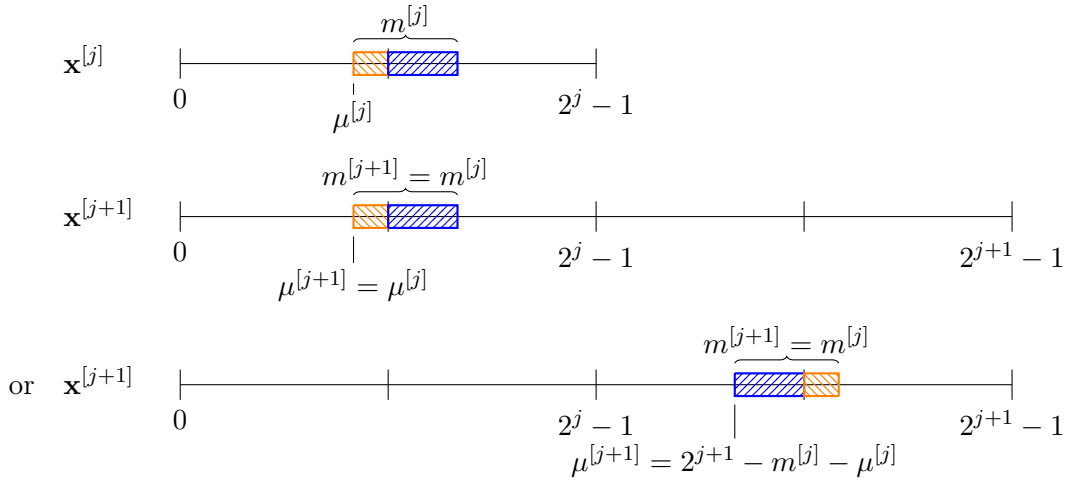

Figure 6.6: Illustration of the two possibilities for the support of $\mathbf{x}^{[j+1]}$ for $\mathbf{x}^{[j]}$ according to Theorem 6.12, case B for $j \in \{j'+1, \dots, J-1\}$ with $m^{[j']} < m^{[j'+1]}$

6.3 Iterative Sparse Recovery Procedures

Lemma 6.6 implies that if $\mathbf{x}^{\widehat{\Pi}}$ is known, the DCT-IIs of all reflected periodizations $\mathbf{x}^{[j]}$ are also known, as they can be obtained by selecting certain entries of $\mathbf{x}^{\widehat{\Pi}}$. Analogously to [PW16a, PW17a, PWCW18] and the methods detailed in Chapter 5, which is based on [BP18c], our goal is to develop an algorithm which recovers $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, with short support of length $m \leq M$ from $\mathbf{x}^{\widehat{\Pi}}$ by successively calculating the reflected periodizations $\mathbf{x}^{[L]}$, $\mathbf{x}^{[L+1]}, \dots, \mathbf{x}^{[J]} = \mathbf{x}$ for some starting index L satisfying $M \leq 2^{L-1}$.

It follows from Theorem 6.12 that there is an important difference between case A and case B. In case A, i.e., when the support of $\mathbf{x}^{[j]}$ is contained in its last M entries, some of the entries of $\mathbf{x}^{[j]}$ might have been obtained as the sum of two nonzero entries of $\mathbf{x}^{[j+1]}$ and thus also of $\mathbf{x}$. On the other hand, in case B, no entries of $\mathbf{x}^{[j]}$ are sums of nonzero entries of $\mathbf{x}^{[j+1]}$, though they might be sums of nonzero entries of $\mathbf{x}$ if $j \leq j'$, where j' is defined as in (6.4). Thus, the nonzero entries of $\mathbf{x}^{[j]}$ and $\mathbf{x}^{[j+1]}$, which are the only relevant vectors for the current iteration step, are the same and there are precisely two possibilities for $\mathbf{x}^{[j+1]}$.

Therefore, in this section we will develop two different methods for calculating $\mathbf{x}^{[j+1]}$: the first one is tailored to case A of Theorem 6.12 and the second one to case B. Both will require a priori knowledge of an upper bound M on the support length m of $\mathbf{x}$.

6.3.1 Recovery Procedure for Case A: Possible Collision

We start by introducing the reconstruction procedure for case A, so let us assume that $j \in \{L, \dots, J-1\}$ with $j = j'$, i.e., $S^{[j]} \subseteq I_{2^j - M, 2^j - 1}$. Then Theorem 6.12, case A yields that $S^{[j+1]} \subsetneq I_{2^j - M, 2^j + M - 1}$ and that the reflected periodization $\mathbf{x}^{[j]}$ may have been obtained by adding nonzero entries of $\mathbf{x}^{[j+1]}$. Thus, the values of the nonzero entries of $\mathbf{x}^{[j]}$ and $\mathbf{x}^{[j+1]}$ are not necessarily the same. We can restrict the possible support of $\mathbf{x}^{[j+1]}$ even further, using that the support of $\mathbf{x}^{[j]}$ has length $m^{[j]} \leq m \leq M$. By Definition 6.4 and Theorem 6.12, case A, the support of the first half of $\mathbf{x}^{[j+1]}$, $S^{[j+1]}_{(0)}$, can have at most length $\widetilde{m}^{[j]} := 2^j - \mu^{[j]} \leq M$. See Figures 6.7 and 6.8 for illustrations.

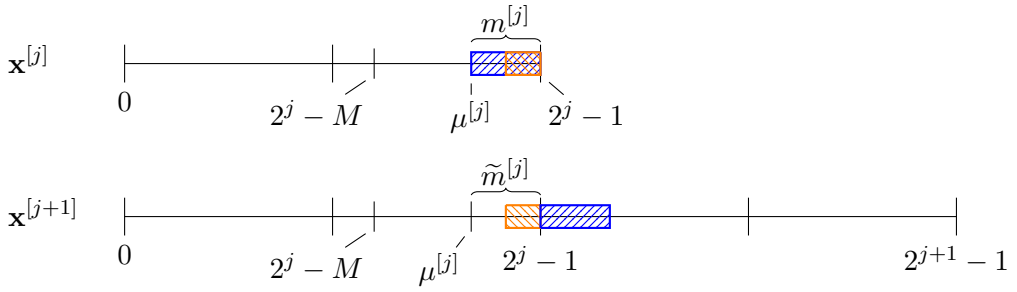

Figure 6.7: Illustration of the support of $\mathbf{x}^{[j]}$ and one possibility for the support of $\mathbf{x}^{[j+1]}$ for $m^{[j]} < m^{[j+1]}$ with $j = j'$

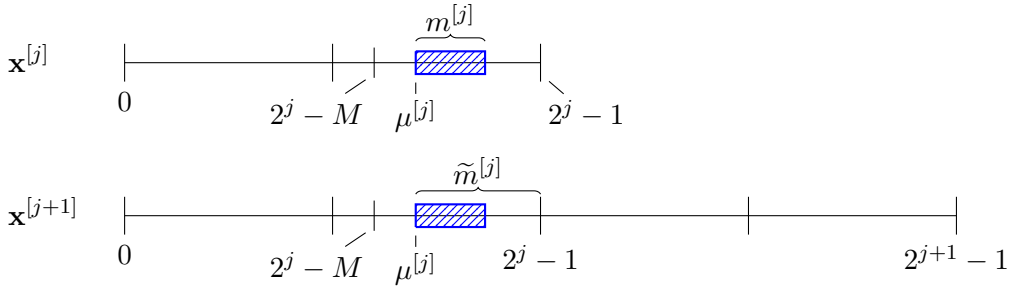

Figure 6.8: Illustration of the support of $\mathbf{x}^{[j]}$ and one possibility for the support of $\mathbf{x}^{[j+1]}$ for $m^{[j]} = m^{[j+1]}$ with $j = j'$

Consequently, we have that

$$S_{(0)}^{[j+1]} \subseteq I_{2^j - \widetilde{m}^{[j]}, 2^j - 1}.$$

By definition of the reflected periodization, $\mathbf{x}_{(1)}^{[j+1]}$ is completely determined by $\mathbf{x}^{[j]}$ and $\mathbf{x}_{(0)}^{[j+1]}$. As the support of $\mathbf{x}_{(0)}^{[j+1]}$ is contained in the short support of length $\widetilde{m}^{[j]}$ of $\mathbf{x}^{[j]}$, we have to recover at most $\widetilde{m}^{[j]}$ nonzero entries of $\mathbf{x}_{(0)}^{[j+1]}$. We will do this, analogously to Section 5.5.1, by considering restrictions of $\mathbf{x}^{[j]}$ and $\mathbf{x}_{(0)}^{[j+1]}$ to vectors of length $2^{\widetilde{K}-1}$, where $\widetilde{m}^{[j]} \leq 2^{\widetilde{K}-1}$, which take all nonzero entries into account. We will use these nonzero entries to show that $\mathbf{x}_{(0)}^{[j+1]}$ and thus $\mathbf{x}^{[j+1]}$ can be calculated essentially by a DCT-IV of length $2^{\widetilde{K}-1}$ and further operations of complexity $\mathcal{O}\left(2^{\widetilde{K}-1}\right)$. In order to do this we also have to employ the vector $\mathbf{x}^{[j]}$ known from the previous iteration step and $2^{\widetilde{K}}$ suitably chosen oddly indexed entries of $\left(\mathbf{x}^{[j+1]}\right)^{\widehat{\Pi}}$.

The efficient computation of $\mathbf{x}^{[j+1]}$ from $\mathbf{x}^{[j]}$ and $\mathbf{x}^{\widehat{\Pi}}$ is based on the following theorem.

Theorem 6.13 (Theorem 3.4 in [BP18a]) *Let $N = 2^J$ with $J \in \mathbb{N}$. Let $\mathbf{x} \in \mathbb{R}^N$ have a short support of length $m \leq M$ and assume that $\mathbf{x}$ satisfies (6.3). Suppose that we have access to all entries of $\mathbf{x}^{\widehat{\Pi}}$. Let $j = j'$ as in (6.4) and set $\widetilde{m}^{[j]} := 2^j - \mu^{[j]}$ and*

$\widetilde{K} \coloneqq \lceil \log_2 \widetilde{m}^{[j]} \rceil + 1$. *Then* $\mathbf{x}^{[j+1]}$ *can be uniquely recovered from* $\mathbf{x}^{[j]}$ *and the* $2^{\widetilde{K}}$ *entries*

$$\left(x^{\widehat{\mathrm{II}}}_{2^{J-j-1}(2\cdot 2^{j-\widetilde{K}}(2p+1)+1)}\right)_{p=0}^{2^{\widetilde{K}-1}-1} \qquad \text{and} \qquad \left(x^{\widehat{\mathrm{II}}}_{2^{J-j-1}(2(2^{j-\widetilde{K}}(2p+1)-1)+1)}\right)_{p=0}^{2^{\widetilde{K}-1}-1}.$$

Proof. It suffices to only consider the oddly indexed entries $\left(x^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1}$ of $\left(\mathbf{x}^{[j+1]}\right)^{\widehat{\mathrm{II}}}$ for $k \in \{0, \dots, 2^j - 1\}$. We obtain from (4.2) and (6.2) that

$$\begin{aligned}
\begin{pmatrix} \left(x^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k} \end{pmatrix}_{k=0}^{2^j-1} \\ \left(x^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1} \end{pmatrix}_{k=0}^{2^j-1} \end{pmatrix} &= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{2^j} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^j} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{2^j} & \mathbf{J}_{2^j} \\ \mathbf{I}_{2^j} & -\mathbf{J}_{2^j} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}^{[j+1]}_{(0)} \\ \mathbf{x}^{[j+1]}_{(1)} \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{2^j} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^j} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}^{[j]} \\ \mathbf{x}^{[j+1]}_{(0)} - \mathbf{J}_{2^j} \mathbf{x}^{[j+1]}_{(1)} \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} \left(\mathbf{x}^{[j]}\right)^{\widehat{\mathrm{II}}} \\ \left(2\mathbf{x}^{[j+1]}_{(0)} - \mathbf{x}^{[j]}\right)^{\widehat{\mathrm{IV}}} \end{pmatrix}, \qquad (6.5)
\end{aligned}$$

where we used that $\mathbf{J}_{2^j} \mathbf{x}^{[j+1]}_{(1)} = \mathbf{x}^{[j]} - \mathbf{x}^{[j+1]}_{(0)}$ by Definition 6.4.

If $j = j'$, Lemma 6.9 and Theorem 6.12, case A imply that the support interval $S^{[j]}$ of $\mathbf{x}^{[j]}$ satisfies

$$S^{[j]} \subseteq I_{\mu^{[j]}, 2^j - 1} \subseteq I_{2^j - M, 2^j - 1}.$$

With $\widetilde{m}^{[j]} \coloneqq 2^j - \mu^{[j]} \leq M$ and $\widetilde{K} \coloneqq \lceil \log_2 \widetilde{m}^{[j]} \rceil + 1$, we obtain

$$S^{[j]} \subseteq I_{2^j - \widetilde{m}^{[j]}, 2^j - 1} \subseteq I_{2^j - 2^{\widetilde{K}-1}, 2^j - 1},$$

and, by definition of the reflected periodization, the support set $S^{[j+1]}$ of $\mathbf{x}^{[j+1]}$ satisfies

$$S^{[j+1]} \subsetneq I_{2^j - \widetilde{m}^{[j]}, 2^j + \widetilde{m}^{[j]} - 1} \subseteq I_{2^j - 2^{\widetilde{K}-1}, 2^j + 2^{\widetilde{K}-1} - 1}. \qquad (6.6)$$

Consequently, both the first half of $\mathbf{x}^{[j+1]}$, $\mathbf{x}^{[j+1]}_{(0)}$, and the second half, $\mathbf{x}^{[j+1]}_{(1)}$, have a short support of length at most $\widetilde{m}^{[j]}$. Note that we always have that $\widetilde{m}^{[j]} \geq m^{[j]}$ and that $\widetilde{m}^{[j]} > m^{[j]}$ is possible if there is no collision. The latter happens if the support of $\mathbf{x}^{[j]}$ is contained in the last M entries, but $2^j - 1 \notin S^{[j]}$, as M is just an upper bound on the support length m of $\mathbf{x}$. The choice of $\widetilde{m}^{[j]}$ allows us to reduce the number of computations necessary to find $\mathbf{x}^{[j+1]}$. Since we only suppose that $x_{\mu^{[J]}} \neq 0$, $x_{\nu^{[J]}} \neq 0$ and $x_{\mu^{[J]}} + x_{\nu^{[J]}} \neq 0$ in (6.3), some of the last $\widetilde{m}^{[j]}$ entries of $\mathbf{x}^{[j]}$ might be zero, despite being obtained by adding two nonzero entries of $\mathbf{x}^{[j+1]}$. However, the first and last support index of $\mathbf{x}^{[j+1]}$ never cancel each other out, so we are always able to find the correct first support index $\mu^{[j]}$, which satisfies either $\mu^{[j]} = \mu^{[j+1]}$ or $\mu^{[j]} = 2^{j+1} - 1 - \nu^{[j+1]}$ by case (iii) in the proof of Lemma 6.9. We need to incorporate all entries of $\mathbf{x}^{[j]}$ that are influenced by possibly nonzero entries of $\mathbf{x}^{[j+1]}$. Hence, if we restrict $\mathbf{x}^{[j]}$ to its last $2^{\widetilde{K}-1} \geq \widetilde{m}^{[j]} = 2^j - \mu^{[j]}$ entries, i.e., to

$$\mathbf{z}^{[j]} \coloneqq \left(x^{[j]}_k\right)_{k=2^j - 2^{\widetilde{K}-1}}^{2^j - 1},$$

we take all of the at most $\widetilde{m}^{[j]}$ entries of $\mathbf{x}^{[j]}$ into account which can be obtained from possibly nonzero entries of $\mathbf{x}^{[j+1]}$ by reflectedly periodizing, as the support of $\mathbf{x}^{[j+1]}$ has to be contained in $I_{2^j-\widetilde{m}^{[j]},\,2^j+\widetilde{m}^{[j]}-1}$. Thus, $\mathbf{z}^{[j]}$ contains all the information of $\mathbf{x}^{[j]}$ necessary for recovering $\mathbf{x}^{[j+1]}$. Analogously, by (6.6), the $2^{\widetilde{K}-1}$-length vectors

$$\mathbf{z}_{(0)}^{[j+1]} \coloneqq \left(x_k^{[j+1]}\right)_{k=2^j-2^{\widetilde{K}-1}}^{2^j-1} \quad \text{and} \quad \mathbf{z}_{(1)}^{[j+1]} \coloneqq \left(x_k^{[j+1]}\right)_{k=2^j}^{2^j+2^{\widetilde{K}-1}-1}$$

take the at most $\widetilde{m}^{[j]}$ nonzero entries of $\mathbf{x}_{(0)}^{[j+1]}$ and $\mathbf{x}_{(1)}^{[j+1]}$ into account. Note that the restricted vectors still satisfy

$$\mathbf{z}^{[j]} = \mathbf{z}_{(0)}^{[j+1]} + \mathbf{J}_{2^{\widetilde{K}-1}} \mathbf{z}_{(1)}^{[j+1]}. \tag{6.7}$$

Therefore, it is enough to derive a fast algorithm for computing $\mathbf{z}_{(0)}^{[j+1]}$ that utilizes $\mathbf{z}^{[j]}$ and some entries of $\left(\mathbf{x}^{[j+1]}\right)^{\widehat{\Pi}}$. In order to do so we restrict the second 2^j equations in (6.5), i.e., the ones depending on $\mathbf{x}_{(0)}^{[j+1]}$, to the vectors $\mathbf{z}^{[j]}$ and $\mathbf{z}_{(0)}^{[j+1]}$, which yields

$$\begin{aligned}
&\left(\left(x^{[j+1]}\right)_{2k+1}^{\widehat{\Pi}}\right)_{k=0}^{2^j-1} \\
&= \frac{1}{\sqrt{2^j}} \left(\cos\left(\frac{(2k+1)(2l'+1)\pi}{4\cdot 2^j}\right)\right)_{k,\,l'=0}^{2^j-1} \left(2\mathbf{x}_{(0)}^{[j+1]} - \mathbf{x}^{[j]}\right) \\
&= \frac{1}{\sqrt{2^j}} \left(\cos\left(\frac{(2k+1)(2l'+1)\pi}{4\cdot 2^j}\right)\right)_{k=0,\,l'=2^j-2^{\widetilde{K}-1}}^{2^j-1} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]}\right) \\
&= \frac{1}{\sqrt{2^j}} \left(\cos\left(\frac{(2k+1)(2^{j+1}-(2l+1))\pi}{4\cdot 2^j}\right)\right)_{k,\,l=0}^{2^j-1,\,2^{\widetilde{K}-1}-1} \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]}\right) \\
&= \frac{1}{\sqrt{2^j}} \left(\cos\left(\frac{(2k+1)2^{j+1}\pi}{4\cdot 2^j}\right)\cos\left(\frac{(2k+1)(2l+1)\pi}{4\cdot 2^j}\right)\right. \\
&\quad \left. + \sin\left(\frac{(2k+1)2^{j+1}\pi}{4\cdot 2^j}\right)\sin\left(\frac{(2k+1)(2l+1)\pi}{4\cdot 2^j}\right)\right)_{k,\,l=0}^{2^j-1,\,2^{\widetilde{K}-1}-1} \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]}\right) \\
&= \frac{1}{\sqrt{2^j}} \left(\cos\left(\frac{(2k+1)\pi}{2}\right)\cos\left(\frac{(2k+1)(2l+1)\pi}{4\cdot 2^j}\right)\right. \\
&\quad \left. + \sin\left(\frac{(2k+1)\pi}{2}\right)\sin\left(\frac{(2k+1)(2l+1)\pi}{4\cdot 2^j}\right)\right)_{k,\,l=0}^{2^j-1,\,2^{\widetilde{K}-1}-1} \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]}\right) \\
&= \frac{1}{\sqrt{2^j}} \left((-1)^k \sin\left(\frac{(2k+1)(2l+1)\pi}{4\cdot 2^j}\right)\right)_{k,\,l=0}^{2^j-1,\,2^{\widetilde{K}-1}-1} \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]}\right),
\end{aligned} \tag{6.8}$$

where we set $l \coloneqq 2^j - 1 - l'$ in the third step. As $\mathbf{z}^{[j]}$ and $\mathbf{z}_{(0)}^{[j+1]}$ both have length $2^{\widetilde{K}-1}$, it suffices to consider $2^{\widetilde{K}-1}$ equations of (6.8). We choose the ones corresponding to the indices $2k_p+1$, where $k_p \coloneqq 2^{j-\widetilde{K}}(2p+1)$, $p \in \left\{0,\dots,2^{\widetilde{K}-1}-1\right\}$. Since we have that

$2k_p + 1 \in \{0, \dots, 2^{j+1} - 1\}$ for all p, we obtain

$$\begin{aligned}
&\sqrt{2^j}(-1)^{2^{j-\widetilde{K}}} \left(\left(x^{[j+1]} \right)^{\widehat{\Pi}}_{2k_p+1} \right)_{p=0}^{2^{\widetilde{K}-1}-1} \\
&= \left(\sin\left(\frac{\left(2^{j-\widetilde{K}+1}(2p+1)+1\right)(2l+1)\pi}{4 \cdot 2^j} \right) \right)_{p,\,l=0}^{2^{\widetilde{K}-1}-1} \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]} \right) \\
&= \left(\sin\left(\frac{(2p+1)(2l+1)\pi}{4 \cdot 2^{\widetilde{K}-1}} \right) \cos\left(\frac{(2l+1)\pi}{4 \cdot 2^j} \right) \right. \\
&\quad \left. + \cos\left(\frac{(2p+1)(2l+1)\pi}{4 \cdot 2^{\widetilde{K}-1}} \right) \sin\left(\frac{(2l+1)\pi}{4 \cdot 2^j} \right) \right)_{p,\,l=0}^{2^{\widetilde{K}-1}-1} \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]} \right). \qquad (6.9)
\end{aligned}$$

Defining the vectors

$$\mathbf{c} \coloneqq \left(\cos\left(\frac{(2l+1)\pi}{4 \cdot 2^j} \right) \right)_{l=0}^{2^{\widetilde{K}-1}-1} \quad \text{and} \quad \mathbf{s} \coloneqq \left(\sin\left(\frac{(2l+1)\pi}{4 \cdot 2^j} \right) \right)_{l=0}^{2^{\widetilde{K}-1}-1}$$

and recalling Definition 4.3 of the sine matrix of type IV, (6.9) can be written as

$$\begin{aligned}
&\sqrt{2^{j-\widetilde{K}+2}}(-1)^{2^{j-\widetilde{K}}} \left(\left(x^{[j+1]} \right)^{\widehat{\Pi}}_{2k_p+1} \right)_{p=0}^{2^{\widetilde{K}-1}-1} \\
&= \left(\mathbf{S}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} \cdot \operatorname{diag}(\mathbf{c}) + \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} \cdot \operatorname{diag}(\mathbf{s}) \right) \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]} \right) \\
&= \left(\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} \operatorname{diag}(\mathbf{s}) + \mathbf{J}_{2^{\widetilde{K}-1}} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} \mathbf{D}_{2^{\widetilde{K}-1}} \operatorname{diag}(\mathbf{c}) \right) \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]} \right) \\
&= \left(\begin{array}{c|c} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} & \mathbf{J}_{2^{\widetilde{K}-1}} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} \end{array} \right) \begin{pmatrix} \operatorname{diag}(\mathbf{s}) & \\ & \mathbf{D}_{2^{\widetilde{K}-1}} \operatorname{diag}(\mathbf{c}) \end{pmatrix} \begin{pmatrix} \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]} \right) \\ \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]} \right) \end{pmatrix}, \qquad (6.10)
\end{aligned}$$

where we used that, by Theorem 4.4 (ii),

$$\mathbf{S}^{\mathrm{IV}}_n = \mathbf{J}_n \mathbf{C}^{\mathrm{IV}}_n \mathbf{D}_n \quad \text{with} \quad \mathbf{D}_n = \operatorname{diag}\left((-1)^k \right)_{k=0}^{n-1} \quad \forall\, n \in \mathbb{N}.$$

Our aim is to find a representation of $\mathbf{z}_{(0)}^{[j+1]}$ depending only on $\mathbf{z}^{[j]}$ and some entries of $\mathbf{x}^{\widehat{\Pi}}$. However, as the first matrix in (6.10), $\left(\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} \mid \mathbf{J}_{2^{\widetilde{K}-1}} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} \right)$, is not a square matrix, we have to consider $2^{\widetilde{K}-1}$ additional equations from (6.8) in order to be able to invert (6.10) and solve it for $\mathbf{z}_{(0)}^{[j+1]}$. Now we choose the equations corresponding to the indices $2k'_p + 1$, where $k'_p \coloneqq 2^{j-\widetilde{K}}(2p+1) - 1$, $p \in \left\{ 0, \dots, 2^{\widetilde{K}-1} - 1 \right\}$. Since also

$2k'_p+1 \in \{0,\dots,2^{j+1}-1\}$ for all p, we find that

$$
\begin{aligned}
&\sqrt{2^{j-\widetilde{K}+2}}\left(\left(x^{[j+1]}\right)^{\widehat{\Pi}}_{2k'_p+1}\right)_{p=0}^{2^{\widetilde{K}-1}-1}\\
&=\frac{(-1)^{k'_p}}{\sqrt{2^{\widetilde{K}-2}}}\left(\sin\left(\frac{\left(2^{j-\widetilde{K}+1}(2p+1)-1\right)(2l+1)\pi}{4\cdot 2^j}\right)\right)_{p,\,l=0}^{2^{\widetilde{K}-1}-1}\mathbf{J}_{2^{\widetilde{K}-1}}\left(2\mathbf{z}^{[j+1]}_{(0)}-\mathbf{z}^{[j]}\right)\\
&=\frac{(-1)^{2^{j-\widetilde{K}}-1}}{\sqrt{2^{\widetilde{K}-2}}}\left(\sin\left(\frac{(2p+1)(2l+1)\pi}{4\cdot 2^{\widetilde{K}-1}}\right)\cos\left(\frac{(2l+1)\pi}{4\cdot 2^j}\right)\right.\\
&\quad\left.-\cos\left(\frac{(2p+1)(2l+1)\pi}{4\cdot 2^{\widetilde{K}-1}}\right)\sin\left(\frac{(2l+1)\pi}{4\cdot 2^j}\right)\right)_{p,\,l=0}^{2^{\widetilde{K}-1}-1}\mathbf{J}_{2^{\widetilde{K}-1}}\left(2\mathbf{z}^{[j+1]}_{(0)}-\mathbf{z}^{[j]}\right)\\
&=\frac{(-1)^{2^{j-\widetilde{K}}-1}}{\sqrt{2^{\widetilde{K}-2}}}\sqrt{2^{\widetilde{K}-2}}\left(\mathbf{S}^{\mathrm{IV}}_{2^{\widetilde{K}-1}}\cdot\operatorname{diag}(\mathbf{c})-\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}}\cdot\operatorname{diag}(\mathbf{s})\right)\mathbf{J}_{2^{\widetilde{K}-1}}\left(2\mathbf{z}^{[j+1]}_{(0)}-\mathbf{z}^{[j]}\right)\\
&=(-1)^{2^{j-\widetilde{K}}}\left(\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}}\cdot\operatorname{diag}(\mathbf{s})-\mathbf{S}^{\mathrm{IV}}_{2^{\widetilde{K}-1}}\cdot\operatorname{diag}(\mathbf{c})\right)\mathbf{J}_{2^{\widetilde{K}-1}}\left(2\mathbf{z}^{[j+1]}_{(0)}-\mathbf{z}^{[j]}\right)\\
&=(-1)^{2^{j-\widetilde{K}}}\left(\begin{array}{c|c}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} & -\mathbf{J}_{2^{\widetilde{K}-1}}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}}\end{array}\right)\begin{pmatrix}\operatorname{diag}(\mathbf{s}) & \\ & \mathbf{D}_{2^{\widetilde{K}-1}}\operatorname{diag}(\mathbf{c})\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{J}_{2^{\widetilde{K}-1}}\left(2\mathbf{z}^{[j+1]}_{(0)}-\mathbf{z}^{[j]}\right)\\ \mathbf{J}_{2^{\widetilde{K}-1}}\left(2\mathbf{z}^{[j+1]}_{(0)}-\mathbf{z}^{[j]}\right)\end{pmatrix}
\end{aligned}
\tag{6.11}
$$

Using Lemma 6.6, we denote by

$$
\mathbf{b}^0 \coloneqq \left(\left(x^{[j+1]}\right)^{\widehat{\Pi}}_{2k_p+1}\right)_{p=0}^{2^{\widetilde{K}-1}-1} = \sqrt{2}^{J-j-1}\left(x^{\widehat{\Pi}}_{2^{J-j-1}(2k_p+1)}\right)_{p=0}^{2^{\widetilde{K}-1}-1}\in\mathbb{R}^{2^{\widetilde{K}-1}} \qquad \text{and}
$$

$$
\mathbf{b}^1 \coloneqq \left(\left(x^{[j+1]}\right)^{\widehat{\Pi}}_{2k'_p+1}\right)_{p=0}^{2^{\widetilde{K}-1}-1} = \sqrt{2}^{J-j-1}\left(x^{\widehat{\Pi}}_{2^{J-j-1}(2k'_p+1)}\right)_{p=0}^{2^{\widetilde{K}-1}-1}\in\mathbb{R}^{2^{\widetilde{K}-1}}
$$

the two $2^{\widetilde{K}-1}$-length vectors of required entries of $\mathbf{x}^{\widehat{\Pi}}$. Combining (6.10) and (6.11) yields

$$
\begin{aligned}
&\sqrt{2^{j-\widetilde{K}+2}}(-1)^{2^{j-\widetilde{K}}}\begin{pmatrix}\mathbf{b}^0\\ \mathbf{b}^1\end{pmatrix}\\
&=\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} & \mathbf{J}_{2^{\widetilde{K}-1}}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}}\\ \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} & -\mathbf{J}_{2^{\widetilde{K}-1}}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}}\end{pmatrix}\begin{pmatrix}\operatorname{diag}(\mathbf{s}) & \\ & \mathbf{D}_{2^{\widetilde{K}-1}}\operatorname{diag}(\mathbf{c})\end{pmatrix}\begin{pmatrix}\mathbf{J}_{2^{\widetilde{K}-1}}\left(2\mathbf{z}^{[j+1]}_{(0)}-\mathbf{z}^{[j]}\right)\\ \mathbf{J}_{2^{\widetilde{K}-1}}\left(2\mathbf{z}^{[j+1]}_{(0)}-\mathbf{z}^{[j]}\right)\end{pmatrix}\\
&=\begin{pmatrix}\mathbf{I}_{2^{\widetilde{K}-1}} & \mathbf{J}_{2^{\widetilde{K}-1}}\\ \mathbf{I}_{2^{\widetilde{K}-1}} & -\mathbf{J}_{2^{\widetilde{K}-1}}\end{pmatrix}\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}}\end{pmatrix}\begin{pmatrix}\operatorname{diag}(\mathbf{s}) & \\ & \mathbf{D}_{2^{\widetilde{K}-1}}\operatorname{diag}(\mathbf{c})\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{J}_{2^{\widetilde{K}-1}}\left(2\mathbf{z}^{[j+1]}_{(0)}-\mathbf{z}^{[j]}\right)\\ \mathbf{J}_{2^{\widetilde{K}-1}}\left(2\mathbf{z}^{[j+1]}_{(0)}-\mathbf{z}^{[j]}\right)\end{pmatrix}.
\end{aligned}
\tag{6.12}
$$

Note that the first matrix in (6.12) is invertible, as

$$\begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}-1}} & \mathbf{J}_{2^{\widetilde{K}-1}} \\ \mathbf{I}_{2^{\widetilde{K}-1}} & -\mathbf{J}_{2^{\widetilde{K}-1}} \end{pmatrix} \cdot \frac{1}{2} \begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}-1}} & \mathbf{I}_{2^{\widetilde{K}-1}} \\ \mathbf{J}_{2^{\widetilde{K}-1}} & -\mathbf{J}_{2^{\widetilde{K}-1}} \end{pmatrix} = \begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}-1}} & \\ & \mathbf{I}_{2^{\widetilde{K}-1}} \end{pmatrix} = \mathbf{I}_{2^{\widetilde{K}}}. \tag{6.13}$$

Furthermore, since $\widetilde{m}^{[j]} \leq M$ and thus $\widetilde{K} \leq L \leq j$ by definition, it follows that

$$\frac{(2l+1)\pi}{4 \cdot 2^j} \in \left(0, \frac{\pi}{4}\right)$$

for all $l \in \left\{0, \ldots, 2^{\widetilde{K}-1} - 1\right\}$. Consequently, the entries of the vectors $\mathbf{c}$ and $\mathbf{s}$ satisfy

$$\cos\left(\frac{(2l+1)\pi}{4 \cdot 2^j}\right) \in \left(\frac{1}{\sqrt{2}}, 1\right) \qquad \text{and} \qquad \sin\left(\frac{(2l+1)\pi}{4 \cdot 2^j}\right) \in \left(0, \frac{1}{\sqrt{2}}\right). \tag{6.14}$$

This implies that the third matrix in (6.12) is invertible as well, since the multiplication of oddly indexed entries of $\mathbf{c}$ with -1, caused by $\mathbf{D}_{2^{\widetilde{K}-1}}$, does not change the absolute value of the determinant of the matrix. Thus, all matrices in (6.12) can be inverted and we obtain that

$$\begin{aligned}
&\begin{pmatrix} \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]}\right) \\ \mathbf{J}_{2^{\widetilde{K}-1}} \left(2\mathbf{z}_{(0)}^{[j+1]} - \mathbf{z}^{[j]}\right) \end{pmatrix} \\
&= \sqrt{2^{j-\widetilde{K}}}(-1)^{2^{j-\widetilde{K}}} \begin{pmatrix} \operatorname{diag}(\tilde{\mathbf{s}}) & \\ & \operatorname{diag}(\tilde{\mathbf{c}})\mathbf{D}_{2^{\widetilde{K}-1}} \end{pmatrix} \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} \end{pmatrix} \\
&\quad \cdot \begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}-1}} & \mathbf{I}_{2^{\widetilde{K}-1}} \\ \mathbf{J}_{2^{\widetilde{K}-1}} & -\mathbf{J}_{2^{\widetilde{K}-1}} \end{pmatrix} \begin{pmatrix} \mathbf{b}^0 \\ \mathbf{b}^1 \end{pmatrix} \\
&= \sqrt{2^{j-\widetilde{K}}}(-1)^{2^{j-\widetilde{K}}} \begin{pmatrix} \operatorname{diag}(\tilde{\mathbf{s}})\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} & \\ & \operatorname{diag}(\tilde{\mathbf{c}})\mathbf{D}_{2^{\widetilde{K}-1}}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} \end{pmatrix} \begin{pmatrix} \mathbf{b}^0 + \mathbf{b}^1 \\ \mathbf{J}_{2^{\widetilde{K}-1}} \left(\mathbf{b}^0 - \mathbf{b}^1\right) \end{pmatrix}
\end{aligned} \tag{6.15}$$

where

$$\tilde{\mathbf{c}} \coloneqq \left(\cos\left(\frac{(2l+1)\pi}{4 \cdot 2^j}\right)^{-1}\right)_{l=0}^{2^{\widetilde{K}-1}-1} \qquad \text{and} \qquad \tilde{\mathbf{s}} \coloneqq \left(\sin\left(\frac{(2l+1)\pi}{4 \cdot 2^j}\right)^{-1}\right)_{l=0}^{2^{\widetilde{K}-1}-1}.$$

For recovering $\mathbf{z}_{(0)}^{[j+1]}$ it suffices to use the second $2^{\widetilde{K}-1}$ equations in (6.15), which yields

$$\mathbf{z}_{(0)}^{[j+1]} = \frac{1}{2}\left(\sqrt{2^{j-\widetilde{K}}}(-1)^{2^{j-\widetilde{K}}} \mathbf{J}_{2^{\widetilde{K}-1}} \operatorname{diag}(\tilde{\mathbf{c}}) \mathbf{D}_{2^{\widetilde{K}-1}} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}-1}} \mathbf{J}_{2^{\widetilde{K}-1}} \left(\mathbf{b}^0 - \mathbf{b}^1\right) + \mathbf{z}^{[j]}\right). \tag{6.16}$$

Then, by (6.7), $\mathbf{z}_{(1)}^{[j+1]}$ can be found in $\mathcal{O}\left(2^{\widetilde{K}-1}\right)$ time as

$$\mathbf{z}_{(1)}^{[j+1]} = \mathbf{J}_{2^{\widetilde{K}-1}} \left(\mathbf{z}^{[j]} - \mathbf{z}_{(0)}^{[j+1]}\right).$$

Thus, by definition of $\mathbf{z}_{(0)}^{[j+1]}$ and $\mathbf{z}_{(1)}^{[j+1]}$, the reflected periodization $\mathbf{x}^{[j+1]}$ is given as

$$x_k^{[j+1]} = \begin{cases} \left(z_{(0)}^{[j+1]}\right)_{k-2^j+2^{\widetilde{K}-1}} & \text{if } k \in \left\{2^j - 2^{\widetilde{K}-1}, \ldots, 2^j - 1\right\}, \\ \left(z_{(1)}^{[j+1]}\right)_{k-2^j} & \text{if } k \in \left\{2^j, \ldots, 2^j + 2^{\widetilde{K}-1} - 1\right\}, \\ 0 & \text{else} \end{cases}$$

for $k \in \left\{0, \ldots, 2^{j+1} - 1\right\}$, since all possibly nonzero entries of $\mathbf{x}^{[j+1]}$ are determined by $\mathbf{z}_{(0)}^{[j+1]}$ and $\mathbf{z}_{(1)}^{[j+1]}$. Note that $\mathbf{D}_{2^{\widetilde{K}-1}}$ is a diagonal matrix and $\mathbf{J}_{2^{\widetilde{K}-1}}$ is a permutation. Thus, (6.16) implies that $\mathbf{z}_{(0)}^{[j+1]}$ can be computed by performing a DCT-IV with a runtime of $\mathcal{O}\left(2^{\widetilde{K}-1} \log 2^{\widetilde{K}-1}\right)$, and $\mathcal{O}\left(2^{\widetilde{K}-1}\right)$ further operations, while using $2^{\widetilde{K}}$ entries of $\mathbf{x}^{\widehat{\Pi}}$. □

Remark 6.14 Note that by choosing to compute $\mathbf{z}_{(0)}^{[j+1]}$ from the second $2^{\widetilde{K}-1}$ equations in (6.15), we avoid inverting $\operatorname{diag}(\mathbf{s})$, which would be numerically less stable than inverting $\operatorname{diag}(\mathbf{c})$, since for large $\widetilde{K}$ its nonzero entries are rather close to zero, whereas all nonzero entries of $\operatorname{diag}(\mathbf{c})$ are greater than $\frac{1}{\sqrt{2}}$. ◊

As by restricting $\mathbf{x}^{[j]}$ and $\mathbf{x}_{(0)}^{[j+1]}$ to vectors of length $2^{\widetilde{K}-1}$ we can recover $\mathbf{x}^{[j+1]}$ in $\mathcal{O}\left(\tilde{m}^{[j]} \log m^{[j]}\right) = \mathcal{O}\left(M \log M\right)$ time using $2^{\widetilde{K}} = \mathcal{O}(M)$ samples of $\mathbf{x}^{\widehat{\Pi}}$, we have reason to believe that the runtime reduction for case A is sufficient to obtain an IDCT-II algorithm with overall runtime that is sublinear in the vector length, provided that we can also find a fast method for case B. Furthermore, the procedure for case A has to be executed at most once by Theorem 6.12. Since we do not know a priori for which $j' \in \{L, \ldots, J-1\}$ this is the case, we need the runtime of the procedure from Theorem 6.13 to be subquadratic in the bound M on the support length m of $\mathbf{x}$ in order to obtain an overall runtime that is subquadratic in M. Note that a priori knowledge of the upper bound M is an integral part of the procedure detailed in this section.

6.3.2 Recovery Procedure for Case B: No Collision

We still have to derive a method for case B of Theorem 6.12, so let us now suppose that $j \in \{L, \ldots, J-1\} \setminus \{j'\}$, where j' is given by (6.4), i.e., that $S^{[j]} \not\subseteq I_{2^j-M,\,2^j-1}$. Then case B of Theorem 6.12 implies that if $S^{[j]} = I_{\mu^{[j]},\,\nu^{[j]}}$, the values of the nonzero entries of $\mathbf{x}^{[j]}$ and $\mathbf{x}^{[j+1]}$ are the same, with $m^{[j+1]} = m^{[j]}$ and

$$S^{[j+1]} = I_{\mu^{[j]},\,\nu^{[j]}} \qquad \text{or} \qquad S^{[j+1]} = I_{2^{j+1}-1-\nu^{[j]},\,2^{j+1}-1-\mu^{[j]}}.$$

Hence, we only need to determine whether the first support index of $\mathbf{x}^{[j+1]}$ is $\mu^{[j+1]} = \mu^{[j]}$ or $\mu^{[j+1]} = 2^{j+1} - 1 - \nu^{[j]}$, i.e.,

$$\mathbf{x}^{[j+1]} = \begin{pmatrix} \mathbf{x}^{[j]} \\ \mathbf{0}_{2^j} \end{pmatrix} \qquad \text{or} \qquad \mathbf{x}^{[j+1]} = \begin{pmatrix} \mathbf{0}_{2^j} \\ \mathbf{J}_{2^j}\mathbf{x}^{[j]} \end{pmatrix}.$$

We can determine which is the correct first support index by employing a nonzero entry of $\left(\mathbf{x}^{[j+1]}\right)^{\widehat{\Pi}}$, similarly to the recovery procedure described in Section 5.5.2. First we show how such a nonzero entry can be found efficiently, for which we will employ the odd Vandermonde matrices defined in Section 4.4.

Lemma 6.15 (Lemma 3.2 in [BP18a]) Let $N = 2^J$ with $J \in \mathbb{N}$. Let $\mathbf{x} \in \mathbb{R}^N$ have a short support of length $m \leq M$ and assume that $\mathbf{x}$ satisfies (6.3). Suppose that we have access to all entries of $\mathbf{x}^{\widehat{\mathrm{II}}}$. Set $L \coloneqq \lceil \log_2 M \rceil + 1$ and $j \in \{L, \dots, J-1\} \setminus \{j'\}$ with j' as in (6.4). Then $\left(\left(x^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1} \right)_{k=0}^{m^{[j]}-1}$ has at least one nonzero entry.

Proof. If $\mathbf{x}^{[j]}$ has the short support $S^{[j]} = I_{\mu^{[j]}, \nu^{[j]}}$ of length $m^{[j]}$ for some $\mu^{[j]} \in I_{0,\, 2^j - m^{[j]}}$ and $\nu^{[j]} = \mu^{[j]} + m^{[j]} - 1$, then by Theorem 6.12, case B, $\mathbf{x}^{[j+1]}$ has the short support $S^{[j+1]}$ of length $m^{[j+1]} = m^{[j]}$ and either

$$\mathbf{x}^{[j+1]} = \begin{pmatrix} \mathbf{x}^{[j]} \\ \mathbf{0}_{2^j} \end{pmatrix} \qquad \text{or} \qquad \mathbf{x}^{[j+1]} = \begin{pmatrix} \mathbf{0}_{2^j} \\ \mathbf{J}_{2^j} \mathbf{x}^{[j]} \end{pmatrix}. \tag{6.17}$$

We want to guarantee the existence of an oddly indexed nonzero entry of $\left(\mathbf{x}^{[j+1]}\right)^{\widehat{\mathrm{II}}}$ by only looking at the first $m^{[j]}$ ones. Recall that it follows from (6.5) that

$$\left(\left(x^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1} \right)_{k=0}^{2^j-1} = \frac{1}{\sqrt{2}} \mathbf{C}^{\mathrm{IV}}_{2^j} \left(2\mathbf{x}^{[j+1]}_{(0)} - \mathbf{x}^{[j]} \right). \tag{6.18}$$

Let us denote the support interval of $\mathbf{x}^{[j+1]}_{(0)}$ by $S^{[j+1]}_{(0)}$ and the support interval of an arbitrary vector $\mathbf{y} \in \mathbb{R}^n$ with short support by $S(\mathbf{y})$. Then (6.17) yields that $S^{[j+1]}_{(0)} = S^{[j]}$ or $S^{[j+1]}_{(0)} = \emptyset$. Consequently, $S^{[j+1]}_{(0)} \subseteq S^{[j]}$ and $S\left(2\mathbf{x}^{[j+1]}_{(0)} - \mathbf{x}^{[j]}\right) \subseteq S^{[j]}$. Restricting (6.18) to the rows corresponding to the first $m^{[j]}$ oddly indexed entries of $\left(\mathbf{x}^{[j+1]}\right)^{\widehat{\mathrm{II}}}$, we find that

$$\begin{aligned}
\left(\left(x^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1} \right)_{k=0}^{m^{[j]}-1} &= \frac{1}{\sqrt{2}} \left(\left(\mathbf{C}^{\mathrm{IV}}_{2^j}\right)_{k,\,l} \right)_{k,\,l=0}^{m^{[j]}-1,\, 2^j-1} \left(2\mathbf{x}^{[j+1]}_{(0)} - \mathbf{x}^{[j]} \right) \\
&= \frac{1}{\sqrt{2^j}} \left(\sum_{l \in S^{[j]}} \cos\left(\frac{(2k+1)(2l+1)\pi}{4 \cdot 2^j} \right) \left(2\mathbf{x}^{[j+1]}_{(0)} - \mathbf{x}^{[j]} \right)_l \right)_{k=0}^{m^{[j]}-1} \\
&= \frac{1}{\sqrt{2^j}} \cdot \mathbf{T}^{[j]} \cdot \left(\left(2\mathbf{x}^{[j+1]}_{(0)} - \mathbf{x}^{[j]} \right)_l \right)_{l \in S^{[j]}},
\end{aligned} \tag{6.19}$$

where

$$\mathbf{T}^{[j]} \coloneqq \left(\cos\left(\frac{(2k+1)(2l+1)\pi}{4 \cdot 2^j} \right) \right)_{k=0,\, l \in S^{[j]}}^{m^{[j]}-1}$$

is the restriction of the cosine matrix of type IV without the normalization factor to the first $m^{[j]}$ rows and the $m^{[j]}$ columns indexed by $S^{[j]}$. Since $\left|S^{[j]}\right| = m^{[j]}$, $\mathbf{T}^{[j]}$ is a quadratic matrix.

We assume now that the claim is false, i.e., that $\left(\left(x^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1} \right)_{k=0}^{m^{[j]}-1} = \mathbf{0}_{m^{[j]}}$. By showing that $\mathbf{T}^{[j]}$ is invertible, which is possible with the help of Chebyshev polynomials, (6.19) will yield a contradiction. Recall that by Lemma 4.14 (v), the Chebyshev

polynomial of the first kind of degree n can be written as

$$T_n(x) = \cos(n \arccos x) =: \sum_{l=0}^{n} \alpha_{n,\,l} x^l$$

if $|x| \leq 1$. Note that it follows from Lemma 4.14 (iii) that we have for any $k \in \mathbb{N}_0$

$$\alpha_{2k+1,\,2l} = 0 \qquad \forall\, l \in \{0, \ldots, k\},$$

since T_{2k+1} is an odd polynomial. Then Lemma 4.14 (vi) and the coefficient representation of the Chebyshev polynomials yield that

$$\begin{aligned}
&\mathbf{T}^{[j]} \\
&= \left(\cos\left(\frac{(2k+1)(2l+1)\pi}{2 \cdot 2^{j+1}} \right) \right)_{k=0,\, l \in S^{[j]}}^{m^{[j]}-1} \\
&= \left(T_{2k+1}\left(t_{2^{j+1},\,l} \right) \right)_{k=0,\, l \in S^{[j]}}^{m^{[j]}-1} \\
&= \left(\sum_{\substack{r'=0 \\ r' \equiv 1 \bmod 2}}^{2k+1} \alpha_{2k+1,\,r'} \cdot {t_{2^{j+1},\,l}}^{r'} \right)_{k=0,\, l \in S^{[j]}}^{m^{[j]}-1} \\
&= \left(\alpha_{2k+1,\,2r+1} \right)_{k,\,r=0}^{m^{[j]}-1} \cdot \left({t_{2^{j+1},\,l}}^{2r+1} \right)_{r=0,\, l \in S^{[j]}}^{m^{[j]}-1} \qquad (6.20) \\
&= \begin{pmatrix}
\alpha_{1,1} & 0 & 0 & \cdots & 0 \\
\alpha_{3,1} & \alpha_{3,3} & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & & 0 \\
\alpha_{2m^{[j]}-1,1} & \alpha_{2m^{[j]}-1,3} & a_{2m^{[j]}-1,5} & \cdots & \alpha_{2m^{[j]}-1,2m^{[j]}-1}
\end{pmatrix}
\begin{pmatrix}
\left(t_{2^{j+1},\,l} \right)_{l \in S^{[j]}}^{T} \\
\left({t_{2^{j+1},\,l}}^{3} \right)_{l \in S^{[j]}}^{T} \\
\vdots \\
\left({t_{2^{j+1},\,l}}^{2m^{[j]}-1} \right)_{l \in S^{[j]}}^{T}
\end{pmatrix} \\
&=: \mathbf{X}_{\mathrm{odd}}^{[j]} \cdot \mathbf{V}^{\mathrm{odd}} \left(\left(t_{2^{j+1},\,l} \right)_{l \in S^{[j]}} \right)^{T}, \qquad (6.21)
\end{aligned}$$

where we set $a_{2k+1,\,2r+1} \coloneqq 0$ for $r \in \{k+1, \ldots, m^{[j]}-1\}$ in (6.20). Recall that $\mathbf{V}^{\mathrm{odd}} \left(\left(t_{2^{j+1},\,l} \right)_{l \in S^{[j]}} \right)$ denotes the odd Vandermonde matrix defined in Definition 4.15. By Lemma 4.14 (ii), the triangular matrix $\mathbf{X}_{\mathrm{odd}}^{[j]}$ in (6.21) is invertible. Furthermore, since $S^{[j]} \subseteq I_{0,\,2^j-1}$, it follows that

$$\frac{(2l+1)\pi}{2 \cdot 2^{j+1}} \in \left(0, \frac{\pi}{2}\right)$$

for all $l \in S^{[j]}$. Consequently, we have that

$$t_{2^{j+1},\,l} = \cos\left(\frac{(2l+1)\pi}{2 \cdot 2^{j+1}} \right) \in (0, 1).$$

This implies that $\left| t_{2^{j+1},\,k} \right| \neq \left| t_{2^{j+1},\,l} \right|$ for all $k \neq l$, $k, l \in S^{[j]}$, as the cosine is bijective on $\left(0, \frac{\pi}{2}\right)$. Hence, the transposed odd Vandermonde matrix in (6.21), $\mathbf{V}^{\mathrm{odd}} \left(\left(t_{2^{j+1},\,l} \right)_{l \in S^{[j]}} \right)^{T}$, is invertible by Lemma 4.16, so $\mathbf{T}^{[j]}$ can also be inverted.

Recall that we assumed that $\big(x^{[j+1]}\big)^{\widehat{\mathrm{II}}}_{2k+1} = 0$ for all $k \in \big\{0, \dots, m^{[j]} - 1\big\}$. Then (6.19) and (6.21) yield

$$\begin{aligned} \mathbf{0}_{m^{[j]}} &= \left(\left(x^{[j+1]} \right)^{\widehat{\mathrm{II}}}_{2k+1} \right)_{k=0}^{m^{[j]}-1} \\ &= \frac{1}{\sqrt{2^j}} \mathbf{T}^{[j]} \left(\left(2\mathbf{x}^{[j+1]}_{(0)} - \mathbf{x}^{[j]} \right)_l \right)_{l \in S^{[j]}} \\ \Leftrightarrow \quad \mathbf{0}_{m^{[j]}} &= \left(\left(2\mathbf{x}^{[j+1]}_{(0)} - \mathbf{x}^{[j]} \right)_l \right)_{l \in S^{[j]}} . \end{aligned} \tag{6.22}$$

However, since $j \neq j'$, we have by (6.17) that either

$$\mathbf{x}^{[j+1]} = \begin{pmatrix} \mathbf{x}^{[j]} \\ \mathbf{0}_{2^j} \end{pmatrix} \qquad \text{or} \qquad \mathbf{x}^{[j+1]} = \begin{pmatrix} \mathbf{0}_{2^j} \\ \mathbf{J}_{2^j} \mathbf{x}^{[j]} \end{pmatrix}.$$

In either case, (6.22) is only possible if $\mathbf{x}^{[j]} = \mathbf{0}_{2^j}$. This is a contradiction, as by assumption $\mathbf{x} \neq \mathbf{0}_N$ has a short support of length m and satisfies (6.3). Hence, there exists an index $k_0 \in \big\{0, \dots, m^{[j]} - 1\big\}$ such that $\big(x^{[j+1]}\big)^{\widehat{\mathrm{II}}}_{2k_0+1} \neq 0$. □

Remark 6.16 For obtaining an efficient and stable implementation of the recovery procedure for case B, we set

$$k_0 \coloneqq \operatorname*{argmax}_{k \in \{0, \dots, m^{[j]}-1\}} \left\{ \left| \sqrt{2}^{J-j-1} x^{\widehat{\mathrm{II}}}_{2^{J-j-1}(2k+1)} \right| \right\},$$

where we use that $\big(\mathbf{x}^{[j+1]}\big)^{\widehat{\mathrm{II}}}$ is given via Lemma 6.6. Then $\big(x^{[j+1]}\big)^{\widehat{\mathrm{II}}}_{2k_0+1} \neq 0$ and it is likely that this entry is not too close to zero, which is supported empirically by the numerical experiments in Section 6.5. ◊

Now that it is guaranteed that at least one of the first $m^{[j]}$ oddly indexed entries of $\big(\mathbf{x}^{[j+1]}\big)^{\widehat{\mathrm{II}}}$ is nonzero, we will show how $\mathbf{x}^{[j+1]}$ can be computed from $\mathbf{x}^{[j]}$ and one such nonzero entry in the following theorem.

Theorem 6.17 (Theorem 3.3 in [BP18a]) *Let $N = 2^J$ with $J \in \mathbb{N}$. Let $\mathbf{x} \in \mathbb{R}^N$ have a short support of length $m \leq M$ and assume that $\mathbf{x}$ satisfies (6.3). Suppose that we have access to all entries of $\mathbf{x}^{\widehat{\mathrm{II}}}$. Set $L \coloneqq \lceil \log_2 M \rceil + 1$, let j' as in (6.4) and let $j \in \{L, \dots, J-1\} \setminus \{j'\}$. Then $\mathbf{x}^{[j+1]}$ can be uniquely recovered from $\mathbf{x}^{[j]}$ and one nonzero entry of $\left(\sqrt{2}^{J-j-1} x^{\widehat{\mathrm{II}}}_{2^{J-j-1}(2k+1)} \right)_{k=0}^{m^{[j]}-1}$.*

Proof. If $\mathbf{x}^{[j]}$ is known and has the short support $S^{[j]} = I_{\mu^{[j]}, \nu^{[j]}}$ of length $m^{[j]}$ for some $\mu^{[j]} \in I_{0, 2^j - m^{[j]}}$ and $\nu^{[j]} = \mu^{[j]} + m^{[j]} - 1$, there are precisely two vectors in $\mathbb{R}^{2^{j+1}}$ that arise from repeatedly applying the reflected periodization to $\mathbf{x}$ and have the given reflected periodization $\mathbf{x}^{[j]}$ by Theorem 6.12, case B. They are

$$\mathbf{u}^0 \coloneqq \begin{pmatrix} \mathbf{x}^{[j]} \\ \mathbf{0}_{2^j} \end{pmatrix} \quad \text{and} \quad \mathbf{u}^1 \coloneqq \begin{pmatrix} \mathbf{0}_{2^j} \\ \mathbf{J}_{2^j} \mathbf{x}^{[j]} \end{pmatrix},$$

with short support

$$S\left(\mathbf{u}^0\right) = I_{\mu^{[j]},\,\nu^{[j]}} \qquad \text{and} \qquad S\left(\mathbf{u}^1\right) = I_{2^{j+1}-1-\nu^{[j]},\,2^{j+1}-1-\mu^{[j]}},$$

see also Figures 6.5 and 6.6. Hence, the vector $\mathbf{u}^0$ has the first support index $\mu^{[j]}$, the vector $\mathbf{u}^1$ has the first support index $2^{j+1} - m^{[j]} - \mu^{[j]}$ and both have a support of length $m^{[j+1]} = m^{[j]}$.

Analogously to the approach in Theorem 5.23, let us now compare the DCT-IIs of $\mathbf{u}^0$ and $\mathbf{u}^1$. Lemma 4.5 yields

$$\begin{aligned}
\begin{pmatrix} \left(\left(u^0\right)^{\widehat{\mathrm{II}}}_{2k}\right)_{k=0}^{2^j-1} \\ \left(\left(u^0\right)^{\widehat{\mathrm{II}}}_{2k+1}\right)_{k=0}^{2^j-1} \end{pmatrix} &= \mathbf{P}_{2^{j+1}} \left(\mathbf{u}^0\right)^{\widehat{\mathrm{II}}} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{2^j} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^j} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{2^j} & \mathbf{J}_{2^j} \\ \mathbf{I}_{2^j} & -\mathbf{J}_{2^j} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}^{[j]} \\ \mathbf{0}_{2^j} \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{2^j} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^j} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}^{[j]} \\ \mathbf{x}^{[j]} \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} \left(\mathbf{x}^{[j]}\right)^{\widehat{\mathrm{II}}} \\ \left(\mathbf{x}^{[j]}\right)^{\widehat{\mathrm{IV}}} \end{pmatrix}
\end{aligned}$$

and

$$\begin{aligned}
\begin{pmatrix} \left(\left(u^1\right)^{\widehat{\mathrm{II}}}_{2k}\right)_{k=0}^{2^j-1} \\ \left(\left(u^1\right)^{\widehat{\mathrm{II}}}_{2k+1}\right)_{k=0}^{2^j-1} \end{pmatrix} &= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{2^j} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^j} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{2^j} & \mathbf{J}_{2^j} \\ \mathbf{I}_{2^j} & -\mathbf{J}_{2^j} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{0}_{2^j} \\ \mathbf{J}_{2^j}\mathbf{x}^{[j]} \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{2^j} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^j} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{J}_{2^j}\left(\mathbf{J}_{2^j}\mathbf{x}^{[j]}\right) \\ -\mathbf{J}_{2^j}\left(\mathbf{J}_{2^j}\mathbf{x}^{[j]}\right) \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} \left(\mathbf{x}^{[j]}\right)^{\widehat{\mathrm{II}}} \\ -\left(\mathbf{x}^{[j]}\right)^{\widehat{\mathrm{IV}}} \end{pmatrix}.
\end{aligned}$$

Consequently, we find that

$$\left(u^1\right)^{\widehat{\mathrm{II}}}_{2k+1} = -\left(u^0\right)^{\widehat{\mathrm{II}}}_{2k+1}, \quad k \in \left\{0, \ldots, 2^j - 1\right\}, \tag{6.23}$$

for all oddly indexed entries of $\left(\mathbf{u}^0\right)^{\widehat{\mathrm{II}}}$ and $\left(\mathbf{u}^1\right)^{\widehat{\mathrm{II}}}$. In order to decide whether $\mathbf{x}^{[j+1]} = \mathbf{u}^0$ or $\mathbf{x}^{[j+1]} = \mathbf{u}^1$, we compare a nonzero entry

$$\left(x^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_0+1} = \sqrt{2}^{J-j-1} x^{\widehat{\mathrm{II}}}_{2^{J-j-1}(2k_0+1)} \neq 0$$

to the corresponding entry of $\mathbf{u}^0$. By Lemma 6.15 such an entry can be found by examining $m^{[j]}$ entries of $\mathbf{x}^{\widehat{\mathrm{II}}}$. If $\left(u^0\right)^{\widehat{\mathrm{II}}}_{2k_0+1} = \left(x^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_0+1}$, it follows that $\mathbf{x}^{[j+1]} = \mathbf{u}^0$ by (6.23),

and if $\left(u^0\right)^{\widehat{\mathrm{II}}}_{2k_0+1} = -\left(x^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_0+1}$, then we must have that $\mathbf{x}^{[j+1]} = \mathbf{u}^1$. Numerically, we set $\mathbf{x}^{[j+1]} = \mathbf{u}^0$ if

$$\left|\left(u^0\right)^{\widehat{\mathrm{II}}}_{2k_0+1} - \sqrt{2}^{J-j-1} x^{\widehat{\mathrm{II}}}_{2^{J-j-1}(2k_0+1)}\right| < \left|\left(u^0\right)^{\widehat{\mathrm{II}}}_{2k_0+1} + \sqrt{2}^{J-j-1} x^{\widehat{\mathrm{II}}}_{2^{J-j-1}(2k_0+1)}\right|,$$

and $\mathbf{x}^{[j+1]} = \mathbf{u}^1$ otherwise. The required entry of $\mathbf{u}^0$ can be computed from $\mathbf{x}^{[j]}$ using $\mathcal{O}\left(m^{[j]}\right) = \mathcal{O}(m)$ operations,

$$\begin{aligned}\left(u^0\right)^{\widehat{\mathrm{II}}}_{2k_0+1} &= \sum_{l=0}^{2^{j+1}-1} \left(\mathbf{C}^{\mathrm{II}}_{2^{j+1}}\right)_{2k_0+1,\,l}\, u^0_l \\ &= \sum_{l=0}^{m^{[j]}-1} \left(\mathbf{C}^{\mathrm{II}}_{2^{j+1}}\right)_{2k_0+1,\,\mu^{[j]}+l}\, x^{[j]}_{\mu^{[j]}+l},\end{aligned}$$

since the support of $\mathbf{x}^{[j]}$ and thus of $\mathbf{u}^0$ is already known from the previous iteration step. Furthermore, the first support index $\mu^{[j+1]}$ of $\mathbf{x}^{[j+1]}$ is given via

$$\mu^{[j+1]} := \begin{cases}\mu^{[j]} & \text{if } \mathbf{x}^{[j+1]} = \mathbf{u}^0, \\ 2^{j+1} - m^{[j]} - \mu^{[j]} & \text{if } \mathbf{x}^{[j+1]} = \mathbf{u}^1,\end{cases}$$

so we know which of the two possible first support indices from Theorem 6.12 case B is attained. □

Recovering the vector $\mathbf{x}^{[j+1]}$ from $\mathbf{x}^{[j]}$ and an oddly indexed nonzero entry of $\left(\mathbf{x}^{[j+1]}\right)^{\widehat{\mathrm{II}}}$ via the procedure for case B has a runtime of $\mathcal{O}\left(m^{[j]}\right) = \mathcal{O}(m)$ and requires at most $m^{[j]} = \mathcal{O}(m)$ samples of $\mathbf{x}^{\widehat{\mathrm{II}}}$. This contributes decidedly to obtaining an IDCT-II algorithm with an overall runtime that is sublinear in the vector length N, since the procedure for case A has to be executed at most once by Theorem 6.12.

Note that, as for the procedure described in Section 6.3.1, a priori knowledge of an upper bound M on the block length m is indispensable.

6.4 Real Sparse Fast IDCT-II for Vectors with Short Support

In Section 6.3 we introduced the iterative methods required for the new sparse IDCT-II for vectors $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, with short support of length $m \leq M$ that satisfy (6.3). In this section we will summarize them into both an algorithm for the case that the support length m of $\mathbf{x}$ is known exactly and an algorithm that only requires an upper bound $M \geq m$ on the support length. We will begin by detailing an algorithm for the case that only an upper bound M on m is known and investigating its theoretical runtime and sampling complexity. Afterwards, we will focus on the special case that the support length m is known exactly, as in that case the more time consuming procedure from Section 6.3.1 only has to be applied when there indeed has been collision of possibly nonzero entries of $\mathbf{x}$. Such an algorithm can be obtained easily by modifying the more general method.

6.4.1 Sparse Fast IDCT-II for Bounded Short Support Length

Let us first consider the case that an upper bound on the support length is known. More precisely, we suppose that $N = 2^J$ with $J \geq 2$ and $\mathbf{x} \in \mathbb{R}^N$ has a short support of unknown length m, but that $M \geq m$ is given a priori. Further, we assume that (6.3) holds for $\mathbf{x}$, i.e., that no relevant information about $\mathbf{x}$ is canceled out in the periodization process, and that we can access all entries of $\mathbf{x}^{\widehat{\mathrm{II}}} \in \mathbb{R}^N$. Utilizing Lemma 6.6, the algorithm begins by computing the initial vector

$$\mathbf{x}^{[L]} = \mathbf{C}_{2^L}^{\mathrm{III}} \left(\sqrt{2}^{J-L} \left(x^{\widehat{\mathrm{II}}}_{2^{J-L}k} \right)_{k=0}^{2^L-1} \right),$$

where $L \coloneqq \lceil \log_2 M \rceil + 1$. This can be done with the help of a fast DCT-III algorithm for vectors with full support, see, e.g., [PT05, Wan84], since the DCT-III is the same as the IDCT-II, as we saw in Theorem 4.2 (ii). By Lemma 6.9 the periodization $\mathbf{x}^{[L]}$ has a short support of length $m^{[L]} \leq M$. For $j \in \{L, \ldots, J-1\}$ we perform the following iteration steps.

1) If the support of $\mathbf{x}^{[j]}$ is contained in $I_{2^j-M,\,2^j-1}$, recover $\mathbf{x}^{[j+1]}$ using the procedure given in Theorem 6.13.

2) If the support of $\mathbf{x}^{[j]}$ is not contained in $I_{2^j-M,\,2^j-1}$, recover $\mathbf{x}^{[j+1]}$ using the procedure given in Theorem 6.17.

By Theorem 6.12, there is at most one index j' such that $S^{[j']} \subseteq I_{2^{j'}-M,\,2^{j'}-1}$, so we have to apply step 1 at most once. The complete procedure is summarized in Algorithm 9.

Remark 6.18 (Remark 4.1 in [BP18a]) For finding the first support index $\mu^{[L]}$ and the support length $m^{[L]}$ in line 2, as well as $\mu^{[j+1]}$ and $m^{[j+1]}$ in line 13 efficiently, we choose a threshold $\varepsilon > 0$ depending on the noise level of the data. In order to determine the support of $\mathbf{x}^{[L]}$, we define the set

$$T^{[L]} \coloneqq \left\{ k \in I_{0,\,2^L-1} : \left| x_k^{[L]} \right| > \varepsilon \right\} =: \{u_1, \ldots, u_P\}$$

of indices corresponding to the significantly large entries of $\mathbf{x}^{[L]}$. This set can be found in $\mathcal{O}\left(2^L\right) = \mathcal{O}\left(M\right)$ time. Then we set

$$\mu^{[L]} \coloneqq u_1 \quad \text{and} \quad m^{[L]} \coloneqq u_P - u_1 + 1.$$

In order to find the support of $\mathbf{x}^{[j+1]}$ in step 1, recall that then $j = j' \in \{L, \ldots, J-1\}$. Thus, it suffices by Theorem 6.12, case A to consider the set

$$T^{[j+1]} \coloneqq \left\{ k \in \left\{ 2^j - 2^{\widetilde{K}-1}, \ldots, 2^j + 2^{\widetilde{K}-1} - 1 \right\} : \left| x_k^{[j+1]} \right| > \varepsilon \right\} =: \{v_1, \ldots, v_Q\},$$

where $Q \leq \widetilde{m}^{[j]} \leq M$. Then $T^{[j+1]}$ can be found in $\mathcal{O}\left(2^{\widetilde{K}}\right) = \mathcal{O}\left(\widetilde{m}^{[j]}\right) = \mathcal{O}(M)$ time as well, and we define

$$\mu^{[j+1]} \coloneqq v_1 \quad \text{and} \quad m^{[j+1]} \coloneqq v_Q - v_1 + 1$$

in line 13. For step 2, i.e., for $j \in \{L, \ldots, J-1\} \setminus \{j'\}$, the first support index $\mu^{[j+1]}$ and the support length $m^{[j+1]}$ are computed in line 20 or line 23. ◊

Algorithm 9 Real Sparse Fast IDCT-II for Vectors with Bounded Short Support Length (Algorithm 1 in [BP18a])

Input: $\mathbf{x}^{\widehat{\Pi}}$, M, where the sought-after $\mathbf{x} \in \mathbb{R}^N$ with $N = 2^J$, $J \in \mathbb{N}$, has an unknown short support of length at most M and satisfies (6.3), and noise threshold $\varepsilon > 0$.

1: $L \leftarrow \lceil \log_2 M \rceil + 1$ and $\mathbf{x}^{[L]} \leftarrow \mathbf{DCT\text{-}III}\left[\sqrt{2}^{J-L}\left(x^{\widehat{\Pi}}_{2^{J-L}k}\right)_{k=0}^{2^L-1}\right]$

2: Find $\mu^{[L]}$ and $m^{[L]}$.

3: **for** j from L to $J-1$ **do**

4: **if** $\mu^{[j]} \geq 2^j - M$ **then**

5: $\widetilde{K} \leftarrow \left\lceil \log_2\left(2^j - \mu^{[j]}\right)\right\rceil + 1$

6: $\mathbf{z}^{[j]} \leftarrow \left(x^{[j]}_{2^j - 2^{\widetilde{K}-1}+k}\right)_{k=0}^{2^{\widetilde{K}-1}-1}$

7: $\mathbf{b}^0 \leftarrow \sqrt{2}^{J-j-1}\left(x^{\widehat{\Pi}}_{2^{J-j-1}\left(2^{j-\widetilde{K}+1}(2p+1)+1\right)}\right)_{p=0}^{2^{\widetilde{K}-1}-1}$

8: $\mathbf{b}^1 \leftarrow \sqrt{2}^{J-j-1}\left(x^{\widehat{\Pi}}_{2^{J-j-1}\left(2^{j-\widetilde{K}+1}(2p+1)-1\right)}\right)_{p=0}^{2^{\widetilde{K}-1}-1}$

9: $\mathbf{z}^{[j+1]}_{(0)} \leftarrow \frac{1}{2}\sqrt{2^{j-\widetilde{K}}}(-1)^{2^{j-\widetilde{K}}}\mathbf{J}_{2^{\widetilde{K}-1}}\operatorname{diag}(\tilde{\mathbf{c}})\mathbf{D}_{2^{\widetilde{K}-1}}\mathbf{DCT\text{-}IV}\left[\mathbf{J}_{2^{\widetilde{K}-1}}\left(\mathbf{b}^0 - \mathbf{b}^1\right)\right] + \frac{1}{2}\mathbf{z}^{[j]}$

10: $\left(\mathbf{z}^{[j+1]}_{(0)}\right)_k \leftarrow \begin{cases}\left(\mathbf{z}^{[j+1]}_{(0)}\right)_k & \text{if } \left|\left(\mathbf{z}^{[j+1]}_{(0)}\right)_k\right| > \varepsilon,\\ 0 & \text{else,}\end{cases} \quad k \in I_{0,\,2^{\widetilde{K}-1}-1}$

11: $\mathbf{z}^{[j+1]}_{(1)} \leftarrow \mathbf{J}_{2^{\widetilde{K}-1}}\left(\mathbf{z}^{[j]} - \mathbf{z}^{[j+1]}_{(0)}\right)$

12: $x^{[j+1]}_k \leftarrow \begin{cases}\left(\mathbf{z}^{[j+1]}_{(0)}\right)_{k-2^j+2^{\widetilde{K}-1}} & \text{if } k \in I_{2^j-2^{\widetilde{K}-1},\,2^j-1},\\ \left(\mathbf{z}^{[j+1]}_{(1)}\right)_{k-2^j} & \text{if } k \in I_{2^j,\,2^j+2^{\widetilde{K}-1}-1},\\ 0 & \text{else,}\end{cases} \quad k \in I_{0,\,2^{j+1}-1}$

13: Find $\mu^{[j+1]}$ and $m^{[j+1]}$.

14: **else**

15: $k_0 \leftarrow \underset{k \in I_{0,\,m^{[j]}-1}}{\operatorname{argmax}} \left\{\left|\sqrt{2}^{J-j-1}x^{\widehat{\Pi}}_{2^{J-j-1}(2k+1)}\right|\right\}$

16: $\alpha \leftarrow \sqrt{2}^{J-j-1}x^{\widehat{\Pi}}_{2^{J-j-1}(2k_0+1)}$.

17: $\left(u^0\right)^{\widehat{\Pi}}_{2k_0+1} \leftarrow \frac{1}{\sqrt{2^j}}\sum_{l=0}^{m^{[j]}-1}\cos\left(\frac{(2k_0+1)\left(2\left(\mu^{[j]}+l\right)+1\right)\pi}{2\cdot 2^{j+1}}\right)x^{[j]}_{\mu^{[j]}+l}$

18: $\nu_t \leftarrow \begin{cases}0 & \text{if } \left|\left(u^0\right)^{\widehat{\Pi}}_{2k_0+1} - \alpha\right| < \left|\left(u^0\right)^{\widehat{\Pi}}_{2k_0+1} + \alpha\right|,\\ 1 & \text{else}\end{cases}$

19: **if** $\nu_t = 0$ **then**

20: $\mu^{[j+1]} \leftarrow \mu^{[j]}$ and $m^{[j+1]} \leftarrow m^{[j]}$

21: $x^{[j+1]}_k \leftarrow \begin{cases}x^{[j]}_k & \text{if } k \in I_{\mu^{[j+1]},\,\mu^{[j+1]}+m^{[j+1]}-1},\\ 0 & \text{else,}\end{cases} \quad k \in I_{0,\,2^{j+1}-1}$

22: **else**

23: $\mu^{[j+1]} \leftarrow 2^{j+1} - m^{[j]} - \mu^{[j]}$ and $m^{[j+1]} \leftarrow m^{[j]}$

24: $x^{[j+1]}_k \leftarrow \begin{cases}x^{[j]}_{2^{j+1}-1-k} & \text{if } k \in I_{\mu^{[j+1]},\ldots,\mu^{[j+1]}+m^{[j+1]}-1},\\ 0 & \text{else,}\end{cases} \quad k \in I_{0,\,2^{j+1}-1}$

25: **end if**

26: **end if**

27: **end for**

Output: $\mathbf{x} = \mathbf{x}^{[J]}$.

6.4.2 Runtime and Sampling Bounds

Having presented our new algorithm we now prove that its runtime and sampling complexity are sublinear in the vector length N and subquadratic in the bound M on the support length. In Section 6.5.2 we will investigate the performance of Algorithm 9 with respect to runtime and noisy input data in some numerical examples, also comparing it to other sparse IDCT-II methods, including Algorithm 8.

Theorem 6.19 (Theorem 4.2 in [BP18a]) *Let $N = 2^J$ with $J \in \mathbb{N}$ and $\mathbf{x} \in \mathbb{R}^N$ have a short support of length $m < N$. Assume that $\mathbf{x}$ satisfies (6.3). Further suppose that only an upper bound $M \geq m$ is known. Then Algorithm 9 has a runtime of $\mathcal{O}\left(M \log M + m \log_2 \frac{N}{M}\right)$ and uses $\mathcal{O}\left(M + m \log_2 \frac{N}{M}\right)$ samples of $\mathbf{x}^{\widehat{\Pi}}$.*

Proof. (i) Computing the initial vector $\mathbf{x}^{[L]}$ in line 1 via a 2^L-length DCT-III has a runtime of $\mathcal{O}\left(2^L \log 2^L\right)$, as mentioned in Remark 4.8. It follows from Remark 6.18 that finding $\mu^{[L]}$ and $m^{[L]}$ in line 2 needs $\mathcal{O}\left(2^L\right)$ operations.

If $j = j'$ with j' as defined in (6.4), we have to apply the recovery step 1. The computation of $\mathbf{z}_{(0)}^{[j+1]}$ in lines 9 and 10 requires a DCT-IV of length $2^{\widetilde{K}-1}$ and further operations of complexity $\mathcal{O}\left(2^{\widetilde{K}-1}\right)$, since $\mathbf{D}_{2^{\widetilde{K}-1}}$ and $\operatorname{diag}(\tilde{\mathbf{c}})$ are diagonal matrices and $\mathbf{J}_{2^{\widetilde{K}-1}}$ is just a permutation. Computing $\mathbf{z}_{(1)}^{[j+1]}$ and $\mathbf{x}^{[j+1]}$ in lines 11 and 12 and finding $\mu^{[j+1]}$ and $m^{[j+1]}$ in line 13 also needs $\mathcal{O}\left(2^{\widetilde{K}-1}\right)$ operations. Note that we can only estimate that $\widetilde{m}^{[j]} = \mathcal{O}(M)$ and thus $2^{\widetilde{K}-1} = \mathcal{O}(M)$, since m is not known apriori and the support of $\mathbf{x}^{[j]}$ can be located anywhere in the interval $I_{2^j-M,\,2^j-1}$. Consequently, lines 6 to 13 have a runtime of $\mathcal{O}\left(2^{\widetilde{K}-1} \log 2^{\widetilde{K}-1}\right) = \mathcal{O}(M \log M)$, which was shown in Section 4.2.

For $j \in \{L, \ldots, J-1\} \setminus \{j'\}$ the support of $\mathbf{x}^{[j]}$ is not contained in the interval $I_{2^j-M,\,2^j-1}$, so we have to apply the recovery step 2. Finding a nonzero entry of $\left(\mathbf{x}^{[j+1]}\right)^{\widehat{\Pi}}$ in lines 15 and 16 requires $\mathcal{O}\left(m^{[j]}\right) = \mathcal{O}(m)$ operations by Lemma 6.15. The execution of lines 17 to 24 has a runtime of $\mathcal{O}\left(m^{[j]}\right)$ as well, since $\mathbf{u}^0$ has a known short support of length $m^{[j]}$ and $\mathbf{x}^{[j+1]}$ has a short support of length $m^{[j+1]} = m^{[j]}$.

Hence, Algorithm 9 has an overall runtime of

$$\begin{aligned}\mathcal{O}\left(2^{\widetilde{K}} \log 2^{\widetilde{K}} + \sum_{\substack{j=L \\ j \neq j'}}^{J-1} m^{[j]}\right) &= \mathcal{O}\left(M \log M + (J-L)m\right) \\ &= \mathcal{O}\left(M \log M + m \log_2 \frac{N}{M}\right).\end{aligned}$$

(ii) The initial vector $\mathbf{x}^{[L]}$ can be computed from 2^L samples of $\mathbf{x}^{\widehat{\Pi}}$ in line 1. If $j = j'$, we need to use $2^{\widetilde{K}} \leq 2^L$ samples in lines 7 and 8. Further, finding an oddly indexed nonzero entry of $\left(\mathbf{x}^{[j+1]}\right)^{\widehat{\Pi}}$ in lines 15 and 16 requires at most $m^{[j]}$ samples of $\mathbf{x}^{\widehat{\Pi}}$ by Lemma 6.15,

which yields a total sampling complexity of

$$\mathcal{O}\left(2^L + 2^{\widetilde{K}} + \sum_{\substack{j=L \\ j\neq j'}}^{J-1} m^{[j]}\right) = \mathcal{O}\left(M + (J-L)m\right)$$
$$= \mathcal{O}\left(M + m\log_2\frac{N}{M}\right).$$

□

Remark 6.20 If the upper bound M on the support length of $\mathbf{x}$ or the support length m, and thus M, approach N, Algorithm 9 has a runtime of $\mathcal{O}(N\log N)$, which is the same order as the runtime of a full length fast IDCT-II. Further, then it requires $\mathcal{O}(N)$ samples of $\mathbf{x}^{\widehat{\Pi}}$, which is the same as the sampling complexity of a full length fast IDCT-II. ◊

6.4.3 Sparse Fast IDCT-II for Exactly Known Short Support Length

Having introduced our new sparse IDCT-II for vectors with bounded short support length, we can now modify Algorithm 9 to better fit the case where the support length m of $\mathbf{x}$ is known exactly, i.e., if $M = m$. Since there is at most one index j' for which the support of $\mathbf{x}^{[j']}$ is contained in the last m entries, the procedure from Theorem 6.13 only has to be applied if $m^{[j']} < m^{[j'+1]}$, i.e., if there was a collision of nonzero entries, or if $\nu^{[j']} = 2^{j'} - 1$, unlike in Algorithm 9.

We can simply replace M by m in Algorithm 9 to obtain the sparse IDCT-II for vectors with exactly known short support length. Then we have that $L \coloneqq \lceil \log_2 m \rceil + 1$ and $\widetilde{m}^{[j']} = 2^{j'} - \mu^{[j']} = m^{[j']} = \mathcal{O}(m)$ and $\widetilde{K} = L$. Note that $m^{[j+1]} = m$ for $j \geq j'$. For this simplification we find the following runtime and sampling complexities.

Corollary 6.21 (Theorem 4.3 in [BP18a]) Let $N = 2^J$ with $J \in \mathbb{N}$ and $\mathbf{x} \in \mathbb{R}^N$ have a short support of length $m < N$. Assume that $\mathbf{x}$ satisfies (6.3). Further suppose that m is known exactly. Then Algorithm 9 has a runtime of $\mathcal{O}\left(m\log m + m\log_2\frac{N}{m}\right)$ and uses $\mathcal{O}\left(m + m\log_2\frac{N}{m}\right)$ samples of $\mathbf{x}^{\widehat{\Pi}}$.

6.5 Numerical Results for the Algorithms from Chapters 5 and 6

In the following section we will test the sparse IDFT and IDCT-II algorithms developed in Chapters 5 and 6 numerically with respect to their runtime and stability for noisy input data. First, we will evaluate Algorithm 7, the sparse IDFT algorithm for vectors with reflected block support, and afterwards we will focus on the two new IDCT-II algorithms for vectors with one-block or short support, Algorithms 8 and 9.

6.5.1 Numerical Results for Algorithm 7

Let us first present some numerical experiments regarding the runtime of Algorithm 7 and its robustness to noise. As our algorithm is deterministic and designed for vectors with reflected block support, we only compare it to Algorithm 2.3 in [PWCW18] and to MATLAB 2016b's `ifft` routine, which is a fast and highly optimized implementation of the fast inverse Fourier transform based on the FFTW library, see [FJ17, The18d, The18b].

The sparse IDFT algorithm in [PWCW18] is suited for the fast reconstruction of an arbitrary m-sparse vector $\mathbf{y}$ from its DFT $\widehat{\mathbf{y}}$ for small sparsities m and is the only general sparse deterministic IDFT for vectors we are aware of.

Similarly as Algorithm 2 in [PW16a] and Algorithm 2.1 in [PW17a], Algorithm 2.3 in [PWCW18] recovers $\mathbf{y}$ iteratively from $\widehat{\mathbf{y}}$ and its periodizations $\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(J)}$. It finds the sparsity m of the vector $\mathbf{y}$ adaptively without a priori knowledge of m. The main idea is to restrict the periodizations $\mathbf{y}^{(j)}$ and $\mathbf{y}^{(j+1)}_{(0)}$ to vectors of length $m^{(j)}$, where $m^{(j)}$ denotes the sparsity of $\mathbf{y}^{(j)}$. Then it was shown that (5.3) can be restricted to $m^{(j)}$ equations such that all occurring matrices are well-conditioned, which requires careful considerations of the chosen equations. Employing additional stabilizing techniques, Algorithm 2.3 in [PWCW18] achieves a runtime of $\mathcal{O}(M^2 \log N)$ if $M^2 < N$.

Deterministic sparse FFT methods like Algorithm 4, Algorithm 6 or the ones introduced in [Iwe10, Iwe13, CLW16] are designed for 2π-periodic functions and rely heavily on the fact that the function in question can be evaluated anywhere. They require very complex sampling schemes, unlike the simple $\mathcal{O}\left(m \log \frac{2N}{m}\right)$ entries of $\widehat{\mathbf{y}} \in \mathbb{C}^{2N}$ necessary for Algorithm 7, and are thus not suited for a comparison. Methods like Algorithms 2 and 3 additionally have sparsity constraints that are not satisfied by reflected block sparsity. Other deterministic sparse IDFT algorithms for vectors like Algorithm 2 in [PW16a] and Algorithm 2.1 in [PW17a] also require $\mathbf{y}$ to have a one-block support and hence cannot be compared to a method that is designed for vectors with reflected block support.

Algorithm 7 and Algorithm 2.3 in [PWCW18] have been implemented in MATLAB 2016b; the codes are freely available in [BP18e, PW17b]. We chose the implementation of Algorithm 2.3 in [PWCW18] which uses Algorithm 4.5 in [PWCW18] to find $m^{(j)}$ equations that yield well-conditioned matrices. Note that neither of the algorithms requires a priori knowledge of the length of the support blocks or the sparsity, but that both Algorithm 7 and Algorithm 2.3 in [PWCW18] require that $\mathbf{y}$ satisfies (5.4), i.e., that

$$y^{(j)}_{k \bmod 2^j} \neq 0 \qquad \forall\, j \in \{0, \ldots, J\}$$

for all $k \in \{0, \ldots, 2N-1\}$ with $y_k \neq 0$ for exact data and

$$\left| y^{(j)}_{k \bmod 2^j} \right| > \varepsilon \qquad \forall\, j \in \{0, \ldots, J\}$$

for all $k \in \{0, \ldots, 2N-1\}$ with $|y_k| > \varepsilon$ for noisy data, where $\varepsilon > 0$ is a threshold depending on the noise level.

Figure 6.9 shows the average runtimes of Algorithm 7 with noise threshold $\varepsilon = 10^{-4}$, Algorithm 2.3 in [PWCW18] and MATLAB's `ifft` applied to $\widehat{\mathbf{y}}$ for 100 randomly generated vectors $\mathbf{y}$ of length $2N = 2^{21}$ with reflected block support of lengths varying between 5 and 50,000. The nonzero entries of the vectors are chosen between 0 and 10. For each vector at most $\lfloor (m-2)/2 \rfloor$ entries in the first support block, excluding the first and last one, are randomly set to 0, and the second half of $\mathbf{y}$ is determined by its symmetry $\mathbf{y} = \mathbf{J}_{2N}\mathbf{y}$.

Applying Algorithm 2.3 in [PWCW18] to $2m$-sparse vectors is very unstable for increasing sparsities m, as it often has to solve a close to singular equation system. This is due to the fact that the algorithm is optimized for general sparse vectors and does not utilize the special structure of $\mathbf{y}$ at all. Hence, we decided to only measure its runtime for block lengths up to $m = 30$. Obviously, any comparison to the highly optimized `ifft` must be flawed; however, we can discern that Algorithm 7 and Algorithm 2.3 in [PWCW18] are both much faster than `ifft` for sufficiently small block lengths. The former algorithm

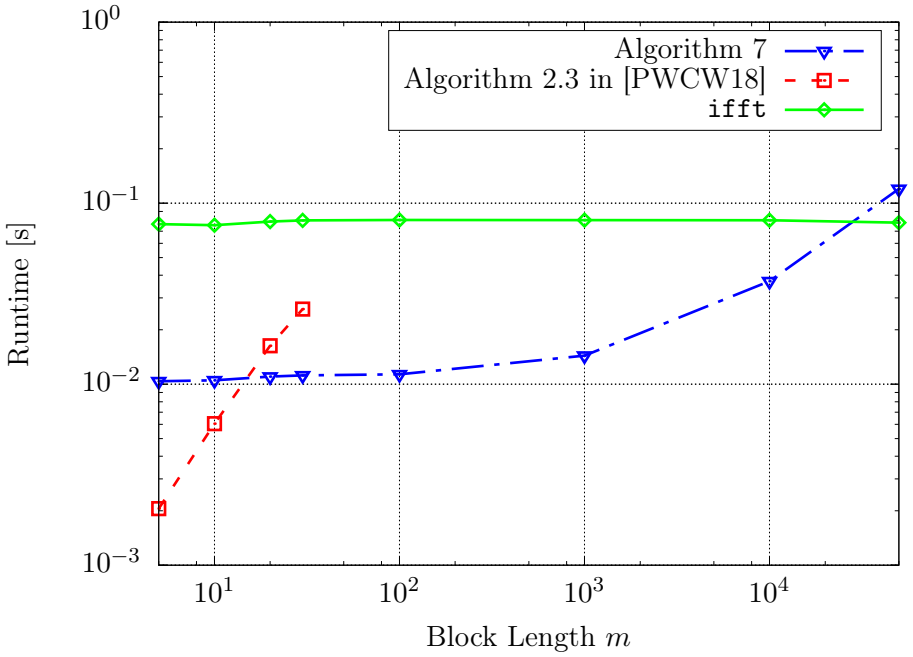

Figure 6.9: Average runtimes of Algorithm 7 with threshold $\varepsilon = 10^{-4}$, Algorithm 2.3 in [PWCW18] and MATLAB's `ifft` for 100 random input vectors with reflected block support of block length m and vector length $2N = 2^{21}$

achieves faster runtimes for block lengths up to $m = 10{,}000$, while the latter does so at least for block lengths up to $m = 30$. Note that by setting at most $\lfloor (m-2)/2 \rfloor \cdot 2$ entries inside the support blocks randomly to 0, the actual sparsity of $\mathbf{y}$ can be almost as low as m. This barely affects the runtime of Algorithm 7, but it decreases the average runtime of Algorithm 2.3 in [PWCW18]. As can be seen from Table 6.1 presenting the average reconstruction errors for exact data, Algorithm 2.3 in [PWCW18] is not accurate for block lengths of $m = 20$ or greater, and, as we found out during the experiments, not even consistently accurate for block lengths up to $m = 10$.

m	Algorithm 7	Algorithm 2.3 in [PWCW18]	`ifft`
5	$4.2 \cdot 10^{-20}$	$3.4 \cdot 10^{-8}$	$3.8 \cdot 10^{-21}$
10	$8.0 \cdot 10^{-20}$	$1.4 \cdot 10^{0}$	$4.8 \cdot 10^{-21}$
20	$2.2 \cdot 10^{-19}$	$3.1 \cdot 10^{7}$	$7.0 \cdot 10^{-21}$
30	$6.6 \cdot 10^{-19}$	$3.9 \cdot 10^{8}$	$8.3 \cdot 10^{-21}$
100	$1.5 \cdot 10^{-18}$	–	$1.5 \cdot 10^{-20}$
1,000	$7.7 \cdot 10^{-14}$	–	$4.7 \cdot 10^{-20}$
10,000	$3.6 \cdot 10^{-12}$	–	$1.5 \cdot 10^{-19}$
50,000	$1.3 \cdot 10^{-11}$	–	$3.5 \cdot 10^{-19}$

Table 6.1: Reconstruction errors for the three IDFT algorithms for exact data

Still, for block lengths up to $m = 10$, this method is much faster than Algorithm 7. For block lengths up to $m = 100$, Algorithm 7 achieves an accuracy close to that of `ifft`, and even for $m = 50{,}000$ its reconstruction error is small.

Next we examine the robustness of the algorithms with respect to noisy data. Since

Algorithm 2.3 in [PWCW18] is not suitable for noisy data due to ill-conditioned equation systems having to be solved, we will only consider Algorithm 7 and MATLAB's `ifft` hereafter. Disturbed Fourier data $\widehat{\mathbf{z}} \in \mathbb{C}^{2N}$ is created by adding uniform noise $\boldsymbol{\eta} \in \mathbb{C}^{2N}$ to the given data $\widehat{\mathbf{y}}$,

$$\widehat{\mathbf{z}} := \widehat{\mathbf{y}} + \boldsymbol{\eta}.$$

We measure the noise with the *signal-to-noise ratio (SNR)*,

$$\mathrm{SNR} := 20 \cdot \log_{10} \frac{\|\widehat{\mathbf{y}}\|_2}{\|\boldsymbol{\eta}\|_2}.$$

Figures 6.10 and 6.11 depict the average reconstruction errors $\|\mathbf{y} - \mathbf{y}'\|_2/(2N)$ for block lengths $m = 100$ and $m = 1{,}000$, where $\mathbf{y}$ denotes the original vector and $\mathbf{y}'$ the reconstruction by the corresponding algorithm applied to $\widehat{\mathbf{z}}$. Note that for noisy data the resulting vector $\mathbf{y}'$ does no longer have an exact reflected block support, but the support blocks have entries that are significantly greater than the noise and can thus be found by the stable support detection procedures presented in Section 5.6.1. The threshold ε

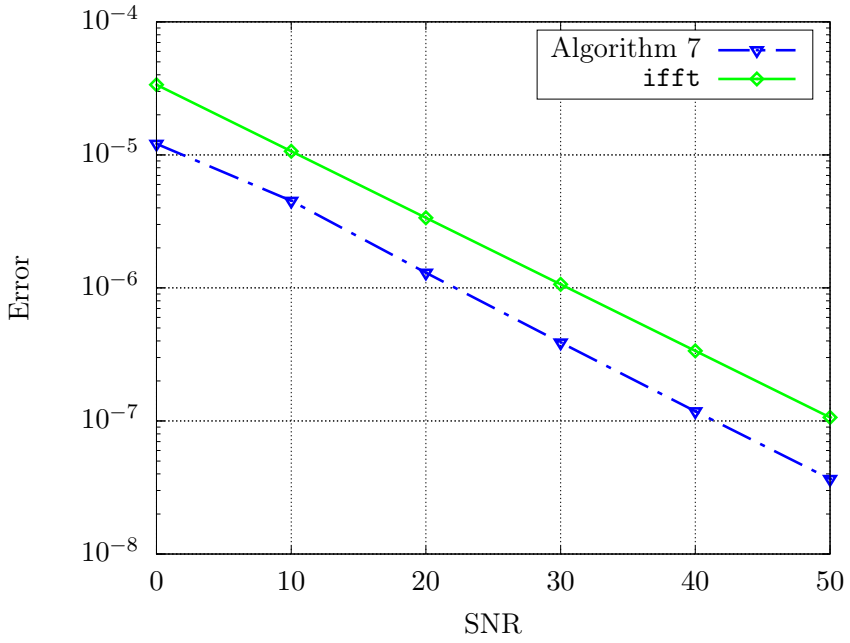

Figure 6.10: Average reconstruction errors $\|\mathbf{y} - \mathbf{y}'\|_2/(2N)$ of Algorithm 7 and `ifft` for 100 random input vectors with reflected block support of block length $m = 100$ and vector length $2N = 2^{21}$

for Algorithm 7 is chosen according to Table 6.2.

SNR	0	10	20	30	40	50
ε	1.70	1.20	0.40	0.19	0.05	0.02

Table 6.2: Threshold ε for Algorithm 7

We cannot recommend general heuristics on how to choose ε for other values of N and m so far, though, as ε depends non-trivially on the entries of $\widehat{\mathbf{y}}$ and the noise

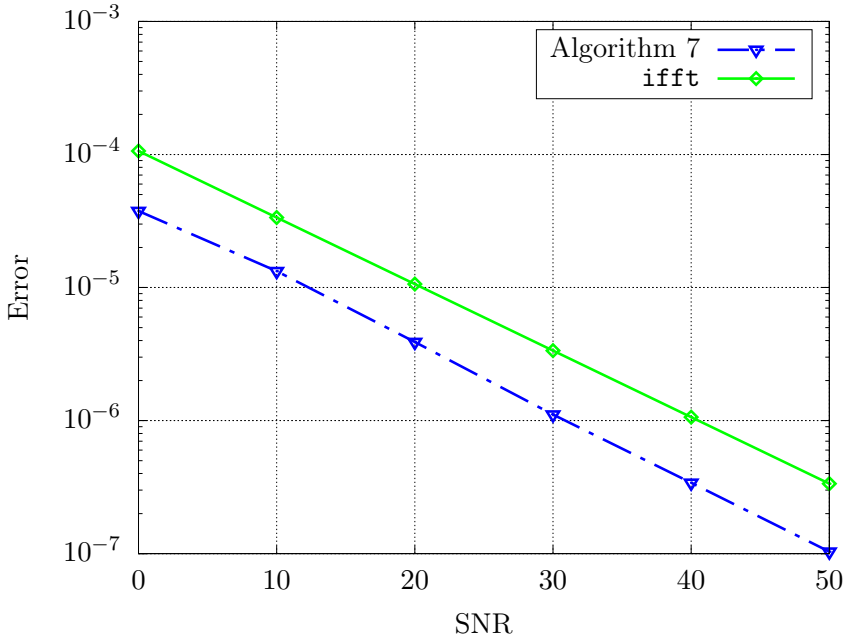

Figure 6.11: Average reconstruction errors $\|\mathbf{y} - \mathbf{y}'\|_2/(2N)$ of Algorithm 7 and `ifft` for 100 random input vectors with reflected block support of block length $m =$ 1,000 and vector length $2N = 2^{21}$

$\boldsymbol{\eta}$, which is added to the given data $\widehat{\mathbf{y}}$. However, ε is essentially only used to detect the support blocks of the periodizations $\mathbf{y}^{(j)}$. Thus, good values for it can be found by testing different parameters ε and comparing the found support indices of $\mathbf{y}'$ and the corresponding entries. If the SNR of the input data can be estimated, ε can be optimized using synthetic data. Too small values for ε result in smaller reconstruction errors, but overestimated support lengths; too large values result in higher reconstruction errors and higher rates of correct recovery. Depending on the application, different choices of ε are viable.

The values in Table 6.2 were found via an attempt to minimize the approximation error and maximize the rate of correct recovery for the given setting. Both for $m = 100$ and $m =$ 1,000 we see that the reconstruction by Algorithm 7 yields a smaller error than the one by `ifft` for all considered noise levels.

Since for vectors with reflected block support the structure is especially important, we also examine whether Algorithm 7 can correctly identify the support blocks of $\mathbf{y}$ for noisy input data. Especially for high noise levels Algorithm 7 tends to overestimate the true length of the support blocks. Table 6.3 shows the rates of correct recovery of the support.

In the second and fourth column we present the rate of correct recovery, where we consider $\mathbf{y}$ to be correctly recovered by $\mathbf{y}'$ if the support of the two blocks of the original vector $\mathbf{y}$ is contained in the support blocks found by the algorithm. In the third and fifth column we additionally require that the block length m' found by Algorithm 7 satisfies $m' \leq 3m$ in order to illustrate whether the block lengths are significantly overestimated or not.

For SNR values of 20 or more our IDFT method has a very high rate of correct recovery in the sense that the original support is contained in the reconstructed one. For these noise levels the block length of $\mathbf{y}'$ is also almost always at most $3m$.

SNR	Rate of Correct Recovery in % Using Algorithm 7 for			
	$m = 100$	$m = 100$ $m' \leq 3m = 300$	$m = 1{,}000$	$m = 1{,}000$ $m' \leq 3m = 3{,}000$
0	70	49	69	47
10	70	70	74	68
20	86	83	93	85
30	98	98	94	93
40	99	98	97	93
50	100	100	99	98

Table 6.3: Rate of correct recovery of the support of $\mathbf{y}$ in percent for Algorithm 7, without bounding m' and with $m' \leq 3m$, for the 100 random input vectors with block length $m = 100$ and $m = 1{,}000$ from Figures 6.10 and 6.11

6.5.2 Numerical Results for Algorithms 8 and 9

In the following section we evaluate the performances of Algorithm 8 and Algorithm 9, in the variant for exactly known support lengths and for bounded short support lengths, with respect to runtime and robustness to noise. To the best of our knowledge most existing sparse IDCT-II algorithms use an approach of computing $\mathbf{x}$ by recovering the vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T$ from $\widehat{\mathbf{y}}$ by an unstructured and thus inefficient $2m$-sparse IDFT. Solely Algorithm 8 utilizes an IDFT approach especially tailored to the structure of $\mathbf{y}$, so we only compare our algorithms to MATLAB 2018a's `idct` routine, which is part of the *Signal Processing Toolbox*, see [The18c]. `idct` is a fast and highly optimized implementation of the fast inverse cosine transform of type II. Note that, compared to the implementation of `idct` in MATLAB 2016b, which we used for the numerical experiments in [BP18a] and in Section 6.5.1, the runtime of `idct` in MATLAB 2018a has reduced by almost half for arbitrary nonnegative vectors of length $N = 2^{20}$ on the machine used for the experiments, which is why the results of the numerical experiments with respect to runtime in this section are different from the ones in [BP18a], Section 6.2. All algorithms have been implemented in MATLAB 2018a, and the code is freely available in [BP18b, BP18d]. Recall that Algorithm 8 does not require any a priori knowledge of the support length, but needs that for $\mathbf{x} \in \mathbb{R}^{2^J}$ the vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T \in \mathbb{R}^{2^{J+1}}$ satisfies

$$\left| \sum_{l=0}^{2^{J+1-j}-1} y_{k+2^j l} \right| > \varepsilon \qquad \forall\, j \in \{0, \ldots, J+1\}$$

for all $|y_k| > \varepsilon$ for a noise threshold $\varepsilon > 0$. Algorithm 9, on the other hand, requires an upper bound $M \geq m$ on the support length and that $\mathbf{x} \in \mathbb{R}^{2^J}$ satisfies

$$\left| x_{\mu[J]} \right| > \varepsilon, \qquad |x_{\nu[J]}| > \varepsilon \qquad \text{and} \qquad \left| x_{\mu[J]} + x_{\nu[J]} \right| > \varepsilon \quad \text{if } m \text{ is even.}$$

Figure 6.12 shows the average runtimes of Algorithm 8, Algorithm 9 for exactly known support lengths, i.e., for $M = m$, and for bounded short support lengths with $M = 3m$, and `idct` applied to $\mathbf{x}^{\widehat{\Pi}}$ for 1,000 randomly generated 2^{20}-length vectors $\mathbf{x}$ with short support of lengths varying between 10 and 500,000. For Algorithms 8 and 9 we use the noise threshold $\varepsilon = 10^{-4}$. The nonzero entries of the vectors are chosen randomly with

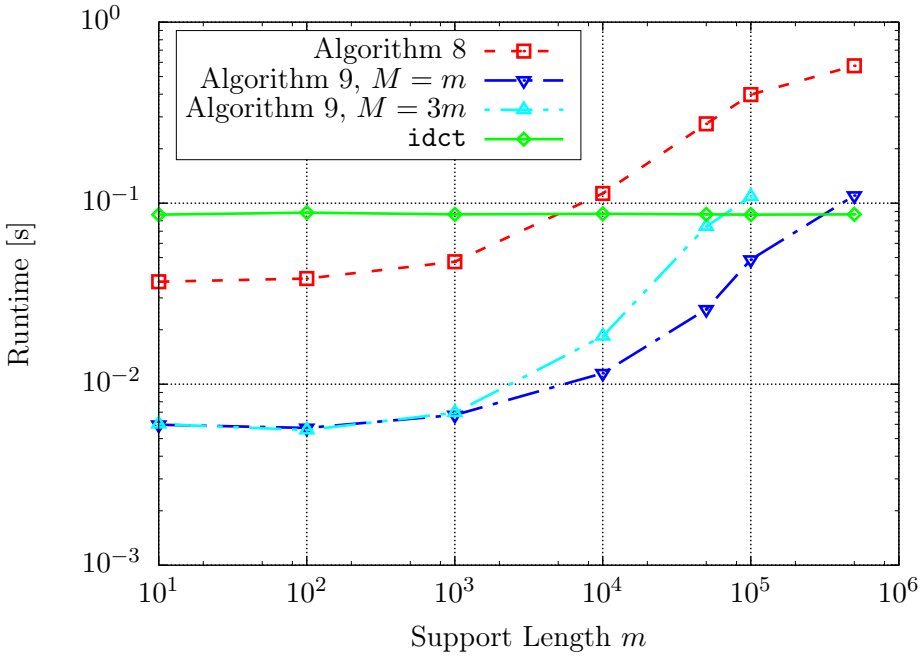

Figure 6.12: Average runtimes of Algorithm 8 and Algorithm 9 for exactly known short support and for bounded short support with $\varepsilon = 10^{-4}$, and MATLAB's `idct` for 1,000 random input vectors with short support of length m, bound $M = 3m$ and vector length $N = 2^{20}$

uniform distribution between 0 and 10, with $x_{\mu^{[J]}}$ and $x_{\nu^{[J]}}$ chosen from $(\varepsilon, 10]$. For each vector at most $\lfloor (m-2)/2 \rfloor$ entries in the support block, excluding the first and last one, are randomly set to 0. Hence, both (5.4) and (6.3) hold. Since for $m = 500{,}000$ we have that $M = 3m > N$, we only execute Algorithm 9 in the variant for bounded support lengths up to $m = 100{,}000$.

Of course the comparison of the sparse IDCT-II algorithms to the highly optimized, support length independent `idct` routine must be flawed; however, one can see that all three sparse IDCT-II procedures are much faster than `idct` for sufficiently small support lengths. Algorithm 8 is faster than `idct` for block lengths up to $m = 1{,}000$. For exactly known support lengths Algorithm 9 achieves smaller runtimes for block lengths up to $m = 100{,}000$, and for bounded support lengths this is the case for block lengths up to $m = 50{,}000$, where the known bound on the block length is $M = 150{,}000$. Both variants of Algorithm 9 are about an order of magnitude faster than Algorithm 8 for block lengths up to $m = 10^4$, and, if m is known exactly, even for all considered block lengths. If $M = 3m$, Algorithm 9 allows for an effective sparsity of three times the actual size in the method for case A of Theorem 6.12, which results in a runtime increasing faster than for the variant where m is known exactly.

Note that by setting $\lfloor (m-2)/2 \rfloor$ entries inside the support to zero, the actual sparsity of $\mathbf{x}$ can be almost as low as $m/2$; however, this barely affects the runtime of any of the considered algorithms, if at all. It follows from Table 6.4, presenting the average reconstruction errors for exact data for all four considered methods, that, while the sparse IDCT-II algorithms only achieve reconstruction errors comparable to those of `idct` if $m \leq 100$, their outputs are still very accurate. Note that, with $\mathbf{x} \in \mathbb{R}^{2^{20}}$, for

m	Algorithm 8	Algorithm 9, $M = m$	Algorithm 9, $M = 3m$	`idct`
10	$1.3 \cdot 10^{-19}$	$1.8 \cdot 10^{-20}$	$1.7 \cdot 10^{-20}$	$7.8 \cdot 10^{-21}$
100	$4.9 \cdot 10^{-18}$	$5.3 \cdot 10^{-20}$	$3.9 \cdot 10^{-20}$	$2.4 \cdot 10^{-20}$
1,000	$4.9 \cdot 10^{-13}$	$7.5 \cdot 10^{-14}$	$4.1 \cdot 10^{-14}$	$7.6 \cdot 10^{-20}$
10,000	$3.9 \cdot 10^{-12}$	$1.0 \cdot 10^{-12}$	$1.4 \cdot 10^{-12}$	$2.4 \cdot 10^{-19}$
50,000	$1.5 \cdot 10^{-11}$	$3.6 \cdot 10^{-12}$	$2.9 \cdot 10^{-12}$	$5.4 \cdot 10^{-19}$
100,000	$2.9 \cdot 10^{-11}$	$7.5 \cdot 10^{-12}$	$7.6 \cdot 10^{-19}$	$7.6 \cdot 10^{-19}$
500,000	$9.6 \cdot 10^{-11}$	$1.7 \cdot 10^{-18}$	–	$1.7 \cdot 10^{-18}$

Table 6.4: Reconstruction errors for the four IDCT-II algorithms for exact data

$M = m = 500{,}000$, we obtain that

$$L = \lceil \log_2 500{,}000 \rceil + 1 = 20,$$

so in line 1, Algorithm 9 computes $\mathbf{x}^{[20]} = \mathbf{x}$ directly via a full length DCT-III, thus resulting in a much lower reconstruction error than for $M = m = 100{,}000$. Similarly, for $M = 3m = 300{,}000$, we also have that $L = 20$, which explains the small reconstruction error for Algorithm 9 in the bounded support case for $m = 100{,}000$.

We also investigate the robustness of Algorithms 8 and 9 for noisy data. Disturbed cosine data $\mathbf{z}^{\widehat{\mathrm{II}}} \in \mathbb{R}^N$ is created by adding uniform noise $\boldsymbol{\eta} \in \mathbb{R}^N$ to the given data $\mathbf{x}^{\widehat{\mathrm{II}}}$,

$$\mathbf{z}^{\widehat{\mathrm{II}}} \coloneqq \mathbf{x}^{\widehat{\mathrm{II}}} + \boldsymbol{\eta}.$$

As in Section 6.5.1, we measure the noise with the signal-to-noise ratio (SNR), given by

$$\mathrm{SNR} \coloneqq 20 \cdot \log_{10} \frac{\left\| \mathbf{x}^{\widehat{\mathrm{II}}} \right\|_2}{\|\boldsymbol{\eta}\|_2}.$$

Figures 6.13 and 6.14 depict the average reconstruction errors $\|\mathbf{x} - \mathbf{x}'\|_2 / N$, where $\mathbf{x}$ denotes the original vector and $\mathbf{x}'$ the reconstruction by the corresponding algorithm applied to $\mathbf{z}^{\widehat{\mathrm{II}}}$ for support lengths $m = 100$ and $m = 1{,}000$.

The threshold parameters ε for Algorithm 8 and both variants of Algorithm 9 are chosen according to Table 6.5. All parameters were obtained in an attempt to minimize the reconstruction error and maximize the rate of correct recovery. Similarly to the choice of ε for Algorithm 7, we cannot recommend good general heuristics for the choice of ε for Algorithms 8 and 9. Usually, Algorithm 8 requires slightly higher noise thresholds than Algorithm 7, which might be caused by the numerical errors induced by the computation of the samples of $\widehat{\mathbf{y}}$ from $\mathbf{x}^{\widehat{\mathrm{II}}}$. For Algorithm 9 the threshold ε is also basically only used to detect the first support indices of the reflected periodizations. Again, all we can suggest is to test different values for ε on synthetic data.

For $m = 100$ and SNR values greater than 10, Algorithm 9 with $M = 3m$ has the smallest reconstruction error, with Algorithm 8 achieving very similar errors. For exactly known support lengths, the reconstruction error of Algorithm 9 is slightly larger and comparable to the one of `idct`, albeit being smaller than the latter. Similar behavior can be observed for $m = 1{,}000$ for these two methods. Algorithms 8 and 9 have comparable

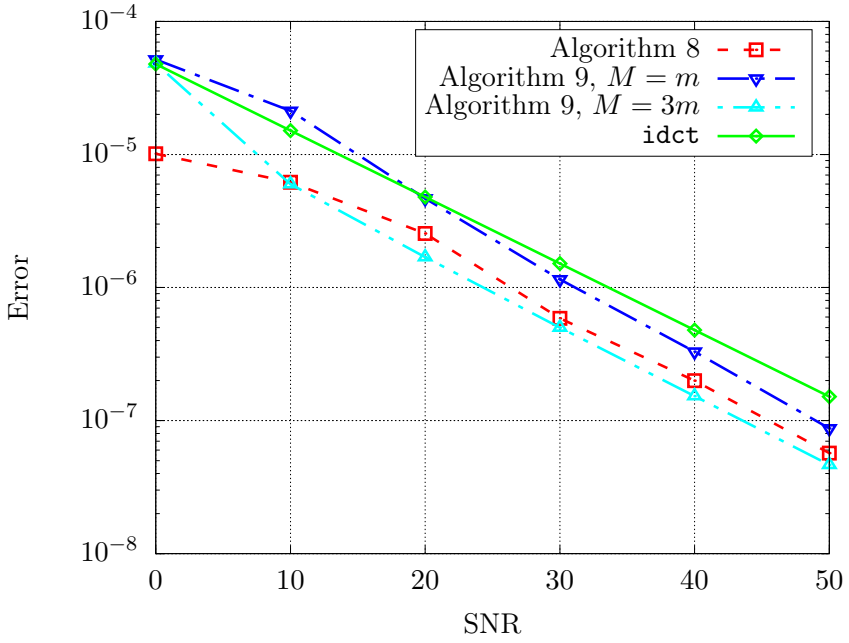

Figure 6.13: Average reconstruction errors $\|\mathbf{x} - \mathbf{x}'\|_2/N$ of Algorithm 8, Algorithm 9 for $M = m$ and $M = 3m$ and `idct` for 1,000 random input vectors with support length m and vector length $N = 2^{20}$

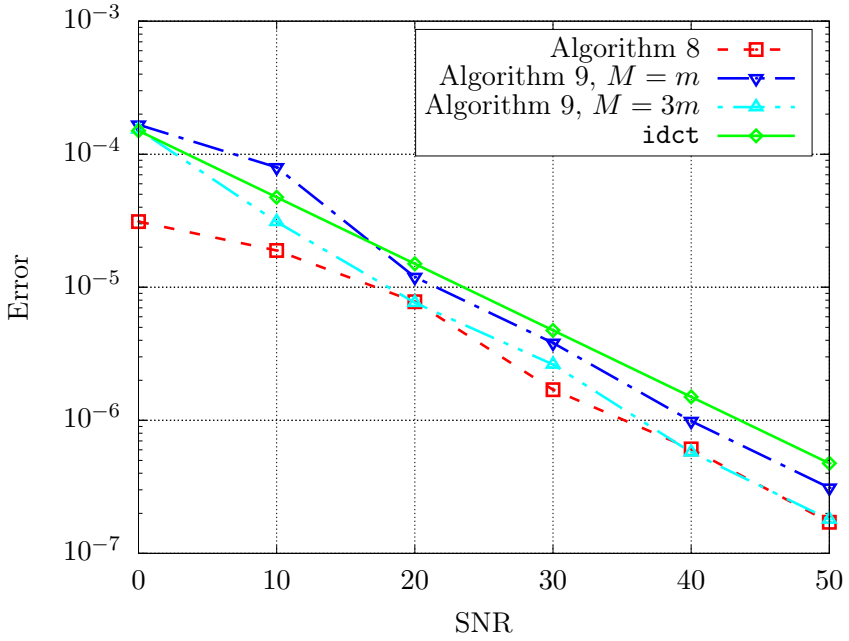

Figure 6.14: Average reconstruction errors $\|\mathbf{x} - \mathbf{x}'\|_2/N$ of Algorithm 8, Algorithm 9 for $M = m$ and $M = 3m$ and `idct` for 1,000 random input vectors with support length m and vector length $N = 2^{20}$

reconstruction errors for $m = 1{,}000$ with no method performing better than the other for all SNR values.

In certain applications the support of $\mathbf{x}$ might be of importance as an output; hence, we

SNR	Alg. 8	Alg. 9, $m = 100$	Alg. 9, $m = 1{,}000$
0	2.50	2.50	2.50
10	1.80	2.00	2.10
20	1.00	1.00	1.50
30	0.50	0.40	0.85
40	0.15	0.15	0.20
50	0.05	0.05	0.10

Table 6.5: Threshold ε for Algorithm 8 and Algorithm 9

also examine whether the sparse IDCT-II algorithms can correctly identify the support for noisy input data. Tables 6.6 and 6.7 show the rates of correct recovery of the support for $m = 100$ and $m = 1{,}000$. As Algorithms 8 and 9 tend to overestimate the support

	Rate of Correct Recovery in % for $m = 100$				
SNR	Alg. 8	Alg. 8, $m' \leq 3m$	Alg. 9, $M = m$	Alg. 9, $M = 3m$	Alg. 9, $M = 3m$, $m' \leq 3m$
0	83.1	77.1	61.6	89.9	0.0
10	97.6	97.4	64.0	98.7	85.4
20	100.0	100.0	95.1	100.0	96.2
30	100.0	100.0	99.3	100.0	98.6
40	100.0	100.0	99.9	100.0	99.4
50	100.0	100.0	100.0	100.0	99.9

Table 6.6: Rate of correct recovery of the support of $\mathbf{x}$ in % for Algorithm 8 and Algorithm 9 for $M = m$ and $M = 3m$, without bounding m' and with $m' \leq 3m$, for 1,000 random input vectors with support length $m = 100$ from Figure 6.13

for noisy data, we consider $\mathbf{x}$ to be correctly recovered by $\mathbf{x}'$ in the second, fourth and fifth column if the support of $\mathbf{x}$ is contained in the support interval returned by the sparse IDCT-II algorithms. In the third and sixth column we additionally require that the support length m' obtained by the procedures satisfies $m' \leq 3m$. Note that if m is known exactly, Algorithm 9 will not overestimate the support length m.

For SNR values of 20 and greater all sparse IDCT-II algorithms have very high rates of correct recovery. Algorithm 9 for bounded short support overestimates the support length by more than a factor three in less than 4% of the cases for SNR values of 20 or more for $m = 100$ and in less than 6 % of the cases for SNR values of 40 or more for $m = 1{,}000$. Algorithm 8 never overestimates the support length for $m = 100$ and in less than 1% of the cases for $m = 1{,}000$, for SNR values of 20 or more.

Summing up the findings gleaned from the numerical experiments described above, we can conclude that Algorithm 9, because of its significantly smaller runtime, is the algorithm of choice for any setting where an a priori bound M on the support length m of the real vector $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, with short support is known. This is true both for noisy and exact data, since Algorithm 8 and Algorithm 9 perform similarly for noisy

SNR	Rate of Correct Recovery in % for $m = 1{,}000$				
	Alg. 8	Alg. 8, $m' \leq 3m$	Alg. 9, $M = m$	Alg. 9, $M = 3m$	Alg. 9, $M = 3m$, $m' \leq 3m$
0	83.1	68.0	51.6	88.0	0.0
10	96.4	95.0	51.6	93.4	53.7
20	100.0	99.7	99.4	100.0	84.5
30	100.0	99.6	100.0	100.0	89.3
40	100.0	99.8	100.0	100.0	94.8
50	100.0	99.8	100.0	100.0	98.1

Table 6.7: Rate of correct recovery of the support of $\mathbf{x}$ in % for Algorithm 8 and Algorithm 9 for $M = m$ and $M = 3m$, without bounding m' and with $m' \leq 3m$, for 1,000 random input vectors with support length $m = 1{,}000$ from Figure 6.14

data, as long as $M > m$. The slightly worse reconstruction error of Algorithm 9 for the case that $M = m$ is caused by the fact that a slightly overestimated support length leaves more room for detecting the first support indices correctly.

However, if no a priori knowledge of the support length is given or if one does not know whether the vector has a short or a one-block support, i.e., it is not clear whether the support is wrapped periodically around the boundary of the vector or not, then Algorithm 9 cannot be applied, but one can still use Algorithm 8, which detects the support length on the fly. The latter algorithm also has a slightly higher rate of correctly recovering the support of $\mathbf{x}$.

In settings where it is more important to obtain the first and last support index correctly than it is to obtain them as fast as possible, the usage of Algorithm 8 is also recommendable, especially since this method tends to overestimate the support length less than Algorithm 9.

7 Real 2D Block Sparse Fast IDCT-II

In Chapters 5 and 6 we derived two different algorithms for recovering a vector $\mathbf{x} \in \mathbb{R}^N$ from its DCT-II transformed vector, $\mathbf{x}^{\widehat{\mathrm{II}}}$, under the assumption that $\mathbf{x}$ has a one-block or a short support, respectively. Many practical applications of the DCT-II concern higher dimensional objects like images or videos. In this chapter we will derive a 2-dimensional IDCT-II algorithm for sparse matrices.

This chapter contains completely new, previously unpublished results, which I developed by myself.

In order to obtain a new sparse 2-dimensional IDCT-II algorithm, we will transfer the ideas presented in Chapter 6 to the 2-dimensional setting of sparse matrices. Instead of a short support we will now consider a block support, meaning that the support of a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, $M = 2^{J_R}$, $N = 2^{J_C}$, is contained in a rectangle of size $m \times n$, where both m and n are small compared to the matrix sizes M and N. The concept of reflected periodizations for vectors can be directly generalized to matrices. Hence, our new algorithm will be based on reconstructing $\mathbf{A}$ iteratively from its DCT-II transformed matrix, $\mathbf{A}^{\widehat{\mathrm{II}}}$, and its reflected periodizations $\mathbf{A}^{[L]}, \mathbf{A}^{[L+1]}, \ldots, \mathbf{A}$ for some suitably chosen starting index L. It will only employ real arithmetic and shorter 2-dimensional DCTs, instead of performing 1-dimensional M- and N-length IDCT-IIs row- and column-wise as described in Section 4.3. Analogously to Chapter 6, our 2-dimensional IDCT-II algorithm will require a priori knowledge of upper bounds $b_R \geq m$ and $b_C \geq n$ on the number of support rows and columns, respectively. As far as we are aware, the algorithm we will develop hereafter is the first existing 2-dimensional sparse IDCT-II algorithm that only uses real arithmetic.

7.1 Preliminaries

Throughout this chapter we will always consider a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, where $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$. Further, we will assume that $\mathbf{A}$ has a block support of size $m \times n$, and that upper bounds $b_R \geq m$ and $b_C \geq n$ are known.

Remark 7.1 A natural idea for deriving a fast 2-dimensional IDCT-II algorithm for matrices with block support would be to apply either Algorithm 8 or Algorithm 9 both row- and column-wise, analogously to one of the methods for obtaining fast full-sized 2-dimensional IDCT-II algorithms from Section 4.3. However, close examination of the definition of the 2-dimensional DCT-II shows that neither of the 1-dimensional sparse IDCT-II methods developed in this thesis is feasible for this problem. Recall that for a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ we have by Definition 4.9 that

$$\mathbf{A}^{\widehat{\mathrm{II}}} = \mathbf{C}_M^{\mathrm{II}} \mathbf{A} {\mathbf{C}_N^{\mathrm{II}}}^T.$$

Let us assume that $\mathbf{A}$ has a block support of size $m \times n$. We will now check whether the preconditions of Algorithms 8 and 9 for reconstructing $\mathbf{A}{\mathbf{C}_N^{\mathrm{II}}}^T$ from $\mathbf{A}^{\widehat{\mathrm{II}}} = \mathbf{C}_M^{\mathrm{II}} \left(\mathbf{A}{\mathbf{C}_N^{\mathrm{II}}}^T \right)$ are satisfied. Since $\mathbf{A}{\mathbf{C}_N^{\mathrm{II}}}^T = \left(\mathbf{C}_N^{\mathrm{II}} \mathbf{A}^T \right)^T$, i.e., a 1-dimensional DCT-II is applied to all

rows of $\mathbf{A}$, it follows that each column of $\mathbf{AC}_N^{\mathrm{II}\,T}$ has a short support of length m. Thus, all but the n rows of $\mathbf{AC}_N^{\mathrm{II}\,T}$ that correspond to the support of the columns only consist of zeros. However, having a short support in all columns of $\mathbf{AC}_N^{\mathrm{II}\,T}$ is not the only condition that has to be satisfied if we want to apply Algorithms 8 and 9 to all columns of $\mathbf{A}^{\widehat{\mathrm{II}}}$. Additionally, Algorithm 8 requires that we have for each column $\mathbf{a}_{\bullet,l}$ of $\mathbf{AC}_N^{\mathrm{II}\,T}$, $l \in \{0,\ldots,N-1\}$, that

$$\left(\mathbf{y}_{\bullet,l}^{(j)}\right)_{k \bmod 2^j} \neq 0 \qquad \forall\, j \in \{0,\ldots,\log_2 N\},$$

if $(\mathbf{y}_{\bullet,l})_k \neq 0$, where $\mathbf{y}_{\bullet,l} \coloneqq (\mathbf{a}_{\bullet,l}{}^T, (\mathbf{J}_N \mathbf{a}_{\bullet,l})^T)^T$ and $\mathbf{y}_{\bullet,l}^{(j)}$ denotes the 2^j-length periodization of $\mathbf{y}_{\bullet,l}$ according to Definition 5.7. For Algorithm 9 it is necessary that we have for the first and last support indices $\mu_l^{[J_R]}$ and $\nu_l^{[J_R]}$ of each column $\mathbf{a}_{\bullet,l} \in \mathbb{R}^{2^{J_R}} = M$ of $\mathbf{AC}_N^{\mathrm{II}\,T} \in \mathbb{R}^{M\times N}$ that

$$\left(\mathbf{a}_{\bullet,l}^{[J_R]}\right)_{\mu_l^{[J_R]}} + \left(\mathbf{a}_{\bullet,l}^{[J_R]}\right)_{\nu_l^{[J_R]}} \neq 0$$

if m is even. Neither of these two conditions can be satisfied without extensive knowledge of the matrix $\mathbf{A}$ we aim to recover. In general, even if $\mathbf{A} \in \mathbb{R}_{\geq 0}^{M\times N}$ or $\mathbf{A} \in \mathbb{R}_{\leq 0}^{M\times N}$, the nonzero entries of $\mathbf{AC}_N^{\mathrm{II}\,T}$ are not all positive or all negative. Consider for example the matrix

$$\mathbf{A} = \begin{pmatrix} 0 & & & & & & & 0 \\ & \ddots & & & & & \iddots & \\ & & 2 & 8 & & & & \\ & & 8 & 2 & & & & \\ & & & & & & & \\ & \iddots & & & & & \ddots & \\ 0 & & & & & & & 0 \end{pmatrix} \in \mathbb{R}^{8\times 8}$$

with block support of size 2×2. Then we obtain that

$$\mathbf{AC}_8^{\mathrm{II}\,T} \approx \begin{pmatrix} 0 & \ldots & & & & & \ldots & 0 \\ \vdots & & & & & & & \vdots \\ 0 & \ldots & & & & & \ldots & 0 \\ 3.54 & -0.59 & -4.62 & 1.67 & 3.54 & -2.49 & -1.91 & 2.94 \\ 3.54 & 0.59 & -4.62 & -1.67 & 3.54 & 2.49 & -1.91 & -2.94 \\ 0 & \ldots & & & & & \ldots & 0 \\ \vdots & & & & & & & \vdots \\ 0 & \ldots & & & & & \ldots & 0 \end{pmatrix}.$$

However, this matrix satisfies neither the non-cancellation condition (5.4) required by Algorithm 8 nor the non-cancellation condition (6.3) for Algorithm 9. For the column indexed by 1 we find that

$$\begin{aligned} \mathbf{a}_{\bullet,1} &\approx (0,0,0,-0.59,0.59,0,0,0)^T, \\ \mathbf{y}_{\bullet,1} = \left(\mathbf{a}_{\bullet,1}{}^T, (\mathbf{J}_N \mathbf{a}_{\bullet,1})^T\right)^T &\approx (0,0,0,-0.59,0.59,0,0,0,0,0,0,0.59,-0.59,0,0,0)^T, \\ \mathbf{y}_{\bullet,1}^{(4)} &= (0,0,0,0,0,0,0,0)^T, \end{aligned}$$

and further, since $\left(\mathbf{a}^{[4]}_{\bullet,1}\right)_{\mu_1^{[4]}} + \left(\mathbf{a}^{[4]}_{\bullet,1}\right)_{\nu_1^{[4]}} \approx -0.59 + 0.59 = 0$,

$$\begin{aligned}\mathbf{a}^{[4]}_{\bullet,1} &\approx (0,0,0,-0.59,0.59,0,0,0)^T,\\ \mathbf{a}^{[3]}_{\bullet,1} &= (0,0,0,0)^T.\end{aligned}$$

Thus, we have to develop new sparse IDCT-II methods for the 2-dimensional case. ◊

In order to obtain a sparse 2-dimensional IDCT-II algorithm, we will adapt techniques used in Chapter 6 for the reconstruction of vectors with short support from their DCT-II to the 2-dimensional setting. First, we need to define a generalization of the definition of a short support to matrices.

Definition 7.2 (Block Support I) Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$, and let $J \coloneqq \min\{J_R, J_C\}$. We say that $\mathbf{A} = (a_{k,l})_{k,l=0}^{M-1,N-1} \in \mathbb{R}^{M\times N}$ has a *block support* of size $m \times n$ if m and n are the minimal integers such that

$$a_{k,l} = 0 \quad \forall\,(k,l) \notin I_{\mu_R^{[J]},\nu_R^{[J]}} \times I_{\mu_C^{[J]},\nu_C^{[J]}},$$

for some $\mu_R^{[J]} \in \{0,\dots,M-m\}$ and $\mu_C^{[J]} \in \{0,\dots,N-n\}$ with $\nu_R^{[J]} \coloneqq \mu_R^{[J]} + m - 1$ and $\nu_C^{[J]} \coloneqq \mu_C^{[J]} + n - 1$. Further, there have to exist indices $k_0, k_1 \in I_{\mu_R^{[J]},\nu_R^{[J]}}$ and $l_0, l_1 \in I_{\mu_C^{[J]},\nu_C^{[J]}}$ such that

$$a_{\mu_R^{[J]},l_0} \neq 0, \quad a_{\nu_R^{[J]},l_1} \neq 0, \quad a_{k_0,\mu_C^{[J]}} \neq 0, \quad \text{and} \quad a_{k_1,\nu_C^{[J]}} \neq 0.$$

The intervals $S_R^{[J]} \coloneqq I_{\mu_R^{[J]},\nu_R^{[J]}}$ and $S_C^{[J]} \coloneqq I_{\mu_C^{[J]},\nu_C^{[J]}}$ are called the *row and column support* of $\mathbf{A}$, respectively. The set $S^{[J]} \coloneqq S_R^{[J]} \times S_C^{[J]}$ is called the *support block*. Further, the indices $\mu_R^{[J]}$ and $\mu_C^{[J]}$ are called the *first row and column support indices*, and $\nu_R^{[J]}$ and $\nu_C^{[J]}$ the *last row and column support indices*.

Remark 7.3 Note that, as in Chapter 6, we do not allow the support block to be wrapped periodically around any of the boundaries of the matrix. Consequently, the support block sizes m and n, and the first and last row and column support indices are uniquely determined. Instead of $S^{[J]}$, $m^{[J]}$ and $n^{[J]}$ etc., we usually write S, m and n.

Analogously to the 1-dimensional cases in Chapters 5 and 6, the block $S^{[J]}$ contains all indices at which the matrix $\mathbf{A}$ has nonzero entries, while $\mathbf{A}$ may also be zero at indices contained in $S^{[J]}$, since we need the support of $\mathbf{A}$ to be a block in $\mathbb{N}_0 \times \mathbb{N}_0$ for some of the theoretical concepts used hereafter. ◊

If we want to transfer the techniques used in Chapter 6 to the 2-dimensional setting, we need to define an analog to the reflectedly periodized vectors $\mathbf{x}^{[j]} \in \mathbb{R}^{2^j}$ of $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, from Definition 6.4. For this we first introduce some more notation to generalize the concept of first and second half of a vector.

Definition 7.4 (Notation) Let $M = 2^{J_R}$ and $N = 2^{J_C}$ for $J_R, J_C \in \mathbb{N}$. For a matrix $\mathbf{A} \in \mathbb{R}^{2^{j_R} \times 2^{j_C}}$, $j_R \in \{1, \dots, J_R\}$, $j_C \in \{1, \dots, J_C\}$, we denote by

$$\mathbf{A}_{(0,0)} \coloneqq (a_{k,l})_{k,l=0}^{2^{j_R-1}-1,\, 2^{j_C-1}-1}, \qquad \mathbf{A}_{(0,1)} \coloneqq (a_{k,l})_{k=0,\, l=2^{j_C-1}}^{2^{j_R-1}-1,\, 2^{j_C}-1},$$
$$\mathbf{A}_{(1,0)} \coloneqq (a_{k,l})_{k=2^{j_R-1},\, l=0}^{2^{j_R}-1,\, 2^{j_C-1}-1}, \qquad \mathbf{A}_{(0,0)} \coloneqq (a_{k,l})_{k=2^{j_R-1},\, l=2^{j_C-1}}^{2^{j_R}-1,\, 2^{j_C}-1},$$

the four quadrants of size $2^{j_R-1} \times 2^{j_C-1}$ of $\mathbf{A}$, i.e.,

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{(0,0)} & \mathbf{A}_{(0,1)} \\ \mathbf{A}_{(1,0)} & \mathbf{A}_{(1,1)} \end{pmatrix}.$$

Definition 7.4 allows us to extend the notion of reflected periodizations to matrices.

Definition 7.5 (Reflected Periodization) Let $M = 2^{J_R}$ and $N = 2^{J_C}$, $J_R, J_C \in \mathbb{N}$. Let $\mathbf{A} \in \mathbb{R}^{M \times N}$. Set $J \coloneqq \min\{J_R, J_C\}$ and $\mathbf{A}^{[J]} \coloneqq \mathbf{A}$. For $j \in \{0, \dots, J-1\}$ let $M^{[j]} \coloneqq 2^{J_R - J + j}$, $N^{[j]} \coloneqq 2^{J_C - J + j}$, and define the *reflected periodization* $\mathbf{A}^{[j]} \in \mathbb{R}^{M^{[j]} \times N^{[j]}}$

$$\mathbf{A}^{[j]} \coloneqq \mathbf{A}_{(0,0)}^{[j+1]} + \mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,0)}^{[j+1]} + \left(\mathbf{A}_{(0,1)}^{[j+1]} + \mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,1)}^{[j+1]} \right) \mathbf{J}_{N^{[j]}}.$$

By definition, for any $j \in \{0, \dots, J-1\}$, the reflected periodization $\mathbf{A}^{[j]} \in \mathbb{R}^{M^{[j]} \times N^{[j]}}$ is obtained by first adding the bottom left quadrant with reverse-ordered rows, $\mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,0)}^{[j+1]}$, to the top left quadrant $\mathbf{A}_{(0,0)}^{[j+1]}$, and the bottom right quadrant with reverse-ordered rows, $\mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,1)}^{[j+1]}$, to the top right quadrant, $\mathbf{A}_{(0,1)}^{[j+1]}$. Then the sum originating from the right half of the matrix is added with reverse-ordered columns, $\left(\mathbf{A}_{(0,1)}^{[j+1]} + \mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,1)}^{[j+1]} \right) \mathbf{J}_{N^{[j]}}$, to the sum arising from the left half of the matrix, $\mathbf{A}_{(0,0)}^{[j+1]} + \mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,0)}^{[j+1]}$. See Figure 7.1 and Example 7.6 for an illustration. One can also visualize the reflected periodization

$$\mathbf{J}_{M^{[j]}} \left(\left(\begin{array}{c|c} \mathbf{A}_{(0,0)}^{[j+1]} & \mathbf{A}_{(0,1)}^{[j+1]} \\ \hline \mathbf{A}_{(1,0)}^{[j+1]} & \mathbf{A}_{(1,1)}^{[j+1]} \end{array} \right) \right) \mathbf{J}_{M^{[j]}}$$

$$\rightsquigarrow \quad \left(\begin{array}{c|c} \mathbf{A}_{(0,0)}^{[j+1]} + \mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,0)}^{[j+1]} & \mathbf{A}_{(0,1)}^{[j+1]} + \mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,1)}^{[j+1]} \end{array} \right) \mathbf{J}_{N^{[j]}}$$

$$\rightsquigarrow \quad \left(\mathbf{A}^{[j]} \right)$$

Figure 7.1: Visualization of the reflected periodization

by imagining folding a piece of paper twice, halving first the vertical edges and then the horizontal ones.

Example 7.6 Let $\mathbf{A} \in \mathbb{R}^{8\times 8}$ with nonzero entries $a_{3,3}, a_{3,4}, a_{4,3}, a_{4,4}$, i.e.,

$$\mathbf{A} = \left(\begin{array}{cccc|cccc} 0 & & & 0 & 0 & & & 0 \\ 0 & & & & & & & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & 0 & 0 & a_{3,3} & a_{3,4} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & a_{4,3} & a_{4,4} & 0 & 0 & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & & & & & & & 0 \\ 0 & & & 0 & 0 & & & 0 \end{array}\right).$$

Performing first a folding of the columns, i.e., adding the bottom half of $\mathbf{A}$ with reverse-ordered rows to the top half of $\mathbf{A}$, we obtain the intermediate matrix

$$\left(\begin{array}{ccc|ccc} 0 & & 0 & 0 & & 0 \\ 0 & & & & & 0 \\ 0 & & 0 & 0 & & 0 \\ 0 & 0 & a_{3,3}+a_{4,3} & a_{3,4}+a_{4,4} & 0 & 0 \end{array}\right).$$

Folding the rows, i.e., adding the right half of the intermediate matrix with reverse-ordered columns to its left half, now yields the reflected periodization $\mathbf{A}^{[2]} \in \mathbb{R}^{4\times 4}$,

$$\mathbf{A}^{[2]} = \left(\begin{array}{ccc|c} 0 & 0 & 0 & 0 \\ 0 & & & 0 \\ 0 & & & 0 \\ 0 & 0 & 0 & a_{3,3}+a_{4,3}+a_{3,4}+a_{4,4} \end{array}\right).$$

$\Diamond$

Remark 7.7 We can also apply the factorization of the cosine matrix of type II from Lemma 4.5 in the 2-dimensional case, which can be used to motivate the definition of the reflected periodization for matrices. Recall that for $N \in \mathbb{N}$ even we have that

$$\mathbf{C}_N^{\mathrm{II}} = \mathbf{P}_N{}^T \left(\begin{array}{c|c} \mathbf{C}_{\frac{N}{2}}^{\mathrm{II}} & \mathbf{0}_{\frac{N}{2}} \\ \hline \mathbf{0}_{\frac{N}{2}} & \mathbf{C}_{\frac{N}{2}}^{\mathrm{IV}} \end{array}\right) \mathbf{T}_N.$$

Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$, and $\mathbf{A} \in \mathbb{R}^{M\times N}$. Then we obtain for all $j \in \{0, \ldots, J-1\}$ that

$$\begin{aligned} &\left(\mathbf{A}^{[j+1]}\right)^{\widehat{\mathrm{II}}} \\ &= \mathbf{C}_{M[j+1]}^{\mathrm{II}} \mathbf{A}^{[j+1]} \mathbf{C}_{N[j+1]}^{\mathrm{II}}{}^T \\ &= \frac{1}{2} \mathbf{P}_{M[j+1]}{}^T \begin{pmatrix} \mathbf{C}_{M[j]}^{\mathrm{II}} & \\ & \mathbf{C}_{M[j]}^{\mathrm{IV}} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{M[j]} & \mathbf{J}_{M[j]} \\ \mathbf{I}_{M[j]} & -\mathbf{J}_{M[j]} \end{pmatrix} \mathbf{A}^{[j+1]} \begin{pmatrix} \mathbf{I}_{N[j]} & \mathbf{I}_{N[j]} \\ \mathbf{J}_{N[j]} & -\mathbf{J}_{N[j]} \end{pmatrix} \\ &\quad \cdot \begin{pmatrix} \mathbf{C}_{N[j]}^{\mathrm{II}}{}^T & \\ & \mathbf{C}_{N[j]}^{\mathrm{IV}}{}^T \end{pmatrix} \mathbf{P}_{N[j+1]} \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}\mathbf{P}_{M^{[j+1]}}{}^{T} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}} & \\ & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \end{pmatrix} \begin{pmatrix} \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} & \mathbf{A}^{[j+1]}_{(0,1)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)} \\ \mathbf{A}^{[j+1]}_{(0,0)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} & \mathbf{A}^{[j+1]}_{(0,1)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)} \end{pmatrix} \\
&\quad \cdot \begin{pmatrix} \mathbf{I}_{N^{[j]}} & \mathbf{I}_{N^{[j]}} \\ \mathbf{J}_{N^{[j]}} & -\mathbf{J}_{N^{[j]}} \end{pmatrix} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^{T} & \\ & \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix} \mathbf{P}_{N^{[j+1]}} \\
&=: \frac{1}{2}\mathbf{P}_{M^{[j+1]}}{}^{T} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}} & \\ & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \end{pmatrix} \begin{pmatrix} \mathbf{A}^{[j]} & \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} \\ \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} & \tilde{\mathbf{A}}^{[j+1]}_{(1,1)} \end{pmatrix} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^{T} & \\ & \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix} \mathbf{P}_{N^{[j+1]}} \\
&= \frac{1}{2}\mathbf{P}_{M^{[j+1]}}{}^{T} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^{T} & \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(0,1)}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ \mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(1,0)}\mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^{T} & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(1,1)}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix} \mathbf{P}_{N^{[j+1]}}, \qquad (7.1)
\end{aligned}
$$

where

$$
\begin{aligned}
\mathbf{A}^{[j]} := \tilde{\mathbf{A}}^{[j+1]}_{(0,0)} &:= \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} + \left(\mathbf{A}^{[j+1]}_{(0,1)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}, \\
\tilde{\mathbf{A}}^{[j+1]}_{(0,1)} &:= \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} - \left(\mathbf{A}^{[j+1]}_{(0,1)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}, \\
\tilde{\mathbf{A}}^{[j+1]}_{(1,0)} &:= \mathbf{A}^{[j+1]}_{(0,0)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} + \left(\mathbf{A}^{[j+1]}_{(0,1)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}, \\
\tilde{\mathbf{A}}^{[j+1]}_{(1,1)} &:= \mathbf{A}^{[j+1]}_{(0,0)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} - \left(\mathbf{A}^{[j+1]}_{(0,1)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}.
\end{aligned} \qquad (7.2)
$$

Using that

$$
\begin{aligned}
&\mathbf{P}_{M^{[j+1]}}\left(\mathbf{A}^{[j+1]}\right)^{\widehat{\mathrm{II}}}\mathbf{P}_{N^{[j+1]}}{}^{T} \\
&= \begin{pmatrix} \left(a^{\widehat{\mathrm{II}}}_{2k,\,2l}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} & \left(a^{\widehat{\mathrm{II}}}_{2k,\,2l+1}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} \\ \left(a^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} & \left(a^{\widehat{\mathrm{II}}}_{2k+1,\,2l+1}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} \end{pmatrix}, \qquad (7.3)
\end{aligned}
$$

we also find that (7.1) is equivalent to

$$
\begin{aligned}
&\begin{pmatrix} \left(a^{\widehat{\mathrm{II}}}_{2k,\,2l}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} & \left(a^{\widehat{\mathrm{II}}}_{2k,\,2l+1}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} \\ \left(a^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} & \left(a^{\widehat{\mathrm{II}}}_{2k+1,\,2l+1}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} \end{pmatrix} \\
&= \frac{1}{2}\begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^{T} & \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(0,1)}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ \mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(1,0)}\mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^{T} & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(1,1)}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix}. \qquad (7.4)
\end{aligned}
$$

$\Diamond$

By (7.1) it is evident that the reflected periodization for matrices, which is given in Definition 7.5, occurs naturally for the 2-dimensional DCT-II if the factorization of $\mathbf{C}^{\mathrm{II}}_{N}$ from Lemma 4.5 is used. The concept of reflected periodizations is of such importance for developing a fast 2-dimensional IDCT-II algorithm, because the DCT-IIs of the reflected periodizations are already completely determined by the DCT-II of $\mathbf{A}$, similarly to the

1-dimensional case in Lemma 6.6.

Lemma 7.8 Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$. Let $\mathbf{A} \in \mathbb{R}^{M \times N}$. Set $J \coloneqq \min\{J_R, J_C\}$ and let $j \in \{0, \dots, J\}$. Then $\left(\mathbf{A}^{[j]}\right)^{\widehat{\mathrm{II}}}$ satisfies

$$\left(\mathbf{A}^{[j]}\right)^{\widehat{\mathrm{II}}} = 2^{J-j} \left(a^{\widehat{\mathrm{II}}}_{2^{J-j}k,\, 2^{J-j}l}\right)^{M^{[j]}-1,\, N^{[j]}-1}_{k,\, l=0}.$$

Proof. We show the lemma by induction. For $j = J$ the claim holds, since $\mathbf{A}^{[J]} = \mathbf{A}$ by definition. Now we assume that the induction hypothesis is satisfied for some index $j+1 \in \{1, \dots, J\}$ and prove that the claim also holds for j. By (7.1) we find that

$$\mathbf{P}_{M^{[j+1]}} \left(\mathbf{A}^{[j+1]}\right)^{\widehat{\mathrm{II}}} \mathbf{P}_{N^{[j+1]}}{}^{T} = \frac{1}{2} \begin{pmatrix} \left(\mathbf{A}^{[j]}\right)^{\widehat{\mathrm{II}}} & \mathbf{C}^{\mathrm{II}}_{M^{[j]}} \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} \mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^{T} & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \tilde{\mathbf{A}}^{[j+1]}_{(1,1)} \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix}.$$

Consequently, the induction hypothesis and (7.3), applied to the upper left quadrant of the right-hand side, yield

$$\begin{aligned} \left(\mathbf{A}^{[j]}\right)^{\widehat{\mathrm{II}}} &= 2 \left(\left(\mathbf{A}^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k,\, 2l} \right)^{M^{[j]}-1,\, N^{[j]}-1}_{k,\, l=0} \\ &= 2 \left(2^{J-(j+1)} a^{\widehat{\mathrm{II}}}_{2^{J-(j+1)}2k,\, 2^{J-(j+1)}2l} \right)^{M^{[j]}-1,\, N^{[j]}-1}_{k,\, l=0} \\ &= 2^{J-j} \left(a^{\widehat{\mathrm{II}}}_{2^{J-j}k,\, 2^{J-j}l} \right)^{M^{[j]}-1,\, N^{[j]}-1}_{k,\, l=0}, \end{aligned}$$

which completes the proof. □

We will prove shortly that, analogously to Chapter 6, the reflected periodizations of a matrix with block support will also have a short support. First, we introduce notation for their supports sets.

Definition 7.9 (Block Support II) Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$, and $J \coloneqq \min\{J_R, J_C\}$. Let $\mathbf{A} \in \mathbb{R}^{M \times N}$ and $j \in \{0, \dots, J-1\}$. We say that $\mathbf{A}^{[j]} \in \mathbb{R}^{M^{[j]} \times N^{[j]}}$ has a *block support* of size $m^{[j]} \times n^{[j]}$ if $m^{[j]}$ and $n^{[j]}$ are the minimal integers such that

$$a^{[j]}_{k,l} = 0 \quad \forall\, (k,l) \notin I_{\mu^{[j]}_R, \nu^{[j]}_R} \times I_{\mu^{[j]}_C, \nu^{[j]}_C},$$

for some $\mu^{[j]}_R \in \{0, \dots, M-m\}$ and $\mu^{[j]}_C \in \{0, \dots, N-n\}$ with $\nu^{[j]}_R \coloneqq \mu^{[j]}_R + m - 1$ and $\nu^{[j]}_C \coloneqq \mu^{[j]}_C + n - 1$. Further, there have to exist indices $k_0, k_1 \in I_{\mu^{[j]}_R, \nu^{[j]}_R}$ and $l_0, l_1 \in I_{\mu^{[j]}_C, \nu^{[j]}_C}$ such that

$$a^{[j]}_{\mu^{[j]}_R, l_0} \neq 0, \quad a^{[j]}_{\nu^{[j]}_R, l_1} \neq 0, \quad a^{[j]}_{k_0, \mu^{[j]}_C} \neq 0 \quad \text{and} \quad a^{[j]}_{k_1, \nu^{[j]}_C} \neq 0.$$

The intervals $S^{[j]}_R \coloneqq I_{\mu^{[j]}_R, \nu^{[j]}_R}$ and $S^{[j]}_C \coloneqq I_{\mu^{[j]}_C, \nu^{[j]}_C}$ are called the *row and column support* of $\mathbf{A}^{[j]}$, respectively. The set $S^{[j]} \coloneqq S^{[j]}_R \times S^{[j]}_C \coloneqq$ is called the *support block*. The indices $\mu^{[j]}_R$ and $\mu^{[j]}_C$ are called the *first row and column support indices*, and $\nu^{[j]}_R$ and $\nu^{[j]}_C$ the *last row and column support indices*.

7.2 Support Structures of Reflectedly Periodized Matrices

Analogously to Chapters 5 and 6, our goal is to recover a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, $M = 2^{J_R}$, $N = 2^{J_C}$, with block support of size $m \times n$ from $\mathbf{A}^{\widehat{\Pi}}$ by successively calculating its reflected periodizations $\mathbf{A}^{[L]}, \mathbf{A}^{[L+1]}, \ldots, \mathbf{A}^{[J]} = \mathbf{A}$. Our 2-dimensional IDCT-II method will be based on a generalization of Algorithm 9, and, even for $M = N$, the reconstruction cannot begin with $\mathbf{A}^{[0]} \in \mathbb{R}$ due to similar support constraints. This means that we will require a priori knowledge of upper bounds b_R and b_C on the row and column support lengths, and have to choose L depending on them.

Transferring the approaches from Algorithm 9 to matrices implies that the $(j-L)$th iteration step consists of efficiently reconstructing $\mathbf{A}^{[j+1]}$ from $\mathbf{A}^{\widehat{\Pi}}$ utilizing that $\mathbf{A}^{[j]}$ is known from the previous step. Thus, we have to investigate how the support of $\mathbf{A}^{[j+1]}$ can look like if the support of $\mathbf{A}^{[j]}$ is given. Furthermore, we require an analog to the non-cancellation condition (6.3) for Algorithm 9.

Remark 7.10 In order to preserve the information necessary to detect the correct block support in all iteration steps, we require the following non-cancellation conditions. For exact data it is necessary by Definition 7.2 that there exist indices $k_0, k_1 \in I_{\mu_R^{[J]}, \nu_R^{[J]}}$ and $l_0, l_1 \in I_{\mu_C^{[J]}, \nu_C^{[J]}}$ such that

$$a_{\mu_R^{[J]}, l_0} \neq 0, \quad a_{\nu_R^{[J]}, l_1} \neq 0, \quad a_{k_0, \mu_C^{[J]}} \neq 0 \quad \text{and} \quad a_{k_1, \nu_C^{[J]}} \neq 0.$$

Additionally, by Definition 7.5, we need that there exist an index $l \in I_{\mu_C^{[J]}, \nu_C^{[J]}}$ such that

$$a_{\mu_R^{[J]}, l} + a_{\nu_R^{[J]}, l} + a_{\mu_R^{[J]}, N-1-l} + a_{\nu_R^{[J]}, N-1-l} \neq 0 \quad \text{if } n \text{ is even}, \tag{7.5}$$

and an index $k \in I_{\mu_R^{[J]}, \nu_R^{[J]}}$ such that

$$a_{k, \mu_C^{[J]}} + a_{k, \nu_C^{[J]}} + a_{M-1-k, \mu_C^{[J]}} + a_{M-1-k, \nu_C^{[J]}} \neq 0 \quad \text{if } m \text{ is even}. \tag{7.6}$$

Conditions (7.5) and (7.6) ensure that in every reflected periodization $\mathbf{A}^{[j]} \in \mathbb{R}^{M^{[j]} \times N^{[j]}}$, $j \in \{L, \ldots, J\}$, there is at least one row from which the correct current row support length can be detected, and one column from which the correct current column support length can be determined. These assumptions are for example satisfied if $\mathbf{A} \in \mathbb{R}^{M \times N}_{\geq 0}$ or if $\mathbf{A} \in \mathbb{R}^{M \times N}_{\leq 0}$.

In practice, i.e., for noisy data, we have to guarantee that for a threshold $\varepsilon > 0$ depending on the noisy level we have, besides

$$\left|a_{\mu_R^{[J]}, l_0}\right| > \varepsilon, \quad \left|a_{\nu_R^{[J]}, l_1}\right| > \varepsilon, \quad \left|a_{k_0, \mu_C^{[J]}}\right| > \varepsilon \quad \text{and} \quad \left|a_{k_1, \nu_C^{[J]}}\right| > \varepsilon,$$

also

$$\left|a_{\mu_R^{[J]}, l} + a_{\nu_R^{[J]}, l} + a_{\mu_R^{[J]}, N-1-l} + a_{\nu_R^{[J]}, N-1-l}\right| > \varepsilon \qquad \text{and}$$
$$\left|a_{k, \mu_C^{[J]}} + a_{k, \nu_C^{[J]}} + a_{M-1-k, \mu_C^{[J]}} + a_{M-1-k, \nu_C^{[J]}}\right| > \varepsilon.$$

$\Diamond$

7.2.1 Support Structure of $\mathbf{A}^{[j]}$ for Given $\mathbf{A}$

We showed in Section 6.2.1 that for a vector $\mathbf{x} \in \mathbb{R}^N$, $N = 2^J$, with short support of length m the reflected periodizations $\mathbf{x}^{[j]} \in \mathbb{R}^{2^j}$ for $j \in \{L, \dots, J-1\}$ have a short support of length $m^{[j]} \leq m$. Thus, we expect similar results for the reflected periodizations of a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, $M = 2^{J_R}$, $N = 2^{J_C}$, with block support of length $m \times n$. Analogously to Chapter 6, we will only consider the reflected periodizations to the level K, where we now set

$$K \coloneqq \max\left\{\lceil \log_2 m \rceil + 1, \lceil \log_2 n \rceil + 1\right\},$$

in order to reduce the collisions of nonzero entries of $\mathbf{A}$ in the reflected periodizations $\mathbf{A}^{[j]}$. This causes our method, like Algorithm 9, to require a priori knowledge of upper bounds b_R and b_C on the number of support rows and columns.

Before we prove the corresponding lemma, let us motivate the claims therein by some examples.

Example 7.11

1. Let $\mathbf{A} \in \mathbb{R}^{16 \times 16}$ satisfy (7.5) and (7.6) with nonzero entries $a_{3,11}$, $a_{3,12}$, $a_{4,11}$ and $a_{4,12}$, i.e., with block support $S^{[4]} = I_{3,4} \times I_{11,12}$. We assume that $m = 2$ and $n = 2$ are known exactly, i.e., that $b_R = b_C = 2$. Then $K = 2$, and $\mathbf{A}$ and its reflected periodizations $\mathbf{A}^{[j]}$ for $j \in \{K, \dots, J\}$ are

$$\mathbf{A} = \mathbf{A}^{[4]} = \left(\begin{array}{ccc|cccccccc}
0 & \dots & 0 & 0 & \dots & & & \dots & & & 0 \\
\vdots & & \vdots & 0 & & & & & & & 0 \\
 & & & 0 & & & 0 & 0 & & & 0 \\
 & & & 0 & 0 & 0 & a_{3,11} & a_{3,12} & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 & a_{4,11} & a_{4,12} & 0 & 0 & 0 \\
 & & & 0 & & & 0 & 0 & & & 0 \\
\vdots & & \vdots & 0 & & & & & & & 0 \\
0 & \dots & 0 & 0 & \dots & & & & & \dots & 0 \\
\hline
0 & \dots & 0 & 0 & \dots & & & & & \dots & 0 \\
\vdots & & \vdots & \vdots & & & & & & & \vdots \\
0 & \dots & 0 & 0 & \dots & & & & & \dots & 0
\end{array}\right)$$

$$\mathbf{A}^{[3]} = \left(\begin{array}{ccc|ccccc}
0 & & & 0 & 0 & & & 0 \\
0 & & & & & & & 0 \\
0 & & & 0 & 0 & & & 0 \\
0 & 0 & 0 & a_{3,12} & a_{3,11} & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & a_{4,12} & a_{4,11} & 0 & 0 & 0 \\
0 & & & 0 & 0 & & & 0 \\
0 & & & & & & & 0 \\
0 & & & 0 & 0 & & & 0
\end{array}\right)$$

$$\mathbf{A}^{[2]} = \begin{pmatrix}
0 & & & & 0 & & \\
0 & & & & 0 & & \\
0 & & & & 0 & & \\
0 & 0 & 0 & a_{3,11} + a_{3,12} + a_{4,11} + a_{4,12} & & &
\end{pmatrix}$$

Here, $\mathbf{A}^{[3]}$ has the block support $S^{[3]} = I_{3,4} \times I_{3,4}$ of size $m^{[3]} \times n^{[3]} \coloneqq 2 \times 2$, and $\mathbf{A}^{[2]}$ has the block support $S^{[2]} = I_{3,3}$ of size $m^{[2]} \times n^{[2]} \coloneqq 1 \times 1$.

2. Let $\mathbf{A} \in \mathbb{R}^{16\times 16}$ satisfy (7.5) and (7.6) with nonzero entries $a_{7,11}$, $a_{7,12}$, $a_{8,11}$ and $a_{8,12}$, i.e., with block support $S^{[4]} = I_{7,8} \times I_{11,12}$. We assume again that $m = 2$ and $n = 2$ are known exactly. Then the reflected periodizations of $\mathbf{A}$ are

$$\mathbf{A} = \mathbf{A}^{[4]} = \left(\begin{array}{ccc|cccccccc} 0 & \dots & 0 & 0 & \dots & & & & \dots & 0 \\ \vdots & & \vdots & \vdots & & & & & & \vdots \\ \vdots & & \vdots & 0 & \dots & & & & \dots & 0 \\ 0 & \dots & 0 & 0 & 0 & 0 & a_{7,11} & a_{7,12} & 0 & 0 & 0 \\ \hline 0 & \dots & 0 & 0 & 0 & 0 & a_{8,11} & a_{8,12} & 0 & 0 & 0 \\ \vdots & & \vdots & 0 & \dots & & & & \dots & 0 \\ \vdots & & \vdots & \vdots & & & & & & \vdots \\ 0 & \dots & 0 & 0 & \dots & & & & \dots & 0 \end{array}\right)$$

$$\mathbf{A}^{[3]} = \left(\begin{array}{ccc|c|c|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & & & & & & & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & & & 0 & 0 & & & 0 \\ \hline 0 & & & 0 & 0 & & & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & 0 & 0 & a_{7,12}+a_{8,12} & a_{7,11}+a_{8,11} & 0 & 0 & 0 \end{array}\right)$$

$$\mathbf{A}^{[2]} = \begin{pmatrix} 0 & 0 & 0 & a_{7,11}+a_{7,12}+a_{8,11}+a_{8,12} \\ 0 & & & 0 \\ 0 & & & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Here, $\mathbf{A}^{[3]}$ has the block support $S^{[3]} = I_{7,7} \times I_{3,4}$ of size $m^{[3]} \times n^{[3]} \coloneqq 1 \times 2$, and $\mathbf{A}^{[2]}$ has the block support $S^{[2]} = I_{0,0} \times I_{3,3}$ of size $m^{[2]} \times n^{[2]} \coloneqq 1 \times 1$. ◊

As in Chapter 6, we have to undo collisions of nonzero entries in the reflected periodizations. For both matrices considered in Example 7.11 the reflected periodizations $\mathbf{A}^{[j]}$ with $j \in \{K, \dots, J\}$ have a block support of size $m^{[j]} \times n^{[j]}$, where $m^{[j]} \leq m$ and $n^{[j]} \leq n$. We generalize this observation in the next lemma.

Lemma 7.12 Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$, and let $J \coloneqq \min\{J_R, J_C\}$. Let $\mathbf{A} \in \mathbb{R}^{M\times N}$ have a block support of size $m \times n$ and assume that $\mathbf{A}$ satisfies (7.5) and (7.6). Set $K \coloneqq \max\{\lceil \log_2 m \rceil + 1, \lceil \log_2 n \rceil + 1\}$ and let $j \in \{K, \dots, J\}$. Then $\mathbf{A}^{[j]}$ has a block support of size $m^{[j]} \times n^{[j]}$ with $m^{[j]} \leq m$ and $n^{[j]} \leq n$.

Proof. We prove the lemma by induction. By assumption, $\mathbf{A} = \mathbf{A}^{[J]}$ has a block support of size $m \times n$. Now we suppose that $\mathbf{A}^{[j+1]}$ has a block support of size $m^{[j+1]} \times n^{[j+1]}$ with $m^{[j+1]} \leq m$ and $n^{[j+1]} \leq n$ for some $j \in \{K, \dots, J-1\}$. More precisely, we assume that $\mathbf{A}^{[j+1]}$ has the support block

$$S^{[j+1]} = I_{\mu_R^{[j+1]}, \nu_R^{[j+1]}} \times I_{\mu_C^{[j+1]}, \nu_C^{[j+1]}},$$

where $\mu_R^{[j+1]} \in \{0, \dots, M^{[j+1]} - m^{[j+1]}\}$ and $\mu_C^{[j+1]} \in \{0, \dots, N^{[j+1]} - n^{[j+1]}\}$, and we set $\nu_R^{[j+1]} \coloneqq \mu_R^{[j+1]} + m^{[j+1]} - 1$ and $\nu_C^{[j+1]} \coloneqq \mu_C^{[j+1]} + n^{[j+1]} - 1$. We have to distinguish four main cases and their subcases.

(i) $S^{[j+1]}$ contains indices from all four quadrants of $\mathbf{A}^{[j+1]}$, i.e., $\{M^{[j]}-1, M^{[j]}\} \times \{N^{[j]}-1, N^{[j]}\} \subseteq S^{[j+1]}$.

Then it follows from Definition 7.5 that

$$\begin{aligned} S^{[j]} &= \left(\left(I_{\mu_R^{[j+1]},\nu_R^{[j+1]}} \cup I_{M^{[j+1]}-1-\nu_R^{[j+1]},\, M^{[j+1]}-1-\mu_R^{[j+1]}}\right) \cap I_{0,\, M^{[j]}-1}\right) \\ &\quad \times \left(\left(I_{\mu_C^{[j+1]},\nu_C^{[j+1]}} \cup I_{N^{[j+1]}-1-\nu_C^{[j+1]},\, N^{[j+1]}-1-\mu_C^{[j+1]}}\right) \cap I_{0,\, N^{[j]}-1}\right) \\ &=: I_{M^{[j]}-m^{[j]},\, M^{[j]}-1} \times I_{N^{[j]}-n^{[j]},\, N^{[j]}-1} \\ &\subsetneq I_{M^{[j]}-m^{[j+1]},\, M^{[j]}-1} \times I_{N^{[j]}-n^{[j+1]},\, N^{[j]}-1}. \end{aligned}$$

Thus, $\mathbf{A}^{[j]}$ has a block support as well. Due to collision of possibly nonzero entries of $\mathbf{A}^{[j+1]}$, the support of $\mathbf{A}^{[j]}$ has less rows and less columns than the support of $\mathbf{A}^{[j+1]}$, i.e., $m^{[j]} < m^{[j+1]}$ and $n^{[j]} < n^{[j+1]}$, see also Figure 7.2.

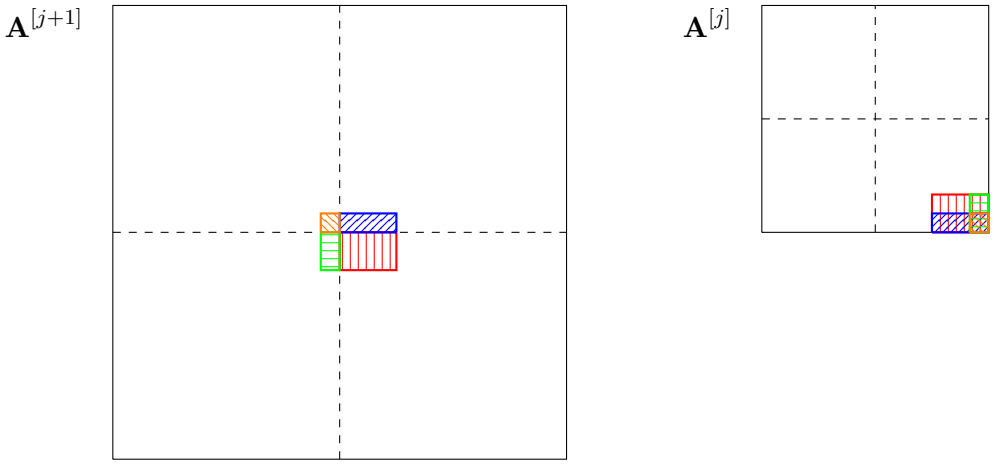

Figure 7.2: Illustration of the support of $\mathbf{A}^{[j]}$ if $\{M^{[j]}-1, M^{[j]}\} \times \{N^{[j]}-1, N^{[j]}\} \subseteq S^{[j+1]}$

(ii) $S^{[j+1]}$ is completely contained in the upper or lower half, but contains indices from both the left and right half of $\mathbf{A}^{[j+1]}$.

(a) $S^{[j+1]} \subseteq I_{0,\, M^{[j]}-1} \times I_{0,\, N^{[j+1]}-1}$ and $\{N^{[j]}-1, N^{[j]}\} \subseteq S_C^{[j+1]}$.

Then it follows from Definition 7.5 that

$$\begin{aligned} S^{[j]} &= I_{\mu_R^{[j+1]},\nu_R^{[j+1]}} \times \left(\left(I_{\mu_C^{[j+1]},\nu_C^{[j+1]}} \cup I_{N^{[j+1]}-1-\nu_C^{[j+1]},\, N^{[j+1]}-1-\mu_C^{[j+1]}}\right) \cap I_{0,\, N^{[j]}-1}\right) \\ &=: I_{\mu_R^{[j]},\nu_R^{[j]}} \times I_{N^{[j]}-n^{[j]},\, N^{[j]}-1} \\ &\subsetneq I_{\mu_R^{[j+1]},\nu_R^{[j+1]}} \times I_{N^{[j]}-n^{[j+1]},\, N^{[j]}-1}. \end{aligned}$$

(b) $S^{[j+1]} \subseteq I_{M^{[j]},\, M^{[j+1]}-1} \times I_{0,\, N^{[j+1]}-1}$ and $\left\{N^{[j]}-1, N^{[j]}\right\} \subseteq S_C^{[j+1]}$.

Then we obtain that

$$\begin{aligned} S^{[j]} &= I_{M^{[j+1]}-1-\nu_R^{[j+1]},\, M^{[j+1]}-1-\mu_R^{[j+1]}} \\ &\quad \times \left(\left(I_{\mu_C^{[j+1]},\, \nu_C^{[j+1]}} \cup I_{N^{[j+1]}-1-\nu_C^{[j+1]},\, N^{[j+1]}-1-\mu_C^{[j+1]}}\right) \cap I_{0,\, N^{[j]}-1}\right) \\ &=: I_{\mu_R^{[j]},\, \nu_R^{[j]}} \times I_{N^{[j]}-n^{[j]},\, N^{[j]}-1} \\ &\subsetneq I_{M^{[j+1]}-1-\nu_R^{[j+1]},\, M^{[j+1]}-1-\mu_R^{[j+1]}} \times I_{N^{[j]}-n^{[j+1]},\, N^{[j]}-1}, \end{aligned}$$

see also Figure 7.3.

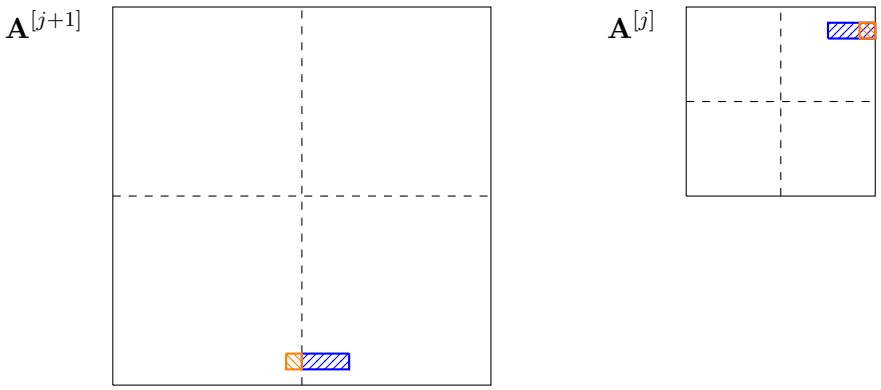

Figure 7.3: Illustration of the supports of $\mathbf{A}^{[j+1]}$ and $\mathbf{A}^{[j]}$ if $\left\{N^{[j]}-1, N^{[j]}\right\} \subseteq S_C^{[j+1]}$ and $S_R^{[j+1]} \subseteq I_{M^{[j]},\, M^{[j+1]}-1}$

Consequently, we find that $\mathbf{A}^{[j+1]}$ has a block support as well. Due to collision of possibly nonzero entries from $\mathbf{A}^{[j+1]}$ in both subcases of case (ii), the support of $\mathbf{A}^{[j]}$ has less columns than the one of $\mathbf{A}^{[j+1]}$, i.e., $n^{[j]} < n^{[j+1]}$, and the supports of $\mathbf{A}^{[j]}$ and $\mathbf{A}^{[j+1]}$ have the same number of rows. i.e., $m^{[j+1]} = m^{[j]}$.

(iii) $S^{[j+1]}$ is completely contained in the left or right half, but contains indices from both the upper and lower half of $\mathbf{A}^{[j+1]}$.

(c) $S^{[j+1]} \subseteq I_{0,\, M^{[j+1]}-1} \times I_{0,\, N^{[j]}-1}$ and $\left\{M^{[j]}-1,\, M^{[j]}\right\} \subseteq S_R^{[j+1]}$.

Then it follows from the definition of the reflected periodization that

$$\begin{aligned} S^{[j]} &= \left(\left(I_{\mu_R^{[j+1]},\, \nu_R^{[j+1]}} \cup I_{M^{[j+1]}-1-\nu_R^{[j+1]},\, M^{[j+1]}-1-\mu_R^{[j+1]}}\right) \cap I_{0,\, M^{[j]}-1}\right) \\ &\quad \times I_{\mu_C^{[j+1]},\, \nu_C^{[j+1]}} \\ &=: I_{M^{[j]}-m^{[j]},\, M^{[j]}-1} \times I_{\mu_C^{[j]},\, \nu_C^{[j]}} \\ &\subsetneq I_{M^{[j]}-m^{[j+1]},\, M^{[j]}-1} \times I_{\mu_C^{[j+1]},\, \nu_C^{[j+1]}}. \end{aligned}$$

(d) $S^{[j+1]} \subseteq I_{0,\,M^{[j+1]}-1} \times I_{N^{[j]},\,N^{[j+1]}-1}$ and $\{M^{[j]}-1,\, M^{[j]}\} \subseteq S_R^{[j+1]}$.

Then we find that

$$\begin{aligned} S^{[j]} &= \left(\left(I_{\mu_R^{[j+1]},\,\nu_R^{[j+1]}} \cup I_{M^{[j+1]}-1-\nu_R^{[j+1]},\,M^{[j+1]}-1-\mu_R^{[j+1]}}\right) \cap I_{0,\,M^{[j]}-1}\right) \\ &\quad \times I_{N^{[j+1]}-1-\nu_C^{[j+1]},\,N^{[j+1]}-1-\mu_C^{[j+1]}} \\ &=: I_{M^{[j]}-m^{[j]},\,M^{[j]}-1} \times I_{\mu_C^{[j]},\,\nu_C^{[j]}} \\ &\subsetneq I_{M^{[j]}-m^{[j+1]},\,M^{[j]}-1} \times I_{N^{[j+1]}-1-\nu_C^{[j+1]},\,N^{[j+1]}-1-\mu_C^{[j+1]}}, \end{aligned}$$

see also Figure 7.4.

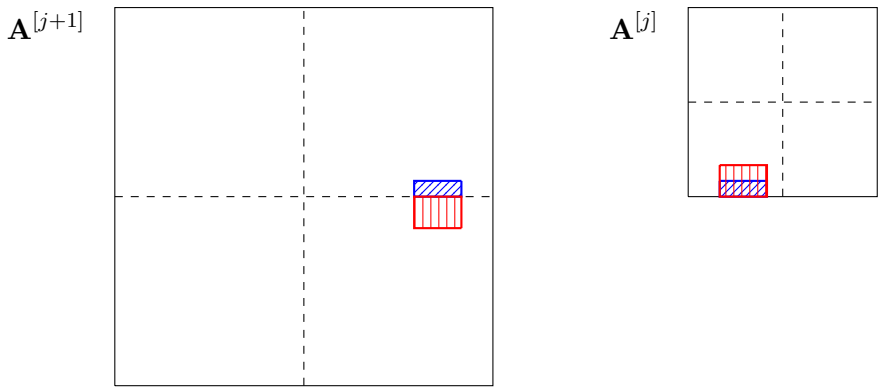

Figure 7.4: Illustration of the supports of $\mathbf{A}^{[j+1]}$ and $\mathbf{A}^{[j]}$ if $\{M^{[j]}-1,\, M^{[j]}\} \subseteq S_R^{[j+1]}$ and $S_C^{[j+1]} \subseteq I_{N^{[j]},\,N^{[j+1]}-1}$

Hence, we obtain that $\mathbf{A}^{[j]}$ also has a block support. Due to collision of possibly nonzero entries of $\mathbf{A}^{[j+1]}$ in both subcases of case (iii), the support of $\mathbf{A}^{[j]}$ has less rows than the one of $\mathbf{A}^{[j+1]}$, i.e., $m^{[j]} < m^{[j+1]}$, and the supports of $\mathbf{A}^{[j]}$ and $\mathbf{A}^{[j+1]}$ have the same number of columns, i.e., $n^{[j+1]} = n^{[j]}$.

(iv) $S^{[j+1]}$ is completely contained in one of the four quadrants of $\mathbf{A}^{[j+1]}$.

(e) $S^{[j+1]} \subseteq I_{0,\,M^{[j]}-1} \times I_{0,\,N^{[j]}-1}$.

Then it follows from Definition 7.5 that $\mathbf{A}^{[j]}$ has a block support with

$$\mathbf{A}^{[j]} = \mathbf{A}_{(0,0)}^{[j+1]} \quad \text{and} \quad S^{[j]} = S^{[j+1]}.$$

(f) $S^{[j+1]} \subseteq I_{0,\,M^{[j]}-1} \times I_{N^{[j]},\,N^{[j+1]}-1}$.

Then $\mathbf{A}^{[j]}$ has a block support with

$$\mathbf{A}^{[j]} = \mathbf{A}_{(0,1)}^{[j+1]} \mathbf{J}_{N^{[j]}} \quad \text{and} \quad S^{[j]} = I_{\mu_R^{[j+1]},\,\nu_R^{[j+1]}} \times I_{N^{[j+1]}-1-\nu_C^{[j+1]},\,N^{[j+1]}-1-\mu_C^{[j+1]}}.$$

(g) $S^{[j+1]} \subseteq I_{M^{[j]},\,M^{[j+1]}-1} \times I_{0,\,N^{[j]}-1}$.

Then $\mathbf{A}^{[j]}$ has a block support with

$$\mathbf{A}^{[j]} = \mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,0)}^{[j+1]} \quad \text{and} \quad S^{[j]} = I_{M^{[j+1]}-1-\nu_R^{[j+1]},\,M^{[j+1]}-1-\mu_R^{[j+1]}} \times I_{\mu_C^{[j+1]},\,\nu_C^{[j+1]}}.$$

(h) $S^{[j+1]} \subseteq I_{M^{[j]},\,M^{[j+1]}-1} \times I_{N^{[j]},\,N^{[j+1]}-1}$.

Then $\mathbf{A}^{[j]}$ has a block support with

$$\mathbf{A}^{[j]} = \mathbf{J}_{M^{[j]}} \mathbf{A}^{[j+1]}_{(1,1)} \mathbf{J}_{N^{[j]}} \quad \text{and}$$
$$S^{[j]} = I_{M^{[j+1]}-1-\nu_R^{[j+1]},\,M^{[j+1]}-1-\mu_R^{[j+1]}} \times I_{N^{[j+1]}-1-\nu_C^{[j+1]},\,N^{[j+1]}-1-\mu_C^{[j+1]}},$$

see also Figure 7.5.

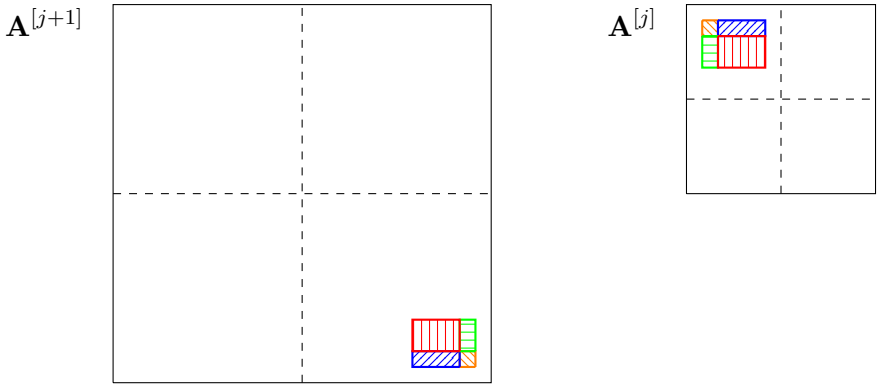

Figure 7.5: Illustration of the support of $\mathbf{A}^{[j]}$ if $S^{[j+1]} \subseteq I_{M^{[j]},\,M^{[j+1]}-1} \times I_{N^{[j]},\,N^{[j+1]}-1}$

Since the support block of $\mathbf{A}^{[j]}$ is just a reflection of the support block of $\mathbf{A}^{[j+1]}$, we have $m^{[j]} = m^{[j+1]}$ and $n^{[j]} = n^{[j+1]}$ in all four subcases of case (iv). □

7.2.2 Support Structure of $\mathbf{A}^{[j+1]}$ for Given $\mathbf{A}^{[j]}$

In our algorithm we want to recover $\mathbf{A}$ from $\mathbf{A}^{\widehat{\Pi}}$ by iteratively computing its reflected periodizations $\mathbf{A}^{[L]}, \mathbf{A}^{[L+1]}, \ldots, \mathbf{A}^{[J]} = \mathbf{A}$ for a suitable starting index L. For this we only assume that upper bounds $b_R \geq m$ and $b_C \geq n$ on the number of support rows and columns of $\mathbf{A}$ are known. If we choose $L := \max\{\lceil \log_2 b_R \rceil + 1, \lceil \log_2 b_C \rceil + 1\}$, we can guarantee correct reconstruction of the supports of all of these reflected periodizations. As in Chapter 6, we have to examine how the support of $\mathbf{A}^{[j+1]}$ looks like if $\mathbf{A}^{[j]}$ is given. Again, we will illustrate some of the possible cases for the support of $\mathbf{A}^{[j+1]}$ by the vectors from Example 7.11, before we will prove the general results.

Example 7.13 (Example 7.11 continued) Throughout this example we will again assume that the support sizes m and n of $\mathbf{A}$ are known exactly. Then it follows that $L = 2$. Additionally, we suppose that the size $M \times N = 16 \times 16$ of $\mathbf{A}$ is known.

1. We consider the reflected periodization

$$\mathbf{A}^{[2]} = \begin{pmatrix} 0 & & & 0 \\ 0 & & & 0 \\ 0 & & & 0 \\ 0 & 0 & 0 & a^{[2]}_{3,3} \end{pmatrix} \in \mathbb{R}^{4\times 4}$$

with block support $S^{[2]} = I_{3,3} \times I_{3,3}$ of size $m^{[2]} \times n^{[2]} := 1 \times 1$, which also occurs in Example 7.11.1. It follows from Definition 7.5 that $\mathbf{A}^{[3]}$ has a block support of size at

most $m \times n = 2 \times 2$, which is of the form

$$\mathbf{A}^{[3]} = \left(\begin{array}{ccc|ccc} 0 & & 0 & 0 & & 0 \\ 0 & & & & & 0 \\ 0 & & 0 & 0 & & 0 \\ 0 & 0 & a^{[3]}_{3,3} & a^{[3]}_{3,4} & 0 & 0 \\ \hline 0 & 0 & a^{[3]}_{4,3} & a^{[3]}_{4,4} & 0 & 0 \\ 0 & & 0 & 0 & & 0 \\ 0 & & & & & 0 \\ 0 & & 0 & 0 & & 0 \end{array}\right),$$

where $a^{[2]}_{3,3} = a^{[3]}_{3,3} + a^{[3]}_{3,4} + a^{[3]}_{4,3} + a^{[3]}_{4,4}$. The nonzero entries of $\mathbf{A}^{[3]}$ can be determined from $\mathbf{A}^{\widehat{\Pi}}$ using the methods we will present in Section 7.3.2. If at least one entry from each row and column of $\mathbf{A}^{[3]}$ is not zero, then $\mathbf{A}^{[3]}$ has the block support $S^{[3]} = I_{3,4} \times I_{3,4}$ of size $m^{[3]} \times n^{[3]} = 2 \times 2$.

2. Now we will consider the reflected periodization $\mathbf{A}^{[3]} \in \mathbb{R}^{8\times 8}$ with block support $S^{[3]} = I_{3,4} \times I_{3,4}$ of size $m^{[3]} \times n^{[3]} = 2 \times 2$, as it occurs in Example 7.11.1. By definition of the reflected periodization, there are precisely four possibilities for $\mathbf{A} = \mathbf{A}^{[4]}$,

$$\begin{aligned} \mathbf{A} &= \begin{pmatrix} \mathbf{A}^{[3]} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} && \text{or} \quad \mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{A}^{[3]}\mathbf{J}_8 \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \\ \text{or} \quad \mathbf{A} &= \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{J}_8\mathbf{A}^{[3]} & \mathbf{0} \end{pmatrix} && \text{or} \quad \mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_8\mathbf{A}^{[3]}\mathbf{J}_8 \end{pmatrix}, \end{aligned}$$

with the corresponding support sets

$$\begin{aligned} S^{[4]} &= S^{[3]} = I_{3,4} \times I_{3,4} && \text{or} \quad S^{[4]} = I_{3,4} \times I_{11,12} \\ \text{or} \quad S^{[4]} &= I_{11,12} \times I_{3,4} && \text{or} \quad S^{[4]} = I_{11,12} \times I_{11,12}. \end{aligned}$$

We will show in Section 7.3.1 how to determined which of these four matrices is the correct one by employing additional nonzero entries of $\mathbf{A}^{\widehat{\Pi}}$.

3. Let us consider the reflected periodization

$$\mathbf{A}^{[2]} = \begin{pmatrix} 0 & 0 & 0 & a^{[2]}_{0,3} \\ 0 & & & 0 \\ 0 & & & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{4\times 4}$$

with block support $S^{[2]} = I_{0,3}$ of size $m^{[2]} \times n^{[2]} = 1 \times 1$, as obtained in Example 7.11.2.

By Definition 7.5, there are two possibilities for the support of $\mathbf{A}^{[3]}$, namely

$$\mathbf{A}^{[3]} = \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & a^{[3]}_{0,3} & a^{[3]}_{0,4} & 0 & 0 & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline & & \mathbf{0} & & & \mathbf{0} & & \end{array}\right)$$

$$\text{or} \qquad \mathbf{A}^{[3]} = \left(\begin{array}{cccc|cccc} & & \mathbf{0} & & & \mathbf{0} & & \\ \hline 0 & & & 0 & 0 & & & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & 0 & 0 & a^{[3]}_{7,3} & a^{[3]}_{7,4} & 0 & 0 & 0 \end{array}\right),$$

with $S^{[3]} = I_{0,0} \times I_{3,4}$ or $S^{[3]} = I_{7,7} \times I_{3,4}$ of size $m^{[3]} \times n^{[3]} = 1 \times 2$. Further, we have that $a^{[2]}_{0,3} = a^{[3]}_{0,3} + a^{[3]}_{0,4}$ in the first case and $a^{[2]}_{0,3} = a^{[3]}_{7,3} + a^{[3]}_{7,4}$ in the second case. Whether the support of $\mathbf{A}^{[3]}$ is contained in its upper or lower half, and which values the nonzero entries of $\mathbf{A}^{[3]}$ take on can be determined with the help of additional nonzero entries of $\mathbf{A}^{\widehat{\Pi}}$, as we will show in Section 7.3.3.

4. Finally, we investigate the matrix

$$\mathbf{A}^{[3]} = \left(\begin{array}{cccc|cccc} & & \mathbf{0} & & & \mathbf{0} & & \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & & & 0 & 0 & & & 0 \\ 0 & 0 & 0 & a^{[3]}_{7,3} & a^{[3]}_{7,4} & 0 & 0 & 0 \end{array}\right) \in \mathbb{R}^{8\times 8}$$

with block support $S^{[3]} = I_{7,7} \times I_{3,4}$ of size $m^{[3]} \times n^{[3]} = 1 \times 2$, as it occurs in Example 7.11.2. By definition of the reflected periodization there are two possibilities for the support of $\mathbf{A} = \mathbf{A}^{[4]}$, either

$$\mathbf{A} = \mathbf{A}^{[4]} = \left(\begin{array}{cccccccc|c} 0 & \dots & & & & & \dots & 0 & \\ \vdots & & & & & & & \vdots & \mathbf{0} \\ 0 & \dots & & & & & \dots & 0 & \\ 0 & 0 & 0 & a_{7,3} & a_{7,4} & 0 & 0 & 0 & \\ \hline 0 & 0 & 0 & a_{8,3} & a_{8,4} & 0 & 0 & 0 & \\ 0 & \dots & & & & & \dots & 0 & \mathbf{0} \\ \vdots & & & & & & & \vdots & \\ 0 & \dots & & & & & \dots & 0 & \end{array}\right)$$

$$\text{or} \qquad \mathbf{A} = \mathbf{A}^{[4]} = \left(\begin{array}{c|cccccccc} & 0 & \dots & & & & \dots & 0 \\ \mathbf{0} & \vdots & & & & & & 0 \\ & 0 & \dots & & & & \dots & \vdots \\ & 0 & 0 & 0 & a_{7,11} & a_{7,12} & 0 & 0 & 0 \\ \hline & 0 & 0 & 0 & a_{8,11} & a_{8,12} & 0 & 0 & 0 \\ & 0 & \dots & & & & \dots & 0 \\ \mathbf{0} & \vdots & & & & & & \vdots \\ & 0 & \dots & & & & \dots & 0 \end{array}\right)$$

$\Diamond$

Example 7.13 shows that even for the two matrices from Example 7.11 we already have to distinguish four cases for the support of $\mathbf{A}^{[j]}$. In the following theorem we generalize the observations made above.

Theorem 7.14 *Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$, and let $J \coloneqq \min\{J_R, J_C\}$. Let $\mathbf{A} \in \mathbb{R}^{M\times N}$ have a block support of size $m\times n$ with known bounds $b_R \geq m$ and $b_C \geq n$, and assume that $\mathbf{A}$ satisfies (7.5) and (7.6). Set $L \coloneqq \max\{\lceil \log_2 b_R\rceil + 1, \lceil \log_2 b_C\rceil + 1\}$.*

A) There is at most one index $j_1 \in \{L, \dots, J\}$ such that

$$S^{[j_1]} \subseteq I_{M^{[j_1]}-b_R,\, M^{[j_1]}-1} \times I_{N^{[j_1]}-b_C,\, N^{[j_1]}-1}.$$

If $j_1 \leq J-1$, we find that

$$S^{[j_1+1]} \subsetneq I_{M^{[j_1]}-b_R,\, M^{[j_1]}+b_R-1} \times I_{N^{[j_1]}-b_C,\, N^{[j_1]}+b_C-1}.$$

B) There is at most one index $j_2 \in \{L, \dots, J\}$ such that

$$S^{[j_2]} \subseteq I_{0,\, M^{[j_2]}-1} \times I_{N^{[j_2]}-b_C,\, N^{[j_2]}-1} \quad \text{and} \quad S_R^{[j_2]} \not\subseteq I_{M^{[j_2]}-b_R,\, M^{[j_2]}-1}.$$

If $j_2 \leq J-1$, we find that

$$\begin{aligned} &S^{[j_2+1]} \subsetneq I_{\mu_R^{[j_2]},\, \nu_R^{[j_2]}} \times I_{N^{[j_2]}-b_C,\, N^{[j_2]}+b_C-1} && \text{or} \\ &S^{[j_2+1]} \subsetneq I_{M^{[j_2+1]}-1-\nu_R^{[j_2]},\, M^{[j_2+1]}-1-\mu_R^{[j_2]}} \times I_{N^{[j_2]}-b_C,\, N^{[j_2]}+b_C-1}, \end{aligned}$$

so $m^{[j_2+1]} = m^{[j_2]}$.

C) There is at most one index $j_3 \in \{L, \dots, J\}$ such that

$$S^{[j_3]} \subseteq I_{M^{[j_3]}-b_R,\, M^{[j_3]}-1} \times I_{0,\, N^{[j_3]}-1} \quad \text{and} \quad S_C^{[j_3]} \not\subseteq I_{N^{[j_3]}-b_C,\, N^{[j_3]}-1}.$$

If $j_3 \leq J-1$, we find that

$$\begin{aligned} &S^{[j_3+1]} \subsetneq I_{M^{[j_3]}-b_R,\, M^{[j_3]}+b_R-1} \times I_{\mu_C^{[j_3]},\, \nu_C^{[j_3]}} && \text{or} \\ &S^{[j_3+1]} \subsetneq I_{M^{[j_3]}-b_R,\, M^{[j_3]}+b_R-1} \times I_{N^{[j_3+1]}-1-\nu_C^{[j_3]},\, N^{[j_3+1]}-1-\mu_C^{[j_3]}}, \end{aligned}$$

so $n^{[j_3+1]} = n^{[j_3]}$.

D) If $j \in \{L, \dots, J-1\} \setminus \{j_1, j_2, j_3\}$, then either

$$\mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{A}^{[j]} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \qquad \text{or} \quad \mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{0} & \mathbf{A}^{[j]}\mathbf{J}_{N^{[j]}} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$
$$\text{or} \quad \mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j]} & \mathbf{0} \end{pmatrix} \qquad \text{or} \quad \mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{J}_{N^{[j]}} \end{pmatrix},$$

so $m^{[j+1]} = m^{[j]}$ and $n^{[j+1]} = n^{[j]}$.

Proof. Recall that $K = \max\{\lceil \log_2 m \rceil + 1, \lceil \log_2 n \rceil + 1\}$ and that $m \le b_R$ and $n \le b_C$. Consequently, $L \ge K$ and $\mathbf{A}^{[j]}$ has a block support of size $m^{[j]} \times n^{[j]}$ with $m^{[j]} \le m$ and $n^{[j]} \le n$ for all $j \in \{L, \dots, J\}$ by Lemma 7.12. We define

$$j_1 \coloneqq \max\left\{ j \in \{L, \dots, J\} : S^{[j]} \subseteq I_{M^{[j]}-b_R,\, M^{[j]}-1} \times I_{N^{[j]}-b_C,\, N^{[j]}-1} \right\}, \tag{7.7}$$

$$j_2 \coloneqq \max\Big\{ j \in \{L, \dots, J\} : S^{[j]} \subseteq I_{0,\, M^{[j]}-1} \times I_{N^{[j]}-b_C,\, N^{[j]}-1} \text{ and } S_R^{[j]} \not\subseteq I_{M^{[j]}-b_R,\, M^{[j]}-1} \Big\}, \tag{7.8}$$

$$j_3 \coloneqq \max\Big\{ j \in \{L, \dots, J\} : S^{[j]} \subseteq I_{M^{[j]}-b_R,\, M^{[j]}-1} \times I_{0,\, N^{[j]}-1} \text{ and } S_C^{[j]} \not\subseteq I_{N^{[j]}-b_C,\, N^{[j]}-1} \Big\}, \tag{7.9}$$

if such indices exist.

(i) If there is no index j_1, claim A is already proven, so we assume that there exists a $j_1 \in \{L, \dots, J\}$ satisfying (7.7). Then we find that

$$\mathbf{A}^{[j_1-1]} = \underbrace{\mathbf{A}^{[j_1]}_{(0,0)}}_{=\mathbf{0}} + \mathbf{J}_{M^{[j_1-1]}} \underbrace{\mathbf{A}^{[j_1]}_{(1,0)}}_{=\mathbf{0}} + \underbrace{\mathbf{A}^{[j_1]}_{(0,1)}}_{=\mathbf{0}} \mathbf{J}_{N^{[j_1-1]}} + \mathbf{J}_{M^{[j_1-1]}} \mathbf{A}^{[j_1]}_{(1,1)} \mathbf{J}_{N^{[j_1-1]}},$$

since by $m^{[j_1]} \le 2^{L-1}$ and $n^{[j_1]} \le 2^{L-1}$ the reflected periodization $\mathbf{A}^{[j_1]}$ can only have nonzero entries in its bottom right quadrant. Hence, we obtain that

$$S^{[j_1-1]} \subseteq I_{0,\, b_R-1} \times I_{0,\, b_C-1}$$

if $j_1 > L$. Inductively, it follows for all $j \in \{L, \dots, j_1 - 2\}$ that

$$\mathbf{A}^{[j]} = \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{J}_{M^{[j]}} \underbrace{\mathbf{A}^{[j+1]}_{(1,0)}}_{=\mathbf{0}} + \underbrace{\mathbf{A}^{[j+1]}_{(0,1)}}_{=\mathbf{0}} \mathbf{J}_{N^{[j]}} + \mathbf{J}_{M^{[j]}} \underbrace{\mathbf{A}^{[j+1]}_{(1,1)}}_{=\mathbf{0}} \mathbf{J}_{N^{[j]}},$$

and thus

$$S^{[j]} \subseteq I_{0,\, b_R-1} \times I_{0,\, b_C-1}.$$

Consequently, j_1 is the unique index satisfying (7.7). Further, for $j \in \{L, \dots, j_1 - 1\}$, the support of $\mathbf{A}^{[j]}$ is contained in the first b_R rows and first b_C columns.

Since $\mathbf{A}^{[j_1+1]}$ has a block support of size $m^{[j_1+1]} \times n^{[j_1+1]}$ with $m^{[j_1+1]} \le m$ and $n^{[j_1+1]} \le n$ by Lemma 7.12 and

$$\mathbf{A}^{[j_1]} = \mathbf{A}^{[j_1+1]}_{(0,0)} + \mathbf{J}_{M^{[j_1]}} \mathbf{A}^{[j_1+1]}_{(1,0)} + \mathbf{A}^{[j_1+1]}_{(0,1)} \mathbf{J}_{N^{[j_1]}} + \mathbf{J}_{M^{[j_1]}} \mathbf{A}^{[j_1+1]}_{(1,1)} \mathbf{J}_{N^{[j_1]}},$$

the support of $\mathbf{A}^{[j_1+1]}$ satisfies

$$S^{[j_1+1]} \subsetneq I_{M^{[j_1+1]}-b_R,\, M^{[j_1+1]}+b_R-1} \times I_{N^{[j_1+1]}-b_C,\, N^{[j_1+1]}+b_C-1}.$$

See Figure 7.6 for an illustration.

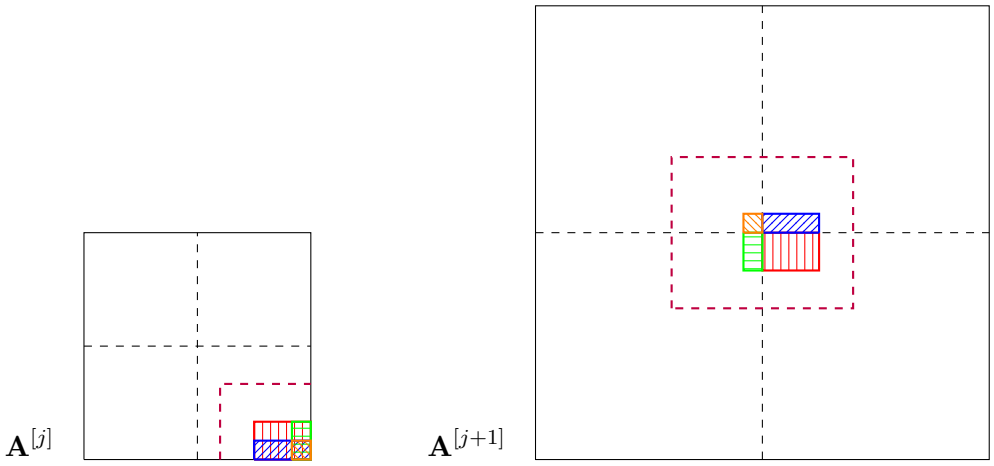

Figure 7.6: Illustration of the support of $\mathbf{A}^{[j+1]}$ if the support of $\mathbf{A}^{[j]}$ is contained in the last b_R rows and b_C columns

(ii) Assume that there exists an index $j_2 \in \{L, \dots, J\}$ as defined in (7.8). Then we have

$$\mathbf{A}^{[j_2-1]} = \underbrace{\mathbf{A}^{[j_2]}_{(0,0)}}_{=\mathbf{0}} + \mathbf{J}_{M^{[j_2-1]}} \underbrace{\mathbf{A}^{[j_2]}_{(1,0)}}_{=\mathbf{0}} + \mathbf{A}^{[j_2]}_{(0,1)} \mathbf{J}_{N^{[j_2-1]}} + \mathbf{J}_{M^{[j_2-1]}} \mathbf{A}^{[j_2]}_{(1,1)} \mathbf{J}_{N^{[j_2-1]}},$$

since by $n^{[j_2]} \leq n \leq b_C \leq 2^{L-1}$ the reflected periodization $\mathbf{A}^{[j_2]}$ cannot have nonzero entries in its left half. Hence, we find that

$$S^{[j_2-1]} \subseteq I_{0,\, M^{[j_2-1]}} \times I_{0,\, b_C-1}$$

if $j_2 > L$. It follows by induction that for all $j \in \{L, \dots, j_2 - 2\}$

$$\mathbf{A}^{[j]} = \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{J}_{M^{[j]}} \mathbf{A}^{[j+1]}_{(1,0)} + \underbrace{\mathbf{A}^{[j+1]}_{(0,1)}}_{=\mathbf{0}} \mathbf{J}_{N^{[j]}} + \mathbf{J}_{M^{[j]}} \underbrace{\mathbf{A}^{[j+1]}_{(1,1)}}_{=\mathbf{0}} \mathbf{J}_{N^{[j]}},$$

and thus

$$S^{[j]} \subseteq I_{0,\, M^{[j]}-1} \times I_{0,\, b_C-1}.$$

Consequently, j_2 is the only index that can satisfy (7.8). For $j \in \{L, \dots, j_2 - 1\}$, the support of $\mathbf{A}^{[j]}$ is contained in the first b_C columns.

Since $\mathbf{A}^{[j_2+1]}$ has a block support of size $m^{[j_2+1]} \times n^{[j_2+1]}$ with $m^{[j_2+1]} \leq m$ and $n^{[j_2+1]} \leq n$ by Lemma 7.12, Definition 7.5 implies that the support of $\mathbf{A}^{[j_2+1]}$ satisfies

$$S^{[j_2+1]} \subsetneq I_{\mu_R^{[j_2]},\, \nu_R^{[j_2]}} \times I_{N^{[j_2]}-b_C,\, N^{[j_2]}+b_C-1} \qquad \text{or}$$
$$S^{[j_2+1]} \subsetneq I_{M^{[j_2+1]}-1-\nu_R^{[j_2]},\, M^{[j_2+1]}-1-\mu_R^{[j_2]}} \times I_{N^{[j_2]}-b_C,\, N^{[j_2]}+b_C-1}.$$

This is depicted in Figure 7.7.

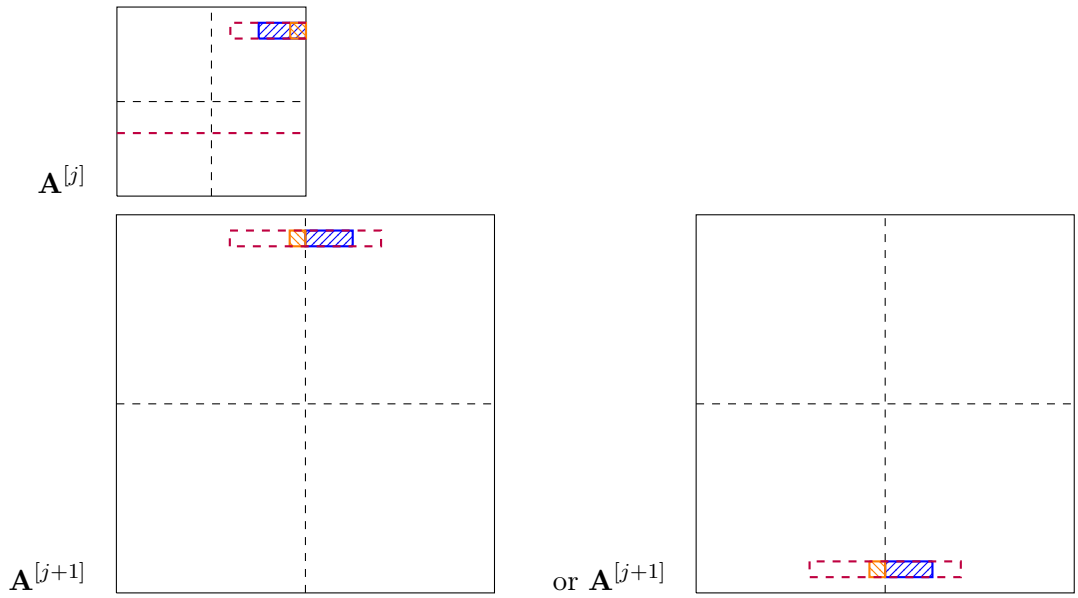

Figure 7.7: Illustration of the support of $\mathbf{A}^{[j+1]}$ if the column support of $\mathbf{A}^{[j]}$ is contained in the last b_C columns and the row support is not contained in the last b_R rows

(iii) Now we assume that there exists an index $j_3 \in \{L, \dots, J\}$ as in (7.9). Then we have

$$\mathbf{A}^{[j_3-1]} = \underbrace{\mathbf{A}^{[j_3]}_{(0,0)}}_{=\mathbf{0}} + \mathbf{J}_{M^{[j_3-1]}} \mathbf{A}^{[j_3]}_{(1,0)} + \underbrace{\mathbf{A}^{[j_3]}_{(0,1)}}_{=\mathbf{0}} \mathbf{J}_{N^{[j_3-1]}} + \mathbf{J}_{M^{[j_3-1]}} \mathbf{A}^{[j_3]}_{(1,1)} \mathbf{J}_{N^{[j_3-1]}},$$

since by $m^{[j_3]} \leq m \leq b_R \leq 2^{L-1}$ the reflected periodization $\mathbf{A}^{[j_3]}$ cannot have nonzero entries in its upper half. Hence, we find that

$$S^{[j_3-1]} \subseteq I_{0,\, b_R-1} \times I_{0,\, N^{[j_3-1]}}$$

if $j_3 > L$. Again, it follows from induction that for all $j \in \{L, \dots, j_3 - 2\}$

$$\mathbf{A}^{[j]} = \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{J}_{M^{[j]}} \underbrace{\mathbf{A}^{[j+1]}_{(1,0)}}_{=\mathbf{0}} + \mathbf{A}^{[j+1]}_{(0,1)} \mathbf{J}_{N^{[j]}} + \mathbf{J}_{M^{[j]}} \underbrace{\mathbf{A}^{[j+1]}_{(1,1)}}_{=\mathbf{0}} \mathbf{J}_{N^{[j]}},$$

and thus

$$S^{[j]} \subseteq I_{0,\, b_R-1} \times I_{0,\, N^{[j]}-1}.$$

Consequently, j_3 is the only index for which (7.9) holds. For $j \in \{L, \dots, j_3 - 1\}$, the support of $\mathbf{A}^{[j]}$ is contained in the first b_R rows.

Since $\mathbf{A}^{[j_3+1]}$ has a block support of size $m^{[j_3+1]} \times n^{[j_3+1]}$ with $m^{[j_3+1]} \leq m$ and $n^{[j_3+1]} \leq n$ by Lemma 7.12, the definition of the reflected periodization yields that the

support of $\mathbf{A}^{[j_3+1]}$ satisfies

$$S^{[j_3+1]} \subsetneq I_{M^{[j_3]}-b_R,\, M^{[j_3]}+b_R-1} \times I_{\mu_C^{[j_3]},\, \nu_C^{[j_3]}} \qquad \text{or}$$
$$S^{[j_3+1]} \subsetneq I_{M^{[j_3]}-b_R,\, M^{[j_3]}+b_R-1} \times I_{N^{[j_3+1]}-1-\nu_C^{[j_3]},\, N^{[j_3+1]}-1-\mu_C^{[j_3]}}.$$

See Figure 7.8 for a visualization.

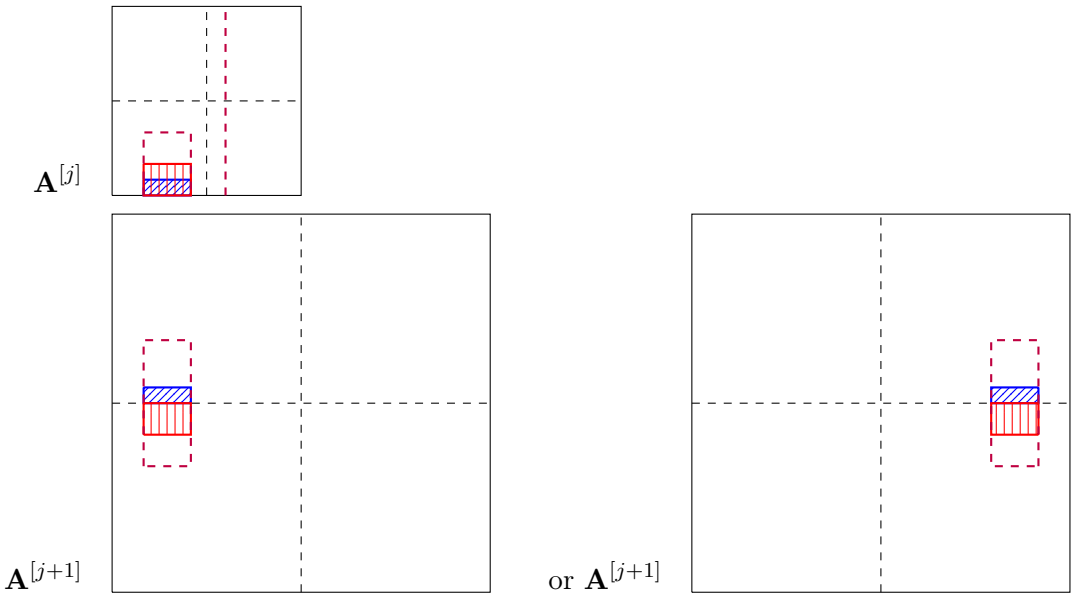

Figure 7.8: Illustration of the support of $\mathbf{A}^{[j+1]}$ if the row support of $\mathbf{A}^{[j]}$ is contained in the last b_R rows and the column support is not contained in the last b_C columns

(iv) For $j \in \{L, \dots, J-1\} \setminus \{j_1, j_2, j_3\}$ we show first that $m^{[j+1]} = m^{[j]}$ and $n^{[j+1]} = n^{[j]}$. The proof of Lemma 7.12 yields that $m^{[j+1]} \geq m^{[j]}$ and $n^{[j+1]} \geq n^{[j]}$. Let us assume that there exists an index $j' \in \{L, \dots, J-1\} \setminus \{j_1, j_2, j_3\}$ with $m^{[j'+1]} > m^{[j']}$ or $n^{[j'+1]} > n^{[j']}$. If $m^{[j'+1]} > m^{[j']}$, then

$$\left\{ M^{[j']} - 1, M^{[j']} \right\} \subseteq S_R^{[j'+1]},$$

as we have to be in case (i) or (iii) of the proof of Lemma 7.12. Since $m^{[j'+1]} \leq m \leq b_R$, this implies that

$$S_R^{[j'+1]} \subsetneq I_{M^{[j']}-b_R,\, M^{[j']}+b_R-1}.$$

Consequently, it follows from Definition 7.5 that

$$S_R^{[j']} \subseteq I_{M^{[j']}-b_R,\, M^{[j']}-1}. \tag{7.10}$$

Depending on the column support, either j_1 or j_3, which are given by (7.7) and (7.9), is the unique index satisfying (7.10). Thus, we obtain that $j' = j_1$ or $j' = j_3$, which is a contradiction to the choice of j'. Hence, we showed that $m^{[j+1]} = m^{[j]}$.

Analogously, if $n^{[j'+1]} > n^{[j']}$, then we have to be in case (i) or (ii) of the proof of

Lemma 7.12, so

$$\left\{N^{[j']}-1, N^{[j']}\right\} \subseteq S_C^{[j'+1]}.$$

It follows from $n^{[j'+1]} \leq n \leq b_C$ that

$$S_C^{[j'+1]} \subsetneq I_{N^{[j']}-b_C,\, N^{[j']}+b_C-1},$$

and thus the definition of the reflected periodization implies that

$$S_C^{[j']} \subseteq I_{N^{[j']}-b_C,\, N^{[j']}-1}. \tag{7.11}$$

Since, depending on the row support, either j_1 or j_2, as defined in (7.7) and (7.8), is the unique index satisfying (7.11), we obtain that $j' = j_1$ or $j' = j_2$. This is a contradiction to the choice of j', so we proved that $n^{[j+1]} = n^{[j]}$.

As $m^{[j+1]} = m^{[j]}$ and $n^{[j+1]} = n^{[j]}$, Definition 7.5 and Lemma 7.12 yield that there exist precisely four matrices of size $M^{[j+1]} \times N^{[j+1]}$ arising from repeatedly applying the reflected periodization to the matrix $\mathbf{A} \in \mathbb{R}^{M\times N}$ with block support that have the given reflected periodization $\mathbf{A}^{[j]}$, namely

$$\begin{aligned}
&\mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{A}^{[j]} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} && \text{or} \quad \mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{0} & \mathbf{A}^{[j]}\mathbf{J}_{N^{[j]}} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \\
\text{or} \quad &\mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j]} & \mathbf{0} \end{pmatrix} && \text{or} \quad \mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{J}_{N^{[j]}} \end{pmatrix}.
\end{aligned}$$

This is depicted in Figure 7.9. □

7.3 Iterative Sparse 2D Recovery Procedures

If we aim to recover a matrix $\mathbf{A} \in \mathbb{R}^{M\times N}$, $M = 2^{J_R}$, $N = 2^{J_C}$, with block support of size $m \times n$ from its DCT-II transformed matrix $\mathbf{A}^{\widehat{\mathrm{II}}}$ by iteratively computing its reflected periodizations $\mathbf{A}^{[L]}, \mathbf{A}^{[L+1]}, \ldots, \mathbf{A}^{[J]} = \mathbf{A}$, then we need to develop recovery procedures for the four cases of Theorem 7.14. All of these methods will require a priori knowledge of upper bounds $b_R \geq m$ and $b_C \geq n$ on the number of support rows and columns of $\mathbf{A}$.

We will first present a recovery procedure for case D of Theorem 7.14 and then one for case A. Afterwards, we will investigate cases B and C, which are closely related, as we can utilize some techniques for them that are easier to introduce for cases D and A.

7.3.1 Recovery Procedure for Case D: No Collision

Let us begin by deriving a recovery procedure for case D of Theorem 7.14. Hence, we assume that $j \in \{L, \ldots, J-1\} \setminus \{j_1, j_2, j_3\}$, with j_1, j_2 and j_3 given by (7.7) to (7.9). Then the support of $\mathbf{A}^{[j]}$ is not contained in its last b_R rows and last b_C columns. We know from Theorem 7.14, case D that the values of the nonzero entries of $\mathbf{A}^{[j]}$ and $\mathbf{A}^{[j+1]}$ are the same, with

$$\begin{aligned}
&\mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{A}^{[j]} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} && \text{or} \quad \mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{0} & \mathbf{A}^{[j]}\mathbf{J}_{N^{[j]}} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \\
\text{or} \quad &\mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j]} & \mathbf{0} \end{pmatrix} && \text{or} \quad \mathbf{A}^{[j+1]} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{J}_{N^{[j]}} \end{pmatrix},
\end{aligned}$$

see also Figure 7.9.

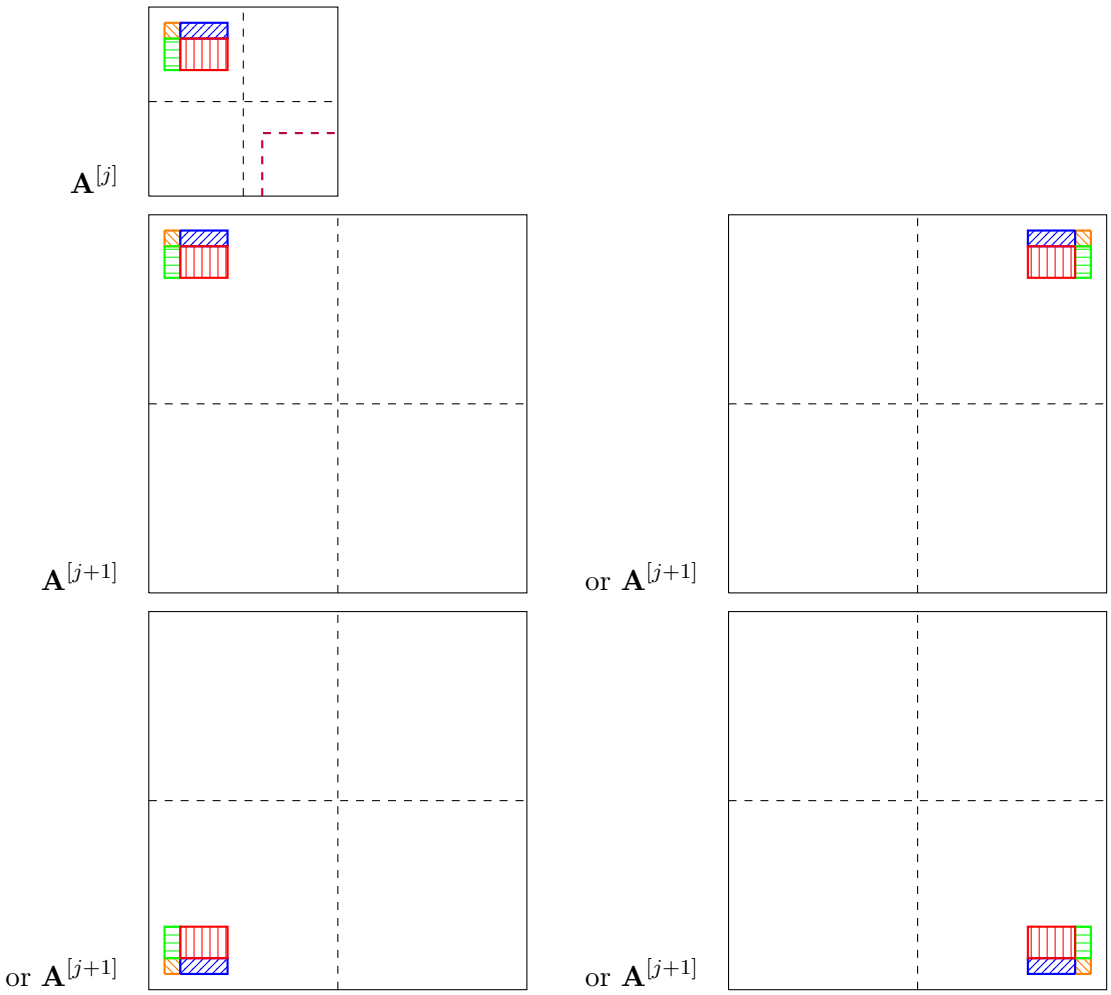

Figure 7.9: Illustration of the support of $\mathbf{A}^{[j+1]}$ if the support of $\mathbf{A}^{[j]}$ is not contained in the last b_R rows and the last b_C columns

Similarly to the procedure for case B of Theorem 6.12 in Chapter 6, it is possible to determine which of these matrices is the correct one by using one nonzero entry of $\left(\left(a^{[j+1]}\right)^{\widehat{\Pi}}_{2k,\,2l+1} \right)_{k,\,l=0}^{m^{[j]}-1,\,n^{[j]}-1}$ and one of $\left(\left(a^{[j+1]}\right)^{\widehat{\Pi}}_{2k+1,\,2l} \right)_{k,\,l=0}^{m^{[j]}-1,\,n^{[j]}-1}$.

Thus, we first show that such nonzero entries exist and can be found efficiently. Analogously to Section 6.3.2, this can be done with the help of odd Vandermonde matrices.

Lemma 7.15 Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with J_R, $J_C \in \mathbb{N}$, and let $J \coloneqq \min\{J_R, J_C\}$. Let $\mathbf{A} \in \mathbb{R}^{M \times N}$ have an unknown block support of size $m \times n$ with known bounds $b_R \geq m$ and $b_C \geq n$, and assume that $\mathbf{A}$ satisfies the non-cancellation conditions (7.5) and (7.6). Let $L \coloneqq \max\{\lceil \log_2 b_R \rceil + 1, \lceil \log_2 b_C \rceil + 1\}$ and $j \in \{L, \ldots, J-1\} \setminus \{j_1, j_2, j_3\}$ with j_1, j_2 and j_3 as in (7.7), (7.8) and (7.9). Suppose that we have access to all entries of $\mathbf{A}^{\widehat{\Pi}}$.

Then the even-odd partial matrix and the odd-even partial matrix,

$$\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1}\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0} \quad \text{and} \quad \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0},$$

of $\left(\mathbf{A}^{[j+1]}\right)^{\widehat{\mathrm{II}}}$ both have at least one nonzero entry.

Proof. The claims can be shown by adapting the proof of Lemma 6.15 (Lemma 3.2 in [BP18a]) to the 2-dimensional setting. Let $\mathbf{A}^{[j]}$ have the block support $S^{[j]} = S_R^{[j]} \times S_C^{[j]}$.

(i) It follows from (7.1) in Remark 7.7 that

$$\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} = \frac{1}{2}\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(0,1)}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}}, \tag{7.12}$$

where

$$\tilde{\mathbf{A}}^{[j+1]}_{(0,1)} \coloneqq \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} - \mathbf{A}^{[j+1]}_{(0,1)}\mathbf{J}_{N^{[j]}} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\mathbf{J}_{N^{[j]}}. \tag{7.13}$$

If we denote by $S(\mathbf{B})$ the block support of a matrix $\mathbf{B} \in M^{[j]} \times N^{[j]}$, the definition of $\tilde{\mathbf{A}}^{[j+1]}_{(0,1)} = \left(\left(\tilde{a}^{[j+1]}_{(0,1)}\right)_{k,\,l}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0}$ yields that

$$S\left(\tilde{\mathbf{A}}^{[j+1]}_{(0,1)}\right) \subseteq S^{[j]} = I_{\mu_R^{[j]},\,\nu_R^{[j]}} \times I_{\mu_C^{[j]},\,\nu_C^{[j]}}.$$

Since the block support $S^{[j]}$ of $\mathbf{A}^{[j]}$ is of size $m^{[j]} \times n^{[j]}$, we can restrict (7.12) to the equations corresponding to the first $m^{[j]}$ rows and the first $n^{[j]}$ columns, obtaining

$$\begin{aligned}
&\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1}\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0} \\
&= \frac{1}{2}\left(\left(\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\right)_{k,\,r}\right)^{m^{[j]}-1,\,M^{[j]}-1}_{k,\,r=0} \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} \left(\left(\mathbf{C}^{\mathrm{IV}}_{N^{[j]}}\right)_{s,\,l}\right)^{N^{[j]}-1,\,n^{[j]}-1}_{s,\,l=0} \\
&= \frac{1}{\sqrt{M^{[j]}N^{[j]}}}\left(\sum_{r=\mu_R^{[j]}}^{\nu_R^{[j]}}\sum_{s=\mu_C^{[j]}}^{\nu_C^{[j]}} \varepsilon_{M^{[j]}}(k)\cos\left(\frac{k(2r+1)\pi}{2M^{[j]}}\right)\left(\tilde{a}^{[j+1]}_{(0,1)}\right)_{r,\,s}\right. \\
&\quad \left.\cdot\cos\left(\frac{(2s+1)(2l+1)\pi}{4N^{[j]}}\right)\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0} \\
&=: \frac{1}{\sqrt{M^{[j]}N^{[j]}}}\operatorname{diag}\left(\left(\varepsilon_{M^{[j]}}(k)\right)_{k=0}^{m^{[j]}-1}\right)\mathbf{T}^{[j]}_{\mathrm{II},\,R}\left(\left(\tilde{a}^{[j+1]}_{(0,1)}\right)_{r,\,s}\right)^{\nu_R^{[j]},\,\nu_C^{[j]}}_{r=\mu_R^{[j]},\,s=\mu_C^{[j]}}\left(\mathbf{T}^{[j]}_{\mathrm{IV},\,C}\right)^{T},
\end{aligned} \tag{7.14}$$

where

$$\mathbf{T}^{[j]}_{\mathrm{II},\,R} \coloneqq \left(\cos\left(\frac{k(2r+1)\pi}{2M^{[j]}}\right)\right)^{m^{[j]}-1,\,\nu_R^{[j]}-1}_{k=0,\,r=\mu_R^{[j]}}$$

is the restriction of the cosine matrix of type II without the normalization factors to the

first $m^{[j]}$ rows and the $m^{[j]}$ columns indexed by the *row support* $S_R^{[j]}$ of $\mathbf{A}^{[j]}$, and

$$\mathbf{T}_{\mathrm{IV},\,C}^{[j]}{}^{T} \coloneqq \left(\cos\left(\frac{(2s+1)(2l+1)\pi}{4N^{[j]}}\right)\right)_{s=\mu_C^{[j]},\,l=0}^{\nu_C^{[j]},\,n^{[j]}-1}$$

is the restriction of the cosine matrix of type IV without the normalization factors to the $n^{[j]}$ rows indexed by the *column support* $S_C^{[j]}$ of $\mathbf{A}^{[j]}$ and the first $n^{[j]}$ rows.

Our next goal is to show that $T_{\mathrm{II},\,R}^{[j]}$ and $T_{\mathrm{IV},\,C}^{[j]}$ are invertible. Both of these claims can be proven by employing Chebyshev polynomials. Recall that by Lemma 4.14 (v) the Chebyshev polynomial of the first kind of degree n can be written as

$$T_n(x) = \cos(n \arccos x) = \sum_{l=0}^{n} \alpha_{n,\,l}\, x^l$$

for $x \in \mathbb{R}$ with $|x| \leq 1$ and $n \in \mathbb{N}_0$. Further, Lemma 4.14 (vi) yields that

$$T_k\left(t_{n,\,l}\right) = \cos\left(\frac{k(2l+1)\pi}{2n}\right) \qquad \forall\, l \in \{0,\ldots,n-1\},\ k \in \mathbb{N}_0.$$

Combining this with the definition of $\mathbf{T}_{\mathrm{II},\,R}^{[j]}$ and the coefficient representation of the Chebyshev polynomials, and setting $\alpha_{k,\,i} \coloneqq 0$ for all $i \in \{k+1,\ldots,m^{[j]}-1\}$, we find

$$\begin{aligned}
\mathbf{T}_{\mathrm{II},\,R}^{[j]} &= \left(T_k\left(t_{M^{[j]},\,r}\right)\right)_{k=0,\,r=\mu_R^{[j]}}^{m^{[j]}-1,\,\nu_R^{[j]}} \\
&= \left(\sum_{i=0}^{k} \alpha_{k,\,i} \cdot {t_{M^{[j]},\,r}}^{i}\right)_{k=0,\,r=\mu_R^{[j]}}^{m^{[j]}-1,\,\nu_R^{[j]}} \\
&= (\alpha_{k,\,i})_{k,\,i=0}^{m^{[j]}-1} \cdot \left({t_{M^{[j]},\,r}}^{i}\right)_{i=0,\,r=\mu_R^{[j]}}^{m^{[j]}-1,\,\nu_R^{[j]}} \\
&= \begin{pmatrix}
\alpha_{0,\,0} & 0 & 0 & \ldots & 0 \\
\alpha_{1,\,0} & \alpha_{1,\,1} & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & & \vdots \\
\alpha_{m^{[j]}-1,\,0} & \alpha_{m^{[j]}-1,\,1} & \alpha_{m^{[j]}-1,\,2} & \cdots & \alpha_{m^{[j]}-1,\,m^{[j]}-1}
\end{pmatrix}
\begin{pmatrix}
(1)_{r\in S_R^{[j]}}^{T} \\
\left(t_{M^{[j]},\,r}\right)_{r\in S_R^{[j]}}^{T} \\
\vdots \\
\left({t_{M^{[j]},\,r}}^{m^{[j]}-1}\right)_{r\in S_R^{[j]}}^{T}
\end{pmatrix} \\
&\eqqcolon \mathbf{X}_R^{[j]} \cdot \mathbf{V}\left(\left(t_{M^{[j]},\,r}\right)_{r\in S_R^{[j]}}\right)^{T}. \qquad (7.15)
\end{aligned}$$

The diagonal matrix $\mathbf{X}_R^{[j]}$ is invertible by (6.20). Further, we have that

$$\frac{(2r+1)\pi}{2M^{[j]}} \in (0,\pi)$$

for all $r \in S_R^{[j]} \subseteq I_{0,\,M^{[j]}-1}$. Hence, we obtain that

$$t_{M^{[j]},\,r} = \cos\left(\frac{(2r+1)\pi}{2M^{[j]}}\right) \in (-1,1),$$

with $t_{M^{[j]},\,r} \neq t_{M^{[j]},\,s}$ for $r \neq s$, $r,s \in S_R^{[j]}$, as the cosine is bijective on $(0,\pi)$. Consequently, $\mathbf{V}\left(\left(t_{M^{[j]},\,r}\right)_{r\in S_R^{[j]}}\right)$ is invertible by Lemma 4.11, and (7.15) yields that $\mathbf{T}^{[j]}_{\mathrm{II},\,R}$ can also be inverted.

We already proved in Section 6.3.2, (6.21), that

$$\begin{aligned}
\mathbf{T}^{[j]}_{\mathrm{IV},\,C} &= \left(T_{2l+1}\left(t_{N^{[j+1]},\,s}\right)\right)_{l=0,\,s\in S_C^{[j]}}^{n^{[j]}-1} \\
&= \left(\alpha_{2l+1,\,2r+1}\right)_{l,\,r=0}^{n^{[j]}-1}\left(t_{N^{[j+1]},\,s}{}^{2i+1}\right)_{i=0,\,s\in S_C^{[j]}}^{n^{[j]}-1} \\
&=: \mathbf{X}^{[j]}_{\mathrm{odd},\,C}\cdot\mathbf{V}^{\mathrm{odd}}\left(\left(t_{N^{[j+1]},\,s}\right)_{s\in S_C^{[j]}}\right)^T. \qquad (7.16)
\end{aligned}$$

Since $S_C^{[j]} \subseteq I_{0,\,N^{[j]}-1}$, both matrices in (7.16) and thus $T^{[j]}_{\mathrm{IV},\,C}$ are invertible.

Let us now assume that all entries of $\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1}\right)_{k,\,l=0}^{m^{[j]}-1,\,n^{[j]}-1}$ are zero. Then we obtain from (7.14), (7.15) and (7.16) that

$$\begin{aligned}
\mathbf{0} &= \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1}\right)_{k,\,l=0}^{m^{[j]}-1,\,n^{[j]}-1} \\
&= \frac{1}{\sqrt{M^{[j]}N^{[j]}}}\operatorname{diag}\left(\left(\varepsilon_{M^{[j]}}(k)\right)_{k=0}^{m^{[j]}-1}\right)\cdot\mathbf{X}^{[j]}_R\cdot\mathbf{V}\left(\left(t_{M^{[j]},\,r}\right)_{r\in S_R^{[j]}}\right)^T \\
&\quad\cdot\left(\left(\tilde{a}^{[j+1]}_{(0,1)}\right)_{r,\,s}\right)_{r\in S_R^{[j]},\,s\in S_C^{[j]}}\cdot\mathbf{V}^{\mathrm{odd}}\left(\left(t_{N^{[j+1]},\,s}\right)_{s\in S_C^{[j]}}\right)\cdot{\mathbf{X}^{[j]}_{\mathrm{odd},\,C}}^T \\
\Leftrightarrow\quad \mathbf{0} &= \left(\left(\tilde{a}^{[j+1]}_{(0,1)}\right)_{r,\,s}\right)_{r\in S_R^{[j]},\,s\in S_C^{[j]}}, \qquad (7.17)
\end{aligned}$$

since all other matrices are invertible. However, we have $j \in \{L,\ldots,J-1\}\setminus\{j_1,j_2,j_3\}$, i.e., only one quadrant of $\mathbf{A}^{[j+1]}$ contains nonzero entries. Theorem 7.14, case D and (7.13) yield that $\tilde{\mathbf{A}}^{[j+1]}_{(0,1)} = \mathbf{A}^{[j]}$ or $\tilde{\mathbf{A}}^{[j+1]}_{(0,1)} = -\mathbf{A}^{[j]}$. Consequently, (7.17) is only possible if $\mathbf{A}^{[j]} = \mathbf{0}$, which is a contradiction to (7.5) and (7.6) and the fact that $\mathbf{A} \neq \mathbf{0}$ has a block support of size $m\times n$. Thus, there exists an index pair $\left(k_{(0,1)},\,l_{(0,1)}\right) \in I_{0,\,m^{[j]}-1}\times I_{0,\,n^{[j]}-1}$ such that $\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)}+1} \neq 0$.

(ii) For $\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)_{k,\,l=0}^{m^{[j]}-1,\,n^{[j]}-1}$ we can proceed analogously. Equation (7.1) in Remark 7.7 yields that

$$\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)_{k,\,l=0}^{m^{[j]}-1,\,n^{[j]}-1} = \frac{1}{2}\mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(1,0)}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^T, \qquad (7.18)$$

where

$$\tilde{\mathbf{A}}^{[j+1]}_{(1,0)} := \mathbf{A}^{[j+1]}_{(0,0)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} + \left(\mathbf{A}^{[j+1]}_{(0,1)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}. \tag{7.19}$$

We restrict (7.18) to the first $m^{[j]}$ rows and the first $n^{[j]}$ columns as well, obtaining

$$\begin{aligned}
&\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0}\\
&= \frac{1}{\sqrt{M^{[j]}N^{[j]}}}\left(\sum_{r=\mu_R^{[j]}}^{\nu_R^{[j]}}\sum_{s=\mu_C^{[j]}}^{\nu_C^{[j]}}\cos\left(\frac{(2k+1)(2r+1)\pi}{4M^{[j]}}\right)\left(\tilde{a}^{[j+1]}_{(1,0)}\right)_{r,\,s}\right.\\
&\quad\left.\cdot\,\varepsilon_{N^{[j]}}(l)\cos\left(\frac{(2s+1)l\pi}{2N^{[j]}}\right)\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0}\\
&=: \frac{1}{\sqrt{M^{[j]}N^{[j]}}}\mathbf{T}^{[j]}_{\mathrm{IV},\,R}\left(\left(\tilde{a}^{[j+1]}_{(1,0)}\right)_{r,\,s}\right)^{\nu_R^{[j]},\,\nu_C^{[j]}}_{r=\mu_R^{[j]},\,s=\mu_C^{[j]}}{\mathbf{T}^{[j]}_{\mathrm{II},\,C}}^{T}\operatorname{diag}\left(\left(\varepsilon_{N^{[j]}}(l)\right)_{l=0}^{n^{[j]}-1}\right),
\end{aligned} \tag{7.20}$$

where

$${\mathbf{T}^{[j]}_{\mathrm{II},\,C}}^{T} := \left(\cos\left(\frac{(2s+1)l\pi}{2N^{[j]}}\right)\right)^{\nu_C^{[j]}-1,\,n^{[j]}-1}_{s=\mu_C^{[j]},\,l=0}$$

is the restriction of the transposed cosine matrix of type II without the normalization factors to the $n^{[j]}$ rows indexed by the *column support* $S_C^{[j]}$ of $\mathbf{A}^{[j]}$ and the first $n^{[j]}$ columns. The matrix

$$\mathbf{T}^{[j]}_{\mathrm{IV},\,R} := \left(\cos\left(\frac{(2k+1)(2r+1)\pi}{4M^{[j]}}\right)\right)^{m^{[j]}-1,\,\nu_R^{[j]}}_{k=0,\,r=\mu_R^{[j]}}$$

is the restriction of the cosine matrix of type IV without the normalization factors to the first $m^{[j]}$ rows and the $m^{[j]}$ columns indexed by the *row support* $S_R^{[j]}$ of the reflected periodization $\mathbf{A}^{[j]}$. Similar to part (i) of the proof we obtain the factorizations

$$\begin{aligned}
\mathbf{T}^{[j]}_{\mathrm{II},\,C} &= (\alpha_{l,\,i})^{n^{[j]}-1}_{l,\,i=0}\cdot\left({t_{N^{[j]},\,s}}^{i}\right)^{n^{[j]}-1,\,\nu_C^{[j]}}_{i=0,\,s=\mu_C^{[j]}}\\
&=: \mathbf{X}^{[j]}_C\cdot\mathbf{V}\left(\left(t_{N^{[j]},\,s}\right)_{s\in S_C^{[j]}}\right)^T.
\end{aligned} \tag{7.21}$$

and

$$\begin{aligned}
\mathbf{T}^{[j]}_{\mathrm{IV},\,R} &= (\alpha_{2k+1,\,2i+1})^{m^{[j]}-1}_{k,\,i=0}\cdot\left({t_{M^{[j+1]},\,r}}^{2i+1}\right)^{m^{[j]}-1}_{i=0,\,r\in S_R^{[j]}}\\
&=: \mathbf{X}^{[j]}_{\mathrm{odd},\,R}\cdot\mathbf{V}^{\mathrm{odd}}\left(\left(t_{M^{[j+1]},\,r}\right)_{r\in S_R^{[j]}}\right)^T,
\end{aligned} \tag{7.22}$$

By analogous arguments as in part (i) of the proof, $\mathbf{T}^{[j]}_{\mathrm{IV},\,R}$ and $\mathbf{T}^{[j]}_{\mathrm{II},\,C}$ are invertible.

Let us now assume that $\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0}$ has no nonzero entries. Then

(7.20), (7.21) and (7.22) yield that

$$\begin{aligned}
\mathbf{0} &= \left(\left(a^{[j+1]}\right)^{\widehat{\Pi}}_{2k+1,\,2l}\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0} \\
&= \frac{1}{\sqrt{M^{[j]}N^{[j]}}}\,\mathbf{X}^{[j]}_{\mathrm{odd},\,R}\cdot\mathbf{V}^{\mathrm{odd}}\left(\left(t_{M^{[j+1]},\,r}\right)_{r\in S^{[j]}_R}\right)^T \\
&\quad\cdot\left(\left(\tilde{a}^{[j+1]}_{(1,0)}\right)_{r,\,s}\right)_{r\in S^{[j]}_R,\,s\in S^{[j]}_C}\mathbf{V}\left(\left(t_{N^{[j]},\,s}\right)_{s\in S^{[j]}_C}\right)\mathbf{X}^{[j]\,T}_{C}\,\mathrm{diag}\left(\left(\varepsilon_{N^{[j]}}(l)\right)_{l=0}^{n^{[j]}-1}\right) \\
\Leftrightarrow\quad \mathbf{0} &= \left(\left(\tilde{a}^{[j+1]}_{(1,0)}\right)_{r,\,s}\right)_{r\in S^{[j]}_R,\,s\in S^{[j]}_C}. \qquad (7.23)
\end{aligned}$$

However, we are in the case where $j \in \{L,\dots,J-1\}\setminus\{j_1,j_2,j_3\}$, so $\tilde{\mathbf{A}}^{[j+1]}_{(1,0)} = \mathbf{A}^{[j]}$ or $\tilde{\mathbf{A}}^{[j+1]}_{(1,0)} = -\mathbf{A}^{[j]}$ by (7.19). Consequently, (7.23) is only possible if $\mathbf{A}^{[j]} = \mathbf{0}$, which is a contradiction to (7.5) and (7.6) and the fact that $\mathbf{A} \neq \mathbf{0}$ has a block support of size $m \times n$. Thus, there exists an index pair $\left(k_{(1,0)},\, l_{(1,0)}\right) \in I_{0,\,m^{[j]}-1} \times I_{0,\,n^{[j]}-1}$ with $\left(a^{[j+1]}\right)^{\widehat{\Pi}}_{2k_{(1,0)}+1,\,2l_{(1,0)}} \neq 0$. □

Remark 7.16 For an efficient and stable implementation of the recovery procedure for case D of Theorem 7.14 using Lemma 7.8 we set

$$\left(k_{(0,1)},\, l_{(0,1)}\right) := \operatorname*{argmax}_{(k,\,l)\in I_{0,\,m^{[j]}-1}\times I_{0,\,n^{[j]}-1}} \left\{\left|2^{J-j-1}a^{\widehat{\Pi}}_{2^{J-j}k,\,2^{J-j-1}(2l+1)}\right|\right\}$$

and

$$\left(k_{(1,0)},\, l_{(1,0)}\right) := \operatorname*{argmax}_{(k,\,l)\in I_{0,\,m^{[j]}-1}\times I_{0,\,n^{[j]}-1}} \left\{\left|2^{J-j-1}a^{\widehat{\Pi}}_{2^{J-j-1}(2k+1),\,2^{J-j}l}\right|\right\}.$$

◊

The following theorem, a 2-dimensional analog to Theorem 5.23, shows how $\mathbf{A}^{[j+1]}$ can be recovered from $\mathbf{A}^{[j]}$ and two nonzero entries of $\left(\mathbf{A}^{[j+1]}\right)^{\widehat{\Pi}}$.

Theorem 7.17 *Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$, and let $J := \min\{J_R, J_C\}$. Let $\mathbf{A} \in \mathbb{R}^{M\times N}$ have a block support of size $m\times n$ with known bounds $b_R \geq m$ and $b_C \geq n$, and assume that $\mathbf{A}$ satisfies (7.5) and (7.6). Let $L := \max\{\lceil\log_2 b_R\rceil + 1, \lceil\log_2 b_C\rceil + 1\}$ and $j \in \{L,\dots,J-1\}\setminus\{j_1,j_2,j_3\}$ with j_1, j_2 and j_3 as in (7.7), (7.8) and (7.9). Suppose that we have access to all entries of $\mathbf{A}^{\widehat{\Pi}}$. Then $\mathbf{A}^{[j+1]}$ can be recovered from $\mathbf{A}^{[j]}$, one nonzero entry of $\left(a^{[j+1]}_{2k,\,2l+1}\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0}$ and one nonzero entry of $\left(a^{[j+1]}_{2k+1,\,2l}\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0}$.*

Proof. Let $\mathbf{A}^{[j]}$ have the block support $S^{[j]} = S^{[j]}_R \times S^{[j]}_C$ of size $m^{[j]} \times n^{[j]}$. It follows from Theorem 7.14, case D that there are four matrices of size $M^{[j+1]} \times N^{[j+1]}$ arising from repeatedly applying the reflected periodization to the block sparse matrix $\mathbf{A}$ that can have the given reflected periodization $\mathbf{A}^{[j]}$, namely

$$\mathbf{U}^{(0,0)} = \begin{pmatrix}\mathbf{A}^{[j]} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}, \qquad \mathbf{U}^{(0,1)} = \begin{pmatrix}\mathbf{0} & \mathbf{A}^{[j]}\mathbf{J}_{N^{[j]}}\\ \mathbf{0} & \mathbf{0}\end{pmatrix},$$

$$\mathbf{U}^{(1,0)} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j]} & \mathbf{0} \end{pmatrix}, \qquad \mathbf{U}^{(1,1)} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{J}_{N^{[j]}} \end{pmatrix}.$$

We compare the DCT-IIs of $\mathbf{U}^{(0,0)}$, $\mathbf{U}^{(0,1)}$, $\mathbf{U}^{(1,0)}$ and $\mathbf{U}^{(1,1)}$. Remark 7.7 yields that

$$\mathbf{P}_{M^{[j+1]}} \left(\mathbf{U}^{(0,0)}\right)^{\widehat{\mathrm{II}}} \mathbf{P}_{N^{[j+1]}}{}^{T} = \frac{1}{2} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T} & \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ \mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\mathbf{A}^{[j]}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T} & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix}, \tag{7.24}$$

$$\mathbf{P}_{M^{[j+1]}} \left(\mathbf{U}^{(0,1)}\right)^{\widehat{\mathrm{II}}} \mathbf{P}_{N^{[j+1]}}{}^{T} = \frac{1}{2} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T} & -\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ \mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\mathbf{A}^{[j]}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T} & -\mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix},$$

$$\mathbf{P}_{M^{[j+1]}} \left(\mathbf{U}^{(1,0)}\right)^{\widehat{\mathrm{II}}} \mathbf{P}_{N^{[j+1]}}{}^{T} = \frac{1}{2} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T} & \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ -\mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\mathbf{A}^{[j]}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T} & -\mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix},$$

$$\mathbf{P}_{M^{[j+1]}} \left(\mathbf{U}^{(1,1)}\right)^{\widehat{\mathrm{II}}} \mathbf{P}_{N^{[j+1]}}{}^{T} = \frac{1}{2} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T} & -\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ -\mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\mathbf{A}^{[j]}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T} & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix}.$$

Consequently, using (7.3), we find

$$\left(u^{(0,1)}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1} = -\left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1} \quad \text{and} \quad \left(u^{(0,1)}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l} = \left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}, \tag{7.25}$$

$$\left(u^{(1,0)}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1} = \left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1} \quad \text{and} \quad \left(u^{(1,0)}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l} = -\left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}, \tag{7.26}$$

$$\left(u^{(1,1)}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1} = -\left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1} \quad \text{and} \quad \left(u^{(1,1)}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l} = -\left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}, \tag{7.27}$$

for all $k \in \{0, \ldots, M^{[j]} - 1\}$ and $l \in \{0, \ldots, N^{[j]} - 1\}$. As either $\mathbf{U}^{[0,0]}$, $\mathbf{U}^{[0,1]}$, $\mathbf{U}^{[1,0]}$ or $\mathbf{U}^{[1,1]}$ has to be $\mathbf{A}^{[j+1]}$, we can determine which of these four matrices is the correct one by comparing nonzero entries $\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1}$ and $\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}$ to the corresponding entries of $\mathbf{U}^{(0,0)}$. Hence, we require

$$\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)}+1} = 2^{J-j-1} a^{\widehat{\mathrm{II}}}_{2^{J-j}k_{(0,1)},\,2^{J-j-1}(2l_{(0,1)}+1)} \neq 0 \qquad \text{and}$$

$$\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}} = 2^{J-j-1} a^{\widehat{\mathrm{II}}}_{2^{J-j-1}(2k_{(1,0)}+1),\,2^{J-j}l_{(1,0)}} \neq 0$$

for some indices $k_{(0,1)}, k_{(1,0)} \in I_{0,\,M^{[j]}-1}$ and $l_{(0,1)}, l_{(1,0)} \in I_{0,\,N^{[j]}-1}$. By Lemma 7.15, both of these entries can be found by examining $\mathcal{O}\left(m^{[j]}n^{[j]}\right)$ entries of $\mathbf{A}^{\widehat{\mathrm{II}}}$. Further, we also need to compute the corresponding entries of $\left(\mathbf{U}^{(0,0)}\right)^{\widehat{\mathrm{II}}}$. Since the support of size $m^{[j]} \times n^{[j]}$ of $\mathbf{A}^{[j]}$ and thus of $\mathbf{U}^{(0,0)}$ is known, we obtain that

$$\begin{aligned} \left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)}+1} &= \left(\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{U}^{(0,0)}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T}\right)_{2k_{(0,1)},\,2l_{(0,1)}+1} \\ &= \sum_{r=\mu_R^{[j]}}^{\nu_R^{[j]}} \sum_{s=\mu_C^{[j]}}^{\nu_C^{[j]}} \left(\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\right)_{2k_{(0,1)},\,r} u^{(0,0)}_{r,\,s} \left(\mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^{T}\right)_{s,\,2l_{(0,1)}+1} \qquad \text{and} \end{aligned}$$

$$\begin{aligned}\left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}} &= \left(\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{U}^{(0,0)}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T}\right)_{2k_{(1,0)}+1,\,2l_{(1,0)}}\\ &= \sum_{r=\mu_R^{[j]}}^{\nu_R^{[j]}}\sum_{s=\mu_C^{[j]}}^{\nu_C^{[j]}}\left(\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\right)_{2k_{(1,0)}+1,\,r} u^{(0,0)}_{r,\,s}\left({\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T}\right)_{s,\,2l_{(1,0)}},\end{aligned}$$

so the required entries of $\left(\mathbf{U}^{(0,0)}\right)^{\widehat{\mathrm{II}}}$ can be calculated in $\mathcal{O}\left(m^{[j]}n^{[j]}\right)$ time as well.

By (7.24), we have that $\mathbf{A}^{[j+1]} = \mathbf{U}^{(0,0)}$ if

$$\begin{aligned}\left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}} &= \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}} && \text{and}\\ \left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}} &= \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}},\end{aligned}$$

by (7.25) that $\mathbf{A}^{[j+1]} = \mathbf{U}^{(0,1)}$ if

$$\begin{aligned}\left(u^{(0,1)}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}} &= -\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}} && \text{and}\\ \left(u^{(0,1)}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}} &= \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}},\end{aligned}$$

by (7.26) that $\mathbf{A}^{[j+1]} = \mathbf{U}^{(1,0)}$ if

$$\begin{aligned}\left(u^{(1,0)}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}} &= \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}} && \text{and}\\ \left(u^{(1,0)}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}} &= -\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}},\end{aligned}$$

and by (7.27) that $\mathbf{A}^{[j+1]} = \mathbf{U}^{(1,1)}$ if

$$\begin{aligned}\left(u^{(1,1)}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}} &= -\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}} && \text{and}\\ \left(u^{(1,1)}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}} &= -\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}}.\end{aligned}$$

Numerically, we define

$$\begin{aligned}\delta^{-}_{(0,1)} &:= \left|\left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}} - \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}}\right|,\\ \delta^{+}_{(0,1)} &:= \left|\left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}} + \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(0,1)},\,2l_{(0,1)+1}}\right|,\\ \delta^{-}_{(1,0)} &:= \left|\left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}} - \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}}\right|,\\ \delta^{+}_{(1,0)} &:= \left|\left(u^{(0,0)}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}} + \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_{(1,0)}+1,\,2l_{(1,0)}}\right|,\end{aligned}$$

and set

$$\mathbf{A}^{[j+1]} \coloneqq \begin{cases} \mathbf{U}^{(0,0)} & \text{if } \delta^-_{(0,1)} < \delta^+_{(0,1)} \text{ and } \delta^-_{(1,0)} < \delta^+_{(1,0)}, \\ \mathbf{U}^{(0,1)} & \text{if } \delta^-_{(0,1)} > \delta^+_{(0,1)} \text{ and } \delta^-_{(1,0)} < \delta^+_{(1,0)}, \\ \mathbf{U}^{(1,0)} & \text{if } \delta^-_{(0,1)} < \delta^+_{(0,1)} \text{ and } \delta^-_{(1,0)} > \delta^+_{(1,0)}, \\ \mathbf{U}^{(1,1)} & \text{if } \delta^-_{(0,1)} > \delta^+_{(0,1)} \text{ and } \delta^-_{(1,0)} > \delta^+_{(1,0)}. \end{cases}$$

Then we obtain for the first row and column support indices that

$$\mu_R^{[j+1]} \coloneqq \begin{cases} \mu_R^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{U}^{(0,0)} \text{ or } \mathbf{A}^{[j+1]} = \mathbf{U}^{(0,1)}, \\ M^{[j+1]} - 1 - \nu_R^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{U}^{(1,0)} \text{ or } \mathbf{A}^{[j+1]} = \mathbf{U}^{(1,1)}, \end{cases} \tag{7.28}$$

and

$$\mu_C^{[j+1]} \coloneqq \begin{cases} \mu_C^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{U}^{(0,0)} \text{ or } \mathbf{A}^{[j+1]} = \mathbf{U}^{(1,0)}, \\ N^{[j+1]} - 1 - \nu_C^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{U}^{(0,1)} \text{ or } \mathbf{A}^{[j+1]} = \mathbf{U}^{(1,1)}. \end{cases} \tag{7.29}$$

Recall that we have already shown that $m^{[j+1]} = m^{[j]}$ and $n^{[j+1]} = n^{[j]}$ in Theorem 7.14, case D. □

7.3.2 Recovery Procedure for Case A: Colliding Rows and Columns

If $j = j_1$, with j_1 as in (7.7), the support $S^{[j]}$ of $\mathbf{A}^{[j]}$ is contained in the last b_R rows and the last b_C columns, i.e.,

$$S^{[j]} \subseteq I_{M^{[j]}-b_R,\, M^{[j]}-1} \times I_{N^{[j]}-b_C,\, N^{[j]}-1}.$$

We know from Theorem 7.14, case A that

$$S^{[j+1]} \subsetneq I_{M^{[j]}-b_R,\, M^{[j]}+b_R-1} \times I_{N^{[j]}-b_C,\, N^{[j]}+b_C-1},$$

and that the support of $\mathbf{A}^{[j+1]}$ contains entries from all four quadrants of $\mathbf{A}^{[j+1]}$. Further, it is possible that nonzero entries of $\mathbf{A}^{[j+1]}$ have been added to compute $\mathbf{A}^{[j]}$, so $\mathbf{A}^{[j]}$ and $\mathbf{A}^{[j+1]}$ do not necessarily have the same nonzero entries. The support $S^{[j]}$ of $\mathbf{A}^{[j]}$ has size $m^{[j]} \times n^{[j]}$ with $m^{[j]} \leq m \leq b_R$ and $n^{[j]} \leq n \leq b_C$. Analogously to Section 6.3.1, it follows from Definition 7.5 that the supports of $\mathbf{A}^{[j+1]}_{(0,0)}$, $\mathbf{A}^{[j+1]}_{(0,1)}$, $\mathbf{A}^{[j+1]}_{(1,0)}$ and $\mathbf{A}^{[j+1]}_{(1,1)}$ have at most size $\widetilde{m}^{[j]} \times \tilde{n}^{[j]}$, where

$$\widetilde{m}^{[j]} \coloneqq M^{[j]} - \mu_R^{[j]} \leq b_R \quad \text{and} \quad \tilde{n}^{[j]} \coloneqq N^{[j]} - \mu_C^{[j]} \leq b_C.$$

See Figure 7.10 for an illustration.

Consequently, we can restrict $\mathbf{A}^{[j]}$ and the four quadrants of $\mathbf{A}^{[j+1]}$ to matrices of size $2^{\widetilde{K}_R-1} \times 2^{\widetilde{K}_C-1}$, where $\widetilde{m}^{[j]} \leq 2^{\widetilde{K}_R-1}$ and $\widetilde{n}^{[j]} \leq 2^{\widetilde{K}_C-1}$, which take all nonzero entries into account. Then it suffices to recover these restrictions.

The following theorem enables us to compute $\mathbf{A}^{[j+1]}$ from $\mathbf{A}^{[j]}$ and $\mathbf{A}^{\widehat{\Pi}}$.

Theorem 7.18 *Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$, and let $J \coloneqq \min\{J_R, J_C\}$. Let $\mathbf{A} \in \mathbb{R}^{M \times N}$ have a block support of size $m \times n$ with known bounds $b_R \geq m$ and $b_C \geq n$, and assume that $\mathbf{A}$ satisfies (7.5) and (7.6). Let $L \coloneqq \max\{\lceil \log_2 b_R \rceil + 1, \lceil \log_2 b_C \rceil + 1\}$ and $j = j_1 \in \{L, \dots, J-1\}$ as in (7.7). Suppose that we have access to all entries of $\mathbf{A}^{\widehat{\Pi}}$. Then $\mathbf{A}^{[j+1]}$ can be recovered from $\mathbf{A}^{[j]}$ and $8 \cdot 2^{\widetilde{K}_R-1} \cdot 2^{\widetilde{K}_C-1}$ entries of $\mathbf{A}^{\widehat{\Pi}}$.*

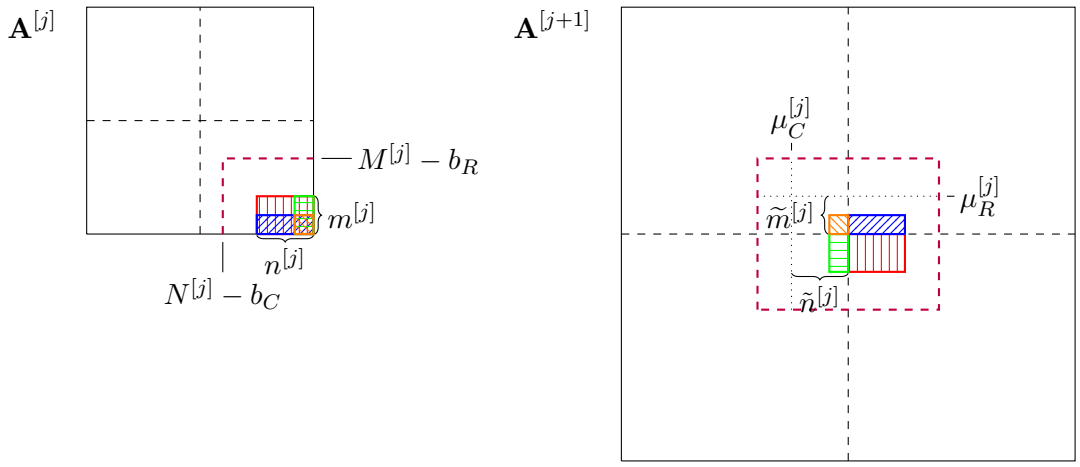

Figure 7.10: Illustration of the support of $\mathbf{A}^{[j]}$ and a possibility for the support of $\mathbf{A}^{[j+1]}$

Proof. Recall that by (7.1)

$$\mathbf{P}_{M^{[j+1]}}\left(\mathbf{A}^{[j+1]}\right)^{\widehat{\mathrm{II}}}\mathbf{P}_{N^{[j+1]}}{}^{T} = \frac{1}{2}\begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{A}^{[j]}\mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^{T} & \mathbf{C}^{\mathrm{II}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(0,1)}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ \mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(1,0)}\mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^{T} & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(1,1)}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix}, \tag{7.30}$$

where

$$\begin{aligned} \mathbf{A}^{[j]} = \tilde{\mathbf{A}}^{[j+1]}_{(0,0)} &= \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} + \left(\mathbf{A}^{[j+1]}_{(0,1)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}, \\ \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} &= \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} - \left(\mathbf{A}^{[j+1]}_{(0,1)} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}, \\ \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} &= \mathbf{A}^{[j+1]}_{(0,0)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} + \left(\mathbf{A}^{[j+1]}_{(0,1)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}, \\ \tilde{\mathbf{A}}^{[j+1]}_{(1,1)} &= \mathbf{A}^{[j+1]}_{(0,0)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} - \left(\mathbf{A}^{[j+1]}_{(0,1)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}. \end{aligned} \tag{7.31}$$

As $\mathbf{A}^{[j]} = \tilde{\mathbf{A}}^{[j+1]}_{(0,0)}$ is known from the previous iteration step and we can only observe $\left(\mathbf{A}^{[j+1]}\right)^{\widehat{\mathrm{II}}}$ from the given data $\mathbf{A}^{\widehat{\mathrm{II}}}$, we have to recover $\tilde{\mathbf{A}}^{[j+1]}_{(0,1)}$, $\tilde{\mathbf{A}}^{[j+1]}_{(1,0)}$ and $\tilde{\mathbf{A}}^{[j+1]}_{(1,1)}$ in order to be able to compute $\mathbf{A}^{[j+1]}_{(0,0)}$, $\mathbf{A}^{[j+1]}_{(0,1)}$, $\mathbf{A}^{[j+1]}_{(1,0)}$ and $\mathbf{A}^{[j+1]}_{(1,1)}$. Solving the equation system (7.31) for the four quadrants of $\mathbf{A}^{[j+1]}$, we find that

$$\begin{aligned} \mathbf{A}^{[j+1]}_{(0,0)} &= \frac{1}{4}\left(\mathbf{A}^{[j]} + \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} + \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} + \tilde{\mathbf{A}}^{[j+1]}_{(1,1)}\right), \\ \mathbf{A}^{[j+1]}_{(0,1)} &= \frac{1}{4}\left(\mathbf{A}^{[j]} - \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} + \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} - \tilde{\mathbf{A}}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}, \\ \mathbf{A}^{[j+1]}_{(1,0)} &= \frac{1}{4}\mathbf{J}_{M^{[j]}}\left(\mathbf{A}^{[j]} + \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} - \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} - \tilde{\mathbf{A}}^{[j+1]}_{(1,1)}\right), \\ \mathbf{A}^{[j+1]}_{(1,1)} &= \frac{1}{4}\mathbf{J}_{M^{[j]}}\left(\mathbf{A}^{[j]} - \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} - \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} + \tilde{\mathbf{A}}^{[j+1]}_{(1,1)}\right)\mathbf{J}_{N^{[j]}}. \end{aligned} \tag{7.32}$$

We now investigate the support of the four quadrants of the matrix

$$\tilde{\mathbf{A}}^{[j+1]} := \begin{pmatrix} \mathbf{A}^{[j]} & \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} \\ \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} & \tilde{\mathbf{A}}^{[j+1]}_{(1,1)} \end{pmatrix}.$$

Note that by (7.31) and the non-cancellation conditions (7.5) and (7.6), all quadrants of $\tilde{\mathbf{A}}^{[j+1]}$ have the same support, namely the support $S^{[j]}$ of $\mathbf{A}^{[j]}$. Let us assume that the support of $\mathbf{A}^{[j]}$ is of size $m^{[j]} \times n^{[j]}$ with first row and column support indices $\mu_R^{[j]}$ and $\mu_C^{[j]}$. As $j = j_1$, Lemma 7.12, case (i) and Theorem 7.14, case A yield that

$$S^{[j]} \subseteq I_{\mu_R^{[j]},\, M^{[j]}-1} \times I_{\mu_C^{[j]},\, N^{[j]}-1} \subseteq I_{M^{[j]}-b_R,\, M^{[j]}-1} \times I_{N^{[j]}-b_C,\, N^{[j]}-1}, \tag{7.33}$$

so the support of all four quadrants of $\tilde{\mathbf{A}}^{[j+1]}$ is contained in the direct product of two intervals on the right-hand side. Analogously to Section 6.3.1, we set

$$\widetilde{m}^{[j]} := M^{[j]} - \mu_R^{[j]} \leq b_R \quad \text{and} \quad \tilde{n}^{[j]} := N^{[j]} - \mu_C^{[j]} \leq b_C.$$

Both intervals can be increased to lengths that are powers of 2 by defining

$$\widetilde{K}_R := \left\lceil \log_2 \widetilde{m}^{[j]} \right\rceil + 1 \quad \text{and} \quad \widetilde{K}_C := \left\lceil \log_2 \tilde{n}^{[j]} \right\rceil + 1.$$

Then we find that

$$\begin{aligned} S^{[j]} &\subseteq I_{M^{[j]}-\widetilde{m}^{[j]},\, M^{[j]}-1} \times I_{N^{[j]}-\tilde{n}^{[j]},\, N^{[j]}-1} \\ &\subseteq I_{M^{[j]}-2^{\widetilde{K}_R-1},\, M^{[j]}-1} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}-1}, \end{aligned} \tag{7.34}$$

and it follows from Definition 7.5 and (7.33) that

$$\begin{aligned} S^{[j+1]} &\subsetneq I_{M^{[j]}-\widetilde{m}^{[j]},\, M^{[j]}+\widetilde{m}^{[j]}-1} \times I_{N^{[j]}-\tilde{n}^{[j]},\, N^{[j]}+\tilde{n}^{[j]}-1} \\ &\subseteq I_{M^{[j]}-2^{\widetilde{K}_R-1},\, M^{[j]}+2^{\widetilde{K}_R-1}-1} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}+2^{\widetilde{K}_C-1}-1}. \end{aligned} \tag{7.35}$$

Note that we always have $\widetilde{m}^{[j]} \geq m^{[j]}$ and $\tilde{n}^{[j]} \geq n^{[j]}$. If there is no collision in the rows or columns, respectively, inequality is possible. By (7.34), we obtain

$$S\left(\tilde{\mathbf{A}}^{[j+1]}_{(0,1)}\right),\, S\left(\tilde{\mathbf{A}}^{[j+1]}_{(1,0)}\right),\, S\left(\tilde{\mathbf{A}}^{[j+1]}_{(1,1)}\right) \subseteq I_{M^{[j]}-2^{\widetilde{K}_R-1},\, M^{[j]}-1} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}-1}$$

if we denote by $S(\mathbf{B})$ the support of a matrix $\mathbf{B}$, see also Figure 7.11.

For incorporating all entries of $\tilde{\mathbf{A}}^{[j+1]}$ that are influenced by possibly nonzero entries of $\mathbf{A}^{[j+1]}$ it suffices to restrict the four quadrants of $\tilde{\mathbf{A}}^{[j+1]}$ to matrices of size $2^{\widetilde{K}_R-1} \times 2^{\widetilde{K}_C-1}$,

$$\widetilde{\mathbf{B}}^{[j+1]}_{(r,s)} := \left(\tilde{a}^{[j+1]}_{k,l}\right)_{k=M^{[j]}-2^{\widetilde{K}_R-1}+rM^{[j]},\, l=N^{[j]}-2^{\widetilde{K}_C-1}+sN^{[j]}}^{M^{[j]}-1+rM^{[j]},\, N^{[j]}-1+sN^{[j]}}$$

for all $r, s \in \{0, 1\}$, and just recover these restrictions, analogously to Section 6.3.1. Note that since $\mathbf{A}^{[j]} = \tilde{\mathbf{A}}^{[j+1]}_{(0,0)}$, we have that $\mathbf{B}^{[j]} := \widetilde{\mathbf{B}}^{[j+1]}_{(0,0)}$ is a restriction of $\mathbf{A}^{[j]}$ to its last $2^{\widetilde{K}_R-1}$ rows and $2^{\widetilde{K}_C-1}$ columns.

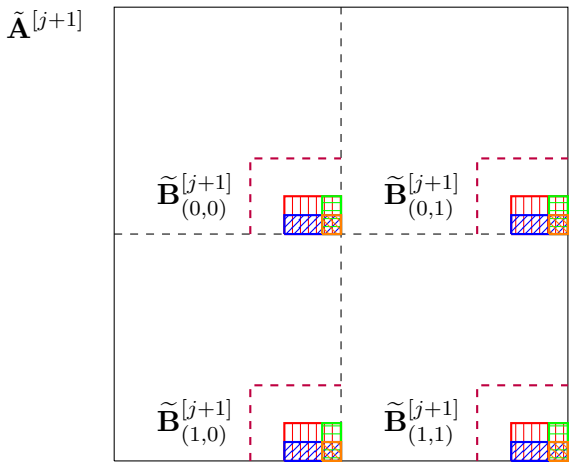

Figure 7.11: Illustration of the support of $\tilde{\mathbf{A}}^{[j+1]}$ and the choice of $\widetilde{\mathbf{B}}^{[j+1]}_{(0,0)}, \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)}, \widetilde{\mathbf{B}}^{[j+1]}_{(1,0)}$ and $\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}$

Furthermore, by (7.35), we also have to restrict the four quadrants of $\mathbf{A}^{[j+1]}$ to matrices of size $2^{\widetilde{K}_R-1} \times 2^{\widetilde{K}_C-1}$ by setting

$$\begin{aligned}
\mathbf{B}^{[j+1]}_{(0,0)} &\coloneqq \left(a^{[j+1]}_{k,l}\right)^{M^{[j]}-1,\, N^{[j]}-1}_{k=M^{[j]}-2^{\widetilde{K}_R-1},\, l=N^{[j]}-2^{\widetilde{K}_C-1}}, \\
\mathbf{B}^{[j+1]}_{(0,1)} &\coloneqq \left(a^{[j+1]}_{k,l}\right)^{M^{[j]}-1,\, N^{[j]}+2^{\widetilde{K}_C-1}-1}_{k=M^{[j]}-2^{\widetilde{K}_R-1},\, l=N^{[j]}}, \\
\mathbf{B}^{[j+1]}_{(1,0)} &\coloneqq \left(a^{[j+1]}_{k,l}\right)^{M^{[j]}+2^{\widetilde{K}_R-1}-1,\, N^{[j]}-1}_{k=M^{[j]},\, l=N^{[j]}-2^{\widetilde{K}_C-1}}, \\
\mathbf{B}^{[j+1]}_{(0,0)} &\coloneqq \left(a^{[j+1]}_{k,l}\right)^{M^{[j]}+2^{\widetilde{K}_R-1}-1,\, N^{[j]}+2^{\widetilde{K}_C-1}-1}_{k=M^{[j]},\, l=N^{[j]}}.
\end{aligned}$$

These four matrices take all of the possibly nonzero entries of $\mathbf{A}^{[j+1]}$ into account, similar to the vectors $\mathbf{z}^{[j+1]}_{(0)}$ and $\mathbf{z}^{[j+1]}_{(1)}$ in Section 6.3.1. Then (7.31) and (7.32) also hold for the restrictions of the quadrants of $\mathbf{A}^{[j+1]}$ and $\tilde{\mathbf{A}}^{[j+1]}$, respectively. Hence, our goal is to recover $\widetilde{\mathbf{B}}^{[j+1]}_{(0,1)}$, $\widetilde{\mathbf{B}}^{[j+1]}_{(1,0)}$ and $\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}$ from $\left(\mathbf{A}^{[j+1]}\right)^{\widehat{\Pi}}$ and $\mathbf{A}^{[j]}$, using as few arithmetical operations and samples from $\mathbf{A}^{\widehat{\Pi}}$ as possible. By (7.32) and $\mathbf{B}^{[j]}$, which is completely given by $\mathbf{A}^{[j]}$, we can then compute $\mathbf{B}^{[j+1]}_{(0,0)}$, $\mathbf{B}^{[j+1]}_{(0,1)}$, $\mathbf{B}^{[j+1]}_{(1,0)}$ and $\mathbf{B}^{[j+1]}_{(1,1)}$. From these restrictions we finally obtain the sought-after matrix $\mathbf{A}^{[j+1]}$.

(i) Computation of $\widetilde{\mathbf{B}}^{[j+1]}_{(0,1)}$.

For the computation of $\widetilde{\mathbf{B}}^{[j+1]}_{(0,1)}$ we proceed analogously to (6.8) in Section 6.3.1. Recall that by definition of $\widetilde{\mathbf{B}}^{[j+1]}_{(0,1)}$, we have that

$$S\left(\widetilde{\mathbf{B}}^{[j+1]}_{(0,1)}\right) \subseteq I_{M^{[j]}-2^{\widetilde{K}_R-1},\, M^{[j]}-1} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}-1}.$$

It follows from (7.30) and (7.3) that

$$
\begin{aligned}
&\sqrt{M^{[j]}N^{[j]}}\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} = \frac{\sqrt{M^{[j]}N^{[j]}}}{2}\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(0,1)}\mathbf{C}^{\mathrm{IV}}_{N^{[j]}}\\
&= \left(\sum_{r'=0}^{M^{[j]}-1}\sum_{s'=0}^{N^{[j]}-1}\varepsilon_{M^{[j]}}(k)\cos\left(\frac{k(2r'+1)\pi}{2M^{[j]}}\right)\left(\tilde{a}^{[j+1]}_{(0,1)}\right)_{r',\,s'}\right.\\
&\quad\left.\cdot\cos\left(\frac{(2s'+1)(2l+1)\pi}{4N^{[j]}}\right)\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0}\\
&= \left(\sum_{r'=M^{[j]}-2^{\widetilde{K}_R-1}}^{M^{[j]}-1}\sum_{s'=N^{[j]}-2^{\widetilde{K}_C-1}}^{N^{[j]}-1}\varepsilon_{M^{[j]}}(k)\cos\left(\frac{k(2r'+1)\pi}{2M^{[j]}}\right)\left(\tilde{a}^{[j+1]}_{(0,1)}\right)_{r',\,s'}\right.\\
&\quad\left.\cdot\cos\left(\frac{(2s'+1)(2l+1)\pi}{4N^{[j]}}\right)\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0}\\
&= \left(\sum_{r=0}^{2^{\widetilde{K}_R-1}-1}\sum_{s=0}^{2^{\widetilde{K}_C-1}-1}\varepsilon_{M^{[j]}}(k)\cos\left(\frac{k\left(2M^{[j]}-(2r+1)\right)\pi}{2M^{[j]}}\right)\left(\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(0,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\right)_{r,\,s}\right.\\
&\quad\left.\cdot\cos\left(\frac{\left(2N^{[j]}-(2s+1)\right)(2l+1)\pi}{4N^{[j]}}\right)\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0}\\
&= \left(\sum_{r=0}^{2^{\widetilde{K}_R-1}-1}\sum_{s=0}^{2^{\widetilde{K}_C-1}-1}\varepsilon_{M^{[j]}}(k)\left(\cos(k\pi)\cos\left(\frac{k(2r+1)\pi}{2M^{[j]}}\right)+\sin(k\pi)\sin\left(\frac{k(2r+1)\pi}{2M^{[j]}}\right)\right)\right.\\
&\quad\left.\cdot\left(\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(0,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\right)_{r,\,s}\cdot(-1)^l\sin\left(\frac{(2s+1)(2l+1)\pi}{4N^{[j]}}\right)\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0}\\
&= \left(\sum_{r=0}^{2^{\widetilde{K}_R-1}-1}\sum_{s=0}^{2^{\widetilde{K}_C-1}-1}\varepsilon_{M^{[j]}}(k)(-1)^k\cos\left(\frac{k(2r+1)\pi}{2M^{[j]}}\right)\left(\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(0,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\right)_{r,\,s}\right.\\
&\quad\left.\cdot(-1)^l\sin\left(\frac{(2s+1)(2l+1)\pi}{4N^{[j]}}\right)\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0}, \qquad (7.36)
\end{aligned}
$$

where we set $r \coloneqq M^{[j]}-1-r'$ and $s \coloneqq N^{[j]}-1-s'$ in the third step, and use (6.8) in the fourth step. Since $\widetilde{\mathbf{B}}^{[j+1]}_{(0,1)}$ is of size $2^{\widetilde{K}_R-1}\times 2^{\widetilde{K}_C-1}$, it suffices to restrict (7.36) to $2^{\widetilde{K}_R-1}$ rows and $2^{\widetilde{K}_C-1}$ columns in order to recover the matrix. From Section 6.3.1 we already know how to restrict the sine terms in (7.36), which correspond to columns. We choose the columns indexed by $2l_q+1$, where $l_q \coloneqq N^{[j]}2^{-\widetilde{K}_C}(2q+1)$ for $q \in \left\{0,\ldots,2^{\widetilde{K}_C-1}-1\right\}$. We restrict the rows, which correspond to the cosine terms, by only considering the ones indexed by $2k''_p$, where we define $k''_p \coloneqq M^{[j]}2^{-\widetilde{K}_R+1}p$, $p \in \left\{0,\ldots,2^{\widetilde{K}_R-1}-1\right\}$. As

$k_p'' \in \{0, \ldots, M^{[j]} - 1\}$ for all p, we find that

$$
\begin{aligned}
&\sqrt{M^{[j]}N^{[j]}} \left(\left(a^{[j+1]} \right)^{\widehat{\mathrm{II}}}_{2k_p'',\, 2l_q+1} \right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\, 2^{\widetilde{K}_C-1}-1} \\
&= \left((-1)^{k_p''} \varepsilon_{M^{[j]}}\left(k_p''\right) \cos\left(\frac{M^{[j]} 2^{-\widetilde{K}_R+1} p(2r+1)\pi}{2M^{[j]}} \right) \right)_{p,\,r=0}^{2^{-\widetilde{K}_R-1}-1} \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \\
&\quad \cdot \left((-1)^{N^{[j]}2^{-\widetilde{K}_C}} \sin\left(\frac{(2s+1)\left(N^{[j]}2^{-\widetilde{K}_C+1}(2q+1)+1\right)\pi}{4N^{[j]}} \right) \right)_{s,\,q=0}^{2^{\widetilde{K}_C-1}-1} \\
&= (-1)^{N^{[j]}2^{-\widetilde{K}_C}} \left(\varepsilon_{2^{\widetilde{K}_R-1}}(p) \cos\left(\frac{p(2r+1)\pi}{2 \cdot 2^{\widetilde{K}_R-1}} \right) \right)_{p,\,r=0}^{2^{\widetilde{K}_R-1}-1} \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \\
&\quad \cdot \left(\cos\left(\frac{(2s+1)\pi}{4N^{[j]}} \right) \sin\left(\frac{(2s+1)(2q+1)\pi}{4 \cdot 2^{\widetilde{K}_C-1}} \right) \right. \\
&\quad \left. + \sin\left(\frac{(2s+1)\pi}{4N^{[j]}} \right) \cos\left(\frac{(2s+1)(2q+1)\pi}{4 \cdot 2^{\widetilde{K}_C-1}} \right) \right)_{s,\,q=0}^{2^{\widetilde{K}_C-1}-1},
\end{aligned}
\tag{7.37}
$$

since

$$2^{\widetilde{K}_R-1} \leq 2^{L-1} = M^{[L]} \cdot 2^{J-J_R-1} \leq M^{[j]} \cdot 2^{-1}$$

for all $j \in \{L, \ldots, J-1\}$. Thus

$$k_p'' = M^{[j]} 2^{-\widetilde{K}_R+1} p = 2^{\alpha} p$$

with $\alpha \in \mathbb{N}$, $\alpha \geq 1$. Hence, we have that $(-1)^{k_p''} = 1$ and $\varepsilon_{M^{[j]}}\left(k_p''\right) = \varepsilon_{2^{\widetilde{K}_R-1}}(p)$ for all p. Defining the vectors

$$\mathbf{c}_C^{[j]} := \left(\cos\left(\frac{(2s+1)\pi}{4N^{[j]}} \right) \right)_{s=0}^{2^{\widetilde{K}_C-1}-1} \quad \text{and} \quad \mathbf{s}_C^{[j]} := \left(\sin\left(\frac{(2s+1)\pi}{4N^{[j]}} \right) \right)_{s=0}^{2^{\widetilde{K}_C-1}-1},$$

analogously to Section 6.3.1, and using Theorem 4.4 (ii), (7.37) can be written as

$$
\begin{aligned}
&\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}}\,(-1)^{N^{[j]}2^{-\widetilde{K}_C}} \left(\left(a^{[j+1]} \right)^{\widehat{\mathrm{II}}}_{2k_p'',\, 2l_q+1} \right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\, 2^{\widetilde{K}_C-1}-1} \\
&= \mathbf{C}^{\mathrm{II}}_{2^{\widetilde{K}_R-1}} \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \left(\operatorname{diag}\left(\mathbf{s}_C^{[j]} \right) \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} + \operatorname{diag}\left(\mathbf{c}_C^{[j]} \right) \mathbf{D}_{2^{\widetilde{K}_C-1}} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \mathbf{J}_{2^{\widetilde{K}_C-1}} \right) \\
&= \mathbf{C}^{\mathrm{II}}_{2^{\widetilde{K}_R-1}} \left(\mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \;\middle|\; \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \right) \\
&\quad \cdot \begin{pmatrix} \operatorname{diag}\left(\mathbf{s}_C^{[j]} \right) & \\ & \operatorname{diag}\left(\mathbf{c}_C^{[j]} \right) \mathbf{D}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \\ \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix}.
\end{aligned}
\tag{7.38}
$$

Since the last matrix in (7.38) is not a square matrix, this means that we did not obtain an invertible matrix factorization. Analogously to Section 6.3.1, we solve this problem by considering $2^{\widetilde{K}_C-1}$ additional columns of $\left(\left(a^{[j+1]} \right)^{\widehat{\mathrm{II}}}_{2k,\, 2l+1} \right)_{k,\,l=0}^{M^{[j]}-1,\, N^{[j]}-1}$. We choose

the same rows as before and the columns corresponding to the indices $2l'_q + 1$, where $l'_q \coloneqq N^{[j]}2^{-\widetilde{K}_C}(2q+1) - 1$, $q \in \left\{0, \dots, 2^{\widetilde{K}_C - 1} - 1\right\}$. Then (6.11) and (7.36) yield

$$\begin{aligned}
&\sqrt{M^{[j]}2^{-\widetilde{K}_R+2}} \left(\left(a^{[j+1]} \right)^{\widehat{\Pi}}_{2k''_p,\, 2l'_q+1} \right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\, 2^{\widetilde{K}_C-1}-1} \\
&= (-1)^{l'_q} \mathbf{C}^{\mathrm{II}}_{2^{\widetilde{K}_R-1}} \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \\
&\quad \cdot \left(\sin\left(\frac{(2s+1)\left(N^{[j]}2^{-\widetilde{K}_C+1}(2q+1) - 1\right)\pi}{4N^{[j]}} \right) \right)_{s,\,q=0}^{2^{\widetilde{K}_C-1}-1} \\
&= \frac{(-1)^{N^{[j]}2^{-\widetilde{K}_C}}}{\sqrt{N^{[j]}2^{-\widetilde{K}_C+2}}} \mathbf{C}^{\mathrm{II}}_{2^{\widetilde{K}_R-1}} \left(\begin{array}{c|c} \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{array} \right) \\
&\quad \cdot \begin{pmatrix} \operatorname{diag}\left(\mathbf{s}^{[j]}_C\right) & \\ & \operatorname{diag}\left(\mathbf{c}^{[j]}_C\right) \mathbf{D}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \\ -\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix}.
\end{aligned} \tag{7.39}$$

Let us denote the matrices of required entries of $\mathbf{A}^{\widehat{\Pi}}$ by

$$\mathbf{E}^{(0,1)}_{(0,0)} \coloneqq \left(\left(a^{[j+1]} \right)^{\widehat{\Pi}}_{2k''_p,\, 2l_q+1} \right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\, 2^{\widetilde{K}_C-1}-1} \quad \text{and}$$

$$\mathbf{E}^{(0,1)}_{(0,1)} \coloneqq \left(\left(a^{[j+1]} \right)^{\widehat{\Pi}}_{2k''_p,\, 2l'_q+1} \right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\, 2^{\widetilde{K}_C-1}-1}.$$

If we combine (7.38) and (7.39), we obtain

$$\begin{aligned}
&(-1)^{N^{[j]}2^{-\widetilde{K}_C}} \sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}} \left(\begin{array}{c|c} \mathbf{E}^{(0,1)}_{(0,0)} & \mathbf{E}^{(0,1)}_{(0,1)} \end{array} \right) \\
&= \mathbf{C}^{\mathrm{II}}_{2^{\widetilde{K}_R-1}} \left(\begin{array}{c|c} \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{array} \right) \\
&\quad \cdot \begin{pmatrix} \operatorname{diag}\left(\mathbf{s}^{[j]}_C\right) & \\ & \operatorname{diag}\left(\mathbf{c}^{[j]}_C\right) \mathbf{D}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \\ \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \mathbf{J}_{2^{\widetilde{K}_C-1}} & -\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \\
&= \mathbf{C}^{\mathrm{II}}_{2^{\widetilde{K}_R-1}} \left(\begin{array}{c|c} \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{array} \right) \\
&\quad \cdot \begin{pmatrix} \operatorname{diag}\left(\mathbf{s}^{[j]}_C\right) & \\ & \operatorname{diag}\left(\mathbf{c}^{[j]}_C\right) \mathbf{D}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}_C-1}} & \mathbf{I}_{2^{\widetilde{K}_C-1}} \\ \mathbf{J}_{2^{\widetilde{K}_C-1}} & -\mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix}.
\end{aligned} \tag{7.40}$$

By (6.13), the last matrix in (7.40) is invertible. As $\tilde{n}^{[j]} \leq b_C$ and thus $2^{\widetilde{K}_C} \leq 2^L \leq N^{[j]}$, $\operatorname{diag}\left(\mathbf{c}^{[j]}_C\right)$ and $\operatorname{diag}\left(\mathbf{s}^{[j]}_C\right)$ are invertible, analogously to (6.14). Consequently, the third matrix in (7.40) is invertible as well, since the multiplication with ± 1, caused by the diagonal matrix $\mathbf{D}_{2^{\widetilde{K}_C-1}}$, does not change the absolute value of the determinant. Thus,

all square matrices in (7.40) can be inverted, and it follows that

$$
\begin{aligned}
&\left(\mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \;\middle|\; \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \right) \\
&= (-1)^{N^{[j]}2^{-\widetilde{K}_C}} \sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}}\, \mathbf{C}^{\mathrm{III}}_{2^{\widetilde{K}_R-1}} \left(\mathbf{E}^{(0,1)}_{(0,0)} \;\middle|\; \mathbf{E}^{(0,1)}_{(0,1)} \right) \\
&\quad \cdot \frac{1}{2} \begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_C-1}} \\ \mathbf{I}_{2^{\widetilde{K}_C-1}} & -\mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \begin{pmatrix} \operatorname{diag}\left(\tilde{\mathbf{s}}^{[j]}_C\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_C-1}} \operatorname{diag}\left(\tilde{\mathbf{c}}^{[j]}_C\right) \end{pmatrix} \\
&= (-1)^{N^{[j]}2^{-\widetilde{K}_C}} \sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+2}}\, \mathbf{C}^{\mathrm{III}}_{2^{\widetilde{K}_R-1}} \\
&\quad \cdot \left(\mathbf{E}^{(0,1)}_{(0,0)} + \mathbf{E}^{(0,1)}_{(0,1)} \;\middle|\; \left(\mathbf{E}^{(0,1)}_{(0,0)} - \mathbf{E}^{(0,1)}_{(0,1)}\right) \mathbf{J}_{2^{\widetilde{K}_C-1}} \right) \\
&\quad \cdot \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \operatorname{diag}\left(\tilde{\mathbf{s}}^{[j]}_C\right) & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \mathbf{D}_{2^{\widetilde{K}_C-1}} \operatorname{diag}\left(\tilde{\mathbf{c}}^{[j]}_C\right) \end{pmatrix},
\end{aligned} \tag{7.41}
$$

where

$$
\tilde{\mathbf{c}}^{[j]}_C \coloneqq \left(\cos\left(\frac{(2s+1)\pi}{4N^{[j]}} \right)^{-1} \right)_{s=0}^{2^{\widetilde{K}_C-1}-1} \quad \text{and} \quad \tilde{\mathbf{s}}^{[j]}_C \coloneqq \left(\sin\left(\frac{(2s+1)\pi}{4N^{[j]}} \right)^{-1} \right)_{s=0}^{2^{\widetilde{K}_C-1}-1}.
$$

The last $2^{\widetilde{K}_C-1}$ columns of (7.41) are sufficient in order to find $\widetilde{B}^{[j+1]}_{(0,1)}$, i.e.,

$$
\begin{aligned}
\widetilde{\mathbf{B}}^{[j+1]}_{(0,1)} &= (-1)^{N^{[j]}2^{-\widetilde{K}_C}} \sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+2}}\, \mathbf{J}_{2^{\widetilde{K}_R-1}} \mathbf{C}^{\mathrm{III}}_{2^{\widetilde{K}_R-1}} \left(\mathbf{E}^{(0,1)}_{(0,0)} - \mathbf{E}^{(0,1)}_{(0,1)}\right) \mathbf{J}_{2^{\widetilde{K}_C-1}} \\
&\quad \cdot \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \mathbf{D}_{2^{\widetilde{K}_C-1}} \operatorname{diag}\left(\tilde{\mathbf{c}}^{[j]}_C\right) \mathbf{J}_{2^{\widetilde{K}_C-1}}.
\end{aligned} \tag{7.42}
$$

Thus, $\tilde{\mathbf{B}}^{[j+1]}_{(0,1)}$ can be computed using $2^{\widetilde{K}_R-1} \cdot 2^{\widetilde{K}_C}$ samples of $\mathbf{A}^{\widehat{\mathrm{II}}}$ by essentially applying $2^{\widetilde{K}_C-1}$ 1-dimensional DCT-IIIs of length $2^{\widetilde{K}_R-1}$ to the columns of the sample matrix and $2^{\widetilde{K}_R-1}$ 1-dimensional DCT-IVs of length $2^{\widetilde{K}_C-1}$ to the rows.

By choosing the last $2^{\widetilde{K}_C-1}$ columns in (7.41), we avoid inverting $\operatorname{diag}\left(\mathbf{s}^{[j]}_C\right)$, whose entries can be close to zero, and instead invert $\operatorname{diag}\left(\mathbf{c}^{[j]}_C\right)$, whose entries are all greater than $\frac{1}{\sqrt{2}}$. This computation is numerically more stable and is similar to the calculation of $\mathbf{z}^{[j+1]}_{(0)}$ in Algorithm 9.

(ii) Computation of $\widetilde{\mathbf{B}}^{[j+1]}_{(1,0)}$

Recall that

$$
S\left(\widetilde{\mathbf{B}}^{[j+1]}_{(1,0)}\right) \subseteq I_{M^{[j]}-2^{\widetilde{K}_R-1},\, M^{[j]}-1} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}-1}.
$$

Analogously to (7.36) in case (i), (7.30) and (7.3) yield that

$$
\begin{aligned}
&\sqrt{M^{[j]}N^{[j]}} \left(\left(a^{[j+1]} \right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l} \right)_{k,\,l=0}^{M^{[j]}-1,\,N^{[j]}-1} \\
&= \frac{\sqrt{M^{[j]}N^{[j]}}}{2} \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} {\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T}
\end{aligned}
$$

$$
\begin{aligned}
&= \left(\sum_{r'=0}^{M^{[j]}-1} \sum_{s'=0}^{N^{[j]}-1} \cos\left(\frac{(2k+1)(2r'+1)\pi}{4M^{[j]}}\right) \left(\tilde{a}^{[j+1]}_{(1,0)}\right)_{r',\,s'} \right. \\
&\quad \left. \cdot\, \varepsilon_{N^{[j]}}(l) \cos\left(\frac{(2s'+1)l\pi}{2N^{[j]}}\right)\right)_{k,\,l=0}^{M^{[j]}-1,\,N^{[j]}-1} \\
&= \left(\sum_{r=0}^{2^{\widetilde{K}_R-1}-1} \sum_{s=0}^{2^{\widetilde{K}_C-1}-1} (-1)^k \sin\left(\frac{(2k+1)(2r+1)\pi}{4M^{[j]}}\right) \left(\mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(1,0)} \mathbf{J}_{2^{\widetilde{K}_C-1}}\right)_{r,\,s} \right. \\
&\quad \left. \cdot \varepsilon_{N^{[j]}}(l)(-1)^l \cos\left(\frac{(2s+1)l\pi}{2N^{[j]}}\right)\right)_{k,\,l=0}^{M^{[j]}-1,\,N^{[j]}-1}. \qquad (7.43)
\end{aligned}
$$

We can restrict (7.43) using the same ideas as in (i) if we switch the roles of the rows and columns. Consequently, we select the rows corresponding to the indices $2k_p+1$, where $k_p \coloneqq M^{[j]}2^{-\widetilde{K}_R} \cdot (2p+1)$, $p \in \left\{0,\dots,2^{\widetilde{K}_R-1}-1\right\}$, and the columns corresponding to the indices $2l''_q$, where $l''_q \coloneqq N^{[j]}2^{-\widetilde{K}_C+1}q$, $q \in \left\{0,\dots,2^{\widetilde{K}_C-1}-1\right\}$. Note that, like k''_p, l''_q is even for all q. Then we obtain

$$
\begin{aligned}
&\sqrt{M^{[j]}N^{[j]}} \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p+1,\,2l''_q} \right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1} \\
&= (-1)^{M^{[j]}2^{-\widetilde{K}_R}} \left(\sin\left(\frac{(2p+1)(2r+1)\pi}{4\cdot 2^{\widetilde{K}_R-1}}\right) \cos\left(\frac{(2r+1)\pi}{4M^{[j]}}\right) \right. \\
&\quad \left. + \cos\left(\frac{(2p+1)(2r+1)\pi}{4\cdot 2^{\widetilde{K}_R-1}}\right) \sin\left(\frac{(2r+1)\pi}{4M^{[j]}}\right)\right)_{k,\,r=0}^{2^{\widetilde{K}_R-1}-1} \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(1,0)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \\
&\quad \cdot \left(\varepsilon_{2^{\widetilde{K}_C-1}}(q) \cos\left(\frac{(2s+1)q\pi}{2\cdot 2^{\widetilde{K}_C-1}}\right)\right)_{s,\,q=0}^{2^{\widetilde{K}_C-1}-1}. \qquad (7.44)
\end{aligned}
$$

Defining the vectors

$$
\mathbf{c}^{[j]}_R \coloneqq \left(\cos\left(\frac{(2r+1)\pi}{4M^{[j]}}\right)\right)_{r=0}^{2^{\widetilde{K}_R-1}-1} \quad \text{and} \quad \mathbf{s}^{[j]}_R \coloneqq \left(\sin\left(\frac{(2r+1)\pi}{4M^{[j]}}\right)\right)_{r=0}^{2^{\widetilde{K}_R-1}-1},
$$

we can write (7.44) as

$$
\begin{aligned}
&\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}}\,(-1)^{M^{[j]}2^{-\widetilde{K}_R}} \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p+1,\,2l''_q} \right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1} \\
&= \left(\mathbf{J}_{2^{\widetilde{K}_R-1}} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} \mathbf{D}_{2^{\widetilde{K}_R-1}} \operatorname{diag}\left(\mathbf{c}^{[j]}_R\right) + \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} \operatorname{diag}\left(\mathbf{s}^{[j]}_R\right)\right) \\
&\quad \cdot \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(1,0)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \mathbf{C}^{\mathrm{III}}_{2^{\widetilde{K}_C-1}} \\
&= \left(\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} \;\middle|\; \mathbf{J}_{2^{\widetilde{K}_R-1}} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} \right) \begin{pmatrix} \operatorname{diag}\left(\mathbf{s}^{[j]}_R\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_R-1}} \operatorname{diag}\left(\mathbf{c}^{[j]}_R\right) \end{pmatrix} \\
&\quad \cdot \begin{pmatrix} \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(1,0)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \\ \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(1,0)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \mathbf{C}^{\mathrm{III}}_{2^{\widetilde{K}_C-1}}. \qquad (7.45)
\end{aligned}
$$

In order to obtain an invertible matrix factorization, we additionally consider the rows indexed by $2k'_p+1$, where $k'_p \coloneqq M^{[j]}2^{-\widetilde{K}_R}(2p+1)-1$, $p \in \left\{0,\dots,2^{\widetilde{K}_R-1}-1\right\}$. Then (7.43) implies that

$$\begin{aligned}
&\sqrt{N^{[j]}2^{-\widetilde{K}_C+2}}\,(-1)^{M^{[j]}2^{-\widetilde{K}_R}}\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k'_p+1,\,2l''_q}\right)^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}_{p,\,q=0}\\
&= \left(\begin{array}{c|c} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & -\mathbf{J}_{2^{\widetilde{K}_R-1}}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} \end{array}\right)\begin{pmatrix} \operatorname{diag}\left(\mathbf{s}^{[j]}_R\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_R-1}}\operatorname{diag}\left(\mathbf{c}^{[j]}_R\right)\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,0)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\\ \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,0)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\mathbf{C}^{\mathrm{III}}_{2^{\widetilde{K}_C-1}}.
\end{aligned}\tag{7.46}$$

Let us denote the matrices of required entries of $\mathbf{A}^{\widehat{\mathrm{II}}}$ by

$$\mathbf{E}^{(1,0)}_{(0,0)} \coloneqq \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p+1,\,2l''_q}\right)^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}_{p,\,q=0} \qquad \text{and}$$
$$\mathbf{E}^{(1,0)}_{(1,0)} \coloneqq \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k'_p+1,\,2l''_q}\right)^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}_{p,\,q=0}.$$

Combining (7.45) and (7.46), we obtain

$$\begin{aligned}
&(-1)^{M^{[j]}2^{-\widetilde{K}_R}}\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}}\begin{pmatrix}\mathbf{E}^{(1,0)}_{(0,0)}\\ \mathbf{E}^{(1,0)}_{(1,0)}\end{pmatrix}\\
&= \begin{pmatrix}\mathbf{I}_{2^{\widetilde{K}_R-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\\ \mathbf{I}_{2^{\widetilde{K}_R-1}} & -\mathbf{J}_{2^{\widetilde{K}_R-1}}\end{pmatrix}\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}}\end{pmatrix}\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}^{[j]}_R\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_R-1}}\operatorname{diag}\left(\mathbf{c}^{[j]}_R\right)\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,0)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\\ \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,0)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\mathbf{C}^{\mathrm{III}}_{2^{\widetilde{K}_C-1}}.
\end{aligned}\tag{7.47}$$

Recall that $\widetilde{m}^{[j]} \le b_R$ and thus $2^{\widetilde{K}_R} \le 2^L \le M^{[j]}$. Then $\operatorname{diag}\left(\mathbf{c}^{[j]}_R\right)$ and $\operatorname{diag}\left(\mathbf{s}^{[j]}_R\right)$, and hence all square matrices in (7.47), are invertible by (6.14). Setting

$$\tilde{\mathbf{c}}^{[j]}_R \coloneqq \left(\cos\left(\frac{(2r+1)\pi}{4M^{[j]}}\right)^{-1}\right)^{2^{\widetilde{K}_R-1}-1}_{r=0} \quad\text{and}\quad \tilde{\mathbf{s}}^{[j]}_R \coloneqq \left(\sin\left(\frac{(2r+1)\pi}{4M^{[j]}}\right)^{-1}\right)^{2^{\widetilde{K}_R-1}-1}_{r=0},$$

we find that

$$\begin{aligned}
&\begin{pmatrix} \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(1,0)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \\ \mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(1,0)} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \\
&= (-1)^{M^{[j]}2^{-\widetilde{K}_R}} \sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}} \begin{pmatrix} \operatorname{diag}\left(\tilde{\mathbf{s}}^{[j]}_R\right) & \\ & \operatorname{diag}\left(\tilde{\mathbf{c}}^{[j]}_R\right) \mathbf{D}_{2^{\widetilde{K}_R-1}} \end{pmatrix} \\
&\quad \cdot \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} \end{pmatrix} \frac{1}{2} \begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}_R-1}} & \mathbf{I}_{2^{\widetilde{K}_R-1}} \\ \mathbf{J}_{2^{\widetilde{K}_R-1}} & -\mathbf{J}_{2^{\widetilde{K}_R-1}} \end{pmatrix} \begin{pmatrix} \mathbf{E}^{(1,0)}_{(0,0)} \\ \mathbf{E}^{(1,0)}_{(1,0)} \end{pmatrix} \mathbf{C}^{\mathrm{II}}_{2^{\widetilde{K}_C-1}} \\
&= (-1)^{M^{[j]}2^{-\widetilde{K}_R}} \sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}} \\
&\quad \cdot \begin{pmatrix} \operatorname{diag}\left(\tilde{\mathbf{s}}^{[j]}_R\right) \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & \\ & \operatorname{diag}\left(\tilde{\mathbf{c}}^{[j]}_R\right) \mathbf{D}_{2^{\widetilde{K}_R-1}} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} \end{pmatrix} \\
&\quad \cdot \begin{pmatrix} \mathbf{E}^{(1,0)}_{(0,0)} + \mathbf{E}^{(1,0)}_{(1,0)} \\ \mathbf{J}_{2^{\widetilde{K}_R-1}} \left(\mathbf{E}^{(1,0)}_{(0,0)} - \mathbf{E}^{(1,0)}_{(1,0)}\right) \end{pmatrix} \mathbf{C}^{\mathrm{II}}_{2^{\widetilde{K}_C-1}}.
\end{aligned} \tag{7.48}$$

Here, it is numerically more stable to compute $\widetilde{\mathbf{B}}^{[j+1]}_{(1,0)}$ from the last $2^{\widetilde{K}_R-1}$ columns, i.e.,

$$\begin{aligned}
\widetilde{\mathbf{B}}^{[j+1]}_{(1,0)} &= (-1)^{M^{[j]}2^{-\widetilde{K}_R}} \sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}} \mathbf{J}_{2^{\widetilde{K}_R-1}} \operatorname{diag}\left(\tilde{\mathbf{c}}^{[j]}_R\right) \mathbf{D}_{2^{\widetilde{K}_R-1}} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} \mathbf{J}_{2^{\widetilde{K}_R-1}} \\
&\quad \cdot \left(\mathbf{E}^{(1,0)}_{(0,0)} - \mathbf{E}^{(1,0)}_{(1,0)}\right) \mathbf{C}^{\mathrm{II}}_{2^{\widetilde{K}_C-1}} \mathbf{J}_{2^{\widetilde{K}_C-1}}.
\end{aligned} \tag{7.49}$$

Then $\widetilde{\mathbf{B}}^{[j+1]}_{(1,0)}$ can be calculated using $2^{\widetilde{K}_R} \cdot 2^{\widetilde{K}_C-1}$ samples of $\mathbf{A}^{\widehat{\mathrm{II}}}$ by essentially applying $2^{\widetilde{K}_C-1}$ 1-dimensional DCT-IVs of length $2^{\widetilde{K}_R-1}$ to the columns of the sample matrix and $2^{\widetilde{K}_R-1}$ 1-dimensional DCT-IIs of length $2^{\widetilde{K}_C-1}$ to its rows.

(iii) Computation of $\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}$

Recall that

$$S\left(\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\right) \subseteq I_{M^{[j]}-2^{\widetilde{K}_R-1},\, M^{[j]}-1} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}-1}.$$

Analogously to (7.36) in case (i) and (7.43) in case (ii), (7.30) and (7.3) yield that

$$\begin{aligned}
&\sqrt{M^{[j]}N^{[j]}} \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l+1} \right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} \\
&= \frac{\sqrt{M^{[j]}N^{[j]}}}{2} \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \tilde{\mathbf{A}}^{[j+1]}_{(1,1)} \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\
&= \left(\sum_{r=0}^{2^{\widetilde{K}_R-1}-1} \sum_{s=0}^{2^{\widetilde{K}_C-1}-1} (-1)^k \sin\left(\frac{(2k+1)(2r+1)\pi}{4M^{[j]}}\right) \left(\mathbf{J}_{2^{\widetilde{K}_R-1}} \widetilde{\mathbf{B}}^{[j+1]}_{(1,1)} \mathbf{J}_{2^{\widetilde{K}_C-1}}\right)_{r,\,s} \right. \\
&\quad \left. \cdot (-1)^l \sin\left(\frac{(2s+1)(2l+1)\pi}{4N^{[j]}}\right) \right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0}.
\end{aligned} \tag{7.50}$$

Since the sine terms appear for rows and columns, we will have to restrict both the rows and the columns of (7.50) as in Section 6.3.1, so that we will have to consider four submatrices in order to be able to compute $\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}$. We begin by selecting the rows corresponding to $2k_p+1$, $p \in \left\{0,\ldots,2^{\widetilde{K}_R-1}-1\right\}$, and the columns corresponding to $2l_q+1$, $q \in \left\{0,\ldots,2^{\widetilde{K}_C-1}-1\right\}$. This yields

$$
\begin{aligned}
&\sqrt{M^{[j]}N^{[j]}}\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p+1,\,2l_q+1}\right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}\\
&= \frac{(-1)^{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}}}{\sqrt{2^{-\widetilde{K}_R-\widetilde{K}_C+4}}}\left(\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}}\operatorname{diag}\left(\mathbf{s}_R^{[j]}\right)+\mathbf{S}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}}\operatorname{diag}\left(\mathbf{c}_R^{[j]}\right)\right)\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}}\\
&\quad\cdot\left(\operatorname{diag}\left(\mathbf{s}_C^{[j]}\right)\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}+\operatorname{diag}\left(\mathbf{c}_C^{[j]}\right)\mathbf{S}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\right)\\
&= \frac{(-1)^{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}}}{\sqrt{2^{-\widetilde{K}_R-\widetilde{K}_C+4}}}\left(\begin{array}{c|c}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}}\end{array}\right)\\
&\quad\cdot\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}_R^{[j]}\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_R-1}}\operatorname{diag}\left(\mathbf{c}_R^{[j]}\right)\end{pmatrix}\begin{pmatrix}\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}}\\ \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}_C^{[j]}\right) & \\ & \operatorname{diag}\left(\mathbf{c}_C^{[j]}\right)\mathbf{D}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\\ \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}.
\end{aligned}
\tag{7.51}
$$

In order to obtain an invertible matrix factorization, we now have to use both additional rows and columns. First, we choose the same rows as before and the columns corresponding to the indices $2l'_q+1$, $q \in \left\{0,\ldots,2^{\widetilde{K}_C-1}-1\right\}$. Analogously to previously considered cases we find that

$$
\begin{aligned}
&(-1)^{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}}\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}}\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p+1,\,2l'_q+1}\right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}\\
&= \left(\begin{array}{c|c}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}}\end{array}\right)\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}_R^{[j]}\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_R-1}}\operatorname{diag}\left(\mathbf{c}_R^{[j]}\right)\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}}\\ \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}_C^{[j]}\right) & \\ & \operatorname{diag}\left(\mathbf{c}_C^{[j]}\right)\mathbf{D}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\\ -\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}.
\end{aligned}
\tag{7.52}
$$

Taking now the rows corresponding to $2k'_p+1$, $p \in \left\{0,\ldots,2^{\widetilde{K}_R-1}-1\right\}$, and the columns

indexed by $2l_q+1$ yields

$$
\begin{aligned}
&(-1)^{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}}\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}}\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k'_p+1,\,2l_q+1}\right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}\\
&=\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & -\mathbf{J}_{2^{\widetilde{K}_R-1}}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}}\end{pmatrix}\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}_R^{[j]}\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_R-1}}\operatorname{diag}\left(\mathbf{c}_R^{[j]}\right)\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\\ \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}_C^{[j]}\right) & \\ & \operatorname{diag}\left(\mathbf{c}_C^{[j]}\right)\mathbf{D}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\\ \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}.
\end{aligned}
\tag{7.53}
$$

Finally, for the rows indexed by $2k'_p+1$ and the columns indexed by $2l'_q+1$, we obtain

$$
\begin{aligned}
&(-1)^{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}}\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}}\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k'_p+1,\,2l'_q+1}\right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}\\
&=\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & -\mathbf{J}_{2^{\widetilde{K}_R-1}}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}}\end{pmatrix}\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}_R^{[j]}\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_R-1}}\operatorname{diag}\left(\mathbf{c}_R^{[j]}\right)\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\\ \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}_C^{[j]}\right) & \\ & \operatorname{diag}\left(\mathbf{c}_C^{[j]}\right)\mathbf{D}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\\ -\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}.
\end{aligned}
\tag{7.54}
$$

Combining (7.51) to (7.54), we find that

$$
\begin{aligned}
&\begin{pmatrix}\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p+1,\,2l_q+1}\right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1} & \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p+1,\,2l'_q+1}\right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}\\ \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k'_p+1,\,2l_q+1}\right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1} & \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k'_p+1,\,2l'_q+1}\right)_{p,\,q=0}^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}\end{pmatrix}\\
&=\frac{(-1)^{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}}}{\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C+4}}}\begin{pmatrix}\mathbf{I}_{2^{\widetilde{K}_R-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\\ \mathbf{I}_{2^{\widetilde{K}_R-1}} & -\mathbf{J}_{2^{\widetilde{K}_R-1}}\end{pmatrix}\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}}\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}_R^{[j]}\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_R-1}}\operatorname{diag}\left(\mathbf{c}_R^{[j]}\right)\end{pmatrix}\begin{pmatrix}\mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\\ \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\operatorname{diag}\left(\mathbf{s}_C^{[j]}\right) & \\ & \operatorname{diag}\left(\mathbf{c}_C^{[j]}\right)\mathbf{D}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\begin{pmatrix}\mathbf{I}_{2^{\widetilde{K}_C-1}} & \mathbf{I}_{2^{\widetilde{K}_C-1}}\\ \mathbf{J}_{2^{\widetilde{K}_C-1}} & -\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix}.
\end{aligned}
\tag{7.55}
$$

We denote the required samples of $\mathbf{A}^{\widehat{\mathrm{II}}}$ by

$$\mathbf{E}^{(1,1)}_{(0,0)} \coloneqq \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p+1,\,2l_q+1}\right)^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}_{p,\,q=0},$$
$$\mathbf{E}^{(1,1)}_{(0,1)} \coloneqq \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p+1,\,2l'_q+1}\right)^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}_{p,\,q=0},$$
$$\mathbf{E}^{(1,1)}_{(1,0)} \coloneqq \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k'_p+1,\,2l_q+1}\right)^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}_{p,\,q=0},$$
$$\mathbf{E}^{(1,1)}_{(1,1)} \coloneqq \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k'_p+1,\,2l'_q+1}\right)^{2^{\widetilde{K}_R-1}-1,\,2^{\widetilde{K}_C-1}-1}_{p,\,q=0}.$$

As all square matrices in (7.55) are invertible, we obtain that

$$\begin{aligned}
&\begin{pmatrix} \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}} \\ \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_R-1}}\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}\mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \\
&= (-1)^{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}}\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}+4}\begin{pmatrix} \operatorname{diag}\left(\tilde{\mathbf{s}}^{[j]}_R\right) & \\ & \operatorname{diag}\left(\tilde{\mathbf{c}}^{[j]}_R\right)\mathbf{D}_{2^{\widetilde{K}_R-1}} \end{pmatrix} \\
&\quad\cdot\begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} \end{pmatrix}\frac{1}{2}\begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}_R-1}} & \mathbf{I}_{2^{\widetilde{K}_R-1}} \\ \mathbf{J}_{2^{\widetilde{K}_R-1}} & -\mathbf{J}_{2^{\widetilde{K}_R-1}} \end{pmatrix}\begin{pmatrix} \mathbf{E}^{(1,1)}_{(0,0)} & \mathbf{E}^{(1,1)}_{(0,1)} \\ \mathbf{E}^{(1,1)}_{(1,0)} & \mathbf{E}^{(1,1)}_{(1,1)} \end{pmatrix} \\
&\quad\cdot\frac{1}{2}\begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_C-1}} \\ \mathbf{I}_{2^{\widetilde{K}_C-1}} & -\mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix}\cdot\begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \end{pmatrix}\begin{pmatrix} \operatorname{diag}\left(\tilde{\mathbf{s}}^{[j]}_C\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_C-1}}\operatorname{diag}\left(\tilde{\mathbf{c}}^{[j]}_C\right) \end{pmatrix} \\
&= (-1)^{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}}\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}} \\
&\quad\cdot\begin{pmatrix} \operatorname{diag}\left(\tilde{\mathbf{s}}^{[j]}_R\right)\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} & \\ & \operatorname{diag}\left(\tilde{\mathbf{c}}^{[j]}_R\right)\mathbf{D}_{2^{\widetilde{K}_R-1}}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}} \end{pmatrix}\begin{pmatrix} \widetilde{\mathbf{E}}^{(1,1)}_{(0,0)} & \widetilde{\mathbf{E}}^{(1,1)}_{(0,1)} \\ \widetilde{\mathbf{E}}^{(1,1)}_{(1,0)} & \widetilde{\mathbf{E}}^{(1,1)}_{(1,1)} \end{pmatrix} \\
&\quad\cdot\begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\operatorname{diag}\left(\tilde{\mathbf{s}}^{[j]}_C\right) & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\mathbf{D}_{2^{\widetilde{K}_C-1}}\operatorname{diag}\left(\tilde{\mathbf{c}}^{[j]}_C\right) \end{pmatrix},
\end{aligned} \tag{7.56}$$

where we set

$$\begin{aligned}
\widetilde{\mathbf{E}}^{(1,1)}_{(0,0)} &\coloneqq \mathbf{E}^{(1,1)}_{(0,0)}+\mathbf{E}^{(1,1)}_{(1,0)}+\mathbf{E}^{(1,1)}_{(0,1)}+\mathbf{E}^{(1,1)}_{(1,1)},\\
\widetilde{\mathbf{E}}^{(1,1)}_{(0,1)} &\coloneqq \left(\mathbf{E}^{(1,1)}_{(0,0)}+\mathbf{E}^{(1,1)}_{(1,0)}-\mathbf{E}^{(1,1)}_{(0,1)}-\mathbf{E}^{(1,1)}_{(1,1)}\right)\mathbf{J}_{2^{\widetilde{K}_C-1}},\\
\widetilde{\mathbf{E}}^{(1,1)}_{(1,0)} &\coloneqq \mathbf{J}_{2^{\widetilde{K}_R-1}}\left(\mathbf{E}^{(1,1)}_{(0,0)}-\mathbf{E}^{(1,1)}_{(1,0)}+\mathbf{E}^{(1,1)}_{(0,1)}-\mathbf{E}^{(1,1)}_{(1,1)}\right),\\
\widetilde{\mathbf{E}}^{(1,1)}_{(1,1)} &\coloneqq \mathbf{J}_{2^{\widetilde{K}_R-1}}\left(\mathbf{E}^{(1,1)}_{(0,0)}-\mathbf{E}^{(1,1)}_{(1,0)}-\mathbf{E}^{(1,1)}_{(0,1)}+\mathbf{E}^{(1,1)}_{(1,1)}\right)\mathbf{J}_{2^{\widetilde{K}_C-1}}.
\end{aligned}$$

It suffices to just consider the bottom-right quadrant of (7.56) in order to compute $\widetilde{\mathbf{B}}^{[j+1]}_{(1,1)}$,

which is numerically more stable. Thus, we find that

$$\begin{aligned}\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]} = (-1)^{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}}\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-\widetilde{K}_C}}\,\mathbf{J}_{2^{\widetilde{K}_R-1}}\operatorname{diag}\left(\tilde{\mathbf{c}}_R^{[j]}\right)\mathbf{D}_{2^{\widetilde{K}_R-1}}\\ \cdot\,\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}}\mathbf{J}_{2^{\widetilde{K}_R-1}}\left(\mathbf{E}^{(1,1)}_{(0,0)}-\mathbf{E}^{(1,1)}_{(1,0)}-\mathbf{E}^{(1,1)}_{(0,1)}+\mathbf{E}^{(1,1)}_{(1,1)}\right)\mathbf{J}_{2^{\widetilde{K}_C-1}}\\ \cdot\,\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\mathbf{D}_{2^{\widetilde{K}_C-1}}\operatorname{diag}\left(\tilde{\mathbf{c}}_C^{[j]}\right)\mathbf{J}_{2^{\widetilde{K}_C-1}}.\end{aligned} \tag{7.57}$$

Consequently, $\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}$ can be calculated using $2^{\widetilde{K}_R}\cdot 2^{\widetilde{K}_C}$ samples of $\mathbf{A}^{\widehat{\Pi}}$ by essentially applying a 2-dimensional DCT-IV of size $2^{\widetilde{K}_R-1}\times 2^{\widetilde{K}_C-1}$.

(iv) Computation of $\mathbf{A}^{[j+1]}$.

The matrix $\mathbf{B}^{[j]}$ is given as a restriction of $\mathbf{A}^{[j]}$, and the restricted matrices $\widetilde{\mathbf{B}}_{(0,1)}^{[j+1]}$, $\widetilde{\mathbf{B}}_{(1,0)}^{[j+1]}$ and $\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}$ can be obtained from (7.42), (7.49) and (7.57). Thus, we can compute $\mathbf{B}_{(0,0)}^{[j+1]}$, $\mathbf{B}_{(0,1)}^{[j+1]}$, $\mathbf{B}_{(1,0)}^{[j+1]}$ and $\mathbf{B}_{(1,1)}^{[j+1]}$ via (7.32), as it also holds for the restrictions of the matrices $\mathbf{A}^{[j+1]}$ and $\tilde{\mathbf{A}}^{[j+1]}$, with

$$\begin{aligned}\mathbf{B}_{(0,0)}^{[j+1]} &= \frac{1}{4}\left(\mathbf{B}^{[j]}+\widetilde{\mathbf{B}}_{(0,1)}^{[j+1]}+\widetilde{\mathbf{B}}_{(1,0)}^{[j+1]}+\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\right),\\ \mathbf{B}_{(0,1)}^{[j+1]} &= \frac{1}{4}\left(\mathbf{B}^{[j]}-\widetilde{\mathbf{B}}_{(0,1)}^{[j+1]}+\widetilde{\mathbf{B}}_{(1,0)}^{[j+1]}-\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\right)\mathbf{J}_{N^{[j]}},\\ \mathbf{B}_{(1,0)}^{[j+1]} &= \frac{1}{4}\mathbf{J}_{M^{[j]}}\left(\mathbf{B}^{[j]}+\widetilde{\mathbf{B}}_{(0,1)}^{[j+1]}-\widetilde{\mathbf{B}}_{(1,0)}^{[j+1]}-\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\right),\\ \mathbf{B}_{(1,1)}^{[j+1]} &= \frac{1}{4}\mathbf{J}_{M^{[j]}}\left(\mathbf{B}^{[j]}-\widetilde{\mathbf{B}}_{(0,1)}^{[j+1]}-\widetilde{\mathbf{B}}_{(1,0)}^{[j+1]}+\widetilde{\mathbf{B}}_{(1,1)}^{[j+1]}\right)\mathbf{J}_{N^{[j]}}.\end{aligned} \tag{7.58}$$

Then $\mathbf{A}^{[j+1]}$ is given by

$$a_{k,l}^{[j+1]} = \begin{cases} \left(b_{(0,0)}^{[j+1]}\right)_{k-M^{[j]}+2^{\widetilde{K}_R-1},\,l-N^{[j]}+2^{\widetilde{K}_C-1}} & \text{if } k\in I_{M^{[j]}-2^{\widetilde{K}_R-1},\,M^{[j]}-1},\\ & \quad l\in I_{N^{[j]}-2^{\widetilde{K}_C-1},\,N^{[j]}-1},\\ \left(b_{(0,1)}^{[j+1]}\right)_{k-M^{[j]}+2^{\widetilde{K}_R-1},\,l-N^{[j]}} & \text{if } k\in I_{M^{[j]}-2^{\widetilde{K}_R-1},\,M^{[j]}-1},\\ & \quad l\in I_{N^{[j]},\,N^{[j]}+2^{\widetilde{K}_C-1}-1},\\ \left(b_{(1,0)}^{[j+1]}\right)_{k-M^{[j]},\,l-N^{[j]}+2^{\widetilde{K}_C-1}} & \text{if } k\in I_{M^{[j]},\,M^{[j]}+2^{\widetilde{K}_R-1}-1},\\ & \quad l\in I_{N^{[j]}-2^{\widetilde{K}_C-1},\,N^{[j]}},\\ \left(b_{(1,1)}^{[j+1]}\right)_{k-M^{[j]},\,l-N^{[j]}} & \text{if } k\in I_{M^{[j]},\,M^{[j]}+2^{\widetilde{K}_R-1}-1},\\ & \quad l\in I_{N^{[j]},\,N^{[j]}+2^{\widetilde{K}_C-1}-1},\\ 0 & \text{otherwise.}\end{cases}$$

Consequently, reconstructing $\mathbf{A}^{[j+1]}$ in case A of Theorem 7.14 requires

$$2^{\widetilde{K}_R-1}\cdot 2^{\widetilde{K}_C}+2^{\widetilde{K}_R}\cdot 2^{\widetilde{K}_C-1}+2^{\widetilde{K}_R}\cdot 2^{\widetilde{K}_C} = 8\cdot 2^{\widetilde{K}_R-1}\cdot 2^{\widetilde{K}_C-1}$$

entries of $\left(\mathbf{A}^{[j+1]}\right)^{\widehat{\Pi}}$. We will show in Section 7.3.5 how to detect the exact first and last row and column support indices efficiently. □

7.3.3 Recovery Procedure for Case B: Colliding Columns

If $j = j_2$, where j_2 is defined as in (7.8), the column support $S_C^{[j]}$ of $\mathbf{A}^{[j]}$ is completely contained in the last b_C columns, but the row support $S_R^{[j]}$ is not contained in the last b_R rows, i.e.,

$$S_C^{[j]} \subseteq I_{N^{[j]}-b_C,\,N^{[j]}-1} \quad \text{and} \quad S_R^{[j]} \not\subseteq I_{M^{[j]}-b_R,\,M^{[j]}-1}.$$

Then it follows from Theorem 7.14, case B that

$$S^{[j+1]} \subsetneq I_{\mu_R^{[j]},\,\nu_R^{[j]}} \times I_{N^{[j]}-b_C,\,N^{[j]}+b_C-1} \qquad \text{or}$$
$$S^{[j+1]} \subsetneq I_{M^{[j+1]}-1-\nu_R^{[j]},\,M^{[j+1]}-1-\mu_R^{[j]}} \times I_{N^{[j]}-b_C,\,N^{[j]}+b_C-1},$$

see also Figure 7.12.

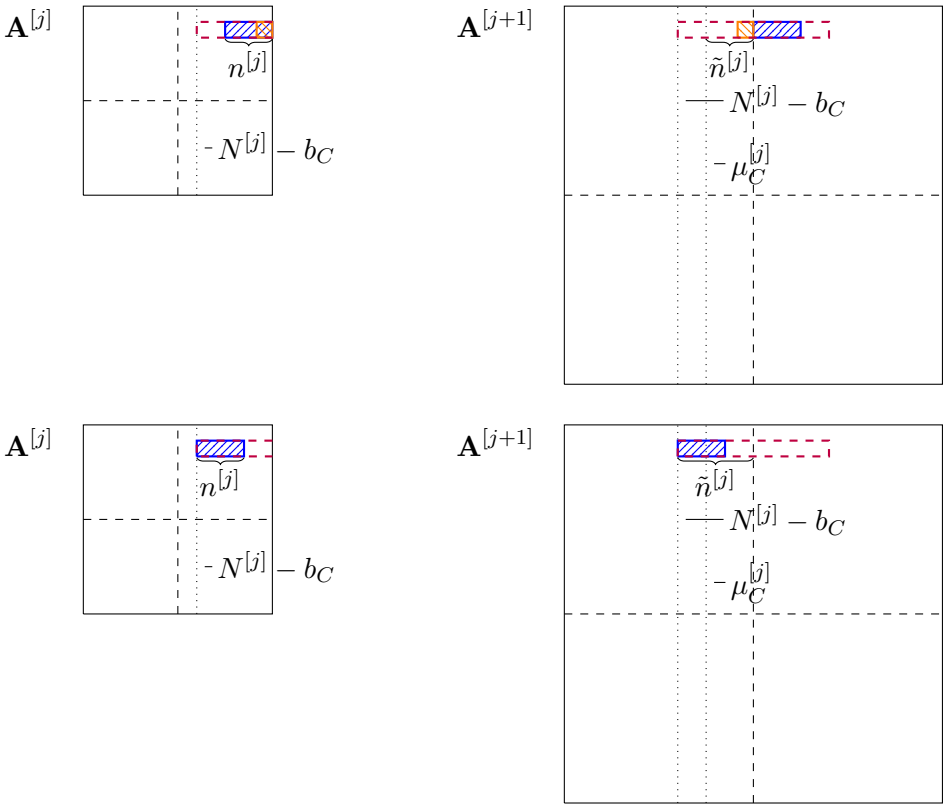

Figure 7.12: Illustration of the support of $\mathbf{A}^{[j]}$ and one possibility for the support of $\mathbf{A}^{[j+1]}$ for $m^{[j]} < m^{[j+1]}$ (top) and $m^{[j]} = m^{[j+1]}$ (bottom)

Hence, there are two possibilities for the support of the rows of $\mathbf{A}^{[j+1]}$, as $S^{[j+1]}$ is either contained in the upper or in the lower half of $\mathbf{A}^{[j+1]}$. Furthermore, we have to undo collisions of nonzero entries, so we do not exactly know the column support of $\mathbf{A}^{[j+1]}$. The following theorem shows how $\mathbf{A}^{[j+1]}$ can be reconstructed from $\mathbf{A}^{[j]}$ and $\mathbf{A}^{\widehat{\Pi}}$ by combining some of the ideas used for cases A and D of Theorem 7.14 with an additional new approach.

Theorem 7.19 *Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$, and let $J \coloneqq \min\{J_R, J_C\}$. Let $\mathbf{A} \in \mathbb{R}^{M\times N}$ have a block support of size $m\times n$ with known bounds $b_R \geq m$ and $b_C \geq n$,*

and assume that $\mathbf{A}$ *satisfies (7.5) and (7.6). Let* $L \coloneqq \max\{\lceil \log_2 b_R \rceil + 1, \lceil \log_2 b_C \rceil + 1\}$ *and* $j = j_2 \in \{L, \dots, J-1\}$ *as in (7.8). Suppose that we have access to all entries of* $\mathbf{A}^{\widehat{\Pi}}$*. Then* $\mathbf{A}^{[j+1]}$ *can be recovered from* $\mathbf{A}^{[j]}$*,* $2^{K_R^{[j]}} \cdot 2^{\widetilde{K}_C}$ *entries of* $\mathbf{A}^{\widehat{\Pi}}$ *and one nonzero entry of* $\left(\left(a^{[j+1]}\right)^{\widehat{\Pi}}_{2k+1,\,2l} \right)_{k,\,l=0}^{m^{[j]}-1,\,n^{[j]}-1}$*, where* $K_R^{[j]} \in \left\{ \lceil \log_2 m \rceil + 1, \dots, \log_2 M^{[j]} \right\}$.

Proof. By assumption and Lemma 7.12, $\mathbf{A}^{[j]}$ has the block support

$$S^{[j]} = S_R^{[j]} \times S_C^{[j]} = I_{\mu_R^{[j]}, \nu_R^{[j]}} \times I_{\mu_C^{[j]}, \nu_C^{[j]}} \subseteq I_{\mu_R^{[j]}, \nu_R^{[j]}} \times I_{N^{[j]}-b_C,\, N^{[j]}-1} \tag{7.59}$$

of size $m^{[j]} \times n^{[j]}$. Recall that it follows from (7.30) that

$$\mathbf{P}_{M^{[j+1]}} \left(\mathbf{A}^{[j+1]} \right)^{\widehat{\Pi}} \mathbf{P}_{N^{[j+1]}}{}^T = \frac{1}{2} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}} \mathbf{A}^{[j]} \mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^T & \mathbf{C}^{\mathrm{II}}_{M^{[j]}} \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} \mathbf{C}^{\mathrm{II}}_{N^{[j]}}{}^T & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \tilde{\mathbf{A}}^{[j+1]}_{(1,1)} \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix}. \tag{7.60}$$

With

$$\tilde{n}^{[j]} \coloneqq N^{[j]} - \mu_C^{[j]} \leq b_C \quad \text{and} \quad \widetilde{K}_C \coloneqq \left\lceil \log_2 \tilde{n}^{[j]} \right\rceil + 1,$$

as in Section 7.3.2, we obtain that

$$S_C^{[j]} \subseteq I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}-1}.$$

Since $j = j_2$, it follows from Definition 7.5 and (7.59) that

$$\begin{aligned} &S^{[j+1]} \subsetneq I_{\mu_R^{[j]}, \nu_R^{[j]}} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}+2^{\widetilde{K}_C-1}-1} \qquad \text{or} \\ &S^{[j+1]} \subsetneq I_{M^{[j+1]}-1-\nu_R^{[j]},\, M^{[j+1]}-1-\mu_R^{[j]}} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}+2^{\widetilde{K}_C-1}-1}, \end{aligned} \tag{7.61}$$

so either $\mathbf{A}^{[j+1]}_{(1,0)} = \mathbf{0}$ and $\mathbf{A}^{[j+1]}_{(1,1)} = \mathbf{0}$, or $\mathbf{A}^{[j+1]}_{(0,0)} = \mathbf{0}$ and $\mathbf{A}^{[j+1]}_{(0,1)} = \mathbf{0}$, see also Figure 7.7. Consequently, there are precisely two possibilities for the support rows of $\mathbf{A}^{[j+1]}$, but we can only deduce that the column support is contained in an interval of length $2^{\widetilde{K}_C}$ starting at $\mu_C^{[j]}$. If the lower half of $\mathbf{A}^{[j+1]}$ is zero, (7.31) and (7.32) can be simplified to

$$\begin{aligned} \mathbf{A}^{[j]} = \tilde{\mathbf{A}}^{[j+1]}_{(0,0)} &= \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{A}^{[j+1]}_{(0,1)} \mathbf{J}_{N^{[j]}} = \tilde{\mathbf{A}}^{[j+1]}_{(1,0)}, \\ \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} &= \mathbf{A}^{[j+1]}_{(0,0)} - \mathbf{A}^{[j+1]}_{(0,1)} \mathbf{J}_{N^{[j]}} = \tilde{\mathbf{A}}^{[j+1]}_{(1,1)}, \end{aligned} \tag{7.62}$$

and

$$\begin{aligned} \mathbf{A}^{[j+1]}_{(0,0)} &= \frac{1}{2} \left(\mathbf{A}^{[j]} + \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} \right), \\ \mathbf{A}^{[j+1]}_{(0,1)} &= \frac{1}{2} \left(\mathbf{A}^{[j]} - \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} \right) \mathbf{J}_{N^{[j]}}. \end{aligned} \tag{7.63}$$

If the top half of $\mathbf{A}^{[j+1]}$ is zero, we find that

$$\begin{aligned} \mathbf{A}^{[j]} = \tilde{\mathbf{A}}^{[j+1]}_{(0,0)} &= \mathbf{J}_{M^{[j]}} \mathbf{A}^{[j+1]}_{(1,0)} + \mathbf{J}_{M^{[j]}} \mathbf{A}^{[j+1]}_{(1,1)} \mathbf{J}_{N^{[j]}} = -\widetilde{\mathbf{A}}^{[j+1]}_{(1,0)}, \\ \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} &= \mathbf{J}_{M^{[j]}} \mathbf{A}^{[j+1]}_{(1,0)} - \mathbf{J}_{M^{[j]}} \mathbf{A}^{[j+1]}_{(1,1)} \mathbf{J}_{N^{[j]}} = -\widetilde{\mathbf{A}}^{[j+1]}_{(1,1)}, \end{aligned} \tag{7.64}$$

and

$$\begin{aligned}\mathbf{A}_{(1,0)}^{[j+1]} &= \frac{1}{2}\mathbf{J}_{M^{[j]}}\left(\mathbf{A}^{[j]} + \tilde{\mathbf{A}}_{(0,1)}^{[j+1]}\right),\\ \mathbf{A}_{(1,1)}^{[j+1]} &= \frac{1}{2}\mathbf{J}_{M^{[j]}}\left(\mathbf{A}^{[j]} - \tilde{\mathbf{A}}_{(0,1)}^{[j+1]}\right)\mathbf{J}_{N^{[j]}}.\end{aligned} \tag{7.65}$$

Note that (7.63) and (7.65) only differ by a row permutation $\mathbf{J}_{M^{[j]}}$. Provided that we can determine whether the support of $\mathbf{A}^{[j+1]}$ is contained in its upper or in its lower half, it suffices to recover $\tilde{\mathbf{A}}_{(0,1)}^{[j+1]}$ in order to be able to compute $\mathbf{A}^{[j+1]}$, as $\mathbf{A}^{[j]}$ is already known from the previous iteration step. Thus, we need to derive a method for reconstructing $\tilde{\mathbf{A}}_{(0,1)}^{[j+1]}$ from $\mathbf{A}^{\widehat{\mathrm{II}}}$ and $\mathbf{A}^{[j]}$, and a method to decide whether the support of $\mathbf{A}^{[j+1]}$ is contained in its upper or in its lower half.

Let us first focus on the reconstruction of $\tilde{\mathbf{A}}_{(0,1)}^{[j+1]}$. It follows from (7.3) and (7.30) that

$$\left(\left(a^{[j+1]}\right)_{2k,\,2l+1}^{\widehat{\mathrm{II}}}\right)_{k,\,l=0}^{M^{[j]}-1,\,N^{[j]}-1} = \frac{1}{2}\mathbf{C}_{M^{[j]}}^{\mathrm{II}}\tilde{\mathbf{A}}_{(0,1)}^{[j+1]}\mathbf{C}_{N^{[j]}}^{\mathrm{IV}}. \tag{7.66}$$

Similarly to the approach taken in Sections 6.3.1 and 7.3.2, we want to reduce the number of rows and columns of $\mathbf{C}_{M^{[j]}}^{\mathrm{II}}$ and $\mathbf{C}_{N^{[j]}}^{\mathrm{IV}}$ required in (7.66), using that $\tilde{\mathbf{A}}_{(0,1)}$ has a block support of size $m^{[j]} \times n^{[j]}$. Thus, we want to obtain an invertible matrix factorization consisting of matrices of size $\mathcal{O}\left({m^{[j]}}^2\right)$ and $\mathcal{O}\left({n^{[j]}}^2\right)$. By definition, the supports of $\tilde{\mathbf{A}}_{(0,1)}^{[j+1]}$ and $\mathbf{A}^{[j]}$ are the same, so, denoting by $S(\mathbf{B})$ the support of a matrix $\mathbf{B}$, we have

$$S\left(\tilde{\mathbf{A}}_{(0,1)}^{[j+1]}\right) \subseteq I_{\mu_R^{[j]},\,\nu_R^{[j]}} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\,N^{[j]}-1}.$$

Since the support of $\mathbf{A}^{[j]}$ is contained in its last b_C columns, we already know from Sections 6.3.1 and 7.3.2 how to restrict the rows and columns of $\mathbf{C}_{N^{[j]}}^{\mathrm{IV}}$. Recall that

$$K_R = \left\lceil \log_2 m^{[j]} \right\rceil + 1 \leq \widetilde{K}_R.$$

Ideally, we would like to subdivide the interval $I_{0,\,M^{[j]}-1}$ into $M^{[j]} \cdot 2^{-K_R}$ intervals of length $2^{K_R} = \mathcal{O}\left(m^{[j]}\right)$ of the form

$$I_{d_R^{[j]} \cdot 2^{K_R},\,\left(d_R^{[j]}+1\right)\cdot 2^{K_R}-1}.$$

If the rows of $\mathbf{A}^{[j]}$ are contained in such a 2^{K_R}-length interval, the multiplication by the cosine matrix of type II in (7.66) can be reduced to a multiplication by invertible matrices of size $2^{K_R} \times 2^{K_R}$, similarly to the restrictions we have seen in Section 7.3.2. However, the row support $S_R^{[j]}$ of $\mathbf{A}^{[j]}$ does not have to be contained in one of these intervals of length 2^{K_R}, for example if $\left\{\frac{M^{[j]}}{2}-1, \frac{M^{[j]}}{2}\right\} \subseteq S_R^{[j]}$. We have not been able to find an invertible matrix factorization of size $2^{K^R} \times 2^{K_R}$ by restricting the rows and columns of $\mathbf{C}_{M^{[j]}}^{\mathrm{II}}$ in (7.66) that correspond to the nonzero entries of $\tilde{\mathbf{A}}_{(0,1)}^{[j+1]}$ and $\mathbf{A}^{[j]}$. Shifting the above intervals by, e.g., 2^{K_R-1}, also does not provide the sought-after factorization.

Instead, we have to content ourselves with an approach that requires more arithmetical operations. Note that there exists a minimal $K_R^{[j]} \in \left\{K_R, \ldots, \log_2 M^{[j]}\right\}$ satisfying that

there is an index $d_R^{[j]} \in \left\{0, \dots, M^{[j]} 2^{-K_R^{[j]}} - 1\right\}$ such that

$$S_R^{[j]} \subsetneq I_{d_R^{[j]} \cdot 2^{K_R^{[j]}}, \left(d_R^{[j]}+1\right) \cdot 2^{K_R^{[j]}} - 1}, \tag{7.67}$$

i.e.,

$$K_R^{[j]} \coloneqq \min_{K \in \left\{K_R, \dots, \log_2 M^{[j]}\right\}} \left\{ \exists\, d_R^{[j]} \in I_{0,\, M^{[j]} 2^{K_R^{[j]}} - 1} : S_R^{[j]} \subsetneq I_{d_R^{[j]} \cdot 2^{K_R^{[j]}}, \left(d_R^{[j]}+1\right) \cdot 2^{K_R^{[j]}} - 1} \right\},$$

Thus, instead of always subdividing $I_{0,\, M^{[j]}-1}$ into intervals of length $2^{K_R} = \mathcal{O}\left(m^{[j]}\right)$, we will divide it into intervals of length $2^{K_R^{[j]}}$, where $K_R^{[j]}$ depends on the location of the row support of $\mathbf{A}^{[j]}$ and $\tilde{\mathbf{A}}_{(0,1)}^{[j+1]}$. The drawback of this approach is that we do not have any a priori knowledge on $K_R^{[j]}$. Both the optimal case $K_R^{[j]} = K_R$ and the worst case $K_R^{[j]} = M^{[j]}$ are possible. Further, j_2 can be as small as L, but also as large as $J-1$, and we do not have any a priori estimates on it. For computing the theoretical runtime of the approach we will present hereafter we can thus only estimate $2^{K_R^{[j]}} = \mathcal{O}\left(\frac{M}{2}\right)$. This value is attained if the support of $\mathbf{A}$ is contained both in its upper and in its lower half, as well as in its last b_C columns, i.e., if

$$I_{\frac{M}{2}-1,\, \frac{M}{2}} \subseteq S_R^{[J]} \qquad \text{and} \qquad S_C^{[J]} \subseteq I_{N-1-b_C,\, N-1},$$

because then $j_2 = J-1$ and $2^{K_R^{[j_2]}} = \frac{M}{2}$. See Figures 7.13 and 7.14 for illustrations.

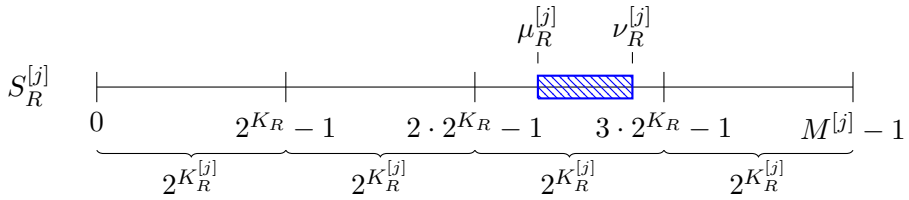

Figure 7.13: Illustration of the subdivision of $I_{0,\, M^{[j]}-1}$ for $2^{K_R^{[j]}} = 2^{K_R} = \frac{1}{4} M^{[j]}$ with $d_R^{[j]} = 2$

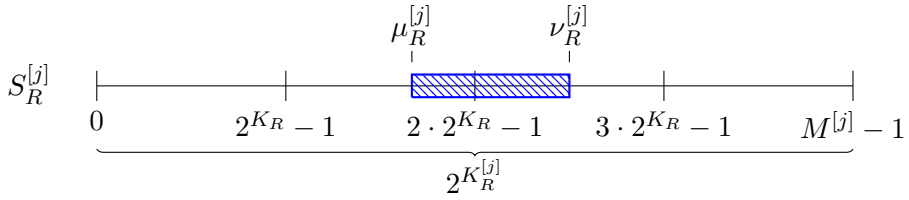

Figure 7.14: Illustration of the subdivision of $I_{0,\, M^{[j]}-1}$ for $2^{K_R^{[j]}} = M^{[j]}$ with $d_R^{[j]} = 0$

Since

$$S^{[j]} = S\left(\tilde{\mathbf{A}}_{0,1}^{[j+1]}\right) \subsetneq I_{d_R^{[j]}\cdot 2^{K_R^{[j]}},\left(d_R^{[j]}+1\right)\cdot 2^{K_R^{[j]}}-1} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\,N^{[j]}-1},$$

let us define the restriction of $\tilde{\mathbf{A}}_{(0,1)}^{[j+1]}$ to the $d_R^{[j]}$th set of $2^{K_R^{[j]}}$ rows and the last $2^{\widetilde{K}_C-1}$ columns as

$$\widetilde{\mathbf{B}}_C^{[j+1]} = \left(\left(\tilde{a}_{(0,1)}^{[j+1]}\right)_{k,\,l}\right)_{k=d_R^{[j]}2^{K_R^{[j]}},\,l=N^{[j]}-2^{\widetilde{K}_C-1}}^{\left(d_R^{[j]}+1\right)2^{K_R^{[j]}}-1,\,N^{[j]}-1}.$$

Utilizing this restriction in the upper right quadrant of (7.60), we obtain from (7.66) and (7.36) that

$$\begin{aligned}
&\sqrt{2M^{[j]}}\left(\left(a^{[j+1]}\right)_{2k,\,2l+1}^{\widehat{\mathrm{II}}}\right)_{k,\,l=0}^{M^{[j]}-1,\,N^{[j]}-1}\\
&= \left(\sum_{r'=0}^{M^{[j]}-1} \varepsilon_{M^{[j]}}(k)\cos\left(\frac{k(2r'+1)\pi}{2M^{[j]}}\right)\left(\tilde{a}_{(0,1)}^{[j+1]}\right)_{r',\,l}\right)_{k,\,l=0}^{M^{[j]}-1,\,N^{[j]}-1}\mathbf{C}_{N^{[j]}}^{\mathrm{IV}}\\
&= \left(\sum_{r'=d_R^{[j]}2^{K_R^{[j]}}}^{\left(d_R^{[j]}+1\right)2^{K_R^{[j]}}-1} \varepsilon_{M^{[j]}}(k)\cos\left(\frac{k(2r'+1)\pi}{2M^{[j]}}\right)\left(\tilde{a}_{(0,1)}^{[j+1]}\right)_{r',\,l}\right)_{k,\,l=0}^{M^{[j]}-1,\,N^{[j]}-1}\mathbf{C}_{N^{[j]}}^{\mathrm{IV}}\\
&= \left(\sum_{r=0}^{2^{K_R^{[j]}}-1} \varepsilon_{M^{[j]}}(k)\cos\left(\frac{k\left(\left(d_R^{[j]}+1\right)2^{K_R^{[j]}+1}-(2r+1)\right)\pi}{2M^{[j]}}\right)\right.\\
&\quad\left.\cdot\left(\tilde{a}_{(0,1)}^{[j+1]}\right)_{\left(d^{[j]}+1\right)2^{K_R^{[j]}}-1-r,\,l}\right)_{k,\,l=0}^{M^{[j]}-1,\,N^{[j]}-1}\mathbf{C}_{N^{[j]}}^{\mathrm{IV}}\\
&= \sqrt{\frac{2}{N^{[j]}}}\left(\sum_{r=0}^{2^{K_R^{[j]}}-1}\sum_{s=0}^{2^{\widetilde{K}_{C^{-1}}}-1} \varepsilon_{M^{[j]}}(k)\left(\cos\left(\frac{k\left(d_R^{[j]}+1\right)\pi}{M^{[j]}2^{-K_R^{[j]}}}\right)\cos\left(\frac{k(2r+1)\pi}{2M^{[j]}}\right)\right.\right.\\
&\quad\left.+\sin\left(\frac{k\left(d_R^{[j]}+1\right)\pi}{M^{[j]}2^{-K_R^{[j]}}}\right)\sin\left(\frac{k(2r+1)\pi}{2M^{[j]}}\right)\right)\left(\mathbf{J}_{2^{K_R^{[j]}}}\widetilde{\mathbf{B}}_C^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_{C^{-1}}}}\right)_{r,\,s}\\
&\quad\left.\cdot(-1)^l\sin\left(\frac{(2s+1)(2l+1)\pi}{4N^{[j]}}\right)\right)_{k,\,l=0}^{M^{[j]}-1,\,N^{[j]}-1},
\end{aligned}\tag{7.68}$$

where $r \coloneqq \left(d_R^{[j]}+1\right)2^{K_R^{[j]}}-1-r'$. As $\widetilde{\mathbf{B}}_C^{[j+1]}$ is of size $2^{K_R^{[j]}}\times 2^{\widetilde{K}_C-1}$, it suffices to use $2^{K_R^{[j]}}$ rows and $2^{\widetilde{K}_C-1}$ columns of (7.68) for its reconstruction. We choose the rows indexed by $2k_p'''$, where $k_p''' \coloneqq M^{[j]}2^{-K_R^{[j]}}p$, $p \in \left\{0,\ldots,2^{K_R^{[j]}}-1\right\}$, since $k_p''' \in \left\{0,\ldots,M^{[j]}-1\right\}$ for all p. As in Sections 6.3.1 and 7.3.2, we use the $2^{\widetilde{K}_C-1}$ columns of (7.68) that correspond

to $2l_q+1$, $q \in \left\{0,\ldots,2^{\widetilde{K}_C-1}-1\right\}$. Using (7.37) and (7.38), this implies

$$
\begin{aligned}
&(-1)^{N^{[j]}2^{-\widetilde{K}_C}}\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_C+2}}\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p''',\,2l_q+1}\right)^{2^{K_R^{[j]}}-1,\,2^{\widetilde{K}_C-1}-1}_{p,\,q=0}\\
&= \left((-1)^{k_p'''}\varepsilon_{M^{[j]}}(k_p''')\left(\cos\left(\frac{M^{[j]}2^{-K_R^{[j]}}p\left(d_R^{[j]}+1\right)\pi}{M^{[j]}2^{-K_R^{[j]}}}\right)\cos\left(\frac{M^{[j]}2^{-K_R^{[j]}}p(2r+1)\pi}{2M^{[j]}}\right)\right.\right.\\
&\quad\left.\left.+\ \sin\left(\frac{M^{[j]}2^{-K_R^{[j]}}p\left(d_R^{[j]}+1\right)\pi}{M^{[j]}2^{-K_R^{[j]}}}\right)\sin\left(\frac{M^{[j]}2^{-K_R^{[j]}}p(2r+1)\pi}{2M^{[j]}}\right)\right)\right)^{2^{K_R^{[j]}}-1}_{p,\,r=0}\\
&\quad\cdot\mathbf{J}_{2^{K_R^{[j]}}}\widetilde{\mathbf{B}}_C^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}}\left(\mathrm{diag}\left(\mathbf{s}_C^{[j]}\right)\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}+\mathrm{diag}\left(\mathbf{c}_C^{[j]}\right)\mathbf{S}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\right)\\
&= \left((-1)^{k_p'''}\varepsilon_{M^{[j]}}(k_p''')\left(\cos\left(p\left(d_R^{[j]}+1\right)\pi\right)\cos\left(\frac{p(2r+1)\pi}{2\cdot 2^{-K_R^{[j]}}}\right)\right.\right.\\
&\quad\left.\left.+\ \sin\left(p\left(d_R^{[j]}+1\right)\pi\right)\sin\left(\frac{p(2r+1)\pi}{2\cdot 2^{-K_R^{[j]}}}\right)\right)\right)^{2^{K_R^{[j]}}-1}_{p,\,r=0}\\
&\quad\cdot\mathbf{J}_{2^{K_R^{[j]}}}\widetilde{\mathbf{B}}_C^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}}\left(\mathrm{diag}\left(\mathbf{s}_C^{[j]}\right)\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}+\mathrm{diag}\left(\mathbf{c}_C^{[j]}\right)\mathbf{S}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\right)\\
&= \left((-1)^{\left(d_R^{[j]}+1+M^{[j]}2^{-K_R^{[j]}}\right)p}\varepsilon_{2^{K_R^{[j]}}}(p)\cos\left(\frac{p(2r+1)\pi}{2\cdot 2^{K_R^{[j]}}}\right)\right)^{2^{K_R^{[j]}}-1}_{p,\,r=0}\mathbf{J}_{2^{K_R^{[j]}}}\widetilde{\mathbf{B}}_C^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}}\\
&\quad\cdot\left(\mathrm{diag}\left(\mathbf{s}_C^{[j]}\right)\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}+\mathrm{diag}\left(\mathbf{c}_C^{[j]}\right)\mathbf{S}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\right)\\
&= \sqrt{2^{K_R^{[j]}-1}}\,\mathrm{diag}\left((-1)^{\left(d_R^{[j]}+1+M^{[j]}2^{-K_R^{[j]}}\right)p}\right)^{2^{K_R^{[j]}}-1}_{p=0}\mathbf{C}^{\mathrm{II}}_{2^{K_R^{[j]}}}\\
&\quad\cdot\left(\ \mathbf{J}_{2^{K_R^{[j]}}}\widetilde{\mathbf{B}}_C^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}}\ \middle|\ \mathbf{J}_{2^{K_R^{[j]}}}\widetilde{\mathbf{B}}_C^{[j+1]}\mathbf{J}_{2^{\widetilde{K}_C-1}}\ \right)\begin{pmatrix}\mathrm{diag}\left(\mathbf{s}_C^{[j]}\right) & \\ & \mathrm{diag}\left(\mathbf{c}_C^{[j]}\right)\mathbf{D}_{2^{\widetilde{K}_C-1}}\end{pmatrix}\\
&\quad\cdot\begin{pmatrix}\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\\ \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}\mathbf{J}_{2^{\widetilde{K}_C-1}}\end{pmatrix},
\end{aligned}
\tag{7.69}
$$

where $\mathbf{s}_C^{[j]}$ and $\mathbf{c}_C^{[j]}$ are defined in Section 7.3.2. As in Sections 6.3.1 and 7.3.2, we also need to consider the columns corresponding to the indices $2l_q'+1$, $q \in \left\{0,\ldots,2^{\widetilde{K}_C-1}-1\right\}$, while using the same $2^{K_R^{[j]}}$ rows as before. It follows from (7.39) that

$$
\begin{aligned}
&(-1)^{N^{[j]}2^{-\widetilde{K}_C}}\sqrt{M^{[j]}N^{[j]}2^{-K_R^{[j]}-\widetilde{K}_C+3}}\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p''',\,2l_q'+1}\right)^{2^{K_R^{[j]}}-1,\,2^{\widetilde{K}_C-1}-1}_{p,\,q=0}\\
&= \mathrm{diag}\left((-1)^{\left(d_R^{[j]}+1+M^{[j]}2^{-K_R^{[j]}}\right)p}\right)^{2^{K_R^{[j]}}-1}_{p=0}\mathbf{C}^{\mathrm{II}}_{2^{K_R^{[j]}}}
\end{aligned}
$$

$$\cdot \left(\begin{array}{c|c} \mathbf{J}_{2^{K_R^{[j]}}} \widetilde{\mathbf{B}}_C^{[j+1]} \mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{K_R^{[j]}}} \widetilde{\mathbf{B}}_C^{[j+1]} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{array} \right) \begin{pmatrix} \operatorname{diag}\left(\mathbf{s}_C^{[j]}\right) & \\ & \operatorname{diag}\left(\mathbf{c}_C^{[j]}\right) \mathbf{D}_{2^{\widetilde{K}_C-1}} \end{pmatrix}$$

$$\cdot \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \\ -\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix}. \tag{7.70}$$

Combining (7.69) and (7.70), we find that

$$\begin{aligned} &(-1)^{N^{[j]}2^{-\widetilde{K}_C}} \sqrt{M^{[j]} N^{[j]} 2^{-K_R^{[j]} - \widetilde{K}_C + 3}} \\ &\cdot \left(\begin{array}{c|c} \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p''', 2l_q+1} \right)_{p,\,q=0}^{2^{K_R^{[j]}}-1,\, 2^{\widetilde{K}_C-1}-1} & \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p''', 2l_q'+1} \right)_{p,\,q=0}^{2^{K_R^{[j]}}-1,\, 2^{\widetilde{K}_C-1}-1} \end{array} \right) \\ &= \operatorname{diag}\left((-1)^{\left(d_R^{[j]} + 1 + M^{[j]} 2^{-K_R^{[j]}}\right) p} \right)_{p=0}^{2^{K_R^{[j]}}-1} \mathbf{C}^{\mathrm{II}}_{2^{K_R^{[j]}}} \\ &\cdot \left(\begin{array}{c|c} \mathbf{J}_{2^{K_R^{[j]}}} \widetilde{\mathbf{B}}_C^{[j+1]} \mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{K_R^{[j]}}} \widetilde{\mathbf{B}}_C^{[j+1]} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{array} \right) \begin{pmatrix} \operatorname{diag}\left(\mathbf{s}_C^{[j]}\right) & \\ & \operatorname{diag}\left(\mathbf{c}_C^{[j]}\right) \mathbf{D}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \\ &\cdot \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}_C-1}} & \mathbf{I}_{2^{\widetilde{K}_C-1}} \\ \mathbf{J}_{2^{\widetilde{K}_C-1}} & -\mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix}. \end{aligned} \tag{7.71}$$

Let us denote the required samples of $\mathbf{A}^{\widehat{\mathrm{II}}}$ by

$$\mathbf{E}^{(C)}_{(0,0)} \coloneqq \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p''', 2l_q+1} \right)_{p,\,q=0}^{2^{K_R^{[j]}}-1,\, 2^{\widetilde{K}_C-1}-1} \qquad \text{and}$$

$$\mathbf{E}^{(C)}_{(0,1)} \coloneqq \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p''', 2l_q'+1} \right)_{p,\,q=0}^{2^{K_R^{[j]}}-1,\, 2^{\widetilde{K}_C-1}-1}.$$

As all square matrices in (7.71) are invertible, see (7.41), it follows that

$$\begin{aligned} &\left(\begin{array}{c|c} \mathbf{J}_{2^{K_R^{[j]}}} \widetilde{\mathbf{B}}_C^{[j+1]} \mathbf{J}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{K_R^{[j]}}} \widetilde{\mathbf{B}}_C^{[j+1]} \mathbf{J}_{2^{\widetilde{K}_C-1}} \end{array} \right) \\ &= (-1)^{N^{[j]}2^{-\widetilde{K}_C}} \sqrt{M^{[j]} N^{[j]} 2^{-K_R^{[j]} - \widetilde{K}_C + 3}}\, \mathbf{C}^{\mathrm{III}}_{2^{K_R^{[j]}}} \operatorname{diag}\left((-1)^{\left(d_R^{[j]} + 1 + M^{[j]} 2^{-K_R^{[j]}}\right) p} \right)_{p=0}^{2^{K_R^{[j]}}-1} \\ &\cdot \left(\begin{array}{c|c} \mathbf{E}^{(C)}_{(0,0)} & \mathbf{E}^{(C)}_{(0,1)} \end{array} \right) \frac{1}{2} \begin{pmatrix} \mathbf{I}_{2^{\widetilde{K}_C-1}} & \mathbf{J}_{2^{\widetilde{K}_C-1}} \\ \mathbf{I}_{2^{\widetilde{K}_C-1}} & -\mathbf{J}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \end{pmatrix} \\ &\cdot \begin{pmatrix} \operatorname{diag}\left(\tilde{\mathbf{s}}_C^{[j]}\right) & \\ & \mathbf{D}_{2^{\widetilde{K}_C-1}} \operatorname{diag}\left(\tilde{\mathbf{c}}_C^{[j]}\right) \end{pmatrix} \end{aligned}$$

$$= (-1)^{N^{[j]}2^{-\widetilde{K}_C}} \sqrt{M^{[j]}N^{[j]}2^{-K_R^{[j]}-\widetilde{K}_C+1}}\, \mathbf{C}^{\mathrm{III}}_{2^{K_R^{[j]}}} \operatorname{diag}\left((-1)^{\left(d_R^{[j]}+1+M^{[j]}2^{-K_R^{[j]}}\right)p}\right)_{p=0}^{2^{K_R^{[j]}}-1}$$
$$\cdot \left(\mathbf{E}^{(C)}_{(0,0)} + \mathbf{E}^{(C)}_{(0,1)} \;\middle|\; \left(\mathbf{E}^{(C)}_{(0,0)} - \mathbf{E}^{(C)}_{(0,1)}\right) \mathbf{J}_{2^{\widetilde{K}_C-1}} \right)$$
$$\cdot \begin{pmatrix} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \operatorname{diag}\left(\tilde{\mathbf{s}}_C^{[j]}\right) & \\ & \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}} \mathbf{D}_{2^{\widetilde{K}_C-1}} \operatorname{diag}\left(\tilde{\mathbf{c}}_C^{[j]}\right) \end{pmatrix}. \tag{7.72}$$

Here, it suffices to take the last $2^{\widetilde{K}_C-1}$ columns of (7.72) for computing $\widetilde{\mathbf{B}}_C^{[j+1]}$, so we find

$$\widetilde{\mathbf{B}}_C^{[j+1]} = (-1)^{N^{[j]}2^{-\widetilde{K}_C}} \sqrt{M^{[j]}N^{[j]}2^{-K_R^{[j]}-\widetilde{K}_C+1}}\, \mathbf{J}_{2^{K_R^{[j]}}} \mathbf{C}^{\mathrm{III}}_{2^{K_R^{[j]}}}$$
$$\cdot \operatorname{diag}\left((-1)^{\left(d_R^{[j]}+1+M^{[j]}2^{-K_R^{[j]}}\right)p}\right)_{p=0}^{2^{K_R^{[j]}}-1} \left(\mathbf{E}^{(C)}_{(0,0)} - \mathbf{E}^{(C)}_{(0,1)}\right) \mathbf{J}_{2^{\widetilde{K}_C-1}} \mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_C-1}}$$
$$\cdot \mathbf{D}_{2^{\widetilde{K}_C-1}} \operatorname{diag}\left(\tilde{\mathbf{c}}_C^{[j]}\right) \mathbf{J}_{2^{\widetilde{K}_C-1}}. \tag{7.73}$$

Using (7.73), we can compute

$$\left(\tilde{a}^{[j+1]}_{(0,1)}\right)_{k,l} = \begin{cases} \left(\tilde{b}_C^{[j+1]}\right)_{k-d_R^{[j]}2^{K_R^{[j]}},\, l-N^{[j]}+2^{\widetilde{K}_C-1}} & \text{if } k \in I_{d_R^{[j]}2^{K_R^{[j]}},\left(d_R^{[j]}+1\right)2^{K_R^{[j]}}-1}, \\ & \quad l \in I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}-1}, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, $\widetilde{\mathbf{B}}_C^{[j+1]}$ can be calculated using $2^{K_R^{[j]}} \cdot 2^{\widetilde{K}_C}$ samples of $\mathbf{A}^{\widehat{\Pi}}$ by essentially applying $2^{\widetilde{K}_C-1}$ 1-dimensional DCT-IIIs of length $2^{K_R^{[j]}}$ to the columns of the sample matrix and $2^{K_R^{[j]}}$ 1-dimensional DCT-IVs of length $2^{\widetilde{K}_C-1}$ to the rows.

We still need to determine whether the support of $\mathbf{A}^{[j+1]}$ is completely contained in its upper or in its lower half. By Theorem 7.14, case B and (7.62) to (7.65), there are precisely two possibilities for $\mathbf{A}^{[j+1]}$, either

$$\mathbf{V}^{(0)} := \frac{1}{2} \left(\begin{array}{c|c} \mathbf{A}^{[j]} + \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} & \left(\mathbf{A}^{[j]} - \tilde{\mathbf{A}}^{[j+1]}_{(0,1)}\right) \mathbf{J}_{N^{[j]}} \\ \hline \mathbf{0} & \mathbf{0} \end{array} \right) \qquad \text{or}$$

$$\mathbf{V}^{(1)} := \frac{1}{2} \left(\begin{array}{c|c} \mathbf{0} & \mathbf{0} \\ \hline \mathbf{J}_{M^{[j]}} \left(\mathbf{A}^{[j]} + \tilde{\mathbf{A}}^{[j+1]}_{(0,1)}\right) & \mathbf{J}_{M^{[j]}} \left(\mathbf{A}^{[j]} - \tilde{\mathbf{A}}^{[j+1]}_{(0,1)}\right) \mathbf{J}_{N^{[j]}} \end{array} \right).$$

Let us define

$$\mathbf{X}^{(+)} := \mathbf{A}^{[j]} + \tilde{\mathbf{A}}^{[j+1]}_{(0,1)} \quad \text{and} \quad \mathbf{X}^{(-)} := \mathbf{A}^{[j]} - \tilde{\mathbf{A}}^{[j+1]}_{(0,1)},$$

and, analogously to Section 7.3.1, compare the DCT-IIs of $\mathbf{V}^{(0)}$ and $\mathbf{V}^{(1)}$. Remark 7.7

yields that

$$\begin{aligned}
&\mathbf{P}_{M^{[j+1]}} \left(\mathbf{V}^{(0)}\right)^{\widehat{\mathrm{II}}} {\mathbf{P}_{N^{[j+1]}}}^T \\
&= \frac{1}{2} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}} \left(\mathbf{X}^{(+)} + \mathbf{X}^{(-)}\right) {\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^T & \mathbf{C}^{\mathrm{II}}_{M^{[j]}} \left(\mathbf{X}^{(+)} - \mathbf{X}^{(-)}\right) \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \left(\mathbf{X}^{(+)} + \mathbf{X}^{(-)}\right) {\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^T & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \left(\mathbf{X}^{(+)} - \mathbf{X}^{(-)}\right) \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix}
\end{aligned}$$

and

$$\begin{aligned}
&\mathbf{P}_{M^{[j+1]}} \left(\mathbf{V}^{(1)}\right)^{\widehat{\mathrm{II}}} {\mathbf{P}_{N^{[j+1]}}}^T \\
&= \frac{1}{2} \begin{pmatrix} \mathbf{C}^{\mathrm{II}}_{M^{[j]}} \left(\mathbf{X}^{(+)} + \mathbf{X}^{(-)}\right) {\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^T & \mathbf{C}^{\mathrm{II}}_{M^{[j]}} \left(\mathbf{X}^{(+)} - \mathbf{X}^{(-)}\right) \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \\ \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \left(-\mathbf{X}^{(+)} - \mathbf{X}^{(-)}\right) {\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^T & \mathbf{C}^{\mathrm{IV}}_{M^{[j]}} \left(-\mathbf{X}^{(+)} + \mathbf{X}^{(-)}\right) \mathbf{C}^{\mathrm{IV}}_{N^{[j]}} \end{pmatrix}.
\end{aligned}$$

Consequently, it follows that

$$\left(v^{(1)}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l} = -\left(v^{(0)}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l} \tag{7.74}$$

for all $k \in \{0, \ldots, M^{[j]} - 1\}$ and $l \in \{0, \ldots, N^{[j]} - 1\}$. Analogously to Section 7.3.1, we can determine whether $\mathbf{V}^{(0)}$ or $\mathbf{V}^{(1)}$ is the correct matrix $\mathbf{A}^{[j+1]}$ by comparing a nonzero entry of $\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0}$ to the corresponding entry of $\left(\mathbf{V}^{(0)}\right)^{\widehat{\mathrm{II}}}$. However, we first need to show that such a nonzero entry can be found efficiently.

Assume that there does not exist such an entry, i.e., that

$$\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0} = \mathbf{0}.$$

We know from (7.23) in Section 7.3.1 that

$$\begin{aligned}
\mathbf{0} &= \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)^{m^{[j]}-1,\,n^{[j]}-1}_{k,\,l=0} \\
&= \frac{1}{\sqrt{M^{[j]} N^{[j]}}} \mathbf{X}^{[j]}_{\mathrm{odd},\,r} \mathbf{V}^{\mathrm{odd}} \left(\left(t_{M^{[j+1]},\,r}\right)_{r \in S^{[j]}_R}\right)^T \\
&\quad \cdot \left(\left(\tilde{a}^{[j+1]}_{(1,0)}\right)_{r,\,s}\right)_{r \in S^{[j]}_R,\, s \in S^{[j]}_C} \mathbf{V} \left(\left(t_{N^{[j]},\,s}\right)_{s \in S^{[j]}_C}\right) {\mathbf{X}^{[j]}_C}^T \operatorname{diag}\left(\left(\varepsilon_{N^{[j]}}(l)\right)^{n^{[j]}-1}_{l=0}\right) \\
\Leftrightarrow \quad \mathbf{0} &= \left(\left(\tilde{a}^{[j+1]}_{(1,0)}\right)_{r,\,s}\right)_{r \in S^{[j]}_R,\, s \in S^{[j]}_C}.
\end{aligned} \tag{7.75}$$

Since $j = j_2$, the support of $\mathbf{A}^{[j+1]}$ is either completely contained in its upper half or in its lower half. If $S^{[j+1]}$ is contained in its upper half, i.e., if $\mathbf{A}^{[j+1]}_{(1,0)} = \mathbf{0}$ and $\mathbf{A}^{[j+1]}_{(1,1)} = \mathbf{0}$, recall that by (7.62)

$$\tilde{\mathbf{A}}^{[j+1]}_{(1,0)} = \mathbf{A}^{[j+1]}_{(0,0)} + \mathbf{A}^{[j+1]}_{(0,1)} \mathbf{J}_{N^{[j]}} = \mathbf{A}^{[j]}.$$

If the support of $\mathbf{A}^{[j+1]}$ is completely contained in its lower half, i.e., if $\mathbf{A}^{[j+1]}_{(0,0)} = \mathbf{0}$ and

$\mathbf{A}^{[j+1]}_{(0,1)} = \mathbf{0}$, then, by (7.64),

$$\tilde{\mathbf{A}}^{[j+1]}_{(1,0)} = -\,\mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,0)} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\mathbf{J}_{N^{[j]}} = -\mathbf{A}^{[j]}.$$

Hence, we have that either $\tilde{\mathbf{A}}^{[j+1]}_{(1,0)} = \mathbf{A}^{[j]}$ or $\tilde{\mathbf{A}}^{[j+1]}_{(1,0)} = -\mathbf{A}^{[j]}$, which implies that (7.75) is equivalent to $\mathbf{A}^{[j]} = \mathbf{0}$. This is a contradiction to (7.5) and (7.6) and the fact that the matrix $\mathbf{A}$ has a block support of size $m \times n$. Consequently, there exists an index pair $(k_C,\, l_C) \in I_{0,\,m^{[j]}-1} \times I_{0,\,n^{[j]}-1}$ such that $\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_C+1,\,2l_C} \neq 0$.

For a stable and efficient implementation of this procedure using Lemma 7.8 we set

$$(k_C,\, l_C) \coloneqq \underset{(k,l)\in I_{0,\,m^{[j]}-1}\times I_{0,\,n^{[j]}-1}}{\operatorname{argmax}} \left\{ \left| 2^{J-j-1} a^{\widehat{\mathrm{II}}}_{2^{J-j-1}(2k+1),\,2^{J-j}l} \right| \right\}.$$

Further, we also need to compute the corresponding entry of $\left(\mathbf{V}^{(0)}\right)^{\widehat{\mathrm{II}}}$. By construction, the support of $\mathbf{V}^{(0)}$ is contained in

$$I_{\mu_R^{[j]},\,\nu_R^{[j]}} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\,N^{[j]}+2^{\widetilde{K}_C-1}-1}.$$

Thus, the required entry of $\left(\mathbf{V}^{(0)}\right)^{\widehat{\mathrm{II}}}$ satisfies

$$\begin{aligned}\left(v^{(0)}\right)^{\widehat{\mathrm{II}}}_{2k_C+1,\,2l_C} &= \left(\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\mathbf{V}^{(0)}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T}\right)_{2k_C+1,\,2l_C} \\ &= \sum_{r=\mu_R^{[j]}}^{\nu_R^{[j]}} \sum_{s=N^{[j]}-2^{\widetilde{K}_C-1}}^{N^{[j]}+2^{\widetilde{K}_C-1}-1} \left(\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\right)_{2k_C+1,\,r} v^{(0)}_{r,\,s} \left({\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T}\right)_{s,\,2l_C}\end{aligned}$$

and can be computed using $\mathcal{O}\left(m^{[j]} \cdot 2^{\widetilde{K}_C}\right)$ operations.

By (7.74), we have that $\mathbf{A}^{[j+1]} = \mathbf{V}^{(0)}$ if

$$\left(v^{(0)}\right)^{\widehat{\mathrm{II}}}_{2k_C+1,\,2l_C} = \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_C+1,\,2l_C},$$

and $\mathbf{A}^{[j+1]} = \mathbf{V}^{(1)}$ if

$$\left(v^{(0)}\right)^{\widehat{\mathrm{II}}}_{2k_C+1,\,2l_C} = -\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_C+1,\,2l_C}.$$

Numerically, we define

$$\delta_C^- \coloneqq \left| \left(v^{(0)}\right)^{\widehat{\mathrm{II}}}_{2k_C+1,\,2l_C} - \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_C+1,\,2l_C} \right| \qquad \text{and}$$
$$\delta_C^+ \coloneqq \left| \left(v^{(0)}\right)^{\widehat{\mathrm{II}}}_{2k_C+1,\,2l_C} + \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_C+1,\,2l_C} \right|,$$

and set

$$\mathbf{A}^{[j+1]} = \begin{cases} \mathbf{V}^{(0)} & \text{if } \delta_C^- < \delta_C^+, \\ \mathbf{V}^{(1)} & \text{if } \delta_C^- > \delta_C^+. \end{cases}$$

Further, the first row support index of $\mathbf{A}^{[j+1]}$ is given by

$$\mu_R^{[j+1]} := \begin{cases} \mu_R^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{V}^{(0)}, \\ M^{[j+1]} - 1 - \nu_R^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{V}^{(1)}. \end{cases} \tag{7.76}$$

Note that $m^{[j+1]} = m^{[j]}$, but that we only know that

$$S_C^{[j+1]} \subseteq I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}+2^{\widetilde{K}_C-1}-1} \tag{7.77}$$

and $n^{[j]} < n^{[j+1]}$. As in Chapter 6 we have to detect the exact first and last column support indices by examining which of the entries corresponding to the indices in (7.77) are nonzero. In Section 7.3.5 we will thus give a 2-dimensional analog to Remark 6.18. □

7.3.4 Recovery Procedure for Case C: Colliding Rows

If $j = j_3$, where j_3 is defined as in (7.9), the row support $S_R^{[j]}$ of $\mathbf{A}^{[j]}$ is completely contained in the last b_R columns, but the column support $S_C^{[j]}$ is not contained in the last b_C columns, i.e.,

$$S_R^{[j]} \subseteq I_{M^{[j]}-b_R,\, M^{[j]}-1} \quad \text{and} \quad S_C^{[j]} \not\subseteq I_{N^{[j]}-b_C,\, N^{[j]}}.$$

Then it follows from Theorem 7.14, case C that

$$\begin{aligned} &S^{[j+1]} \subsetneq I_{M^{[j]}-b_R,\, M^{[j]}+b_R-1} \times I_{\mu_C^{[j]},\, \nu_C^{[j]}} && \text{or} \\ &S^{[j+1]} \subsetneq I_{M^{[j]}-b_R,\, M^{[j]}+b_R-1} \times I_{N^{[j+1]}-1-\nu_C^{[j]},\, N^{[j+1]}-1-\mu_C^{[j]}}, \end{aligned}$$

see also Figure 7.15.

Analogously to Section 7.3.3, there are two possibilities for the column support $S_C^{[j+1]}$ of $\mathbf{A}^{[j+1]}$, as $S^{[j+1]}$ is either contained in the left or the right half of $\mathbf{A}^{[j+1]}$. Due to collision of nonzero entries, the row support $S_R^{[j+1]}$ is not known exactly. Switching the roles of rows and columns, we can proceed similarly to Section 7.3.3. The main theorem for reconstructing $\mathbf{A}^{[j+1]}$ from $\mathbf{A}^{[j]}$ and $\mathbf{A}^{\widehat{\Pi}}$ in case C of Theorem 7.14 presents itself as an analog to Theorem 7.19. We will briefly sketch the proof to introduce the setup and notation necessary for formulating the complete 2-dimensional IDCT-II algorithm for block sparse matrices in Section 7.4.

Theorem 7.20 *Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$, and let $J := \min\{J_R, J_C\}$. Let $\mathbf{A} \in \mathbb{R}^{M\times N}$ have a block support of size $m\times n$ with known bounds $b_R \geq m$ and $b_C \geq n$, and assume that $\mathbf{A}$ satisfies (7.5) and (7.6). Let $L := \max\{\lceil \log_2 b_R \rceil + 1, \lceil \log_2 b_C \rceil + 1\}$ and $j = j_3 \in \{L, \dots, J-1\}$ as in (7.9). Suppose that we have access to all entries of $\mathbf{A}^{\widehat{\Pi}}$. Then $\mathbf{A}^{[j+1]}$ can be recovered from $\mathbf{A}^{[j]}$, $2^{\widetilde{K}_R} \cdot 2^{K_C^{[j]}}$ entries of $\mathbf{A}^{\widehat{\Pi}}$ and one nonzero entry of $\left(\left(a^{[j+1]}\right)^{\widehat{\Pi}}_{2k,\,2l+1}\right)_{k,\,l=0}^{m^{[j]}-1,\, n^{[j]}-1}$, where $K_C^{[j]} \in \left\{\lceil \log_2 n \rceil + 1, \dots, \log_2 N^{[j]}\right\}$.*

Proof. By assumption and Lemma 7.12, $\mathbf{A}^{[j]}$ has the block support

$$S^{[j]} = S_R^{[j]} \times S_C^{[j]} = I_{\mu_R^{[j]},\, \nu_R^{[j]}} \times I_{\mu_C^{[j]},\, \nu_C^{[j]}} \subseteq I_{M^{[j]}-b_R,\, M^{[j]}-1} \times I_{\mu_C^{[j]},\, \nu_C^{[j]}}$$

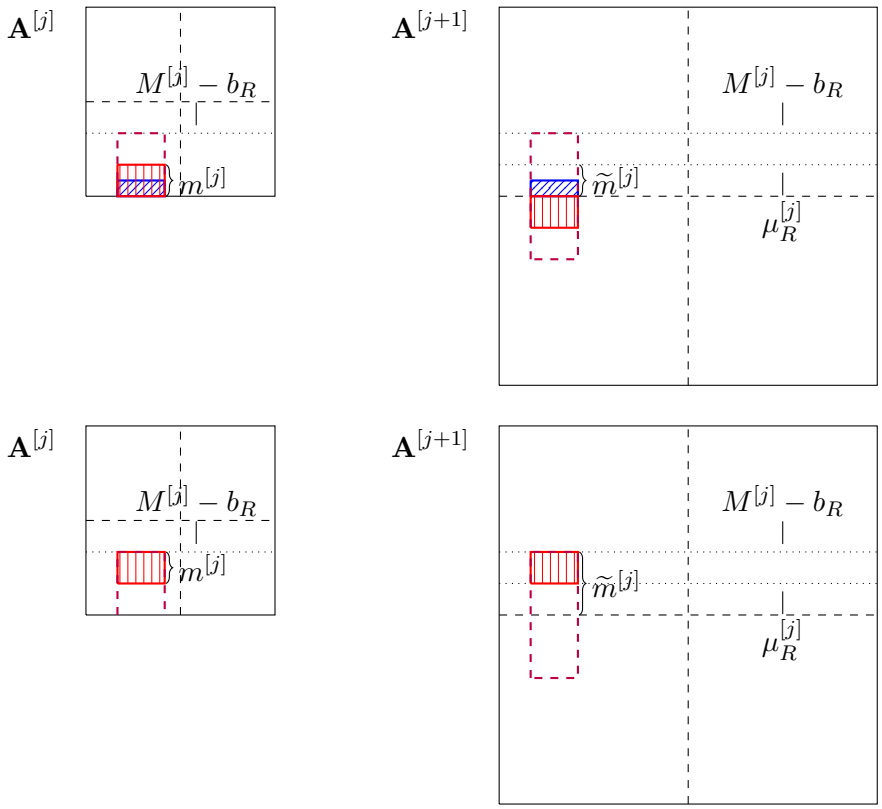

Figure 7.15: Illustration of the support of $\mathbf{A}^{[j]}$ and one possibility for the support of $\mathbf{A}^{[j+1]}$ for $n^{[j]} < n^{[j+1]}$ (top) and $n^{[j]} = n^{[j+1]}$ (bottom)

of size $m^{[j]} \times n^{[j]}$. Analogously to Section 7.3.3, we now have that

$$\begin{aligned} S^{[j+1]} &\subsetneq I_{M^{[j]}-2^{\tilde{K}_R-1},\, M^{[j]}+2^{\tilde{K}_R-1}-1} \times I_{\mu_C^{[j]},\, \nu_C^{[j]}} \qquad \text{or} \\ S^{[j+1]} &\subsetneq I_{M^{[j]}-2^{\tilde{K}_R-1},\, M^{[j]}+2^{\tilde{K}_R-1}-1} \times I_{N^{[j+1]}-1-\nu_C^{[j]},\, N^{[j+1]}-1-\mu_C^{[j]}}. \end{aligned} \tag{7.78}$$

If the right half of $\mathbf{A}^{[j+1]}$ is zero, (7.31) can be simplified to

$$\begin{aligned} \mathbf{A}^{[j]} = \tilde{\mathbf{A}}_{(0,0)}^{[j+1]} &= \mathbf{A}_{(0,0)}^{[j+1]} + \mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,0)}^{[j+1]} = \tilde{\mathbf{A}}_{(0,1)}^{[j+1]} \quad \text{and} \\ \tilde{\mathbf{A}}_{(1,0)}^{[j+1]} &= \mathbf{A}_{(0,0)}^{[j+1]} - \mathbf{J}_{M^{[j]}} \mathbf{A}_{(1,0)}^{[j+1]} = \tilde{\mathbf{A}}_{(1,1)}^{[j+1]}. \end{aligned} \tag{7.79}$$

Hence, it follows from (7.32) that

$$\mathbf{A}^{[j+1]} = \frac{1}{2} \left(\begin{array}{c|c} \mathbf{A}^{[j]} + \tilde{\mathbf{A}}_{(1,0)}^{[j+1]} & \mathbf{0} \\ \hline \mathbf{J}_{M^{[j]}} \left(\mathbf{A}^{[j]} - \tilde{\mathbf{A}}_{(1,0)}^{[j+1]} \right) & \mathbf{0} \end{array} \right) =: \mathbf{W}^{(0)}. \tag{7.80}$$

If the left half of $\mathbf{A}^{[j+1]}$ is zero, we find that

$$\begin{aligned}\mathbf{A}^{[j]} = \tilde{\mathbf{A}}^{[j+1]}_{(0,0)} &= \mathbf{A}^{[j+1]}_{(0,1)}\mathbf{J}_{N^{[j]}} + \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\mathbf{J}_{N^{[j]}} = -\tilde{\mathbf{A}}^{[j+1]}_{(0,1)},\\ \tilde{\mathbf{A}}^{[j+1]}_{(1,0)} &= \mathbf{A}^{[j+1]}_{(0,1)}\mathbf{J}_{N^{[j]}} - \mathbf{J}_{M^{[j]}}\mathbf{A}^{[j+1]}_{(1,1)}\mathbf{J}_{N^{[j]}} = -\tilde{\mathbf{A}}^{[j+1]}_{(1,1)},\end{aligned} \tag{7.81}$$

and

$$\mathbf{A}^{[j+1]} = \frac{1}{2}\left(\begin{array}{c|c} \mathbf{0} & \left(\mathbf{A}^{[j]} + \tilde{\mathbf{A}}^{[j+1]}_{(1,0)}\right)\mathbf{J}_{N^{[j]}} \\ \hline \mathbf{0} & \mathbf{J}_{M^{[j]}}\left(\mathbf{A}^{[j]} - \tilde{\mathbf{A}}^{[j+1]}_{(1,0)}\right)\mathbf{J}_{N^{[j]}} \end{array}\right) =: \mathbf{W}^{(1)}. \tag{7.82}$$

Similarly to case B, it suffices to recover $\tilde{\mathbf{A}}^{[j+1]}_{(1,0)}$ in order to compute $\mathbf{A}^{[j+1]}$ if we can determine whether the support of $\mathbf{A}^{[j+1]}$ is contained in the left or right half. Recall that by (7.60)

$$\left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k+1,\,2l}\right)^{M^{[j]}-1,\,N^{[j]}-1}_{k,\,l=0} = \frac{1}{2}\mathbf{C}^{\mathrm{IV}}_{M^{[j]}}\tilde{\mathbf{A}}^{[j+1]}_{(1,0)}{\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T}. \tag{7.83}$$

Using a subdivision of $I_{0,\,N^{[j]}-1}$ into intervals of length $2^{K_C^{[j]}}$, analogously to (7.67), with

$$K_C^{[j]} = \min_{K\in\left\{K_C,\ldots,\log_2 N^{[j]}\right\}}\left\{\exists\, d_C^{[j]} \in I_{0,\,N^{[j]}2^{K_C^{[j]}}-1} : S_C^{[j]} \subsetneq I_{d_C^{[j]}\cdot 2^{K_C^{[j]}},\,\left(d_C^{[j]}+1\right)\cdot 2^{K_C^{[j]}}-1}\right\},$$

we define the restriction

$$\widetilde{\mathbf{B}}^{[j+1]}_R := \left(\left(\tilde{a}^{[j+1]}_{(1,0)}\right)_{k,\,l}\right)^{M^{[j]}-1,\,\left(d_C^{[j]}+1\right)2^{K_C^{[j]}-1}}_{k=M^{[j]}-2^{\widetilde{K}_R-1},\,l=d_C^{[j]}2^{K_C^{[j]}}}$$

of $\tilde{\mathbf{A}}^{[j+1]}_{(1,0)}$ to the last $2^{\widetilde{K}_R-1}$ rows and the $d_C^{[j]}$th set of $2^{K_C^{[j]}}$ columns. Then, analogously to (7.68) to (7.73), where we use the $2^{K_C^{[j]}}$ columns of (7.83) indexed by $l'''_q := N^{[j]}2^{-K_C^{[j]}}q$, $q \in \left\{0,\ldots,2^{K_C^{[j]}}-1\right\}$, and the rows corresponding to k_p and k''_p, we find that

$$\begin{aligned}\widetilde{\mathbf{B}}^{[j+1]}_R =& (-1)^{M^{[j]}2^{-\widetilde{K}_R}}\sqrt{M^{[j]}N^{[j]}2^{-\widetilde{K}_R-K_C^{[j]}+1}}\,\mathbf{J}_{2^{\widetilde{K}_R-1}}\,\mathrm{diag}\left(\tilde{\mathbf{c}}^{[j]}_R\right)\mathbf{D}_{2^{\widetilde{K}_R-1}}\\ &\cdot\mathbf{C}^{\mathrm{IV}}_{2^{\widetilde{K}_R-1}}\mathbf{J}_{2^{\widetilde{K}_R-1}}\left(\mathbf{E}^{(R)}_{(0,0)} - \mathbf{E}^{(R)}_{(1,0)}\right)\left(\mathrm{diag}(-1)^{\left(d_C^{[j]}+1+N^{[j]}2^{-K_C^{[j]}}\right)q}\right)^{2^{K_C^{[j]}}-1}_{q=0}\\ &\cdot\mathbf{C}^{\mathrm{II}}_{2^{K_C^{[j]}}}\mathbf{J}_{2^{K_C^{[j]}}}.\end{aligned} \tag{7.84}$$

Here, we denote the required samples of $\mathbf{A}^{\widehat{\mathrm{II}}}$ by

$$\mathbf{E}^{(R)}_{(0,0)} := \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_p+1,\,2l'''_q}\right)^{2^{\widetilde{K}_R-1}-1,\,2^{K_C^{[j]}}-1}_{p,\,q=0} \qquad \text{and}$$

$$\mathbf{E}^{(R)}_{(1,0)} := \left(\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k'_p+1,\,2l'''_q}\right)^{2^{\widetilde{K}_R-1}-1,\,2^{K_C^{[j]}}-1}_{p,\,q=0}.$$

Hence, $\widetilde{\mathbf{B}}_R^{[j+1]}$ can be computed using $2^{\widetilde{K}_R} \cdot 2^{K_C^{[j]}}$ samples of $\mathbf{A}^{\widehat{\mathrm{II}}}$ by essentially applying $2^{K_C^{[j]}}$ 1-dimensional DCT-IIs of length $2^{\widetilde{K}_R - 1}$ to the columns of the sample matrix and $2^{\widetilde{K}_R - 1}$ 1-dimensional DCT-IVs of length $2^{K_C^{[j]}}$ to the rows. From (7.84) we can compute

$$\left(\tilde{a}_{(1,0)}^{[j+1]}\right)_{k,\,l} = \begin{cases} \left(\tilde{b}_R^{[j+1]}\right)_{k - M^{[j]} + 2^{\widetilde{K}_R - 1},\, l - d_C^{[j]} 2^{K_C^{[j]}}} & \text{if } k \in I_{M^{[j]} - 2^{\widetilde{K}_R - 1},\, M^{[j]} - 1}, \\ & \quad\; l \in I_{d_C^{[j]} 2^{K_C^{[j]}},\, \left(d_C^{[j]} + 1\right) 2^{K_C^{[j]}} - 1}, \\ 0 & \text{otherwise.} \end{cases}$$

Analogously to Section 7.3.3, we still have to find out whether the support of $\mathbf{A}^{[j+1]}$ is completely contained in its left or in its right half. We can determine whether $\mathbf{A}^{[j+1]}$ is $\mathbf{W}^{(0)}$ or $\mathbf{W}^{(1)}$ by comparing their DCT-IIs at an evenly indexed row and an oddly indexed column such that the corresponding entry of $\left(\mathbf{A}^{[j+1]}\right)^{\widehat{\mathrm{II}}}$ is nonzero, since

$$\left(w^{(1)}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1} = -\left(w^{(0)}\right)^{\widehat{\mathrm{II}}}_{2k,\,2l+1}$$

for all $k \in \{0, \ldots, M^{[j]} - 1\}$ and $l \in \{0, \ldots, N^{[j]} - 1\}$. As in Section 7.3.3, it can be shown that there exists an index pair $(k_R,\, l_R) \in I_{0,\,m^{[j]}-1} \times I_{0,\,n^{[j]}-1}$ such that $\left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_R,\,2l_R+1} \neq 0$. For a stable and efficient implementation of the method we set

$$(k_R,\, l_R) \coloneqq \operatorname*{argmax}_{(k,\,l) \in I_{0,\,m^{[j]}-1} \times I_{0,\,n^{[j]}-1}} \left\{ \left| 2^{J-j-1} a^{\widehat{\mathrm{II}}}_{2^{J-j}k,\; 2^{J-j-1}(2l+1)} \right| \right\}.$$

Because $S\left(\mathbf{W}^{(0)}\right) = S^{[j]}$, we obtain for the corresponding entry of $\left(\mathbf{W}^{(0)}\right)^{\widehat{\mathrm{II}}}$ that

$$\begin{aligned} \left(w^{(0)}\right)^{\widehat{\mathrm{II}}}_{2k_R,\,2l_R+1} &= \left(\mathbf{C}^{\mathrm{II}}_{M^{[j]}} \mathbf{W}^{(0)} {\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T}\right)_{2k_R,\,2l_R+1} \\ &= \sum_{r = M^{[j]} - 2^{\widetilde{K}_R - 1}}^{M^{[j]} + 2^{\widetilde{K}_R - 1} - 1} \sum_{s = \mu_C^{[j]}}^{\nu_C^{[j]}} \left(\mathbf{C}^{\mathrm{II}}_{M^{[j]}}\right)_{2k_R,\,r} w^{(0)}_{r,\,s} \left({\mathbf{C}^{\mathrm{II}}_{N^{[j]}}}^{T}\right)_{s,\,2l_R+1}, \end{aligned}$$

so it can be computed using $\mathcal{O}\left(2^{\widetilde{K}_R} \cdot n^{[j]}\right)$ operations. Then we set

$$\begin{aligned} \delta_R^- &\coloneqq \left| \left(w^{(0)}\right)^{\widehat{\mathrm{II}}}_{2k_R,\,2l_R+1} - \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_R,\,2l_R+1} \right| \qquad \text{and} \\ \delta_R^+ &\coloneqq \left| \left(w^{(0)}\right)^{\widehat{\mathrm{II}}}_{2k_R,\,2l_R+1} + \left(a^{[j+1]}\right)^{\widehat{\mathrm{II}}}_{2k_R,\,2l_R+1} \right|, \end{aligned}$$

and let

$$\mathbf{A}^{[j+1]} = \begin{cases} \mathbf{W}^{(0)} & \text{if } \delta_R^- < \delta_R^+, \\ \mathbf{W}^{(1)} & \text{if } \delta_R^- > \delta_R^+. \end{cases}$$

Further, the first column support index of $\mathbf{A}^{[j+1]}$ satisfies

$$\mu_C^{[j+1]} \coloneqq \begin{cases} \mu_C^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{W}^{(0)}, \\ N^{[j+1]} - 1 - \nu_C^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{W}^{(1)}. \end{cases} \tag{7.85}$$

Note that $n^{[j+1]} = n^{[j]}$ but that, analogously to Section 7.3.3, the first and last row support index of $\mathbf{A}^{[j+1]}$ are not known, and that we only have that

$$S_R^{[j+1]} \subsetneq I_{M^{[j]}-2^{\widetilde{K}_R-1},\, M^{[j]}+2^{\widetilde{K}_R-1}-1}$$

and $m^{[j]} < m^{[j+1]}$. We will show in Section 7.3.5 how to find the exact first and last row support indices by a 2-dimensional analog to Remark 6.18.

□

7.3.5 Detecting the Support Sets

All of the reconstruction procedures introduced above rely heavily on the fact that the reflected periodization $\mathbf{A}^{[j]}$ and its support $S^{[j]}$ are known from the previous iteration step. Hence, we still have to develop methods for efficiently finding the first and last support indices. Let $\varepsilon > 0$ be a threshold depending on the noise level of the data.

Note that if $j \in \{L, \dots, J-1\} \setminus \{j_1, j_2, j_3\}$, where j_1, j_2, j_3 are given by (7.7) to (7.9), the first and last row and column support indices are already completely determined by (7.28) and (7.29).

Case A: Colliding Rows and Columns

If $j = j_1$ as in (7.7), i.e., if the support $S^{[j]}$ of $\mathbf{A}^{[j]}$ is contained in its last b_R rows and b_C columns, we know from (7.35) that

$$S^{[j+1]} \subsetneq I_{M^{[j]}-2^{\widetilde{K}_R-1},\, M^{[j]}+2^{\widetilde{K}_R-1}-1} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}+2^{\widetilde{K}_C-1}-1}.$$

Hence, it suffices to consider the set

$$Z^{[j+1]} := \left\{(k,\, l) \in I_{M^{[j]}-2^{\widetilde{K}_R-1},\, M^{[j]}+2^{\widetilde{K}_R-1}-1} \times I_{N^{[j]}-2^{\widetilde{K}_C-1},\, N^{[j]}+2^{\widetilde{K}_C-1}-1} : \left|a_{k,\,l}^{[j+1]}\right| > \varepsilon\right\}$$

in order to determine the first and last row and column support indices. We obtain that

$$\begin{aligned}
\mu_R^{[j+1]} &:= \min\left\{k : (k,\, l) \in Z^{[j+1]}\right\}, & \mu_C^{[j+1]} &:= \min\left\{l : (k,\, l) \in Z^{[j+1]}\right\},\\
\nu_R^{[j+1]} &:= \max\left\{k : (k,\, l) \in Z^{[j+1]}\right\}, & \nu_C^{[j+1]} &:= \max\left\{l : (k,\, l) \in Z^{[j+1]}\right\},\\
m^{[j+1]} &:= \nu_R^{[j+1]} - \mu_R^{[j+1]} + 1, & n^{[j+1]} &:= \nu_C^{[j+1]} - \mu_C^{[j+1]} + 1.
\end{aligned}$$

Thus, $Z^{[j+1]}$ and the first and last row and column support indices can be found in

$$\mathcal{O}\left(2^{\widetilde{K}_R} \cdot 2^{\widetilde{K}_C}\right) = \mathcal{O}\left(b_R b_C\right)$$

time.

Case B: Colliding Columns

If $j = j_2$ as in (7.8), i.e., if the column support of $\mathbf{A}^{[j]}$ is completely contained in the last b_C columns, but the row support is not contained in the last b_R rows, we know from (7.61) and (7.76) that

$$\mu_R^{[j+1]} := \begin{cases} \mu_R^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{V}^{(0)},\\ M^{[j+1]} - 1 - \nu_R^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{V}^{(1)}, \end{cases}$$

and

$$S_C^{[j+1]} \subsetneq I_{N^{[j]}-2^{\tilde{K}_C-1},\, N^{[j]}+2^{\tilde{K}_C-1}-1}.$$

Consequently, the first and last column support indices can be found from the set

$$Z_C^{[j+1]} \coloneqq \left\{ l \in I_{N^{[j]}-2^{\tilde{K}_C-1},\, N^{[j]}+2^{\tilde{K}_C-1}-1} : \exists\, k \in I_{\mu_R^{[j+1]},\, \nu_R^{[j+1]}} : \left| a_{k,l}^{[j+1]} \right| > \varepsilon \right\}.$$

Then we define

$$\mu_C^{[j+1]} \coloneqq \min\left\{ l \in Z_C^{[j+1]} \right\}, \qquad \nu_C^{[j+1]} \coloneqq \max\left\{ l \in Z_C^{[j+1]} \right\},$$
$$n^{[j+1]} \coloneqq \nu_C^{[j+1]} - \mu_C^{[j+1]} + 1.$$

Hence, we can find $Z_C^{[j+1]}$ and the first and last column support index using

$$\mathcal{O}\left(2^{\tilde{K}_C}\right) = \mathcal{O}\left(b_C\right)$$

operations.

Case C: Colliding Rows

If $j = j_3$ as in (7.9), i.e., if the row support of $\mathbf{A}^{[j]}$ is completely contained in the last b_R rows, but the column support is not contained in the last b_C columns, (7.78) and (7.85) yield that

$$\mu_C^{[j+1]} \coloneqq \begin{cases} \mu_C^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{W}^{(0)}, \\ N^{[j+1]} - 1 - \nu_C^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{W}^{(1)}, \end{cases}$$

and

$$S_R^{[j+1]} \subsetneq I_{M^{[j]}-2^{\tilde{K}_R-1},\, M^{[j]}+2^{\tilde{K}_R-1}-1}.$$

Analogously to case B, we define the set

$$Z_R^{[j+1]} \coloneqq \left\{ k \in I_{M^{[j]}-2^{\tilde{K}_R-1},\, M^{[j]}+2^{\tilde{K}_R-1}-1} : \exists\, l \in I_{\mu_C^{[j+1]},\, \nu_C^{[j+1]}} : \left| a_{k,l}^{[j+1]} \right| > \varepsilon \right\}$$

and let

$$\mu_R^{[j+1]} \coloneqq \min\left\{ k \in Z_R^{[j+1]} \right\}, \qquad \nu_R^{[j+1]} \coloneqq \max\left\{ k \in Z_R^{[j+1]} \right\},$$
$$m^{[j+1]} \coloneqq \nu_R^{[j+1]} - \mu_R^{[j+1]} + 1.$$

Consequently, determining $Z_R^{[j+1]}$ and the first and last row support index requires

$$\mathcal{O}\left(2^{\tilde{K}_R}\right) = \mathcal{O}\left(b_R\right)$$

operations.

Detecting the Support of $\mathbf{A}^{[L]}$

Our method begins by computing the initial matrix $\mathbf{A}^{[L]} \in M^{[L]} \times N^{[L]}$ directly from $\mathbf{A}^{\widehat{\Pi}}$, so we also have to detect its support. As we do not have any a priori knowledge of

$S^{[L]}$, we have to consider the set

$$Z^{[L]} \coloneqq \left\{(k,\, l) \in I_{0,\, M^{[L]}-1} \times I_{0,\, N^{[L]}-1} : \left|a_{k,\,l}^{[L]}\right| > \varepsilon\right\}.$$

Then we let

$$\begin{aligned}
\mu_R^{[L]} &\coloneqq \min\left\{k : (k,\, l) \in Z^{[L]}\right\}, & \mu_C^{[L]} &\coloneqq \min\left\{l : (k,\, l) \in Z^{[L]}\right\},\\
\nu_R^{[L]} &\coloneqq \max\left\{k : (k,\, l) \in Z^{[L]}\right\}, & \nu_C^{[L]} &\coloneqq \max\left\{l : (k,\, l) \in Z^{[L]}\right\},\\
m^{[L]} &\coloneqq \nu_R^{[L]} - \mu_R^{[L]} + 1, & n^{[L]} &\coloneqq \nu_C^{[L]} - \mu_C^{[L]} + 1.
\end{aligned}$$

Thus, $Z^{[L]}$ and the first and last row and column support indices can be obtained in $\mathcal{O}\left(M^{[L]}N^{[L]}\right)$ time.

7.4 A 2D Sparse Fast IDCT-II for Block Sparse Matrices

We can now combine the procedures developed in Section 7.3 to obtain our new deterministic 2-dimensional IDCT-II algorithm for block sparse matrices.

We suppose that $\mathbf{A}^{\widehat{\mathrm{II}}} \in \mathbb{R}^{M\times N}$ is given, where $M = 2^{J_R}$, $N = 2^{J_C}$ and $J_R, J_C \in \mathbb{N}$. Further, we assume that $\mathbf{A}$ has a block support of unknown size $m \times n$, but that bounds $b_R \geq m$ and $b_C \geq n$ are known. We also suppose that $\mathbf{A}$ satisfies (7.5) and (7.6), and that we can access all entries of $\mathbf{A}^{\widehat{\mathrm{II}}}$. Setting $J \coloneqq \max\{J_R, J_C\}$, our method starts with calculating the matrix

$$\mathbf{A}^{[L]} = \mathbf{C}_{M^{[L]}}^{\mathrm{III}} \left(2^{J-L}\left(a_{2^{J-L}k,\, 2^{J-L}l}^{\widehat{\mathrm{II}}}\right)_{k,\,l=0}^{M^{[L]}-1,\, N^{[L]}-1}\right) {\mathbf{C}_{N^{[L]}}^{\mathrm{III}}}^{T},$$

where

$$L \coloneqq \max\left\{\lceil \log_2 b_R\rceil + 1, \lceil \log_2 b_C\rceil + 1\right\},$$

directly via a fast full-sized 2-dimensional DCT-III, which can be computed with the help of the row-column method mentioned in Section 4.3. Then the support of $\mathbf{A}^{[L]}$ has to be detected as we described in Section 7.3.5. Recalling that the indices j_1, j_2 and j_3 are given by (7.7) to (7.9), we have to execute the following iteration steps for $j \in \{L, \ldots, J-1\}$.

1) If the support of $\mathbf{A}^{[j]}$ is contained in the last b_R rows and in the last b_C columns, reconstruct $\mathbf{A}^{[j+1]}$ using the recovery procedure from Theorem 7.18.

2) If the support of $\mathbf{A}^{[j]}$ is not contained in the last b_R rows, but in the last b_C columns, reconstruct $\mathbf{A}^{[j+1]}$ using the recovery procedure from Theorem 7.19.

3) If the support of $\mathbf{A}^{[j]}$ is contained in the last b_R rows, but not in the last b_C columns, reconstruct $\mathbf{A}^{[j+1]}$ using the recovery procedure from Theorem 7.20.

4) If the support of $\mathbf{A}^{[j]}$ is neither contained in the last b_R rows nor in the last b_C columns, reconstruct $\mathbf{A}^{[j+1]}$ using the recovery procedure from Theorem 7.17.

It follows from Theorem 7.14 that there is at most one index j_1 such that we have to apply step 1, at most one index j_2 such that we have to apply step 2 and at most one index j_3 such that we have to apply step 3. The complete procedure is summarized in Algorithm 10.

Algorithm 10 2-Dimensional Real Sparse Fast IDCT-II for Matrices with Block Support

Input: $\mathbf{A}^{\widehat{\mathrm{II}}}$, b_R, b_C, where the sought-after matrix $\mathbf{A} \in \mathbb{R}^{M\times N}$ with $M = 2^{J_R}$, $N = 2^{J_C}$, $J_R, J_C \in \mathbb{N}$, has an unknown block support of size at most $b_R \times b_C$ and satisfies (7.5) and (7.6), and noise threshold $\varepsilon > 0$.

$J \leftarrow \min\{J_R, J_C\}$

$L \leftarrow \max\{\lceil \log_2 b_R \rceil + 1, \lceil \log_2 b_C \rceil + 1\}$

$\mathbf{A}^{[L]} \leftarrow \mathbf{DCT-III}\left[2^{J-L}\left(a^{\widehat{\mathrm{II}}}_{2^{J-L}k,\,2^{J-L}l}\right)_{k,\,l=0}^{M^{[L]}-1,\,N^{[L]}-1}\right]$

Find $\mu_R^{[L]}$, $\mu_C^{[L]}$, $m^{[L]}$ and $n^{[L]}$.

for j from L to $J-1$ **do**

  if $\mu_R^{[j]} \geq M^{[j]} - b_R$ and $\mu_C^{[j]} \geq N^{[j]} - b_C$ **then**

    Calculate $\mathbf{A}^{[j+1]}$ using Theorem 7.18.

    Find $\mu_R^{[j+1]}$, $\mu_C^{[j+1]}$, $m^{[j+1]}$ and $n^{[j+1]}$.

  else if $\mu_R^{[j]} < M^{[j]} - b_R$ and $\mu_C^{[j]} \geq N^{[j]} - b_C$ **then**

    Calculate $\mathbf{A}^{[j+1]}$ using Theorem 7.19.

    $m^{[j+1]} \leftarrow m^{[j]}$ and $\mu_R^{[j+1]} \leftarrow \begin{cases} \mu_R^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{V}^{(0)}, \\ M^{[j+1]} - 1 - \nu_R^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{V}^{(1)}. \end{cases}$

    Find $\mu_C^{[j+1]}$ and $n^{[j+1]}$.

  else if $\mu_R^{[j]} \geq b_R$ and $\mu_C^{[j]} < N^{[j]} - b_C$ **then**

    Calculate $\mathbf{A}^{[j+1]}$ using Theorem 7.20.

    $n^{[j+1]} \leftarrow n^{[j]}$ and $\mu_C^{[j+1]} \leftarrow \begin{cases} \mu_C^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{W}^{(0)}, \\ N^{[j+1]} - 1 - \nu_C^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{W}^{(1)}. \end{cases}$

    Find $\mu_R^{[j+1]}$ and $m^{[j+1]}$.

  else if $\mu_R^{[j]} < M^{[j]} - b_R$ and $\mu_C^{[j]} < N^{[j]} - b_C$ **then**

    Calculate $\mathbf{A}^{[j+1]}$ using Theorem 7.17.

    $\mu_R^{[j+1]} \leftarrow \begin{cases} \mu_R^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{U}^{(0,0)} \text{ or } \mathbf{A}^{[j+1]} = \mathbf{U}^{(0,1)}, \\ M^{[j+1]} - 1 - \nu_R^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{U}^{(1,0)} \text{ or } \mathbf{A}^{[j+1]} = \mathbf{U}^{(1,1)}. \end{cases}$

    $\mu_C^{[j+1]} \leftarrow \begin{cases} \mu_C^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{U}^{(0,0)} \text{ or } \mathbf{A}^{[j+1]} = \mathbf{U}^{(1,0)}, \\ N^{[j+1]} - 1 - \nu_C^{[j]} & \text{if } \mathbf{A}^{[j+1]} = \mathbf{U}^{(0,1)} \text{ or } \mathbf{A}^{[j+1]} = \mathbf{U}^{(1,1)}. \end{cases}$

    $m^{[j+1]} \leftarrow m^{[j]}$ and $n^{[j+1]} \leftarrow n^{[j]}$.

  end if

end for

Output: $\mathbf{A} = \mathbf{A}^{[J]}$.

In the following theorem we prove that the runtime and the sampling complexity of Algorithm 10 are both sublinear in the matrix size MN.

Theorem 7.21 *Let $M = 2^{J_R}$ and $N = 2^{J_C}$ with $J_R, J_C \in \mathbb{N}$, and let $J \coloneqq \min\{J_R, J_C\}$. Let $\mathbf{A} \in \mathbb{R}^{M\times N}$ have a block support of size $m\times n$ with known bounds $b_R \geq m$ and $b_C \geq n$, and assume that $\mathbf{A}$ satisfies (7.5) and (7.6). Let $L \coloneqq \max\{\lceil \log_2 b_R \rceil + 1, \lceil \log_2 b_C \rceil + 1\}$ and $b \coloneqq \max\{b_R, b_C\}$. Suppose that we have access to all entries of $\mathbf{A}^{\widehat{\mathrm{II}}} \in \mathbb{R}^{M\times N}$. Then Algorithm 10 has a runtime of*

$$\mathcal{O}\left(b_R b_C \log_2(b_R b_C) + b^2 \frac{MN}{\min\{M,N\}^2} \log\left(\frac{b^2 MN}{\min\{M,N\}^2}\right) + \frac{M}{2} b_C \log_2\left(\frac{M}{2} b_C\right) + b_R \frac{N}{2} \log_2\left(b_R \frac{N}{2}\right) + mn \log_2 \frac{\min\{M,N\}}{b}\right)$$

and requires

$$\mathcal{O}\left(b_r b_C + b^2 \frac{MN}{\min\{M,N\}^2} + mn \log_2 \frac{\min\{M,N\}}{b} + \frac{M}{2} b_C + b_R \frac{N}{2}\right)$$

samples of $\mathbf{A}^{\widehat{\mathrm{II}}}$.

If $N = \mathcal{O}(M)$ *and* $b_C = \mathcal{O}(b_R)$*, then the above runtime and sampling complexities simplify to*

$$\mathcal{O}\left(b^2 \log_2 b^2 + b^2 \log_2 \frac{M}{b} + \frac{M}{2} b \log_2 \left(\frac{M}{2} b\right)\right) \quad \textit{and} \quad \mathcal{O}\left(b^2 + b^2 \log_2 \frac{M}{b} + \frac{M}{2} b\right).$$

Proof. Computing the initial matrix $\mathbf{A}^{[L]}$ in line 3 using a 2-dimensional DCT-III of size $M^{[L]} \times N^{[L]}$ with $M^{[L]} = \frac{M}{2^{J-L}}$ and $N^{[L]} = \frac{N}{2^{J-L}}$ requires

$$\mathcal{O}\left(M^{[L]} N^{[L]} \log\left(M^{[L]} N^{[L]}\right)\right) = \mathcal{O}\left(b^2 \frac{MN}{\min\{M,N\}^2}\right)$$

operations, since $2^L = \mathcal{O}(b)$. The first and last row and column support indices of $\mathbf{A}^{[L]}$ in line 4 can be found with the method described in Section 7.3.5, which has a runtime of $\mathcal{O}\left(M^{[L]} N^{[L]}\right)$.

If $j = j_1$ as in (7.7), i.e., if we apply step 1, we use the method from Theorem 7.18. By (7.42), (7.49) and (7.57), it requires the application of $2^{\widetilde{K}_C - 1}$ 1-dimensional DCTs of types II and IV of length $2^{\widetilde{K}_R - 1}$ to columns, and of $2^{\widetilde{K}_R - 1}$ 1-dimensional DCTs of types II and IV of length $2^{\widetilde{K}_C - 1}$ to rows. It follows from (7.58) that additional operations of complexity $\mathcal{O}\left(2^{\widetilde{K}_R - 1} \cdot 2^{\widetilde{K}_C - 1}\right)$ have to be performed, as all other occurring matrices are either diagonal or permutations. Using the row-column approach detailed in Section 4.3 for the computation of the DCTs, line 7 has a runtime of

$$\mathcal{O}\left(2^{\widetilde{K}_R - 1} 2^{\widetilde{K}_C - 1} \log_2\left(2^{\widetilde{K}_R - 1} 2^{\widetilde{K}_C - 1}\right)\right) = \mathcal{O}\left(b_R b_C \log_2\left(b_R b_C\right)\right).$$

If we apply step 2, then $j = j_2$ as in (7.8), and we have to employ the technique described in Theorem 7.19. By (7.73), calculating $\widetilde{B}_C^{[j+1]}$ needs the application of 1-dimensional DCT-IIIs of length $2^{K_R^{[j]}}$ to $2^{\widetilde{K}_C - 1}$ columns, the application of 1-dimensional DCT-IVs of length $2^{\widetilde{K}_C - 1}$ to $2^{K_R^{[j]}}$ rows, and further operations of complexity $2^{K_R^{[j]}} \times 2^{\widetilde{K}_C - 1}$, as all other matrices are either diagonal or permutations. Finding the index pair (k_C, l_C) and the entry $\left(v^{(0)}\right)^{\widehat{\mathrm{II}}}_{k_C, l_C}$ requires $\mathcal{O}\left(m^{[j]} b_C\right)$ additional operations. Note that we can only estimate that $2^{K_R^{[j]}} = \mathcal{O}\left(M^{[j]}\right) = \mathcal{O}\left(\frac{M}{2}\right)$, since we do not know the location of the row support of $\mathbf{A}^{[j_2]}$ a priori and also do not have an estimate for j_2 besides $J - 1$. Hence, with the row-column approach for the DCTs, line 10 has a runtime of

$$\mathcal{O}\left(2^{K_R^{[j]}} 2^{\widetilde{K}_C - 1} \log_2\left(2^{K_R^{[j]}} 2^{\widetilde{K}_C - 1}\right)\right) = \mathcal{O}\left(\frac{M}{2} b_C \log_2\left(\frac{M}{2} b_C\right)\right).$$

The first column support index $\mu_C^{[j+1]}$ and the column support length $n^{[j+1]}$ can be computed as detailed in Section 7.3.5, which yields a runtime of $\mathcal{O}\left(b_C\right)$.

If $j = j_3$ as given by (7.9), we have to apply step 3. Employing the technique from

Theorem 7.20, lines 14 and 16 analogously require

$$\mathcal{O}\left(2^{\widetilde{K}_R-1}2^{K_C^{[j]}}\log_2\left(2^{\widetilde{K}_R-1}2^{K_C^{[j]}}\right)\right)=\mathcal{O}\left(b_R\frac{N}{2}\log_2\left(b_R\frac{N}{2}\right)\right)$$

and $\mathcal{O}\left(b_R\right)$ operations, respectively.

If $j\in\{L,\ldots,J-1\}\backslash\{j_1,j_2,j_3\}$, we always have to apply step 4, for which the approach from Theorem 7.17 is used. By Lemma 7.15, the nonzero entries $\left(a^{[j+1]}\right)^{\widehat{\Pi}}_{2k_{(0,1)},\,2l_{(0,1)}+1}$ and $\left(a^{[j+1]}\right)^{\widehat{\Pi}}_{2k_{(1,0)}+1,\,2l_{(1,0)}}$ can be found in $\mathcal{O}\left(m^{[j]}n^{[j]}\right)$ time. Calculating the corresponding entries of $\left(\mathbf{U}^{(0,0)}\right)^{\widehat{\Pi}}$ requires the same number of operations, so line 18 has a runtime of

$$\mathcal{O}\left(m^{[j]}n^{[j]}\right).$$

Recall that, if they exist, j_1, j_2 and j_3 are unique. Thus, Algorithm 10 has an overall runtime of

$$\begin{aligned}
&\mathcal{O}\Bigg(M^{[L]}N^{[L]}\log\left(M^{[L]}N^{[L]}\right)+b_Rb_C\log_2\left(b_Rb_C\right)\\
&\quad+2^{K_R^{[j_2]}}b_C\log_2\left(2^{K_R^{[j_2]}}b_C\right)+b_R2^{K_C^{[j_3]}}\log_2\left(b_R2^{K_C^{[j_3]}}\right)+\sum_{\substack{j=L\\ j\notin\{j_1,j_2,j_3\}}}^{J-1}m^{[j]}n^{[j]}\Bigg)\\
&=\mathcal{O}\Bigg(b^2\frac{MN}{\min\{M,N\}^2}\log\left(b^2\frac{MN}{\min\{M,N\}^2}\right)+b_Rb_C\log_2\left(b_Rb_C\right)+(J-L)mn\\
&\quad+\frac{M}{2}b_C\log_2\left(\frac{M}{2}b_C\right)+b_R\frac{N}{2}\log_2\left(b_R\frac{N}{2}\right)\Bigg)\\
&=\mathcal{O}\Bigg(b_Rb_C\log_2\left(b_Rb_C\right)+b^2\frac{MN}{\min\{M,N\}^2}\log\left(\frac{b^2MN}{\min\{M,N\}^2}\right)\\
&\quad+\frac{M}{2}b_C\log_2\left(\frac{M}{2}b_C\right)+b_R\frac{N}{2}\log_2\left(b_R\frac{N}{2}\right)+mn\log_2\frac{\min\{M,N\}}{b}\Bigg).
\end{aligned}$$

If we know a priori that the number of matrix rows M and the number of matrix columns N, and the bounds b_R and b_C are of a similar size, i.e., if we have that $N=\mathcal{O}(M)$ and $b_C=\mathcal{O}\left(b_R\right)=\mathcal{O}(b)$, then the above runtime simplifies to

$$\mathcal{O}\left(b^2\log_2b^2+b^2\log_2\frac{M}{b}+\frac{M}{2}b\log_2\left(\frac{M}{2}b\right)\right).$$

These assumptions are for example met by commonly used image formats, where the ratio $\frac{N}{M}$ is usually between 1 and 2 if only a small subimage with similar column-to-row ratio is not zero.

In order to compute the initial matrix $\mathbf{A}^{[L]}$ in line 3, we need $M^{[L]}N^{[L]}$ samples of $\mathbf{A}^{\widehat{\Pi}}$. By Theorem 7.18, line 7 requires $8\cdot2^{\widetilde{K}_R-1}2^{\widetilde{K}_C-1}$ samples of $\mathbf{A}^{\widehat{\Pi}}$. Further, it follows from Theorems 7.19 and 7.20 that lines 10 and 14 have sampling complexities of $\mathcal{O}\left(2^{K_R^{[j_2]}}2^{\widetilde{K}_C}+m^{[j_2]}n^{[j_2]}\right)$ and $\mathcal{O}\left(2^{\widetilde{K}_R}2^{K_C^{[j_3]}}+m^{[j_3]}n^{[j_3]}\right)$, respectively. Finally, line 18

requires $m^{[j]}n^{[j]}$ samples of $\mathbf{A}^{\widehat{\mathrm{II}}}$. Hence, Algorithm 10 has a sampling complexity of

$$\begin{aligned}
&\mathcal{O}\left(M^{[L]}N^{[L]} + b_R b_C + 2^{K_R^{[j_2]}} b_C + mn + b_R 2^{K_C^{[j_3]}} + mn + \sum_{\substack{j=L \\ j\notin\{j_1,j_2,j_3\}}}^{J-1} m^{[j]}n^{[j]}\right) \\
&= \mathcal{O}\left(b^2 \frac{MN}{\min\{M,N\}^2} + b_R b_C + (J-L)mn + \frac{M}{2} b_C + b_R \frac{N}{2} + mn\right) \\
&= \mathcal{O}\left(b_r b_C + b^2 \frac{MN}{\min\{M,N\}^2} + mn \log_2 \frac{\min\{M,N\}}{b} + \frac{M}{2} b_C + b_R \frac{N}{2}\right).
\end{aligned}$$

Again, if $N = \mathcal{O}(M)$ and $b_C = \mathcal{O}(b_R) = \mathcal{O}(b)$, this simplifies to

$$\mathcal{O}\left(b^2 + b^2 \log_2 \frac{M}{b} + \frac{M}{2} b\right).$$

□

Remark 7.22 Note that in practice the runtime of Algorithm 10 depends on whether and at which level j steps 2 and 3 have to be executed, and also on the location of the row support of $\mathbf{A}^{[j_2]}$ and the location of the column support of $\mathbf{A}^{[j_3]}$. If j_2 and j_3 exist and are approximately L, then $2^{K_R^{[j_2]}} = \mathcal{O}(b_R)$ and $2^{K_C^{[j_3]}} = \mathcal{O}(b_C)$. Thus, the algorithm requires

$$\mathcal{O}\left(b^2 \log_2 b^2 + b^2 \log_2 \frac{M}{b}\right)$$

arithmetical operations. Such a runtime is also obtained if $K_R^{[j_2]}$ and $K_C^{[j_3]}$ are approximately $\widetilde{K}_R$ and $\widetilde{K}_C$, respectively, since $\widetilde{K}_R = \mathcal{O}(b_R)$ and $\widetilde{K}_C = \mathcal{O}(b_C)$. However, even if, e.g., $j_2 = J-1$ and $j_3 = J-2$ with $2^{K^{[j_2]}} = \frac{M}{2}$ and $2^{K^{[j_3]}} = \frac{N}{4}$, the theoretical runtime of Algorithm 10 proven in Theorem 7.21 is still smaller than the runtime

$$\mathcal{O}(MN \log_2(MN))$$

of a 2-dimensional IDCT-II of size $M \times N$ from Section 4.3. If j_2 and j_3 exist and are approximately L, or if $K_R^{[j_2]} \approx \widetilde{K}_R$ and $K_C^{[j_3]} \approx \widetilde{K}_C$, the number of required samples is

$$\mathcal{O}\left(b^2 + mn \log_2 \frac{M}{b}\right),$$

and if, e.g., $j_2 = J-1$ and $j_3 = J-2$ with $2^{K^{[j_2]}} = \frac{M}{2}$ and $2^{K^{[j_3]}} = \frac{N}{4}$, Algorithm 10 still uses less than MN samples of $\mathbf{A}^{\widehat{\mathrm{II}}}$. In order to obtain a really efficient 2-dimensional IDCT-II algorithm for block sparse matrices with the runtime of

$$\mathcal{O}\left(b^2 \log_2 b^2 + b^2 \log_2 \frac{M}{b}\right)$$

we initially hoped for, faster procedures for cases B and C of Theorem 7.14 have to be developed. Algorithm 10 is a good first algorithm for the problem of recovering block sparse matrices from their DCT-II, but it can still be improved. Nevertheless, even in the worst case Algorithm 10 achieves a runtime that is sublinear in the matrix size MN and subquadratic in the sparsity mn. ◊

Part IV

Conclusion

Conclusion

This thesis consists of two parts, the first one focusing on sparse FFT algorithms for 2π-periodic functions and the second one investigating sparse IDCT-II algorithms for vectors and matrices. In the first part of this thesis we began by introducing two new deterministic sparse FFT algorithms for reconstructing 2π-periodic functions with short frequency support from samples. Both can be obtained as different simplifications of the sampling schemes used for the general B-sparse Algorithm 2 in [Iwe10] if the given support structure is utilized. Analogously to Algorithm 2 in [Iwe10], our new sparse FFT algorithms need a priori knowledge of an upper bound B on the support length. Estimates for the theoretical runtime and sampling complexities of both algorithms provided in Chapter 2 showed that they are sublinear in the bandwidth N of the function we aim to recover, with runtimes of

$$\mathcal{O}\left(B \log B \frac{\log^2 \frac{N}{B}}{\log \log \frac{N}{B}}\right) \qquad \text{and} \qquad \mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B} \log^2\left(\frac{B+\log N}{\log B}\right)\right),$$

respectively. Thus, they both improve on the $\mathcal{O}(N \log N)$ runtime of the FFT. Further, the runtime and sampling complexities scale subquadratically in the sparsity, which makes them faster than all existing deterministic FFT algorithms for arbitrary B-sparse functions, whose runtimes are quadratic in the sparsity. Numerical experiments supported these claims empirically and also showed the robustness of both methods with respect to noisy input data.

In Chapter 3 we extended the methods from [Iwe10, Iwe13] to functions with more complex sparsity constraints, namely polynomially structured sparse functions, where the energetic frequencies are generated by evaluating n polynomials of degree at most d at B consecutive points. We derived the, as far as we are aware, first deterministic FFT algorithm for this class of functions and proved that its runtime is sublinear in the bandwidth N of the function in question, with

$$\mathcal{O}\left(\frac{Bd^2n^3 \log^5 N}{\log^2(2dn)}\right).$$

Our algorithm requires a priori knowledge of upper bounds on n, d and B, analogously to Algorithm 2 in [Iwe10]. Furthermore, we proved a theoretical error estimate of

$$\|\mathbf{c}(N) - \mathbf{x}_R\|_2 \leq \left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{Bn}(N)\right\|_2 + \sqrt{Bn}\left(\varepsilon + \frac{3}{dn}\left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{2Bn}(N)\right\|_1\right)$$

for an approximately polynomially structured sparse function $f + \eta$ with bandwidth N and noise η such that $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. For the class of functions with block sparse frequency support, which is probably the most practically useful subclass of the functions with polynomially structured sparsity, we adapted our method to obtain an even lower runtime. This, to the best of our knowledge, first FFT algorithm for block sparse functions also achieves runtime and sampling complexities that scale subquadratically in

the sparsity, with a runtime of

$$\mathcal{O}\left(\frac{Bn^2 \log^4 N}{\log^1(2n)}\right).$$

We concluded the chapter by numerical experiments highlighting the performance of the algorithm for block sparse functions with respect to runtime and noisy input data.

In the second part of this thesis we investigated the related problem of deterministically reconstructing a vector $\mathbf{x} \in \mathbb{R}^{2^{J-1}}$ from its DCT-II transformed vector $\mathbf{x}^{\widehat{\mathrm{II}}}$. In Chapter 5 we derived an algorithm that recovers a vector $\mathbf{x}$ with one-block support, i.e., the support may be wrapped periodically around the boundary of $\mathbf{x}$. Our approach utilizes that $\mathbf{x}^{\widehat{\mathrm{II}}}$ is completely determined by the Fourier transform of the auxiliary vector $\mathbf{y} = (\mathbf{x}^T, (\mathbf{J}_N \mathbf{x})^T)^T$ of double length. Since $\mathbf{y}$ has a reflected block support if $\mathbf{x}$ has a one-block support, we first developed an algorithm for iteratively recovering $\mathbf{y}$ from its periodizations $\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(J)} = \mathbf{y}$ and $\widehat{\mathbf{y}}$. The notion of periodizations was first introduced in [PW16a] for an IFFT algorithm for reconstructing vectors with one-block support. Applying our new sparse IFFT method then yields a sparse IDCT-II algorithm for recovering vectors with one-block support. Both our new sparse IFFT and our new sparse IDCT-II have runtimes of $\mathcal{O}\left(m \log m \log \frac{2N}{m}\right)$, which are sublinear in the length of the vector and subquadratic in the sparsity. Furthermore, they do not require any a priori information on the support length of $\mathbf{x}$. As far as we are aware, these are the first deterministic IFFT and IDCT-II algorithms, respectively, that are tailored to the specific support structures of reflected block support and one-block support.

As the DCT-II is a real transform that can be computed in a fast way using only real arithmetic, we focused on finding a second sparse IDCT-II that only requires real arithmetic in Chapter 6. Instead of employing periodizations of the auxiliary vector $\mathbf{y}$ and IFFTs, we developed the new concept of reflected periodizations. These are a DCT-II specific analog to the periodizations from [PW16a] used in the DFT case. For recovering a vector $\mathbf{x} \in \mathbb{R}^{2^J}$ from its reflected periodizations the vector must have a short support of length m, i.e., the support can no longer be wrapped periodically around the boundary of the vector. Moreover, this approach requires a priori knowledge of an upper bound M on m. With the help of the reflected periodizations, $\mathbf{x}$ can be recovered iteratively from $\mathbf{x}^{[L]}, \mathbf{x}^{[L+1]}, \ldots, \mathbf{x}^{[J]} = \mathbf{x}$ and $\mathbf{x}^{\widehat{\mathrm{II}}}$, where the starting index L has to satisfy $2^{L-1} \geq M$. Our new real sparse IDCT-II algorithm, which is, to the best of our knowledge, the first deterministic sparse IDCT-II algorithm that only uses real arithmetic, achieves a runtime of $\mathcal{O}\left(M \log M + m \log \frac{N}{M}\right)$. Thus, its runtime is both sublinear in the vector length N and subquadratic in the sparsity bound M. We concluded the chapter with numerical comparisons of our two new sparse IDCT-II methods with respect to runtime and noisy input data.

Many of the applications for which the DCT-II is typically used, like digital image and video compression, are higher dimensional, but also sparse. In Chapter 7 we investigated the special case of block sparse matrices, where the support is contained in a rectangle of size $m \times n$. For such matrices we derived a new 2-dimensional IDCT-II algorithm that is based on a generalization of the techniques used in Chapter 6. Instead of employing reflectedly periodized vectors, we now recover a matrix $\mathbf{A} \in \mathbb{R}^{2^{J_R} \times 2^{J_C}}$ iteratively from its reflectedly periodized matrices $\mathbf{A}^{[L]}, \mathbf{A}^{[L+1]}, \ldots, \mathbf{A}^{[J]} = \mathbf{A}$ and its DCT-II, $\mathbf{A}^{\widehat{\mathrm{II}}}$. Analogously to Algorithm 9 from Chapter 6, our algorithm requires a priori knowledge of upper bounds on the support sizes m and n. As far as we are aware this is the first deterministic 2-dimensional IDCT-II algorithm for block sparse matrices that only uses real

arithmetic. Unfortunately, until now, our method has the worst case runtime estimate

$$\mathcal{O}\left(b^2 \log_2 b^2 + b^2 \log_2 \frac{M}{b} + \frac{M}{2} b \log_2 \left(\frac{M}{2} b\right)\right)$$

for a matrix of size $M \times M$ with block support of size $m \times m$ and upper bound $b \geq m$, which is attained for certain matrices. Initially, we hoped to obtain a runtime of

$$\mathcal{O}\left(b^2 \log_2 b^2 + b^2 \log_2 \frac{M}{b}\right),$$

and, in practice, we expect that this runtime is achieved for most matrices M. Nevertheless, the theoretical runtime which we were able to prove is sublinear in the matrix size M^2 and subquadratic in its sparsity bound b^2.

The in-depth study of the problems discussed in this thesis brings up some open questions. The key concept used in our algorithm for 2π-periodic functions with polynomially structured sparsity is the restriction of the input function to frequencies satisfying congruency conditions. The choice of the restriction is intertwined with the choice of Algorithm 3 in [Iwe13] as the sparse FFT algorithm that is applied to the restrictions. Other sparse FFT methods will require other types of restrictions, which in turn may allow for different sparsity structures. As of yet there has been no investigation of other combinations of sparse FFTs, restrictions and the resulting possible frequency structures. We believe this to be an interesting topic of research. Furthermore, the idea of restricting a complex structure to several sparser subproblems and applying existing methods to the subproblems for reducing the runtime, is rather universal. Similar approaches could be useful in many contexts, perhaps even in the sparse DCT setting we considered in the second part of this thesis.

We solely focused on the IDCT-II, or, equivalently, the DCT-III in Chapters 5 to 7. It would be very interesting to investigate whether the approaches taken in Chapters 5 and 6 can be transferred to other types of the DCT, particularly the IDCT-III or DCT-II. Since DCT-II and DCT-III are, as their respective inverse transforms, very closely connected, we are convinced that similar approaches as the ones introduced in this thesis can also yield fast sparse IDCT-III algorithms. The investigation of other types of sparsity, for example two or more support intervals, or general sparsity, could also prove to be fruitful.

Another highly interesting topic for future research is the runtime optimization of Algorithm 10 for the reconstruction of block sparse matrices from their DCT-II. So far, we provided very efficient procedures for cases A and D of Theorem 7.14. For cases B and C we initially expected runtimes of

$$\mathcal{O}\left(m b_C \log\left(m b_C\right)\right) \qquad \text{and} \qquad \mathcal{O}\left(b_R n \log\left(b_R n\right)\right),$$

respectively. We have not been able to find invertible matrix factorizations allowing such complexities yet, but we are confident that it should be possible. If procedures with such a runtime could be found, the overall runtime of Algorithm 10 would reduce to

$$\mathcal{O}\left(b^2 \log_2 b^2 + b^2 \log_2 \frac{M}{b}\right)$$

for a matrix of size $M \times M$ with block support of size $m \times m$ and upper bound $b \geq m$. This is the order of runtime we initially hoped to achieve. Investigating other types of sparsity could also prove to be rewarding, especially if corresponding 1-dimensional

algorithms have already been found. Furthermore, the sparse 2-dimensional IDCT-II algorithm has not yet been implemented and no numerical experiments have been made.

Summing up, all of our new sparse deterministic FFT or IDCT-II algorithms have applications in several areas of signal and image processing. They can be shown both theoretically and numerically to be faster than comparable existing methods. With the exception of the two algorithms for 2π-periodic functions with short frequency support, all methods introduced in this thesis are, as far as we are aware, the first existing deterministic algorithms for the respective sparsity structure. As all of our methods are deterministic, they will always return accurate estimates of the function or vector that we aim to recover, which we also supported by numerical experiments for all algorithms except the 2-dimensional IDCT-II.

Bibliography

[AF82] N. Ahmed and M. D. Flickner. Some considerations of the discrete cosine transform. In *16th Asilomar Conference on Circuits, Systems and Computers, Pacific Grove*, pages 295–299, 1982.

[AGS03] A. Akavia, S. Goldwasser, and S. Safra. Proving hard-core predicates using list decoding. *FOCS*, 44:146–159, 2003.

[Aka10] A. Akavia. Deterministic sparse Fourier approximation via fooling arithmetic progressions. In *Proc. 23rd COLT*, pages 381–393, 2010.

[Aka14] A. Akavia. Deterministic sparse Fourier approximation via approximating arithmetic progressions. *IEEE Trans. Inform. Theory*, 60(3):1733–1741, 2014.

[AR75] N. Ahmed and K. R. Rao. *Orthogonal transforms for digital signal processing*. Springer-Verlag, 1975.

[Atk89] K. Atkinson. *An Introduction to Numerical Analysis*. Wiley, 1989.

[BDF+11] J. Bourgain, S. Dilworth, K. Ford, S. Konyagin, and D. Kutzarova et al. Explicit constructions of RIP matrices and related problems. *Duke Math. J.*, 159(1):145–185, 2011.

[Ber67] G. D. Bergland. The Fast Fourier Transform Recursive Equations for Arbitrary Length Records. *Math. Comp.*, 21(98):236–238, 1967.

[Ber68] G. D. Bergland. A Fast Fourier Transform Algorithm Using Base 8 Iterations. *Math. Comp.*, 22(98):275–279, 1968.

[Bey95] G. Beylkin. On the Fast Fourier Transform of Functions with Singularities. *Appl. Comput. Harmon. Anal.*, 2(4):363 – 381, 1995.

[Bit17a] S. Bittens. Alternative SFFT for Functions with Short Frequency Support, implemented in MATLAB. `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`, 2017. Accessed Mar 28, 2019.

[Bit17b] S. Bittens. SFFT for Functions with Short Frequency Support , implemented in MATLAB. `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`, 2017. Accessed Nov 29, 2018.

[Bit17c] S. Bittens. Sparse FFT for Functions with Short Frequency Support. *Dolomites Res. Notes Approx.*, 10:43–55, 2017.

[Blu70] L. I. Bluestein. A Linear Filtering Approach to the Computation of Discrete Fourier Transform. *IEEE Trans. Audio Electroacoust.*, 18(4):451–455, 1970.

[Bos06] S. Bosch. *Algebra (6. Aufl.)*. Berlin: Springer, 2006.

[BP18a] S. Bittens and G. Plonka. Real Sparse Fast DCT for Vectors with Short Support. `https://arxiv.org/abs/1807.07397`, 2018.

[BP18b] S. Bittens and G. Plonka. Real Sparse Fast DCT for Vectors with Short Support, implemented in MATLAB. `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`, 2018. Accessed Mar 28, 2019.

[BP18c] S. Bittens and G. Plonka. Sparse fast DCT for vectors with one-block support. *Numer. Algorithms* (2018). `http://doi.org/10.1007/s11075-018-0620-1`, 2018.

[BP18d] S. Bittens and G. Plonka. Sparse Fast DCT for Vectors with One-block Support, implemented in MATLAB. `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`, 2018. Accessed Nov 29, 2018.

[BP18e] S. Bittens and G. Plonka. Sparse FFT for Vectors with Reflected Block Support, implemented in MATLAB. `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`, 2018. Accessed Nov 29, 2018.

[BR00] V. Britanak and K. R. Rao. Two-dimensional DCT/DST universal computational structure for $2^m \times 2^n$ block sizes. *IEEE Trans. Signal Process.*, 48(11):3250–3255, 2000.

[BYR06] V. Britanak, P. Yip, and K. R. Rao. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations.* Elsevier Science, 2006.

[BZI17a] S. Bittens, R. Zhang, and M. A. Iwen. A Deterministic Sparse FFT for Functions with Structured Fourier Sparsity - A deterministic sparse FFT algorithm for functions with block-structured (FAST) and polynomially structured Fourier sparsity, implemented in MATLAB. `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`, 2017. Accessed Dec 14, 2018.

[BZI17b] S. Bittens, R. Zhang, and M. A. Iwen. FAST - A Deterministic Sparse FFT for Functions with Structured Fourier Sparsity - A deterministic sparse FFT algorithm (FAST) for functions with block-structured Fourier sparsity, implemented in C++. `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`, 2017. Accessed Dec 14, 2018.

[BZI19] S. Bittens, R. Zhang, and M. A. Iwen. A deterministic sparse FFT for functions with structured Fourier sparsity. *Adv. Comput. Math.*, 45(2):519–561, 2019.

[CCW16] B. Choi, A. Christlieb, and Y. Wang. Multi-dimensional Sublinear Sparse Fourier Algorithm. `https://arxiv.org/abs/1606.07407`, 2016.

[CG99] E. Chu and A. George. *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms.* Computational Mathematics. Taylor & Francis, 1999.

[CH91] S. C. Chan and K. L. Ho. A new two-dimensional fast cosine transform algorithm. *IEEE Trans. Signal Process.*, 39(2):481–485, 1991.

[CI16] M. Cheraghchi and P. Indyk. Nearly optimal deterministic algorithm for sparse Walsh-Hadamard transform. *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 298–317, 2016.

[CIK18] B. Choi, M. A. Iwen, and F. Krahmer. Sparse Harmonic Transforms: A New Class of Sublinear-time Algorithms for Learning Functions of Many Variables. `https://arxiv.org/abs/1808.04932`, 2018.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 3rd Edition.* MIT Press, 2009.

[CLW16] A. Christlieb, D. Lawlor, and Y. Wang. A multiscale sub-linear time Fourier algorithm for noisy data. *Appl. Comput. Harmon. Anal.*, 40(3):553–574, 2016.

[CSF77] W. H. Chen, C. Smith, and S. Fralick. A Fast Computational Algorithm for the Discrete Cosine Transform. *IEEE Trans. Commun.*, 25(9):1004–1009, 1977.

[CT65] J. W. Cooley and J. W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comput.*, 19:297–301, 1965.

[DG90] P. Duhamel and C. Guillemot. Polynomial transform computation of the 2-D DCT. *International Conference on Acoustics, Speech, and Signal Processing*, 3:1515–1518, 1990.

[DH84] P. Duhamel and H. Hollmann. 'Split radix' FFT algorithm. *Electronics Letters*, 20:14–16, 1984.

[DR93] A. Dutt and V. Rokhlin. Fast Fourier Transforms for Nonequispaced Data. *SIAM J. Sci. Comput.*, 14(6):1368–1393, 1993.

[FB96] P. Feng and Y. Bresler. Spectrum-blind minimum-rate sampling and reconstruction of multiband signals. *Proc. 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 3:1688–1691, 1996.

[Fei90] E. Feig. Fast scaled-DCT algorithm. *Proc. SPIE 1244, Image Processing Algorithms and Techniques.* `https://doi.org/10.1117/12.19490`, 1990.

[FJ17] M. Frigo and S. G. Johnson. FFTW 3.3.6. `http://www.fftw.org/`, 2017.

[Fol92] G. B. Folland. *Fourier Analysis and Its Applications.* Advanced Mathematics Series. Wadsworth & Brooks/Cole Advanced Books & Software, 1992.

[FR13] S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing.* Birkhäuser Basel, 2013.

[FW92] E. Feig and S. Winograd. Fast algorithms for the discrete cosine transform. *IEEE Trans. Signal Process.*, 40(9):2174–2193, 1992.

[GGI+02] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal Sparse Fourier Representations via Sampling. *Proc. 34th Annual ACM Symposium on Theory of Computing*, pages 152–161, 2002.

[GIIS14] A. C. Gilbert, P. Indyk, M. A. Iwen, and L. Schmidt. Recent Developments in the Sparse Fourier Transform: A compressed Fourier transform for big data. *IEEE Signal Process. Mag.*, 31(5):91–100, 2014.

[GMS05] A. C. Gilbert, M. Muthukrishnan, and M. Strauss. Improved time bounds for near-optimal space Fourier representations. *Proc. SPIE 5914, Wavelets XI, 59141A*, `https://doi.org/10.1117/12.615931`, 2005.

[Goo58] I. J. Good. The Interaction Algorithm and Practical Fourier Analysis. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 20(2):361–372, 1958.

[GS66] W. M. Gentleman and G. Sande. Fast Fourier Transforms: For Fun and Profit. In *Proc. 1966 Fall Joint Computer Conference*, pages 563–578, 1966.

[HAKI12] H. Hassanieh, F. Adib, D. Katabi, and P. Indyk. Faster GPS via the sparse Fourier transform. *Proc. 18th Annual International Conference on Mobile Computing and Networking*, pages 353–364, 2012.

[HIK17] X. Hu, M. A. Iwen, and H. Kim. Rapidly computing sparse Legendre expansions via sparse Fourier transforms. *Numer. Algorithms*, 74(4):1029–1059, 2017.

[HIKP12a] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Nearly Optimal Sparse Fourier Transform. In *Proc. 44th Annual ACM Symposium on Theory of Computing*, pages 563–578, 2012.

[HIKP12b] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. SFFT: Sparse Fast Fourier Transform. `https://groups.csail.mit.edu/netmit/sFFT/code.html`, 2012. Accessed Dec 14, 2018.

[HIKP12c] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Simple and Practical Algorithm for Sparse Fourier Transform. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1183–1194, 2012.

[HKPV13] S. Heider, S. Kunis, D. Potts, and M. Veit. A sparse Prony FFT. In *Proc. 10th International Conference on Sampling Theory and Applications*, pages 572–575, 2013.

[HSA+14] H. Hassanieh, L. Shi, O. Abari, E. Hamed, and D. Katabi. Ghz-wide sensing and decoding using the sparse Fourier transform. *Proc. IEEE Conference on Computer Communications*, pages 2256–2264, 2014.

[HW60] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers.* Clarendon Press Oxford, 4th edition, 1960.

[IGS07] M. A. Iwen, A. C. Gilbert, and M. Strauss. Empirical Evaluation of a Sub-Linear Time Sparse DFT Algorithm. *Commun. Math. Sci.*, 5(4):981–998, 2007.

[IKP14] P. Indyk, M. Kapralov, and E. Price. (Nearly) sample-optimal sparse Fourier transform. *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 480–499, 2014.

[IS08] M. A. Iwen and C. V. Spencer. Improved Bounds for a Deterministic Sublinear-Time Sparse Fourier Algorithm. *Proc. 42nd Annual Conference on Information Systems*, pages 458–463, 2008.

[Iwe10] M. A. Iwen. Combinatorial Sublinear-Time Fourier Algorithms. *Found. Comput. Math.*, 10(3):303–338, 2010.

[Iwe13] M. A. Iwen. Improved Approximation Guarantees for Sublinear-Time Fourier Algorithms. *Appl. Comput. Harmon. Anal.*, 34(1):57–82, 2013.

[Jai79] A. K. Jain. A Sinusoidal Family of Unitary Transforms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1(4):356–365, 1979.

[Kar47] K. Karhunen. *Über lineare Methoden in der Wahrscheinlichkeitsrechnung*, volume 37 of *Annales Academiae Scientiarum Fennicae: Ser. A 1.* Sana, 1947.

[Kit80] H. Kitajima. A Symmetric Cosine Transform. *IEEE Trans. Comput.*, C-29(4):317–323, 1980.

[KSK77] H. Kitajima, T. Sato, and T. Kurobe. Comparison of the discrete cosine and Fourier transforms as possible substitutes for the Karhunen-Loéve transform. *Trans. IECE Japan*, E60:279–283, 1977.

[Lan05] S. Lang. *Algebra*. Graduate Texts in Mathematics. Springer New York, 2005.

[LKM+06] J. Laska, S. Kirolos, Y. Massoud, R. Baraniuk, A. C. Gilbert, M. A. Iwen, and M. Strauss. Random sampling for analog-to-information conversion of wideband signals. In *Proc. IEEE/CAS Workshop on Design, Applications, Integration and Software*, pages 119–122, 2006.

[Loé48] M. Loéve. Fonctions Aléatoires de Second Ordre. *Processus Stochastiques et Mouvement Brownien*. P. Lévy, Ed. Hermann, Paris, 1948.

[LWC13] D. Lawlor, Y. Wang, and A. Christlieb. Adaptive sub-linear time Fourier algorithms. *Adv. Adapt. Data Anal.*, 5(1):1350003, 2013.

[Man92] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *Proc. International Colloquium on Automata, Languages and Programming*, pages 261–272, 1992.

[ME09] M. Mishali and Y. C. Eldar. Blind multiband signal reconstruction: Compressed sensing for analog signals. *IEEE Trans. Signal Process.*, 57(3):993–1009, 2009.

[ME10] M. Mishali and Y. C. Eldar. From Theory to Practice: Sub-Nyquist Sampling of Sparse Wideband Analog Signals. *IEEE J. Sel. Top. Signal Process.*, 4(2):375–391, 2010.

[MEDS11] M. Mishali, Y. C. Eldar, O. Dounaevsky, and E. Shoshan. Xampling: Analog to digital at sub-Nyquist rates. *IET Circuits Devices Syst.*, 5(1):8–20, 2011.

[MET08] M. Mishali, Y. C. Eldar, and J. A. Tropp. Efficient sampling of sparse wideband analog signals. In *Proc. IEEE 25th Convention of Electrical and Electronics Engineers*, pages 290–294, 2008.

[Mor73] J. Morgenstern. Note on a Lower Bound on the Linear Complexity of the Fast Fourier Transform. *J. ACM*, 20(2):305–306, 1973.

[Mor16] L. Morotti. Explicit universal sampling sets in finite vector spaces. *Appl. Comput. Harmon. Anal.*, 43:354–369, 2016.

[MV07] H. L. Montgomery and R. C. Vaughan. *Multiplicative Number Theory I: Classical Theory*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2007.

[MZIC18] S. Merhi, R. Zhang, M. A. Iwen, and A. Christlieb. A New Class of Fully Discrete Sparse Fourier Transforms: Faster Stable Implementations with Guarantees. *Fourier Anal. Appl.*, `https://doi.org/10.1007/s00041-018-9616-4`, 2018.

[Nus82] H. J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*. Springer Series in Information Sciences. Springer-Verlag, 1982.

[PD96] J. Prado and P. Duhamel. A polynomial-transform based computation of the 2-D DCT with minimum multiplicative complexity. *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 3:1347–1350, 1996.

[PM03] M. Püschel and J. Moura. The Algebraic Approach to the Discrete Cosine and Sine Transforms and Their Fast Algorithms. *SIAM J. Comput.*, 32(5):1280–1316, 2003.

[PPST19] G. Plonka, D. Potts, G. Steidl, and M. Tasche. *Numerical Fourier Analysis*. Birkhäuser Basel, 2019.

[PST01] D. Potts, G. Steidl, and M. Tasche. Fast Fourier Transforms for Nonequispaced Data: A Tutorial. *Modern Sampling Theory: Mathematics and Applications*, Boston, pages 247–270, 2001.

[PT05] G. Plonka and M. Tasche. Fast and numerically stable algorithms for discrete cosine transforms. *Linear Algebra Appl.*, 394:309 – 345, 2005.

[PT14] G. Plonka and M. Tasche. Prony methods for recovery of structured functions. *GAMM-Mitt.*, 37(2):239–258, 2014.

[PT16] D. Potts and M. Tasche. Reconstruction of sparse Legendre and Gegenbauer expansions. *BIT*, 56(3):1019–1043, 2016.

[PTV16] D. Potts, M. Tasche, and T. Volkmer. Efficient Spectral Estimation by MUSIC and ESPRIT with Application to Sparse FFT. *Front. Appl. Math. Stat.*, 2:1, 2016.

[PW16a] G. Plonka and K. Wannenwetsch. A deterministic sparse FFT algorithm for vectors with small support. *Numer. Algorithms*, 71(4):889–905, 2016.

[PW16b] G. Plonka and K. Wannenwetsch. Sparse FFT (small support), implemented in MATLAB. `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`, 2016. Accessed Nov 29, 2018.

[PW17a] G. Plonka and K. Wannenwetsch. A sparse fast Fourier algorithm for real non-negative vectors. *J. Comput. Appl. Math.*, 321:532 – 539, 2017.

[PW17b] G. Plonka and K. Wannenwetsch. Deterministic Sparse FFT, implemented in MATLAB. `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`, 2017. Accessed Nov 29, 2018.

[PWCW18] G. Plonka, K. Wannenwetsch, A. Cuyt, and W.-s. Lee. Deterministic sparse FFT for M-sparse vectors. *Numer. Algorithms*, 78(1):133–159, 2018.

[Rad68] C. M. Rader. Discrete Fourier Transforms When the Number of Data Samples Is Prime. *Proc. IEEE*, 56:1107 – 1108, 1968.

[RSR69] L. R. Rabiner, R. W. Schafer, and C. M. Rader. The Chirp-z Transform Algorithm. *IEEE Trans. on Audio Electroacoust.*, 17:86–92, 1969.

[RY90] K. R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, 1990.

[Sch02] M. Schatzman. *Numerical Analysis: A Mathematical Introduction*. Clarendon Press, 2002.

[SH86] N. Suehiro and M. Hatori. Fast algorithms for the discrete Fourier transform and for other transforms. *IEEE Trans. Acoust. Speech Signal Process.*, pages 642–644, 1986.

[SI13] B. Segal and M. A. Iwen. Improved sparse Fourier approximation results: faster implementations and stronger guarantees. *Numer. Algorithms*, 63(2):239–263, 2013.

[SI17] B. Segal and M. A. Iwen. Michigan State University's Sparse FFT Repository - GFFT - Improved sparse Fourier approximation results: faster implementations and stronger guarantees. `https://users.math.msu.edu/users/markiwen/Code.html`, 2017. Accessed Mar 12, 2019.

[ST91] G. Steidl and M. Tasche. A Polynomial Approach to Fast Algorithms for Discrete Fourier-Cosine and Fourier-Sine Transforms. *Math. Comput.*, 56:281–296, 1991.

[Ste92] G. Steidl. Fast radix-p discrete cosine transform. *Appl. Algebra Eng., Commun. Comput.*, 3(1):39–46, 1992.

[Ste98] G. Steidl. A note on fast Fourier transforms for nonequispaced grids. *Adv. Computat. Math.*, 9(3):337–352, 1998.

[Str99] G. Strang. The Discrete Cosine Transform. *SIAM Review*, 41(1):135–147, 1999.

[The18a] The GAP Group. GAP's documentation of `ChineseRem`. `https://www.gap-system.org/Manuals/doc/ref/chap14.html`, accessed Nov 28, 2018.

[The18b] The MathWorks. MATLAB's documentation of `fft`. `https://www.mathworks.com/help/matlab/ref/fft.html`, accessed Nov 28, 2018.

[The18c] The MathWorks. MATLAB's documentation of `idct`. `https://www.mathworks.com/help/signal/ref/idct.html`, accessed Nov 28, 2018.

[The18d] The MathWorks. MATLAB's documentation of `ifft`. `https://www.mathworks.com/help/matlab/ref/ifft.html`, accessed Nov 28, 2018.

[Wan83] Z. Wang. Reconsideration of "A Fast Computational Algorithm for the Discrete Cosine Transform". *IEEE Trans. Commun.*, 31(1):121–123, 1983.

[Wan84] Z. Wang. Fast Algorithms for the Discrete W Transform and for the Discrete Fourier Transform. *IEEE Trans. Acoust. Speech Signal Process.*, 32(4):803–816, 1984.

[Wan16] K. Wannenwetsch. Deterministic Sparse FFT Algorithms. Dissertation. `http://hdl.handle.net/11858/00-1735-0000-002B-7C10-0`, 2016.

[Win78] S. Winograd. On Computing the Discrete Fourier Transform. *Math. Comput.*, 32(141):175–199, 1978.

[WP89a] H. R. Wu and F. J. Paoloni. A structural approach to two-dimensional direct fast discrete cosine transform algorithms. *Proc. of the International Symposium on Computer Architechtures and DSP*, 1989.

[WP89b] H. R. Wu and F. J. Paoloni. A two-dimensional fast cosine transform algorithm. *Proc. International Conference on Image Processing*, pages 50–54, 1989.

[WP91] H. R. Wu and F. J. Paoloni. A two-dimensional fast cosine transform algorithm based on Hou's approach. *IEEE Trans. Signal Process.*, 39:544–546, 1991.

[Yav68] R. Yavne. An Economical Method for Calculating the Discrete Fourier Transform. In *Proc. 1968 Fall Joint Computer Conference, Part I*, pages 115–125, 1968.

[YG12] P. K. Yenduri and A. C. Gilbert. Compressive, collaborative spectrum sensing for wideband cognitive radios. *Proc. International Symposium on Wireless Communication Systems*, pages 531–535, 2012.

[YRR+12] P. K. Yenduri, A. Z. Rocca, A. S. Rao, S. Naraghi, M. P. Flynn, and A. C. Gilbert. A Low-Power Compressive Sampling Time-Based Analog-to-Digital Converter. *IEEE J. Em. Sel. Top. C.*, 2(3):502–515, 2012.