UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA
PROGRAMA DE PÓS-GRADUAÇÃO EM ESTATÍSTICA

LUAN PORTELLA DA SILVA

LOW-COMPLEXITY RECURSIVE DISCRETE FOURIER TRANSFORM APPROXIMATIONS FOR SPECTRAL ESTIMATION AND AUTOCORRELATION COMPUTATION

Recife
2023

LUAN PORTELLA DA SILVA

LOW-COMPLEXITY RECURSIVE DISCRETE FOURIER TRANSFORM APPROXIMATIONS FOR SPECTRAL ESTIMATION AND AUTOCORRELATION COMPUTATION

Tese apresentada ao Programa de Pós-Graduação em Estatística do Centro de Ciências Exatas e da Natureza da Universidade Federal de Pernambuco, como requisito parcial à obtenção do título de doutor em Estatística. Área de Concentração: Estatística aplicada

Orientador: Dr. Renato J. Cintra

Co-Orientador: Dr. Fábio M. Bayer

Recife

2023

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

S586l Silva, Luan Portella da
Low-complexity recursive discrete fourier transform approximations for spectral estimation and autocorrelation computation / Luan Portella da Silva. – 2023.
157 f.: il., fig., tab.

Orientador: Renato José de Sobral Cintra.
Tese (Doutorado) – Universidade Federal de Pernambuco. CCEN, Estatística, Recife, 2023.
Inclui referências e apêndice.

1. Estatística aplicada. 2. Transformada discreta de Fourier. 3. Algoritmos rápidos. 4. Transformadas aproximadas. 5. Processamento de sinais. 6. Autocorrelação. I. Cintra, Renato José de Sobral (orientador). II. Título.

310 CDD (23. ed.) UFPE- CCEN 2023 - 76

LUAN PORTELLA DA SILVA

LOW-COMPLEXITY RECURSIVE DISCRETE FOURIER TRANSFORM APPROXIMATIONS FOR SPECTRAL ESTIMATION AND AUTOCORRELATION COMPUTATION

Tese apresentada ao Programa de Pós-Graduação em Estatística da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Estatística. Área de concentração: Estatística aplicada.

Aprovado em: 27 de março de 2023.

BANCA EXAMINADORA

Prof. Dr. Renato José de Sobral Cintra
Orientador, UFPE

Prof. Dr. Fábio Mariano Bayer
Coorientador, UFSM

Prof. Dr. Vitor de Andrade Coutinho
Examinador Externo à Instituição, UFRPE

Prof. Dr. Thiago Lopes Trugillo da Silveira
Examinador Externo à Instituição, UFRGS

Profa. Dra. Bruna Gregory Palm
Examinadora Externa à Instituição
Blekinge Institute of Technology (BTH), Suécia

Prof. Dr. Ricardo Menezes Campello de Souza, UFPE
Examinador Externo ao Programa

ABSTRACT

The importance of the discrete Fourier transform (DFT) stems from its rich physical interpretation and its mathematical principles. In signal processing, the DFT plays a key role in spectral estimation, filtering, and fast signal convolutions. In order to reduce the computational cost of the DFT, a series of algorithms called fast Fourier transforms (FFT) have been developed. Capable of reducing the multiplicative complexity, the FFT has allowed the widespread use of the DFT. However, even with the reduced arithmetic complexity derived from the FFT, the DFT computation can still be an obstacle in applications with restrictive conditions, such as energy consumption, chip occupancy area, and time. If small inaccuracies are allowed under such conditions, the DFT computation can be approximated. The present work approaches four different topics related to the DFT estimation. First, based on iterations of Cooley-Tukey's radix-N algorithm, approximate transforms for signals of lengths $N\hat{}2\hat{}n$ are proposed. Second, an approximate version of the Good-Thomas algorithm capable of performing the DFT calculation without multiplications is presented. Thirdly, using the canonical signed digit (CSD) representation, we present approximations for the transformation and twiddle factor matrices to also propose a multiplication-free Cooley-Tukey algorithm. Finally, a low-complexity estimator is proposed to calculate the autocorrelation of a given signal based on the properties of the DFT. All proposals include (i) construction of fast algorithms, (ii) evaluation of arithmetic complexity, and (iii) error analysis.

Keywords: discrete Fourier transform; fast algorithms; approximate transforms; signal processing; autocorrelation.

RESUMO

A importância da transformada discreta de Fourier (DFT) decorre da sua rica interpretação física e de seus princípios matemáticos. Em processamento de sinais, a DFT desempenha um papel fundamental em estimação espectral, filtragem e convolução rápidas de sinais. Para reduzir o custo computacional da DFT, uma série de algoritmos, denominados transformadas rápidas de Fourier (FFT) têm sido desenvolvidos. Capazes de reduzir a complexidade multiplicativa, os algoritmos rápidos permitiram que o uso da DFT fosse difundido. No entanto, mesmo com a redução da complexidade aritmética oriunda das FFTs, o cômputo da DFT pode ser um obstáculo em aplicações que apresentam condições restritivas, como consumo de energia, área de ocupação no chip e tempo de processamento. Se pequenos desvios de acurácia forem permitidos em tais condições, o cálculo da DFT pode ser realizado de forma aproximada. O presente trabalho aborda quatro diferentes tópicos relacionados com a estimação da DFT. Primeiramente, baseado em iterações do algoritmo Cooley-Tukey de base N, são propostas transformadas aproximadas para sinais de comprimento $N\hat{}2\hat{}n$. Segundo, uma versão aproximada do algoritmo de Good-Thomas capaz de realizar o cálculo da DFT sem necessidade de multiplicações é apresentada. Terceiro, utilizando a representação em dígito de sinal canônico (CSD), nós apresentamos aproximações para as matrizes de transformação e fatores de rotação com o intuito de também propor um algoritmo de Cooley-Tukey livre de multiplicações. Por último, um estimador de baixa complexidade é proposto para o cálculo da autocorrelação baseado nas propriedades da DFT. Todas as propostas contêm (i) construção de algoritmos rápidos, (ii) avaliação da complexidade aritmética e (iii) análise de erro.

Palavras-chave: transformada discreta de Fourier; algoritmos rápidos; transformadas aproximadas; processamento de sinais; autocorrelação.

LIST OF ABBREVIATIONS AND ACRONYMS

ACF	Sample autocorrelation function
AI	Artificial intelligence
APFA	Approximate prime factor algorithm
AR	Autoregressive
CSD	Canonical signed digit
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
DSP	Digital signal processing
FFT	Fast Fourier transform
IDFT	Inverse discrete Fourier transform
IoT	Internet of things
MAPE	Mean absolute percentage error
PFA	Prime factor algorithm
PSD	Power spectrum density

LIST OF SYMBOLS

Symbol	Description
$\mathbf{F}_N$	Matrix representation of the N-point DFT
j	Imaginary unity, $j \triangleq \sqrt{-1}$
$\mathbf{\Omega}_N$	$N \times N$ twiddle factor matrix
$\mathbf{C}$	An arbitrary matrix
$\mathbf{D}$	An arbitrary matrix
$\mathscr{M}_{\mathbb{C}}(N)$	Set of square matrices of order N with complex entries
$f_{\mathrm{CTS}}(\cdot)$	Cooley-Tukey scaling function
$\hat{\mathbf{F}}_N$	Matrix representation of an approximation for the N-point DFT
$M_R(\cdot)$	Real multiplicative complexity
$\hat{\mathbf{F}}_N^i$	Matrix representation of an approximation for the N-point DFT obtained by the scaling method (defined in Section 3.1.1)
$\mathbf{B}_m$	Butterfly-structure (defined in Section 3.2.1)
$\hat{\mathbf{F}}_N^*$	Matrix representation of an approximation for the N-point DFT obtained by optimization problems (defined in Section 4.1)
α	Real number referred to as expansion factor
$\mathscr{D}$	Interval of values that α can assume
$g(\cdot)$	An integer function
$\hat{\mathbf{T}}_N$	A low-complexity matrix generated by an integer matrix function and an α (defined in Section 4.1)
$\hat{\mathbf{S}}_N$	A diagonal matrix obtained from $\hat{\mathbf{T}}_N$ (defined in Section 4.1)
$\mathscr{P}$	A low-complexity subspace
$\mathscr{M}_N(\mathscr{P})$	Set of all square matrices of order N generated by $\mathscr{P}$
α^*	Expansion factor obtained by solving an optimization problem (defined in Section 4.1.2)

$\hat{\mathbf{T}}_N^*$	A low-complexity matrix generated by an integer matrix function and an α^* (defined in Section 4.1.2)
$\hat{\mathbf{S}}_N^*$	A diagonal matrix obtained from $\hat{\mathbf{T}}_N^*$ (defined in Section 4.1.2)
$\varepsilon(\cdot)$	Total error energy
$\|\|\cdot\|\|_{\mathrm{F}}$	Frobenius norm
$M(\cdot)$	Mean absolute percentage error
$\phi(\cdot)$	Deviation from orthogonality
$\hat{\mathbf{F}}'_N$	Matrix representation of an approximation for the N-point DFT obtained by optimization problems with the diagonal matrix approximate by CSD (defined in Section 4.3.4)
$\hat{\mathbf{F}}^*_{N\text{-i}}$	Matrix representation of a hybrid approximation for the N-point DFT obtained by optimization problems (defined in Section 4.4.6)
$\hat{\mathbf{F}}'_{N\text{-i}}$	Matrix representation of a hybrid approximation for the N-point DFT obtained by optimization problems with the diagonal matrix approximate by CSD (defined in Section 4.4.6)
$H_m(\cdot,\cdot)$	Frequency response of the m-th row(defined in Section 4.5.1)
$\mathrm{bin}(\cdot)$	Function to convert a decimal number to binary
$\mathrm{csd}(\cdot)$	Function to convert a binary number to CSD
$\mathrm{crop}(\cdot,i)$	Function to crop the CSD representantion according to $(i+1)$-nonzero digits
$\hat{\mathbf{F}}_N^{(i)}$	Matrix representation of an approximation for the N-point DFT obtained by cropping th CSD representation of $\mathbf{F}_N$ according to $(i+1)$-nonzero digits (defined in Section 5.1.2)
$\hat{\mathbf{\Omega}}_{32}^{(i)}$	Matrix representation of an approximation for the $N\times N$ twiddle factor matrix obtained by cropping th CSD representation of $\mathbf{\Omega}_N$ according to $(i+1)$-nonzero digits (defined in Section 5.1.2)
$\boldsymbol{\theta}$	Complete autocorrelation
$\hat{\boldsymbol{\theta}}$	Autocorrelation estimator

$\mathbf{z}$	Standardized signal vector
$\hat{\boldsymbol{\rho}}$	Autocorrelation estimator of a standardized signal
$\mathbf{P}$	Periodogram
$\mathscr{F}(\cdot)$	Function that denotes the operation of taking the DFT of the argument
$\mathscr{F}^{-1}(\cdot)$	Function that denotes the operation of taking the IDFT of the argument
$\mathbf{z}_{\text{aug}}$	Standardized augmented signal
$\hat{\boldsymbol{\rho}}_M^{\star}$	Low-complexity autocorrelation estimator

CONTENTS

1 INTRODUCTION

In this chapter, we present the motivation and state-of-the-art for the research topics. At the end of this chapter, we detail the main goals, document structure, and computational resources employed to conduct the simulations, obtain, and evaluate the approximations.

1.1 MOTIVATION

The discrete Fourier transform (DFT) is a central tool in signal processing [1], finding applications in a very large number of contexts, such as spectral estimation [2], filtering [3], data compression [4], machine learning [5], and fast convolution [6], to cite but a few. The widespread usage of the DFT is due to its rich physical interpretation [7] and the existence of efficient algorithms for its computation [8]. The DFT presents good properties that allow its immediate applicability in a variety of problems. For example, we can mention: its relationship with the correlation (estimation of the auto- and cross-correlation) [9, p.44]; the circular convolution, which can be implemented in terms of the DFT in popular software such as Matlab [10], Python [11], and R [12]; the linearity that enables us to separate and choose the components of a signal which is the principle of filtering [7, p. 72]; and its use to accurately compute derivatives [13, p. 63], fractional derivatives [14], partial differential equations, and integrals [7, p. 113].

In general, the resources to apply the DFT are limited at some level either by time (e.g., real-time applications [15]), energy (e.g. portable devices [16]), or space (chip area [17]). Therefore, reducing the computational cost is a key procedure to make a DFT-based tool implementable in hardware [18].

For this purpose, a series of algorithms have been developed to reduce the computational cost of the DFT. Although there are different ways to evaluate the cost/performance of the algorithms, the number of mathematical operations is usually considered because it is independent of the available technology [3, p. 748]. In terms of arithmetic complexity, the direct computation of the 1D N-point DFT is an operation in $\mathcal{O}(N^2)$— which

is prohibitively expensive [19] for most applications. Efficient algorithms [7, 8, 20] collectively known as fast Fourier transforms (FFTs) [21] are capable of evaluating the DFT with much fewer numerical operations placing, for example, the resulting complexity in $\mathcal{O}(N\log_2 N)$ [20] when the signal length is a power of two or approximately $N(N_1+N_2)$ when N can be expressed as the product of two relatively prime numbers, N_1 and N_2 [21, p. 81].

FFT algorithms have contributed to the development of numerous digital signal processing (DSP) applications from which we can highlight speech [22–24] and image processing [25–27], statistical signal analysis [28–30], and communication systems [31–33]. Despite the substantial reduction in arithmetic operations provided by the FFT algorithms, the remaining complexity can still be significant in contexts were severe restrictions in computational power and/or in energy autonomy [34] are present. Such restrictive conditions are met in the framework of wireless communication [35, 36], embedded systems [37, 38], and the internet of things (IoT) [39, 40]. Even in scenarios with powerful hardware available, computing time can be a restrictive factor, as is the case in artificial intelligence (AI) that works with problems involving billions to trillions of data points [41, 42] in applications like gaming (interactive virtual environments) [43], finances (stock marketing) [44], and vehicle control systems (to prevent accidents) [45]. In the context of image and video coding, where the DFT-related [46] discrete cosine transform (DCT) is the tool of choice [47], scenarios of extreme resource limitations are present in image fusion [48] and diffusion [49], spatial decorrelation in video tracking [50], unmanned aerial vehicles [51], and low-powered devices in real-time applications in general [15, 52, 53]. Although FFT algorithms made real-time digital signal processing possible [54], under such restrictive conditions, time and power consumption must be alleviated [55]. A possible solution is the trade-off between accuracy and low-cost computation, where a degree of error is tolerated [56]. This scenario might concern the design of low-end technology such as digital signal processors [57] or high-end technology (beyond 5G) especially in wideband services of wireless networks [55].

To address such issues, designers of DCT-based methods resort to matrix ap-

proximation techniques [47, 58] aiming at deriving integer matrices whose complexity is regarded as extremely low [59–61], while retaining the desired properties of the DCT computed by the definition [60]. A number of DCT approximations have been successfully developed such as the SDCT [62], the approximations by Bouguezel–Ahmad–Swamy [63,64], and the Cintra–Bayer approximations [59,65,66]. Such approximations make use of trivial multiplications, as for example multiplications by -2, -1, 0, $+1$, $+2$ that cane be implemented in hardware by simple additions or bit-shifting operations. The cost to perform a multiplication, particularly of floating-point, depends on the specifics of the hardware. However, it is well known that even on modern hardware a double-precision floating-point multiplication requires several clock cycles restricting the implementation of transforms on low-powered hardware [53, 67]. In general, approximate transforms are multiplierless methods [68] that require only a reduced number of addition operations [59] becoming an alternative for such scenarios.

Inspired by such approximation-based methodology, in [69], a suite of multiplierless DFT approximations was derived for N=3, 5, 7, 8, 16, and 32 [70–72]. These DFT approximations were demonstrated to provide spectral estimates close to the DFT computation by definition while requiring only additions. Such approximations can be used as building blocks to obtain approximate transforms of larger blocklengths [73–75] and for the approximate estimation in applications that use the DFT [76–78].

In general, data blocklengths that are powers of two are most commonly used [21, p. 5]. In addition to their simpler factorization, it is often possible to use FFT algorithms recursively. On the other hand, prime lengths can be used in algorithms such as the prime factor algorithm (PFA) [79, 80], which do not need twiddle factors [81]. Twiddle factors are auxiliary complex multiplications by roots of unity that are unavoidable when the transform length is composite with non-coprime factors [81].

The non-power of two DFTs [82–84] is a promising field in 5G broadcasting which usually presents input of the form $2^n \times 3^m$(n and m are positive integers) [85]. Such DFTs are also useful when the signal length can not be chosen (digital radio mondiale [86], channel equalization [87, 88], or specific convolutions), or when it is irrelevant (encryp-

tion [89] and beam-forming [90]).

In light of the above discussion, it is possible to notice that technological trends suggest an ever-increasing demand for faster and more efficient algorithms and approximate transforms. The need to process massive amounts of data in real-time and/or under restrictive conditions keeps this line of research attractive [91].

1.2 STATE-OF-THE-ART

Broadly, finding good approximate transforms is a challenging task, because it is often posed as an integer non-linear matrix optimization problem with a large number of variables [92]. Thus, as N increases, obtaining good approximations becomes an exceedingly demanding problem to be solved [75]. As a consequence, designers of DFT approximations make use of indirect methods such as (i) mathematical relationships between small-sized and large-sized DFT matrices [20], (ii) matrix functional recursions [93] and, (iii) matrix decompositions [94]. The systematic derivation of good DFT approximations for large blocksizes is still an open problem and technical advances occur in a case-by-case fashion due to the inherent numerical difficulties of finding good integer matrices. In the following, we present the main works on DFT approximations and their obtaining methods found in the literature.

In [69], the author proposed multiplierless approximations using Pareto optimality [95, p. 19], integer functions [65], and matrix parameterization [96] along with a collection of factorizations to minimize the remaining additions. In this work, approximations for odd lengths $N = 3$, 5, and 7 and power-of-two lengths $N = 8$, 16, and 32 were considered. The approximation for $N = 8$ contributed to the papers in [70, 97–99] and the patent in [100]; for $N = 16$ contributed to [72, 99]; and for $N = 32$ was used to obtain approximations for the 1024-point DFT [73, 74] and one million-point DFT approximation [75].

In [101], multiplierless approximations for the 3-, 5-, 7-, 8-, 9-, and 16-point DFTs were proposed using scaling factors [47, p. 274]. The approach involved choosing the factors based on the desired accuracy. The author also presented frameworks for $N =$ 30, 72, 240, and 504 employing the PFA and the Winograd small fast Fourier transform

algorithm [102].

In competition with [72], approximations for the 16- and 32-point DFT were proposed in [103]. These approximations combined twiddle factor merging [104] and common subexpression to share duplicated multiplications using the Cooley–Tukey radix-2 decimation-in-time [21, p. 72]. Although this technique reduces the approximation error, multiplications are still required.

In [105], the authors expanded the low-complexity subspace (set of values that the coefficients of the approximate can assume) to propose an approximation for the 8-point DFT and compete with [70]. This approximation would later be used in [106] to obtain an approximation for the 16-point DFT using the Cooley-Tukey algorithm.

A method to obtain multiplierless DFT approximations is to use the Cooley-Tukey radix-2 or radix-4 algorithm (which requires trivial multiplications in the fundamental blocks) and approximate the twiddle factor matrices. This is the case of the work presented in [107] and [108]. In the first, the authors used a truncated representation of the canonical signed digit (CSD) [109] to find the nearest neighbors for the twiddle factor matrix coefficients. This method was employed to obtain approximations for $N = 32$, 64, 128, and 256. In the second, the twiddle factors were approximated using sum-of-powers-of-two [110] and CSD, and the work considered $N = 8$, 16, 32, 64, and 128.

The growing demand for new applications in the fields of IoT, IA, big data, and machine learning allowed approximation techniques to gain popularity and it is expected that the use of these methods will be more common than the traditional ones [91].

1.3 MAIN GOALS

Our main objective is to develop recursive algorithms capable of performing spectral estimation using the DFT and the estimation of the autocorrelation with low arithmetic complexity .

Specifically, we aim at:

1. Propose a scaling method based on the Cooley-Tukey algorithm with reduced multiplicative complexity;

2. Propose a multiplication-free algorithm based on the Good-Thomas algorithm;
3. Propose a method to obtain an approximation for the blocklengths used in the algorithm proposed in Item 2;
4. Propose a multiplication-free algorithm based on the Cooley-Tukey algorithm;
5. Explore efficient numerical representations for the required matrices in Item 4;
6. Propose low-complexity estimators for the autocorrelation based on the algorithms proposed in item 4;
7. Propose factorizations for the approximate matrices to reduce the remaining complexity;
8. Evaluate the performance of the proposed approximations using error measures;
9. Compare the proposed approximations against competing methods found in the current literature.

1.4 DOCUMENT STRUCTURE

The rest of the document is organized as follows. Chapter 2 contains the required mathematical tools for the proposals presented in the following chapters. Chapters 3, 4, 5, and 6 detail a proposed method each, with their methodology, assessment metrics, fast algorithms, and computational complexity. Chapter 7 concludes the work and summarizes the important points.

In Chapter 3, we present a version of the Cooley-Tukey algorithm in which low-complexity transforms are applied to input signals that can be expressed in length of $N^{(2^n)}$, where n is a positive integer. We applied the proposed method to the 2^{20}-point DFT.

In Chapter 4, we introduce an approach to obtain multiplierless transforms. Then, we combine the proposed methodology with the prime factor algorithm to maintain the entire computation of the DFT approximation free of multiplications. The method is demonstrated for the 1023-point DFT.

In Chapter 5, we use the CSD to obtain approximations for the transform and twiddle factor matrices required in the Cooley-Tukey algorithm. The method allows approximate computation of the DFT for signals with length 2^m without multiplications.

We employed such approach to present approximations for the 1024-point DFT.

In Chapter 6, we propose a low-complexity estimator to compute the autocorrelation function based on the properties of the DFT. The approximations obtained in Chapter 5 are used to estimate the autocorrelation parameters of autoregressive processes.

Finally, in Chapter 7, we present some concluding remarks, main contributions and published works of each line of research are listed, and the next steps to be taken are provided.

1.5 COMPUTATIONAL RESOURCES

The optimization and simulation problems were solved in C (Chapter 3) and R (Chapters 4, 5, and 6) languages on a machine with the following specifications: a computer equipped with Hexa-core 4.5 GHz Intel(R) Core(R) I7-9750H CPU, 32 GB RAM running Ubuntu 20.04 LTS 64-bit, and GPU GeForce RTX 2060. In parallel, we also used a virtual machine from the Google Cloud Platform with the following specifications: 8 cores 3.8 GHz Intel (Cascade Lake) with 32 GB RAM running Ubuntu 20.04 LTS 64-bit.

2 MATHEMATICAL BACKGROUND

In this chapter, we review essential tools and concepts required in the next chapters. First, the DFT and inverse discrete Fourier transform (IDFT) are presented. Second, the Cooley-Tukey and Good-Thomas algorithms are presented in both traditional and matrix forms. Third, an algorithm for computing complex multiplications is explained in detail.

2.1 THE DISCRETE FOURIER TRANSFORM

Discrete Fourier transform (DFT) DFT The DFT and IDFT are linear transformations. While the DFT maps an N-point discrete signal $\mathbf{x} = [x[0]x[1]\dots x[N-1]]^\top$ into an output signal $\mathbf{X} = [X[0]X[1]\dots X[N-1]]^\top$, the IDFT does the opposite process. The DFT and IDFT [3, p. 750] are given, respectively, by

$$X[k] \triangleq \sum_{i=0}^{N-1} \omega_N^{ik} \cdot x[i], \quad k = 0,1,\dots,N-1, \tag{2.1}$$

$$x[i] = \frac{1}{N} \sum_{k=0}^{N-1} \omega_N^{-ik} \cdot X[k], \quad i = 0,1,\dots,N-1, \tag{2.2}$$

where $X[k]$ is the kth DFT coefficient, $\omega_N \triangleq e^{-j\frac{2\pi}{N}}$ is the Nth root of unity, and $j \triangleq \sqrt{-1}$.

The DFT and the IDFT can also be expressed in matrix format according to the following expressions:

$$\mathbf{X} \triangleq \mathbf{F}_N \cdot \mathbf{x}, \tag{2.3}$$

$$\mathbf{x} = \frac{1}{N} \overline{\mathbf{F}_N} \cdot \mathbf{X}, \tag{2.4}$$

where the bar symbol denotes the complex conjugate [94, p. 79] and $\mathbf{F}_N$ is the DFT matrix defined by

$$\mathbf{F}_N = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \dots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \dots & \omega_N^{2(N-1)} \\ 1 & \omega_N^3 & \omega_N^6 & \dots & \omega_N^{3(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \dots & \omega_N^{(N-1)(N-1)} \end{bmatrix}.$$

The exact computation of the DFT in computers is impossible due to finite arithmetic precision which leads to round-off errors [111, p. 600]. Therefore, when the exact computation of the DFT is mentioned is referred to the computation of the DFT and IDFT by the definitions established in Sections 2.1 and 2.2.

2.2 COOLEY-TUKEY ALGORITHM

Let $N = N_1 \cdot N_2$ represent the blocklength of the input signal where N_1 and N_2 are integers. To rewrite (2.1) into a two-dimensional transform consider the following indexing:

$$\begin{aligned} i &= i_1 + N_1 i_2, \\ k &= N_2 k_1 + k_2, \end{aligned}$$

where $i_1, k_1 = 0, \ldots, N_1 - 1$, and $i_2, k_2 = 0, \ldots, N_2 - 1$. Then,

$$X[N_2 k_1 + k_2] = \sum_{i_2=0}^{N_2-1} \sum_{i_1=0}^{N_1-1} \omega_N^{(i_1 + N_1 i_2)(k_2 + N_2 k_1)} x[i_1 + N_1 i_2]. \tag{2.5}$$

Expanding the product in the exponent we obtain

$$X[N_2 k_1 + k_2] = \sum_{i_1=0}^{N_1-1} \sum_{i_2=0}^{N_2-1} \left[\omega_N^{(i_1 N_2 k_1)} \omega_N^{(i_1 k_2)} \omega_N^{(N_1 N_2 i_2 k_1)} \omega_N^{(N_1 i_2 k_2)} \right] x[i_1 + N_1 i_2] \tag{2.6}$$

$$= \sum_{i_1=0}^{N_1-1} \omega_N^{(i_1 N_2 k_1)} \left[\omega_N^{(i_1 k_2)} \sum_{i_2=0}^{N_2-1} \omega_N^{(N_1 N_2 i_2 k_1)} \omega_N^{(N_1 i_2 k_2)} x[i_1 + N_1 i_2] \right]. \tag{2.7}$$

Since ω_N has multiplicative order equals to $N = N_1 \cdot N_2$, then we have that $\omega_N^{(N_1 N_2 i_2 k_1)} = 1$. Therefore, the expression (2.7) is reduced to

$$X[N_2 k_1 + k_2] = \sum_{i_1=0}^{N_1-1} \omega_N^{(i_1 N_2 k_1)} \left[\omega_N^{(i_1 k_2)} \sum_{i_2=0}^{N_2-1} \omega_N^{(N_1 i_2 k_2)} x[i_1 + N_1 i_2] \right]. \tag{2.8}$$

The mapping detailed in (2.8) applied to the signal components is known as address shuffling [21, p. 69] and the terms $\omega_N^{(i_1 k_2)}$ are referred to as twiddle factors [81, 112].

As detailed in (2.8), the Cooley-Tukey algorithm [81, 113] is traditionally presented in terms of a mathematical formalism based on: (i) Fourier-kernel weighted summations and (ii) index-variable substitutions to account for Fourier transformation and the

1D to 2D array indexing. However, a matrix-based presentation [21, 74] can be adopted which can be useful because popular software packages such as Matlab, R, Octave, and Julia are matrix oriented. In this way, by explicitly following the Cooley-Tukey algorithm, the N-point DFT can be computed by means of:

1. address-shuffling the 1D input column vector into a 2D $N_1 \times N_2$ array;
2. computing the N_2-point DFT of each array column of the 2D $N_1 \times N_2$ array using FFT calls;
3. element-wise multiply the resulting matrix by the twiddle-factors;
4. computing the N_1-point DFT of each row of the resulting 2D $N_1 \times N_2$ array using FFT calls; and
5. undoing the address shuffling to convert the obtained 2D $N_1 \times N_2$ array into the final 1D output column vector.

The 1D to 2D mapping can be accomplished by means of the inverse vectorization operator [114–116] which obeys the following mapping:

$$\text{invvec}\left(\begin{bmatrix} x[0] & x[1] & \cdots & x[N-1] \end{bmatrix}^{\top}\right) = \begin{bmatrix} x[0] & x[N_1] & \cdots & x[N_1(N_2-1)] \\ x[1] & x[N_1+1] & \cdots & x[N_1(N_2-1)+1] \\ \vdots & \vdots & \ddots & \vdots \\ x[N_1-1] & x[2N_1-1] & \cdots & x[N-1] \end{bmatrix}. \tag{2.9}$$

Based on the 1D to 2D mapping in (2.9) we can show that the N-point DFT can be represented in the following matrix expression based on the Cooley-Tukey algorithm:

$$\mathbf{X} = \mathbf{F}_N \cdot \mathbf{x} = \text{vec}\left(\left\{\mathbf{F}_{N_1} \cdot \left[\mathbf{\Omega}_{N_2 \times N_1} \circ \left(\mathbf{F}_{N_2} \cdot (\text{invvec}(\mathbf{x}))^{\top}\right)\right]^{\top}\right\}^{\top}\right), \tag{2.10}$$

where $\mathbf{x}$ and $\mathbf{X}$ are N-point vectors, $\text{vec}(\cdot)$ is the matrix vectorization operator [94, p. 239], $\circ$ is the Hadamard element-wise multiplication [94, p. 251], the superscript $^{\top}$ denotes simple transposition (non-Hermitian), and $\mathbf{\Omega}_{N_2 \times N_1}$ is the twiddle-factor matrix given by $\mathbf{\Omega}_{N_2 \times N_1} = (\omega_N^{i \cdot k})_{i=0,1,\ldots,N_1-1, k=0,1,\ldots,N_2-1}$.

A specific case of the Cooley-Tukey algorithm is the radix-N algorithm when $N_2 = N_1$. In this way, considering the transposition properties, $\mathbf{\Omega}_{N_2 \times N_1} = \mathbf{\Omega}_{N_1 \times N_2}^{\top}$, $\mathbf{F}_{N_1} =$

$\mathbf{F}_{N_1}^{\top}$, and $\mathbf{F}_{N_2} = \mathbf{F}_{N_2}^{\top}$, (2.10) can be written as

$$\mathbf{X} = \text{vec}\left(\left[\mathbf{\Omega}_{N_2 \times N_1} \circ \left(\mathbf{F}_{N_2} \cdot (\text{invvec}(\mathbf{x}))^{\top}\right)\right] \cdot \mathbf{F}_{N_1}\right). \tag{2.11}$$

The inner DFT call corresponds to row-wise transformation of invvec($\mathbf{x}$), whereas the outer DFT performs column-wise transformations on the resulting intermediate computation.

2.3 PRIME FACTOR ALGORITHM

Comparable to the more popular Cooley-Tukey FFT [21, p. 80], the PFA is a factorization-based FFT capable of computing the N-point DFT, where $N = N_1 \times N_2$, with N_1 and N_2 relatively prime, i.e., $\gcd(N_1, N_2) = 1$, being $\gcd(\cdot,\cdot)$ the greatest common divisor of two integers operator. The method is based on a number-theoretical re-indexing [8, p. 144] of the input signal coefficients into a two-dimensional array [3, p. 846] which is based on the Chinese remainder theorem [21, p. 58].

The PFA mapping requires a shuffle of the input and output as follows:

$$i = i_1 N_2 n_2 + i_2 N_1 n_1 \pmod{N}, \tag{2.12}$$

$$k = N_2 k_1 + N_1 k_2 \pmod{N}, \tag{2.13}$$

where n_1 and n_2 are integers that satisfy $(n_1 \cdot N_1 + n_2 \cdot N_2) \mod N = 1$ [8, p. 167].

Rewriting (2.1) with the indexing in (2.12), we obtain:

$$X[N_2 k_1 + N_1 k_2] = \sum_{i_2=0}^{N_2-1} \sum_{i_1=0}^{N_1-1} \omega_N^{(i_1 N_2 n_2 + i_2 N_1 n_1)(N_2 k_1 + N_1 k_2)} x[i_1 N_2 n_2 + i_2 N_1 n_1]. \tag{2.14}$$

Then, expanding the product in the exponent

$$X[N_2 k_1 + N_1 k_2] = \sum_{i_1=0}^{N_1-1} \sum_{i_2=0}^{N_2-1} \left[\omega_N^{(i_1 N_2 n_2 N_2 k_1)} \omega_N^{(i_1 N_2 n_2 N_1 k_2)} \omega_N^{(i_2 N_1 n_1 N_2 k_1)} \omega_N^{(i_2 N_1 n_1 N_1 k_2)}\right] x[i_1 N_2 n_2 + i_2 N_1 n_1]. \tag{2.15}$$

Dropping the terms of ω_N that involve $N_1 N_2$, (2.15) is reduced to

$$X[N_2 k_1 + N_1 k_2] = \sum_{i_1=0}^{N_1-1} \omega_N^{(i_1 N_2^2 n_2 k_1)} \left[\sum_{i_2=0}^{N_2-1} \omega_N^{(i_2 N_1^2 n_1 k_2)} x[i_1 N_2 n_2 + i_2 N_1 n_1]\right]. \tag{2.16}$$

In matrix format, the PFA can be computed according to the following description:

1. Obtain n_1 and n_2 that satisfy $(n_1 \cdot N_1 + n_2 \cdot N_2) \mod N = 1$ [8, p. 167];
2. Map $\mathbf{x}$ into a block of size $N_1 \times N_2$ according to the following 1D to 2D rearrangement of elements:

$$\text{map}\left(\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}\right) = \begin{bmatrix} x[0] & x[N_1 n1] & \dots & x[(N_2-1)N_1 n1] \\ x[N_2 n_2] & x[N_1 n1 + N_2 n_2] & \dots & x[N_1 n1 + (N_2-1)N_1 n1] \\ \vdots & \vdots & \ddots & \vdots \\ x[(N_1-1)N_2 n_2] & x[(N_1-1)N_1 n1 + N_2 n_2] & \cdots & x[(N_1-1)N_2 n_2 + (N_2-1)N_1 n1] \end{bmatrix};$$

3. Compute the N_2-point DFT of each column of the 2D array obtained in Step 2;
4. Compute the N_1-point DFT of each row of the resulting 2D array from Step 3;
5. Reconstruct the vector $\mathbf{X}$ from the resulting block according to the following mapping:

$$\text{invmap}\left(\begin{bmatrix} X[0] & X[N_1] & \cdots & X[(N_2-1)N_1] \\ X[N_2] & X_{[N_1+N_2]} & \cdots & X[N_2+(N_2-1)N_1] \\ \vdots & \vdots & \ddots & \vdots \\ X[(N_1-1)N_2] & X[(N_1-1)N_2+N_1] & \cdots & X[(N_1-1)N_2+(N_2-1)N_1] \end{bmatrix}\right) = \begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \vdots \\ X[N-1] \end{bmatrix}.$$

All index operations are performed in modulo N arithmetic, ensuring the correct size of the arrays. The algorithm can be synthesized as follows:

$$\mathbf{X} = \text{invmap}\left(\mathbf{F}_{N_1} \cdot \left[\mathbf{F}_{N_2} \cdot (\text{map}(\mathbf{x}))^{\top}\right]^{\top}\right). \tag{2.17}$$

Notice that if N_1 or N_2 can be decomposed into relatively prime factors, then the algorithm can be reapplied. The N_1- and N_2-point transformations are referred to as ground transformations.

2.4 FAST ALGORITHM TO COMPUTE COMPLEX MULTIPLICATIONS

Let c and f represent complex elements, then they can be expressed in the following forms [94, p. 2]:

$$c = a + jb,$$
$$f = d + je,$$

where a, b, d, and e are real numbers. A simple way to express the complex multiplication between c and f is:

$$c \cdot f = (a + jb) \cdot (d + je) \tag{2.18}$$
$$= ad + j(ae) + j(bd) - be \tag{2.19}$$
$$= ad - be + j(ae + bd). \tag{2.20}$$

If the complex multiplications are computed according to (2.20), then four real multiplications and two real additions are required. This approach is useful when multiplications are trivial. However, if the multiplications are not trivial, then we can reduce them by performing the computation as follows:

$$c \cdot f = e(a-b) + a(d-e) + je(a-b) + jb(d+e) \tag{2.21}$$
$$= e(a-b) + a(d-e) + j(e(a-b) + b(d+e)). \tag{2.22}$$

We can take advantage of the redundant term $e(a-b)$ which reduces one multiplication compared to (2.20). Therefore, the arithmetic cost of (2.22) is three real multiplications and five real additions.

Rewriting (2.22) in matrix form and splitting the real and imaginary of (2.22), we have

$$\begin{bmatrix} c \\ f \end{bmatrix} = \begin{bmatrix} e(a-b) + a(d-e) \\ e(a-b) + b(d+e) \end{bmatrix}. \tag{2.23}$$

Notice that (2.23) can be factored as follows:

$$\begin{bmatrix} e(a-b)+a(d-e) \\ e(a-b)+b(d+e) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} a(d-e) \\ b(d+e) \\ e(a-b) \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} (d-e) & 0 & 0 \\ 0 & (d+e) & 0 \\ 0 & 0 & e \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}. \tag{2.24}$$

If d and e are constants, then $d-e$ and $d+e$ are also constants and can be computed off-line [21, p. 3]. Therefore, according to (2.24), one complex multiplication can be translated into three real multiplications and three real additions.

3 RADIX-N ALGORITHM FOR COMPUTING N^{2^n}-POINT DFT APPROXIMATIONS

In this chapter, we introduce a variant of the Cooley-Tukey FFT algorithm [81, 112, 113] in which instead of breaking down an N-point transformation into smaller blocks, our method maps an N-point transformation into an N^2-point transformation with reduced multiplicative complexity, achieved by trading computational precision. This approach is classified as a scaling method and it can be iterated, furnishing large-sized transformations. Large-sized FFT algorithms (i.e., millions-billions of frequency bins) are necessary for some challenging DSP applications like radar [97, 99, 117], sonar [118–120], radio astronomy [121–123], microwave imaging [124–126], and spectrum sensing [16, 127–129]. In areas such as machine learning and AI [130–134], where extremely large amounts of data often need to be processed, a judicious trade-off between precision and multiplicative complexity can generate substantial gains in computation speed [135].

The Cooley-Tukey algorithm is considered "the most important numerical algorithm in our lifetime" by algorithms expert Gilbert Strang [136]. This opinion is perhaps based on the seemingly endless number of applications where the frequency domain representation of a discrete domain signal has to be computed in a fast, efficient, and numerically meaningful way.

The Cooley-Tukey algorithm applies to band-limited and sampled discrete-domain signals regardless of the spectral region of support in the Nyquist interval. That is, the method can be used for both sparse and non-sparse signals. However, many applications only require processing of extremely sparse signals; for example, the chip realization in [137] assumes 0.1% sparsity—that is, 99.9% of the spectral bins are empty—allowing sparse Fourier transforms to be efficiently realized using integrated circuits. There has been extensive work on the design and realization of sparse FFTs in the last 10 years [138–140], with applications in DSP that assume high levels of sparsity, usually below 6% of the spectral bins [138, 141, 142]. Unfortunately, the requirements for sparsity prevent these algorithms to be used in applications where sparsity is either too low (more than 10% occupancy [143]) or cannot be assumed at all, in which case, dense FFTs must

still be used. Clearly, the realization of the FFT using factorizations of the DFT matrix leads to significant performance improvements; however, practical realities of chip area, power consumption, size, weight, and cost prevent direct implementation of dense FFTs for very large numbers of frequency bins. In cases where signals are not sparse, and realizing a traditional FFTs in custom integrated circuits is not practically feasible, a compromise must be reached where one gives up performance in terms of numerical precision to realize a lower complexity implementation on chip.

Consider the N^{2^n}-point DFT where n is a nonnegative integer and N is the input length, and, in this chapter, it is a power of two. The Cooley-Tukey algorithm allows the use of an N-point DFT as a radix building block to realize the larger N^{2^n}-point DFT, where the *exact* sparse factorization of the DFT matrix reduces the multiplier complexity from $\mathcal{O}(N^{2^{n+1}})$ down to $\mathcal{O}(N^{2^n}\log_2 N^{2^n})$ [p. 779] [3] with no computational errors introduced as a consequence of factorization [21]. We consider N-point DFT methods—exact or approximate—as fundamental building blocks for the proposed methods. In particular, we extend the work proposed in [74] to address the million-point DFT which is the discussed subject in [123, 141, 144]. The approximate DFT building block accrues a certain computational error that prevents an exact realization of the exact DFT; however, for many practical applications, small errors in the DFT computation can be tolerated without much practical significance, thereby allowing for the adoption of approximate computing methods as an alternative approach for achieving large-sized DFTs in real-time, albeit in an approximate sense.

3.1 APPROXIMATE DFT SCALING

In the context of discrete transforms, a scaling method is a procedure that maps a transform with smaller blocksize to a larger one. Several results are available for the DCT [145–149], typically mapping the N-point DCT into the $2N$-point DCT [150, 151]. Scaling methods are important because they facilitate the design of fast algorithms of large blocksizes. Indeed, if a fast algorithm (an efficient factorization) for the smaller blocksize transform is available, then a fast algorithm for the large ("scaled up") blocksize transform

becomes immediately available. Such an interpretation of the Cooley-Tukey algorithm in a "reverse" way is one of the main points of this proposal. In other words, instead of breaking a blocksize transform down into smaller ones, we use the Cooley-Tukey algorithm as a means to generate/design transformations of large blocksize.

For the DFT-case, the Cooley-Tukey algorithm can scale up an N-point transform square matrix into an N^2-point square matrix as described in Algorithm 1.

Algorithm 1: Pseudo-algorithm for the CooleyTukeyScaling($\mathbf{C}_N$)

Input: $\mathbf{C}_N$; ▷ Enter an $N \times N$ matrix.
Output: $\mathbf{D}_{N^2}$; ▷ The algorithm returns an $N^2 \times N^2$ matrix.

$\mathbf{D}_{N^2} \leftarrow$ empty; ▷ Create an empty output matrix structure.
for $k \leftarrow 1$ *to* N^2 **do**
  $\mathbf{x} \leftarrow \mathbf{e}_k$; ▷ Generate the standard basis over $\mathbb{R}^{N^2}$.
  $\mathbf{X} \leftarrow \mathrm{vec}\left(\left[\mathbf{\Omega}_N \circ \left(\mathbf{C}_N \cdot (\mathrm{invvec}(\mathbf{x}))^\top\right)\right] \cdot \mathbf{C}_N^\top\right)$; ▷ Transform of each basis element.
  $\mathbf{D}_{N^2} \leftarrow \left[\mathbf{D}_{N^2} \mid \mathbf{X}\right]$; ▷ Concatenate the result to obtain the N^2 output matrix.
end
return $\mathbf{D}_{N^2}$;

As a matrix mapping, Algorithm 1 could be described according to the following matrix function:

$$f_{\mathrm{CTS}} : \mathscr{M}_{\mathbb{C}}(N) \to \mathscr{M}_{\mathbb{C}}(N^2)$$
$$\mathbf{C}_N \mapsto \mathbf{D}_{N^2} = \mathrm{CooleyTukeyScaling}(\mathbf{C}_N),$$

where $f_{\mathrm{CTS}}(\cdot)$ is the Cooley-Tukey scaling and $\mathscr{M}_{\mathbb{C}}(N)$ denotes the set of square matrices of order N with complex entries. If $\mathbf{C}_N$ is the N-point DFT matrix ($\mathbf{C}_N = \mathbf{F}_N$), then we obtain that the output $\mathbf{D}_{N^2}$ is the N^2-point DFT matrix ($\mathbf{D}_{N^2} = \mathbf{F}_{N^2}$). In symbols, $f_{\mathrm{CTS}}(\mathbf{F}_N) = \mathbf{F}_{N^2}$.

The scaling is induced by the Cooley-Tukey algorithm and the factorization of the resulting N^2-point DFT approximate matrix is obtained effortlessly as a useful side-effect of the method. Therefore, by setting $\mathbf{C}_N = \mathbf{F}_N$, we obtain a fast algorithm for the

large blocksize transformation $\mathbf{F}_{N^2}$. This is a desirable result in the context of discrete transforms. The point becomes even more relevant in the context of approximate discrete transforms. Indeed, consider the approximate DFT case. The current literature offers no direct procedure to obtain large DFT approximations capable of good performance, let alone a fast algorithm for them. Therefore, by selecting as the input matrix for the above procedure—not the exact N-point DFT—but an N-point DFT approximation we can directly obtain an N^2-point DFT approximate matrix in a very systematic manner. Then, in symbols, we have:

$$f_{\mathrm{CTS}}(\hat{\mathbf{F}}_N) = \hat{\mathbf{F}}_{N^2}.$$

Usually, obtaining good approximate transforms is not an easy task that involves solving a constrained multi-criteria, multi-variable optimization problem of quadratic complexity [92, 95, 152]. As a matter of fact, our method can be iterated to obtain larger transforms. For example, consider two iterations of the proposed algorithm:

$$f_{\mathrm{CTS}}\left(f_{\mathrm{CTS}}(\hat{\mathbf{F}}_N)\right) = f_{\mathrm{CTS}}\left(\hat{\mathbf{F}}_{N^2}\right) = \hat{\mathbf{F}}_{N^4}.$$

The proposed method can be further iterated to obtain large blocksize approximate transform fully equipped with a fast algorithm.

In the next sections, we present the definition of the approximate DFT scaling method and its multiplicative complexity.

3.1.1 Definition

The approximate DFT scaling consists of using the Cooley-Tukey algorithm and replacing the N_1- and N_2-point DFT matrix $\mathbf{F}_N$ by any N_1- and N_2-point approximate DFT matrix $\hat{\mathbf{F}}_N$, respectively, such as the DFT approximations described in [69]. Mathematically, it is given by

$$\hat{\mathbf{X}} = \mathrm{vec}\left(\left\{\hat{\mathbf{F}}_{N_1} \cdot \left[\mathbf{\Omega}_{N_2\times N_1} \circ \left(\hat{\mathbf{F}}_{N_2} \cdot (\mathrm{invvec}(\mathbf{x}))^\top\right)\right]^\top\right\}^\top\right). \tag{3.1}$$

In this first moment, we consider the radix-N algorithm with ground transform approximations of blocklength N and the twiddle factor matrix computed in the exact

form to control the error propagation in the scaling method. In the next chapters, full approximations are presented in which the ground transforms and twiddle factor matrices (when required) are approximated.

The non-linear approximate DFT scaling to obtain a N^2-point approximate DFT from ground approximate transform of blocklength N is given by

$$\hat{\mathbf{X}}_1 = \mathrm{vec}\left(\left[\boldsymbol{\Omega}_N \circ \left(\hat{\mathbf{F}}_N \cdot (\mathrm{invvec}(\mathbf{x}))^\top\right)\right] \cdot \hat{\mathbf{F}}_N^\top\right). \tag{3.2}$$

In addition, a suite of hybrid approximations is also considered where the column- or row-wise DFT is maintained according to the DFT definition. The hybrid transformations are given by

$$\hat{\mathbf{X}}_2 = \mathrm{vec}\left(\left[\boldsymbol{\Omega}_N \circ \left(\hat{\mathbf{F}}_N \cdot (\mathrm{invvec}(\mathbf{x}))^\top\right)\right] \cdot \mathbf{F}_N\right), \tag{3.3}$$

$$\hat{\mathbf{X}}_3 = \mathrm{vec}\left(\left[\boldsymbol{\Omega}_N \circ \left(\mathbf{F}_N \cdot (\mathrm{invvec}(\mathbf{x}))^\top\right)\right] \cdot \hat{\mathbf{F}}_N^\top\right). \tag{3.4}$$

Vectors $\hat{\mathbf{X}}_1$, $\hat{\mathbf{X}}_2$, and $\hat{\mathbf{X}}_3$ are different approximations for $\mathbf{X}$. Eqs. (3.2) to (3.4) imply therefore different approximate DFT matrices. The explicit matrix form of such N-point approximate transformations can be obtained by submitting the vectors of the standard (canonical) basis to (3.2), (3.3), or (3.4). Once the $N \times N$ transformation matrices required to the N-point transforms are obtained the process can be iterated to derive the $N^2 \times N^2$ approximate transformation matrices. Thus after $n = 1, 2, \ldots$ iterations, an approximate transformation matrix of size $N^{2^n-1} \times N^{2^n-1}$ is obtained. This approach can be understood as a non-linear scaling method where an N-point transformation is mapped onto an N^2-point transformation. The transformation matrices used before the iterations (the fundamental blocks) are defined as ground transformations.

3.1.2 Multiplicative Complexity

In order to minimize the resulting multiplicative complexity and error propagation, $\hat{\mathbf{F}}_N$ is preferably chosen to be a multiplierless DFT approximation close to the $\mathbf{F}_N$ whenever multiplication-free computation is a goal. Unlike the DCT scaling methods [150, 153], in which an N-point transformation is mapped onto a $2 \cdot N$-point transformation, the

above scaling approach does not necessarily result in multiplierless scaled approximate transforms. Although $\hat{\mathbf{F}}_N$ can be a multiplierless matrix, the matrix $\mathbf{\Omega}_N$ contains $(N-1)^2 = N^2 - 2\cdot N + 1$ non-trivial complex multiplicands. Since roughly N^2 multiplications are required (which is comparable to the size of the resulting scaled transform N^2), the method has linear multiplicative complexity. Therefore, (3.2) to (3.4) generate scaled approximations with, albeit small, non-null multiplicative complexity.

Considering the above discussion, we can establish a recursive equation for the real multiplicative complexity. Let a be the number of real multiplications required to perform a complex multiplication and μ be the real multiplicative complexity of the exact DFT $\mathbf{F}_N$ in the hybrid algorithms. Then, the real multiplicative complexity of the approximate DFT scaling in (3.2) to (3.4) can be obtained by the following equation:

$$M_{R_1}\left(N^{2^{n+1}}\right) = N^{2^n}\cdot M_{R_1}\left(N^{2^n}\right) + a\cdot\left(N^{2^{n+1}} - 2\cdot N^{2^n} + 1\right) + N^{2^n}\cdot M_{R_2}\left(N^{2^n}\right), \tag{3.5}$$

where $n = 0, 1, 2, \ldots$. While the first and the last terms in the right-hand side of (3.5) refers respectively to the inner and outer DFT multiplicative complexity and the remaining term refers to the twiddle factor complexity [21].

In this first moment, it is considered that at least one of the DFT approximations is multiplierless, than only one term is required, $M_{R_1}\left(N^{2^n}\right)$ or $M_{R_2}\left(N^{2^n}\right)$. If we used (3.2) and multiplierless approximations of the DFT, both terms are removed. Thus, without loss of generality, for $n = 0$, (3.5) can be expressed as

$$M_{R_1}\left(N^2\right) = N\cdot M_{R_1}(N) + a\cdot\left(N^2 - 2\cdot N + 1\right), \tag{3.6}$$

where $M_{R_1}(N) = \mu$. If $\hat{\mathbf{F}}_N$ is fully replaced by a multiplierless approximation then $\mu = 0$.

For $n = 1$, the number of real multiplications is given by

$$M_{R_1}\left(N^4\right) = N^2\cdot M_{R_1}\left(N^2\right) + a\cdot\left(N^4 - 2\cdot N^2 + 1\right) + N^2\cdot M_{R_1}\left(N^2\right). \tag{3.7}$$

For $n = 2$,

$$M_{R_1}\left(N^8\right) = N^4\cdot M_{R_1}\left(N^4\right) + a\cdot\left(N^8 - 2\cdot N^4 + 1\right) + N^4\cdot M_{R_1}\left(N^4\right). \tag{3.8}$$

For $n = 3$,

$$M_{R_1}\left(N^{16}\right) = N^8 \cdot M_{R_1}\left(N^8\right) + a \cdot (N^{16} - 2 \cdot N^8 + 1) + N^8 \cdot M_{R_1}\left(N^8\right), \tag{3.9}$$

and so on.

With a few algebraic manipulations, (3.6), (3.7), (3.8), and (3.9) can be rewritten respectively as

$$M_{R_1}\left(N^2\right) = a \cdot (N^2) + (\mu - 2 \cdot a) \cdot N + a, \tag{3.10}$$

$$M_{R_1}\left(N^4\right) = 3 \cdot a \cdot N^4 \cdot (N^2) + (\mu - 2 \cdot a) \cdot 2 \cdot N^3 + a, \tag{3.11}$$

$$M_{R_1}\left(N^8\right) = 7 \cdot a \cdot N^8 \cdot (N^2) + (\mu - 2 \cdot a) \cdot 4 \cdot N^7 + a, \tag{3.12}$$

$$M_{R_1}\left(N^{16}\right) = 15 \cdot a \cdot N^{16} \cdot (N^2) + (\mu - 2 \cdot a) \cdot 8 \cdot N^{15} + a. \tag{3.13}$$

Therefore, with initial condition $M_{R_1}(N) = \mu$ the recursion is satisfied by:

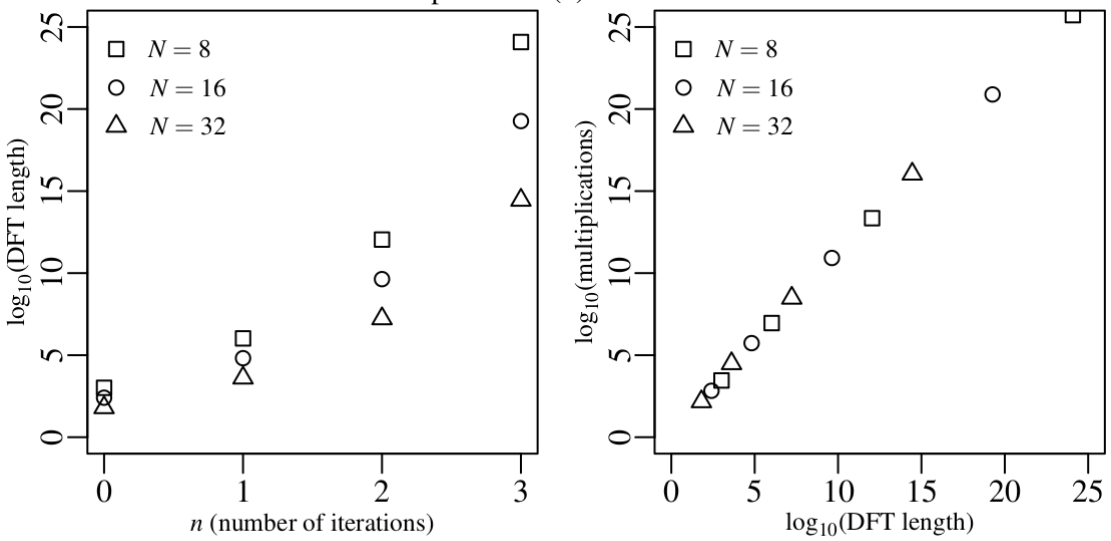
$$M_{R_1}\left(N^{2^n}\right) = (2^n - 1) \cdot N^{2^n} \cdot a + (\mu - 2 \cdot a) \cdot 2^{n-1} \cdot N^{2^n - 1} + a, \tag{3.14}$$

$$= (2^n - 1) \cdot N^{2^n} \cdot a + 2^n \cdot \frac{\mu - 2 \cdot a}{2} \cdot N^{2^n - 1} + a, \tag{3.15}$$

where $n = 1, 2, 3, \ldots$. Figure 1 shows for some selected blocklengths the behavior of the discussed scaling in terms of the number of iterations, DFT length, and the number of real multiplications, where N represents the blocklength of the ground transform. We considered (3.2) using multiplierless ground transformations and $a = 3$.

Let $L_n = N^{2^n}$ be the blocklength of the scaled transform after n iterations of the scaling procedure, starting with a ground transformation matrix of size N. Then (3.14) can

Figure 1 – DFT length in terms of the number of iterations (a) and the number of real multiplications (b) for selected values of N



a b

Source: Author (2023).

be written as

$$M_{R_1}(L_n) = (\log_N L_n - 1)\cdot L_n \cdot a - (2\cdot a - \mu)\cdot \frac{\log_N L_n}{2}\cdot \frac{L_n}{N} + a \tag{3.16}$$

$$= \left(\frac{\log_2 L_n}{\log_2 N} - 1\right)\cdot L_n \cdot a + (\mu + 2\cdot a)\cdot \frac{\log_2 L_n}{\log_2 N}\cdot \frac{L_n}{2N} + a \tag{3.17}$$

$$= \frac{\log_2 L_n}{\log_2 N}\cdot L_n \cdot a - (2\cdot a - \mu)\cdot \frac{\log_2 L_n}{\log_2 N}\cdot \frac{L_n}{2N} + a - a\cdot L_n \tag{3.18}$$

$$= L_n \cdot \left[\frac{\log_2 L_n}{\log_2 N}\left(a - \frac{2\cdot a - \mu}{2\cdot N}\right) - a\right] + a \tag{3.19}$$

$$= L_n \cdot \left[\frac{\log_2 L_n}{\frac{1}{2^n}\log_2 L_n}\left(a - \frac{2\cdot a - \mu}{2\cdot L_n^{2^{-n}}}\right) - a\right] + a \tag{3.20}$$

$$= L_n \left[\log_2 L_n^{\beta} - a\right] + a \tag{3.21}$$

$$= \beta \cdot L_n \cdot \log_2 L_n - a\cdot L_n + a, \tag{3.22}$$

where $\beta = \frac{2^n}{\log_2 L_n}\left(a - \frac{2\cdot a - \mu}{2\cdot L_n^{2^{-n}}}\right)$. If $L_n = N^{2^n}$ then $N = L_n^{2^{(-n)}}$ and it can generate complex and/or negative roots, but as it is a counting measure, only integer values are considered.

Notice that since $\beta < 1$, the growth of $L_n \cdot \log_2 L_n$ is attenuated and if N is a power of two, $N = 2^m$, $m \geq 3$, then $\beta = \frac{1}{m}\left(a - \frac{2\cdot a - \mu}{2^{m+1}}\right)$. Therefore, β does not depend on the number of iterations n. The multiplierless 32-point DFT approximation ($m = 5$ and $a = 3$)

used in [74] results in $\beta = 0.58125$. Based on the same setting for $N = 8, 16, 64, 128, 256$, we have $\beta \approx 0.875, 0.703, 0.492, 0.425, 0.374$, respectively. The theoretical minimal of multiplicative complexity for specific values are listed in [19, p. 137]. Considering $N = 32$ as the size of the ground transformation, such as the approximate DFT in [69], the repeated application of the above procedure generates approximations of sizes: 1024 [74], 1048576 ($\approx 1.05 \times 10^6$), and 1099511627776 ($\approx 1.1 \times 10^{12}$), for $n = 1, 2, 3$, respectively. In this work, we examine the case $N = 32$ and $n = 2$. In other words, we aim at deriving and assessing a DFT approximation of size $1048576 = 32^4$.

3.2 32^4-POINT DFT APPROXIMATION

The work in [74] proposed the use of (3.2), (3.3), or (3.4) to obtain approximations to the 1024-point DFT. To propose transforms with different trade-offs in computational complexity and computational accuracy, the authors in [74] employed the 32-point approximation to defined three different approximations, either approximating (i) the inner 32-point DFT block (over the rows), (ii) the outer 32-point DFT block (over the columns), or (iii) approximating both the inner and outer 32-point DFT blocks.

Based on such 32-point DFT approximation, two iterations of the proposed scaling method leads to a million-point approximation: the 32^4-point DFT approximation. It results in three different 32^4-point DFT approximations, referred to as $\hat{\mathbf{F}}^{\mathrm{I}}_{32^4}$, $\hat{\mathbf{F}}^{\mathrm{II}}_{32^4}$, and $\hat{\mathbf{F}}^{\mathrm{III}}_{32^4}$ obtained respectively by (3.2), (3.3), and (3.4).

3.2.1 Arithmetic Complexity

The arithmetic complexity expressed by the count of required arithmetic operations (multiplications and additions) is the main and standard figure of merit for assessing the computational costs of discrete transforms and associate fast algorithms [3,21,154,155]. For the assessment of the arithmetic complexity, we assumed the following conditions: (i) complex-valued input data, (ii) non-trivial multiplicands only, and (iii) that one complex multiplication requires three real multiplications ($a = 3$) and three real additions, as detailed in Section 2.4.

The ground transformation to $N = 32$ detailed in [74] can be easily obtained by $\hat{\mathbf{F}}_{32} = \text{round}(\mathbf{F}_{32})$, where round$(\cdot)$ is the rounding function as implemented in Matlab [156]. To present the sparse matrix factorization of $\hat{\mathbf{F}}_{32}$, consider the two following butterfly-structure. The first structure is given by

$$\mathbf{B}_m = \begin{bmatrix} \mathbf{I}_{(m-1)/2} & & \bar{\mathbf{I}}_{(m-1)/2} \\ & 1 & \\ -\bar{\mathbf{I}}_{(m-1)/2} & & \mathbf{I}_{(m-1)/2} \end{bmatrix},$$

where m is an odd integer greater than 1, $\mathbf{I}_m$ is the identity matrix of order m, and $\bar{\mathbf{I}}_{m/2}$ is the backward identity matrix [157, p. 33] of order $m/2$. The second structure for when m is an even integer is given by

$$\mathbf{B}_m = \begin{bmatrix} \mathbf{I}_{m/2} & \bar{\mathbf{I}}_{m/2} \\ -\bar{\mathbf{I}}_{m/2} & \mathbf{I}_{m/2} \end{bmatrix}.$$

The approximation $\hat{\mathbf{F}}_{32}$ needs only 348 real additions and no multiplications by means of the following factorization into sparse matrices:

$$\hat{\mathbf{F}}_{32} = \mathbf{G}_7 \cdot \mathbf{G}_6 \cdot \mathbf{G}_5 \cdot \mathbf{G}_4 \cdot \mathbf{G}_3 \cdot \mathbf{G}_2 \cdot \mathbf{G}_1 \cdot \mathbf{G}_0,$$

where

$$\mathbf{G}_0 = \begin{bmatrix} \mathbf{B}_{17} & & \\ & \mathbf{B}_{15} & \end{bmatrix}, \quad \mathbf{G}_1 = \begin{bmatrix} \mathbf{I}_{16} & \begin{bmatrix} 0 & \mathbf{I}_{15} \end{bmatrix} \\ \begin{bmatrix} 0 & \mathbf{I}_{15} \end{bmatrix} & \begin{bmatrix} 1 & \\ & -\mathbf{I}_{15} \end{bmatrix} \end{bmatrix},$$

$$\mathbf{G}_2 = \begin{bmatrix} \mathbf{B}_9 & & \\ & \mathbf{B}_7 & \\ & & \mathbf{I}_{16} \end{bmatrix}, \quad \mathbf{G}_3 = \begin{bmatrix} \mathbf{B}_5 & & & & \\ & 1 & & & \\ & & \mathbf{B}_3 & & \\ & & & 1 & \\ & & & & \mathbf{B}_3 \\ & & & & & \mathbf{E}_1 \end{bmatrix},$$

$$\mathbf{G}_4 = \begin{bmatrix} \mathbf{B}_3 & & & & & \\ & \mathbf{B}_2 & & & & \\ & & \mathbf{B}_4 & & & \\ & & & \mathbf{B}_4 & & \\ & & & & \mathbf{B}_2 & \\ & & & & & \mathbf{E}_2 \end{bmatrix}, \quad \mathbf{G}_5 = \begin{bmatrix} \mathbf{B}_2 & & \\ & \mathbf{I}_{15} & \\ & & \mathbf{E}_3 \end{bmatrix},$$

$$
\mathbf{G}_6 = \begin{bmatrix} \mathbf{I}_{16} & \\ & \begin{bmatrix}
1 &&&&&&&&&&&&& 1 && \\
& 1 &&&&&&& 1 &&&&&&& \\
&& -1 &&&&& 1 &&&&&&&& \\
&&& 1 &&&&&&&&&&&& \\
&&&& 1 &&&&&&&&&&& \\
&&&&& 1 & 1 &&&&&&&&& \\
&&&&& 1 & -1 &&&&&&&&& \\
&& 1 &&&&& 1 &&&&&&&& \\
& 1 &&&&&&& -1 &&&&&&& \\
&&&&&&&&& 1 & 1 &&&&& \\
&&&&&&&&& 1 & -1 &&&&& \\
&&&&&&&&&&& 1 &&&& \\
&&&&&&&&&&&& 1 &&& 1 \\
1 &&&&&&&&&&&&& -1 && \\
&&&&&&&&&&&&&& 1 & \\
&&&&&&&&&&&& 1 &&& -1
\end{bmatrix} \end{bmatrix},
$$

$$
\mathbf{G}_7 = \left[\begin{array}{cccccccccccccccc:cccccccccccccccc}
1 \\
&&&&&&&&&&&&&&&&&&& -j &&&&&&&& 1 \\
&&&&&& 1 &&&& -j \\
&&&&&&&&&&&&&&&&&&&&&&& -j &&&&& -1 \\
&&& 1 &&&&&&&&&& j \\
&&&&&&&&&&&&&&&&& -j &&&&&&&& 1 \\
&&&&& -1 &&&& -j \\
&&&&&&&&&&&&&&&& -1 &&&&&& -j \\
&& 1 &&&&&&&&&&&&& -j \\
&&&&&&&&&&&&&&&&&&&&& -j &&&&&&&& -1 \\
&&&&&&&& 1 &&&& -j \\
&&&&&&&&&&&&&&&&&&&&&&&& -j && -1 \\
&&&& -1 &&&&&&&&&& j \\
&&&&&&&&&&&&&&&&&& -j &&&&&&&&&&&&& -1 \\
&&&&&&& 1 &&&& j \\
&&&&&&&&&&&&&&&&&&&& -j &&&&&&&&&& -1 \\
\hdashline
& 1 \\
&&&&&&&&&&&&&&&&&&&& j &&&&&&&&&& -1 \\
&&&&&&& 1 &&&& -j \\
&&&&&&&&&&&&&&&&&& j &&&&&&&&&&&&& -1 \\
&&&& -1 &&&&&&&&&& -j \\
&&&&&&&&&&&&&&&&&&&&&&&& j && -1 \\
&&&&&&&& 1 &&&& j \\
&&&&&&&&&&&&&&&&&&&&& j &&&&&&&& -1 \\
&& 1 &&&&&&&&&&&&& j \\
&&&&&&&&&&&&&&&& -1 &&&&&& j \\
&&&&& -1 &&&& j \\
&&&&&&&&&&&&&&&&& j &&&&&&&& 1 \\
&&& 1 &&&&&&&&&& -j \\
&&&&&&&&&&&&&&&&&&&&&&& j &&&&& -1 \\
&&&&&& 1 &&&& j \\
&&&&&&&&&&&&&&&&&&& j &&&&&&&& 1
\end{array}\right],
$$

and $\mathbf{E}_1$, $\mathbf{E}_2$, and $\mathbf{E}_3$ are given by

$$\mathbf{E}_1 = \begin{bmatrix}
1 & & & & & & & & & & & & & & & 1 \\
& 1 & & & & & & & & & & & & & & \\
& & 1 & & & & & & & & & & & & & \\
& & & 1 & & & & & & & & & & & & \\
& & & & 1 & & & & 1 & & & & & & & \\
& & & & & 1 & & & & & & & & & & \\
& & & & & & 1 & & & & & & & & & \\
& & & & & & & 1 & & & & & & & & \\
& & & & & 1 & & & -1 & & & & & & & \\
& & & & & & & & & 1 & & & & & & \\
& & & & & & & & & & 1 & & & & & \\
& & & & & & & & & & & 1 & & & & \\
1 & & & & & & & & & & & & -1 & & & \\
& & & & & & & & & & & & & 1 & & \\
& & & & & & & & & & & & & & 1 & \\
& & & & & & & & & & & & & & & 1
\end{bmatrix},$$

$$\mathbf{E}_2 = \begin{bmatrix}
1 & & & & & & & & & & & & & & & \\
& -1 & & & & & & & & & & & & & & 1 \\
& & 1 & & & & & & & & & & & & & \\
& & & 1 & & & & & & 1 & & & & & & \\
& & & & 1 & & 1 & & 1 & & & & & & & \\
& & & & & 1 & & 1 & & & & & & & & \\
& & & & 1 & & -1 & & & & & & & & & \\
& & & & & 1 & & -1 & & & & & & & & \\
& & & 1 & & & & & -1 & & & & & & & \\
& & & & & & & & & 1 & & & & & & \\
& & & & & & & & & & 1 & & 1 & & & \\
& & & & & & & & & & & 1 & & 1 & & 1 \\
& & & & & & & & & & 1 & & -1 & & & \\
& & & & & & & & & & & 1 & & -1 & & \\
& 1 & & & & & & & & & & & & & 1 & \\
& & & & & & & & & & & 1 & & & & -1
\end{bmatrix},$$

and

$$\mathbf{E}_3 = \begin{bmatrix}
1 & & & & 1 & -1 & & & & & & & & & & \\
& 1 & & & & & & & & & & & & & & \\
& & 1 & 1 & & & & & & & & & & & & \\
& & 1 & -1 & & & & & & & & & & & & \\
1 & & & & -1 & & & & & & & & & & & \\
& & & & & 1 & & & & & & & & & & \\
1 & & & & & & 1 & & & & & & & & & \\
& & & & & & & 1 & & & & & & & & \\
& & & & & & & & 1 & & & 1 & & -1 & & \\
& & & & & & & & & 1 & & & & & & \\
& & & & & & & & & & 1 & & & 1 & & \\
& & & & & & & & & & & 1 & & & & \\
& & & & & & & & 1 & & & & -1 & & & \\
& & & & & & & & & 1 & & & & -1 & & \\
& & & & & & & & 1 & & & & & & 1 &
\end{bmatrix}.$$

The algorithms to compute the 32^4-point DFT according to (3.2) to (3.4) requires 131072 calls of the twiddle factor matrix of size $N = 32$ and one call of the twiddle factor matrix of size $N = 1024$. Thus, the twiddle factors imply $(2883 \times 131072) + 3139587 = 9043971$ real multiplications and $(2883 \times 131072) + 3139587 = 9043971$ real additions for each approximation.

Denoted by $\mathbf{F}_{32}$, the exact 32-point DFT is computed by the radix-2 Cooley-Tukey algorithm, requiring 88 multiplications and 408 additions [21]. To compute the exact 32^4-point DFT it is sufficient to use the exact methods—$\mathbf{F}_{32}$ instead of $\hat{\mathbf{F}}_{32}$. Thus, we need

131072 calls of $\mathbf{F}_{32}$ and twiddle factor multiplications. Therefore, the resulting arithmetic costs of the exact 32^4-point DFT using the radix-2 Cooley-Tukey FFT are: $(88 \times 131072) + 9043971 = 20578307$ real multiplications and $(408 \times 131072) + 9043971 = 62521347$ real additions.

The approximations $\hat{\mathbf{F}}^{\mathrm{II}}_{32^4}$ and $\hat{\mathbf{F}}^{\mathrm{III}}_{32^4}$ present the same complexity with differences only in the arrangement of the ground approximations. These two approximations require 65536 calls of $\hat{\mathbf{F}}_{32}$, 65536 calls of $\mathbf{F}_{32}$, and multiplications by twiddle factors. Thus, we have a total of $(88 \times 65536) + 9043971 = 14811139$ real multiplications and $(348 \times 65536) + (408 \times 65536) + 9043971 = 58589187$ real additions.

For $\hat{\mathbf{F}}^{\mathrm{I}}_{32^4}$, all ground transformations were approximations. Thus, the only source of multiplications are the twiddle factors which represent 9043971 multiplications. Due to the 131072 calls of $\hat{\mathbf{F}}_{32}$, $\hat{\mathbf{F}}^{\mathrm{I}}_{32^4}$ requires $(348 \times 131072) + 9043971 = 54657027$ real additions. A summary of the arithmetic complexity is shown in Table 1. While the approximation $\hat{\mathbf{F}}^{\mathrm{I}}_{32^4}$ reduces $\approx$56% of the multiplications, the approximations $\hat{\mathbf{F}}^{\mathrm{II}}_{32^4}$ and $\hat{\mathbf{F}}^{\mathrm{III}}_{32^4}$ reduce $\approx$28% when compared to the DFT computed by the Cooley-Tukey FFT algorithm.

3.2.2 Error Analysis

We evaluate the approximations according to their numerical errors relative to the exact N-point DFT, where $N = 32^4$. We adopted a Monte Carlo simulation experiment where the spectrum of the chosen test signals presents constant, unit magnitude. The experiment consists of generating a vector $\mathbf{U} = \begin{bmatrix} U_0 & U_1 & \dots & U_{N-1} \end{bmatrix}^\top$, where $U_n = \exp(j\omega_n)$, and ω_n are independent and identically distributed samples from a uniform distribution in the interval $[-\pi, \pi)$ for $n = 0, 1, \dots, N-1$. Then we compute the inverse DFT of $\mathbf{U}$ as $\mathbf{x} = \mathbf{F}^{-1}_{32^4} \cdot \mathbf{U}$. We use $\mathbf{x}$ to compute the corresponding transformed signals $\mathbf{X}^{\mathrm{I}}$, $\mathbf{X}^{\mathrm{II}}$, and $\mathbf{X}^{\mathrm{III}}$ according to the 32^4-point DFT approximations $\hat{\mathbf{F}}^{\mathrm{I}}_{32^4}$, $\hat{\mathbf{F}}^{\mathrm{II}}_{32^4}$, and $\hat{\mathbf{F}}^{\mathrm{III}}_{32^4}$. If a specific 32^4-point DFT approximation $\hat{\mathbf{F}}^{i}_{32^4}$ for $i \in \{\mathrm{I}, \mathrm{II}, \mathrm{III}\}$ is close to the exact 32^4-point DFT, then its transformed signal $\mathbf{X}^i$ is expected to be close to the original signal $\mathbf{U}$. This process is repeated $R = 1000$ times, generating, therefore, $\mathbf{U}_r$, $\mathbf{x}_r$, $\mathbf{X}^{\mathrm{I}}_r, \mathbf{X}^{\mathrm{II}}_r, \mathbf{X}^{\mathrm{III}}_r$ for $r = 0, 1, \dots, R-1$.

For comparing each of the three different 32^4-point DFT approximations, the average relative error norm is computed as follows:

$$\varepsilon = \frac{1}{R}\sum_{r=0}^{R-1}\frac{\|\mathbf{U}_r - \mathbf{X}_r^i\|^2}{\|\mathbf{U}_r\|^2}, \tag{3.23}$$

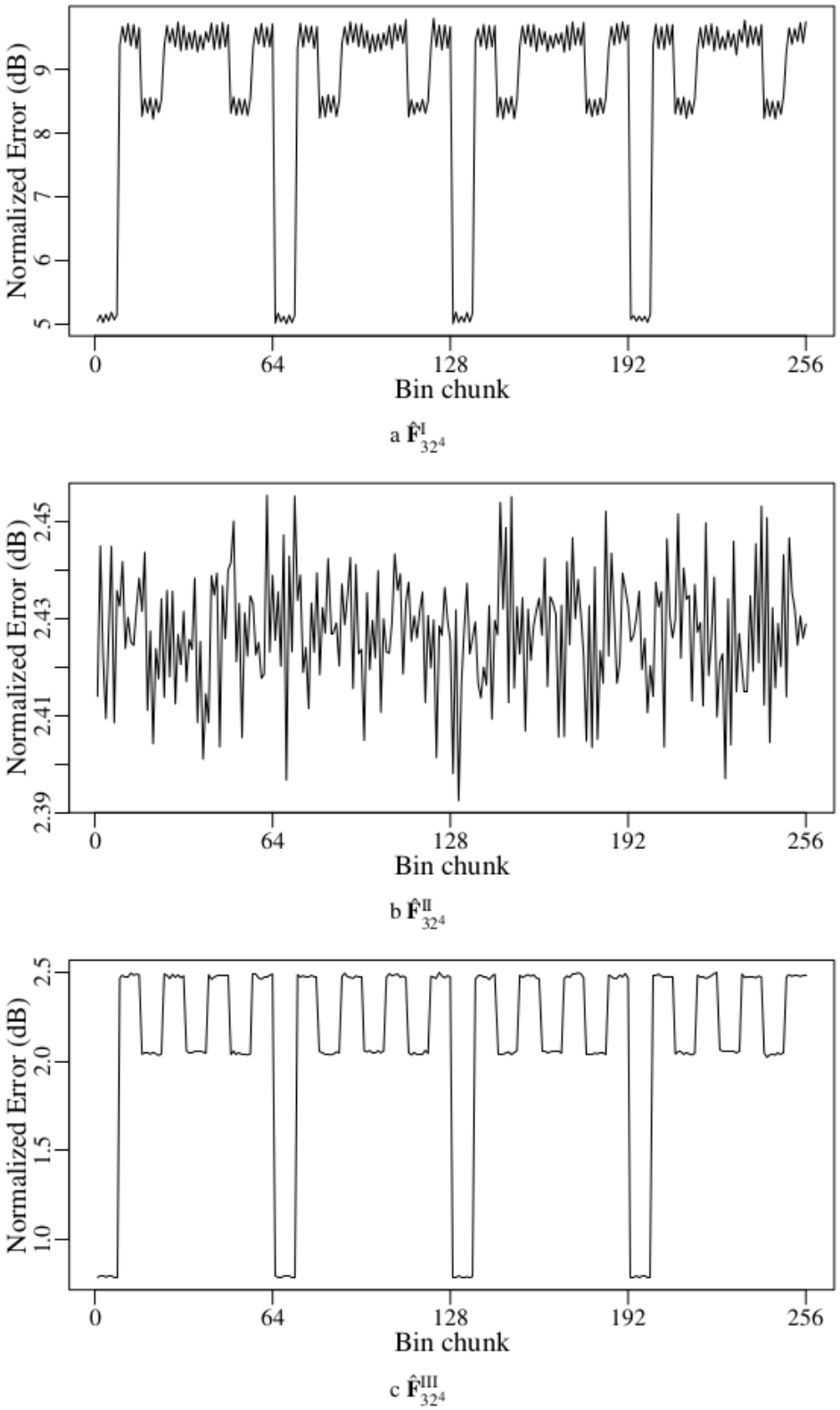
where $i \in \{\mathrm{I}, \mathrm{II}, \mathrm{III}\}$. Table 1 shows the error measure values which were computed using IEEE 754 standard double floating-point precision [158]. In addition, to understand the error behavior computed in (3.23), we divided the 1048576 bins into 256 chunks. Each chunk contains 4096 sequential bins, where the least performing bin of each chunk is collected and plotted in Figure 2. Thus we provide a worst-case scenario analysis. The evaluation shown in Figure 2 were designed to be a descriptive analysis with easy reproducibility. The test signals whose spectrum has constant and unit magnitude are useful to directly evidence any distortion in magnitude that might be relevant in the analysis of the signal.

The DFT computation of $\hat{\mathbf{F}}^{\mathrm{I}}_{32^4}$ presents an approximation error of 0.9899, but reduces the total number of multiplications in $\approx$56% and the additions in $\approx$12% when compared with the radix-2 Cooley-Tukey algorithm. According to Figure 2a, the bin error is at most $\approx$9.8 dB. On the other hand, the approximations $\hat{\mathbf{F}}^{\mathrm{II}}_{32^4}$ and $\hat{\mathbf{F}}^{\mathrm{III}}_{32^4}$ present error equal to 0.2823 and reduces $\approx$28% of multiplications and $\approx$6% of additions compared with the radix-2 Cooley-Tukey algorithm. Although, the approximations present the same error, they have different behavior. While $\hat{\mathbf{F}}^{\mathrm{II}}_{32^4}$ presents maximum bin error of $\approx$ 2.46, $\hat{\mathbf{F}}^{\mathrm{III}}_{32^4}$ has $\approx$ 2.5. Figures 2b and 2c show a greater variability of $\hat{\mathbf{F}}^{\mathrm{III}}_{32^4}$, which is due to the position of the twiddle factors matrix in the algorithm.

3.3 CHAPTER CONCLUSIONS

In this chapter, we propose a recursive scaling method generalizing the proposal in [74]. We have provided a multiplicative complexity analysis for the proposed scaling method including hybrid cases. The hybrid approximations can be useful in signals that are sparsely sampled in the frequency domain [159], ,particularly for high-dimensional signals [160] or nonequispaced discrete Fourier transformation [161, 162] to reduce com-

Figure 2 – Least performing bin of each chunk of 4096 bins for the proposed approximations



Source: Author (2023).

Table 1 – Arithmetic complexity and error evaluation

	Arithmetic operation		Reduction (%)		Error
Transform	Mult.	Add.	Mult.	Add.	ε
Exact	20578307	62521347	-	-	0
$\hat{\mathbf{F}}^{\mathrm{I}}_{32^4}$	9043971	54657027	56.05	12.58	0.9899
$\hat{\mathbf{F}}^{\mathrm{II}}_{32^4}$	14811139	58589187	28.03	6.29	0.2823
$\hat{\mathbf{F}}^{\mathrm{III}}_{32^4}$	14811139	58589187	28.03	6.29	0.2823

Source: Author (2023).

putational cost while maintaining reasonable accuracy. We fully worked out the 32^4-point approximation detailing the trade-off between performance in terms of numerical errors and arithmetic complexity. The proposed method was able to reduce $\approx$56% of the multiplications required to compute the DFT by the radix-2 Cooley-Tukey algorithm.

4 APPROXIMATE PRIME FACTOR ALGORITHM AND A MULTIPLIERLESS 1023-POINT DFT APPROXIMATION

The 32-point DFT approximation proposed in [69] was employed as the fundamental block of the 1024-point DFT approximation introduced in [74]. The methodology described in [74] revisits the Cooley-Tukey algorithm and effectively extends a given 32-point DFT approximation resulting in a 32^2-point DFT approximation. This extension is possible because the Cooley-Tukey algorithm can be structured as a two-dimensional mapping, allowing the computation of the 1024-point DFT to be performed by 2×32 instantiations of the 32-point DFT [81]. However, from the previous chapter, we can see that even considering multiplierless 32-point approximations, the resulting 1024-point approximations proposed in [74] still require multiplications. Indeed, the Cooley-Tukey-based approximations inherit the twiddle factors present in the exact formulation of the traditional Cooley-Tukey algorithm [8]. The Cooley-Tukey-based 1024-point DFT approximations in [74] require 5699 real multiplications for the hybrid algorithms and 2883 for the full-multiplierless ground transformation algorithm. Although the proposed approximations reduce respectively 44% and 72% of the real multiplications (when compared with the Cooley-Tukey Radix-2 FFT [21, p. 76]), such remaining operations may still be a hindrance in extremely resource-constrained scenarios.

In this chapter, a framework for deriving large DFT approximations that are fully multiplierless is proposed. Specifically, we focus on the design of a DFT approximation capable of directly competing with the method introduced in [74]. The 1024-point DFT provides a sampling rate that is usually enough in terms of resolution for many applications. The 1023-point DFT has a sampling rate close to the 1024-point DFT, and it can be exploited by the prime-factor algorithm (PFA) [79, 80], also known as the Good-Thomas algorithm which has distinct number-theoretical properties that eliminate intermediate computations such as twiddle factors. Since the Good-Thomas algorithm requires the lengths of the fundamental blocks to be coprimes, we initially provide a methodology to obtain approximations for any blocklength.

4.1 APPROXIMATE DFT METHODOLOGY

Being alternatives to the exact transformations, approximate transforms possess a low computational cost and exhibit similar mathematical properties and performance to their exact counterparts [163]. In this context, an approximate DFT matrix $\hat{\mathbf{F}}_N^*$ can be derived by solving the following optimization problem:

$$\hat{\mathbf{F}}_N^* = \arg\min_{\hat{\mathbf{F}}_N} \text{error}(\hat{\mathbf{F}}_N, \mathbf{F}_N), \tag{4.1}$$

where $\text{error}(\cdot)$ is an adopted error measure and $\hat{\mathbf{F}}_N$ is a candidate approximation obtained from a suitable search space.

Approximate transforms can be derived from low-complexity matrices [58] according to an orthogonalization process referred to as polar decomposition [164]. Such approach consists of two matrices: a low-complexity matrix and a real-valued diagonal matrix. Thus, a candidate approximation $\hat{\mathbf{F}}_N$ for the exact transformation $\mathbf{F}_N$ can be written as

$$\hat{\mathbf{F}}_N = \mathbf{S}_N \cdot \mathbf{T}_N, \tag{4.2}$$

where $\mathbf{T}_N$ is a low-complexity matrix and $\mathbf{S}_N$ is a diagonal matrix expressed by

$$\mathbf{S}_N = \text{diag}\left(\sqrt{\left[\text{diag}\left(\mathbf{T}_N \cdot \frac{1}{N}\mathbf{T}_N^H\right)\right]^{-1}}\right), \tag{4.3}$$

$\text{diag}(\cdot)$ is a function that returns a diagonal matrix, if the argument is a vector; or a vector with the diagonal elements, if the argument is a matrix, the superscript ^{H} denotes the Hermitian operation [94, p. 82], and $\sqrt{\cdot}$ is the matrix square root operation [165]. Therefore, a suitable choice of $\mathbf{T}_N$ is central to the above approach. To solve the optimization problem in 4.1, it is necessary to define the search space.

4.1.1 Search Space

The low-complexity matrices $\mathbf{T}_N$ are taken from the search space given by the matrix space $\mathscr{M}_N(\mathscr{P})$, which is the set of all $N \times N$ matrices with entries over a set of low-

complexity multipliers $\mathscr{P}$, referred to as low-complexity subspace. Popular choices for $\mathscr{P}$ are $\{-1,0,1\}$ and $\{-1,-\frac{1}{2},0,\frac{1}{2},1\}$, which contain only trivial multipliers [21, p. 71].

The set $\mathscr{M}_N(\mathscr{P})$ can be extremely large. For instance, $\mathscr{M}_N(\mathscr{P})$ contains $3^{64} \approx 3.43 \times 10^{30}$ elements (distinct matrices) for $N = 8$ and $\mathscr{P} = \{-1,0,1\}$. Therefore, we propose, as a working search space, a subset of $\mathscr{M}_N(\mathscr{P})$ given by the expansion factor methodology [47, p. 274]. Thus, low-complexity matrices $\mathbf{T}_N$ can be generated according to the following expression:

$$\mathbf{T}_N = g(\alpha \cdot \mathbf{F}_N), \tag{4.4}$$

where $g(\cdot)$ is an entry-wise integer matrix function, such as rounding, truncation, ceil, and floor functions [58] and α is a real number referred to as the expansion factor [166]. To ensure that the integer function $g(\cdot)$ returns only values defined over $\mathscr{P}$, the values of α are judiciously restricted to an interval $\mathscr{D}$ given by

$$\alpha_{min} \leq \alpha \leq \alpha_{max}, \tag{4.5}$$

where $\alpha_{min} = \inf\{\alpha \in \mathbb{R}_+ : g(\alpha \cdot \gamma_{max}) \neq 0\}$ and $\alpha_{max} = \sup\{\alpha \in \mathbb{R}_+ : g(\alpha \cdot \gamma_{max}) = \max(\mathscr{P})\}$, being $\inf\{\cdot\}$ the infimum operator [167, p. 82], $\sup\{\cdot\}$ the supremum operator [168, p. 12] $\gamma_{max} = \max_{m,n}(\Re(|f_{m,n}|), \Im(|f_{m,n}|))$ and $f_{m,n}$, the (m,n)th entry of $\mathbf{F}_N$. The symmetries of $\mathbf{F}_N$ allow us to restrict the analysis to $\alpha \geq 0$ and since the entries of $\mathbf{F}_N$ are bounded by the unity, $\gamma_{max} = 1$.

4.1.2 Optimization Problem and Objective Function

The general optimization problem shown in (4.1) can be rewritten by employing (4.2), (4.3), and (4.4). Thus, the proposed optimization problem is given by

$$\alpha^* = \arg\min_{\alpha \in \mathscr{D}} \text{error}(\mathbf{S}_N \cdot g(\alpha \cdot \mathbf{F}_N), \mathbf{F}_N), \tag{4.6}$$

and the optimal approximation is furnished by

$$\hat{\mathbf{F}}_N^* = \mathbf{S}_N^* \cdot \mathbf{T}_N^*, \tag{4.7}$$

where $\mathbf{T}_N^* = g(\alpha^* \cdot \mathbf{F}_N)$ and $\mathbf{S}_N^*$ is computed from $\mathbf{T}_N^*$ as detailed in (4.3), *mutatis mutandis*.

Now we aim at specifying the error function in (4.6). As shown in the literature [73, 169, 170], usual choices for such function are: (i) the total error energy [66]; (ii) the mean absolute percentage error (MAPE) [171, p. 79]; and (iii) the deviation from orthogonality [65, 172]. These functions are detailed below.

(i) The total error energy is defined by

$$\varepsilon(\hat{\mathbf{F}}_N) = \pi \cdot ||\mathbf{F}_N - \hat{\mathbf{F}}_N||_{\mathrm{F}}^2,$$

where $||\cdot||_{\mathrm{F}}$ represents the Frobenius norm [173, p. 115];

(ii) The MAPE of the transformation matrix is obtained by

$$M(\hat{\mathbf{F}}_N) = 100 \cdot \frac{1}{N^2} \cdot \sum_{m=1}^{N} \sum_{n=1}^{N} \left| \frac{f_{m,n} - \hat{f}_{m,n}}{f_{m,n}} \right|,$$

where $\hat{f}_{m,n}$ is the (m,n)th entry of $\hat{\mathbf{F}}_N$;

(iii) The deviation from orthogonality [65] is defined by:

$$\phi(\hat{\mathbf{F}}_N) = 1 - \frac{||\operatorname{diag}(\hat{\mathbf{F}}_N \cdot \hat{\mathbf{F}}_N^H)||_{\mathrm{F}}}{||\hat{\mathbf{F}}_N \cdot \hat{\mathbf{F}}_N^H||_{\mathrm{F}}}.$$

Small values of $\phi(\cdot)$ indicate proximity to orthogonality. Orthogonal matrices have null deviation.

Combining the above error functions in a single optimization problem, we obtain the following multicriteria problem [92, 174]:

$$\alpha^* = \underset{\alpha \in \mathscr{D}}{\arg\min} \left\{ \varepsilon\left(\mathbf{S}_N \cdot g(\alpha \cdot \mathbf{F}_N)\right), M\left(\mathbf{S}_N \cdot g(\alpha \cdot \mathbf{F}_N)\right), \phi\left(\mathbf{S}_N \cdot g(\alpha \cdot \mathbf{F}_N)\right) \right\}. \tag{4.8}$$

4.2 APPROXIMATE PFA

In this section, we introduce the approximate prime factor algorithm (APFA) and two variations of the method: (i) the unscaled APFA and (ii) the hybrid APFA.

4.2.1 APFA Definition

Under the assumption of the PFA, we compute an N-point DFT approximation as follows

$$\hat{\mathbf{X}} = \operatorname{invmap}\left(\hat{\mathbf{F}}_{N_1}^* \cdot \left[\hat{\mathbf{F}}_{N_2}^* \cdot (\operatorname{map}(\mathbf{x}))^\top \right]^\top \right), \tag{4.9}$$

where $\hat{\mathbf{F}}^*_{N_1}$ and $\hat{\mathbf{F}}^*_{N_2}$ are approximations of $\mathbf{F}_{N_1}$ and $\mathbf{F}_{N_2}$, respectively (cf. (2.17)) being $N = N_1 \times N_2$ and $\gcd(N_1, N_2) = 1$.

If the approximations used in (4.9) admit the format expressed in (4.7), then the above equation can be rewritten as

$$\hat{\mathbf{X}} = \text{invmap}\left(\mathbf{S}^*_{N_1} \cdot \mathbf{T}^*_{N_1} \cdot \left[\mathbf{S}^*_{N_2} \cdot \mathbf{T}^*_{N_2} \cdot (\text{map}(\mathbf{x}))^\top\right]^\top\right). \tag{4.10}$$

Notice that $\mathbf{S}^*_{N_1}$ and $\mathbf{S}^*_{N_2}$ are real diagonal matrices that can be factored out from the mapping operator as follows:

$$\hat{\mathbf{X}} = \mathbf{S} \cdot \text{invmap}\left(\mathbf{T}^*_{N_1} \cdot \left[\mathbf{T}^*_{N_2} \cdot (\text{map}(\mathbf{x}))^\top\right]^\top\right), \tag{4.11}$$

where $\mathbf{S}$ is a diagonal matrix given by $\mathbf{S} = \text{diag}\left[\text{invmap}\left(\text{diag}(\mathbf{S}^*_{N_1}) \cdot \text{diag}(\mathbf{S}^*_{N_2})^\top\right)\right]$.

4.2.2 Unscaled APFA

Because the matrix $\mathbf{S}$ is a diagonal matrix, its effect to the approximate DFT computation consists of scaling each spectral component individually. Depending on the context in which the DFT is applied, the scaling can be embedded, absorbed, parallel computed, or even neglected when the unscaled spectrum is sufficient [175–177]. The unscaled N-point DFT approximation is obtained by

$$\tilde{\mathbf{X}} = \text{invmap}\left(\mathbf{T}^*_{N_1} \cdot \left[\mathbf{T}^*_{N_2} \cdot (\text{map}(\mathbf{x}))^\top\right]^\top\right). \tag{4.12}$$

Thus, we have the following relationship between (4.11) and (4.12):

$$\hat{\mathbf{X}} = \mathbf{S} \cdot \tilde{\mathbf{X}}. \tag{4.13}$$

Since the elements of the diagonal matrix $\mathbf{S}$ are real numbers, we can approximate them. Although the same approach can be applied in Cooley-Tukey algorithms, due to the twiddle factor, the matrix $\mathbf{S}$ would have complex elements which require more arithmetic operations to be implemented and also insert more error when approximated.

4.2.3 Hybrid APFA

We also provide hybrid algorithms approximating only part of the DFT computation. The diagonal matrix $\mathbf{S}$ can also be factored out in the hybrid algorithms. Thus, we obtain two possible approximations.

First, we keep the row-wise N_1-point DFT exact while the column-wise N_2-point DFT is approximated. This algorithm is given by

$$\hat{\mathbf{X}}_1 = \mathbf{S} \cdot \text{invmap}\left(\left\{ \mathbf{F}_{N_1} \cdot \left[\mathbf{T}^*_{N_2} \cdot (\text{map}(\mathbf{x})) \right]^\top \right\}^\top \right), \tag{4.14}$$

where $\mathbf{S} = \text{diag}\left[\text{invmap}\left(\mathbf{1}_{N_1} \cdot \text{diag}(\mathbf{S}^*_{N_2})^\top \right) \right]$ and $\mathbf{1}_r$ is a column vector of ones with length equals to r. Second, the column-wise N_2-point DFT is maintained exact and the row-wise N_1-point DFT is approximated. Then, the N-point DFT approximation is calculated by

$$\hat{\mathbf{X}}_2 = \mathbf{S} \cdot \text{invmap}\left(\left\{ \mathbf{T}^*_{N_1} \cdot [\mathbf{F}_{N_2} \cdot (\text{map}(\mathbf{x}))]^\top \right\}^\top \right), \tag{4.15}$$

where $\mathbf{S} = \text{diag}\left[\text{invmap}\left(\text{diag}(\mathbf{S}^*_{N_1}) \cdot (\mathbf{1}_{N_2})^\top \right) \right]$.

4.3 NUMERICAL RESULTS

In this section, we advance two results. First, we employ the prime factor algorithm detailed in Section 4.2 to obtain approximations for the 1023-point DFT. Second, we apply the methodology described in Section 4.1 to obtain approximations for the 31-, 11- and 3-point DFT which are required for 1023-point DFT approximations.

4.3.1 1023-point DFT Approximation

Invoking (4.9) for $N_1 = 31$ and $N_2 = 33$, we introduce a 1023-point DFT approximation according to the following equation:

$$\hat{\mathbf{X}} = \text{invmap}\left(\hat{\mathbf{F}}^*_{31} \cdot \left[\hat{\mathbf{F}}^*_{33} \cdot (\text{map}(\mathbf{x}))^\top \right]^\top \right). \tag{4.16}$$

The term in square brackets in (4.16) requires several calls of a 33-point DFT approximation. Because $N_2 = 33 = 11 \times 3$ is suitable for the proposed APFA formalism, a 33-point

DFT approximation can be obtained based on approximations for the 3- and 11-point DFTs, as follows:

$$\hat{\mathbf{Y}} = \text{invmap}\left(\hat{\mathbf{F}}_{11}^{*} \cdot \left[\hat{\mathbf{F}}_{3}^{*} \cdot (\text{map}(\mathbf{y}))^{\top}\right]^{\top}\right), \tag{4.17}$$

where $\mathbf{y}$ is a 33-point column vector corresponding individually to the rows of map($\mathbf{x}$), so in this case there are 31 possibly distinct vectors. As shown in (4.11), the scaling matrix $\mathbf{S}$ can be calculated separately and a 1023-point DFT approximation can be rewritten as

$$\hat{\mathbf{X}} = \mathbf{S} \cdot \text{invmap}\left(\hat{\mathbf{T}}_{31}^{*} \cdot \left[\hat{\mathbf{T}}_{33}^{*} \cdot (\text{map}(\mathbf{x}))^{\top}\right]^{\top}\right), \tag{4.18}$$

and

$$\hat{\mathbf{Y}} = \text{invmap}\left(\hat{\mathbf{T}}_{11}^{*} \cdot \left[\hat{\mathbf{T}}_{3}^{*} \cdot (\text{map}(\mathbf{y}))^{\top}\right]^{\top}\right). \tag{4.19}$$

Notice that the diagonal matrix $\mathbf{S}$ encompasses the intermediate diagonals $\mathbf{S}_3^*$, $\mathbf{S}_{11}^*$, and $\mathbf{S}_{31}^*$ and is given by

$$\mathbf{S} = \text{diag}\Big\{\text{invmap}\Big[\text{diag}(\mathbf{S}_{31}^{*}) \cdot \text{invmap}\Big(\text{diag}(\mathbf{S}_{11}^{*}) \cdot \text{diag}(\mathbf{S}_{3}^{*})^{\top}\Big)^{\top}\Big]\Big\}, \tag{4.20}$$

where $\mathbf{S}_{31}^*$, $\mathbf{S}_{11}^*$, and $\mathbf{S}_3^*$ are the diagonal matrices required by the DFT approximations $\hat{\mathbf{F}}_{31}^*$, $\hat{\mathbf{F}}_{11}^*$, and $\hat{\mathbf{F}}_3^*$, respectively, as described in (4.7).

4.3.2 Design Parameters

The algorithm detailed in the previous section requires approximations to the 31-, 11-, and 3-point DFT. To obtain such approximations, we numerically apply the methodology described in Section 4.1 for which $g(\cdot)$ and $\mathscr{P}$ must be specified. As suggested in [170], we select $\mathscr{P} = \{-1, -\frac{1}{2}, 0, \frac{1}{2}, 1\}$ as the set of low-complexity multipliers. Among the integer functions mentioned, the round function, as implemented in Matlab/Octave [156, 178], is reported to offer superior performance when compared with other integer functions [65, 66, 170]. In Figures 3, 4, and 5 are presented the error measures (y axis) of the matrices obtained by integer functions considering α values (x axis). In Appendix A, we provide the same analysis for $\mathscr{P} = \{-1, 0, 1\}$. Thus, as detailed in [179], we adopted

the following round-to-multiple function:

$$g(x) = \frac{1}{2} \cdot \text{round}(2 \cdot x) \in \mathscr{P}. \tag{4.21}$$

The related α search space is $\mathscr{D} = [0.26, 1.25]$ (cf. (4.5)). The α step used was 10^{-5} providing a total of 42, 16, and 6 different approximations for the 31-, 11-, and 3-point DFT, respectively. Smaller α steps do not alter the results.

4.3.3 31-, 11-, and 3-point DFT Approximations

According to Figures 3, 4, and 5, orthogonal matrices were not found, so priority was given to matrices that had lower values of MAPE and error energy. In this way, the optimal expansion factors α^* are in the intervals $[1.08859, 1.15141]$, $[0.99240, 1.14528]$, and $[0.86603, 1.25000]$ for the 31-, 11-, and 3-point DFT approximations, respectively. Therefore, we selected $\alpha^* = \frac{9}{8}$ for convenience. Then, the low-complexity matrices $\mathbf{T}_{31}^*$, $\mathbf{T}_{11}^*$, and $\mathbf{T}_3^*$ are given by

$$\mathbf{T}_{31}^* = \frac{1}{2} \cdot \text{round}(2 \cdot \frac{9}{8} \cdot \mathbf{F}_{31}),$$

$$\mathbf{T}_{11}^* = \frac{1}{2} \cdot \text{round}(2 \cdot \frac{9}{8} \cdot \mathbf{F}_{11}),$$

and

$$\mathbf{T}_3^* = \frac{1}{2} \cdot \text{round}(2 \cdot \frac{9}{8} \cdot \mathbf{F}_3).$$

From (4.3), we obtain

$$\mathbf{S}_{31}^* = \begin{bmatrix} 1 & \\ & \sqrt{\frac{31}{38}} \cdot \mathbf{I}_{30} \end{bmatrix},$$

$$\mathbf{S}_{11}^* = \begin{bmatrix} 1 & \\ & \sqrt{\frac{11}{13}} \cdot \mathbf{I}_{10} \end{bmatrix},$$

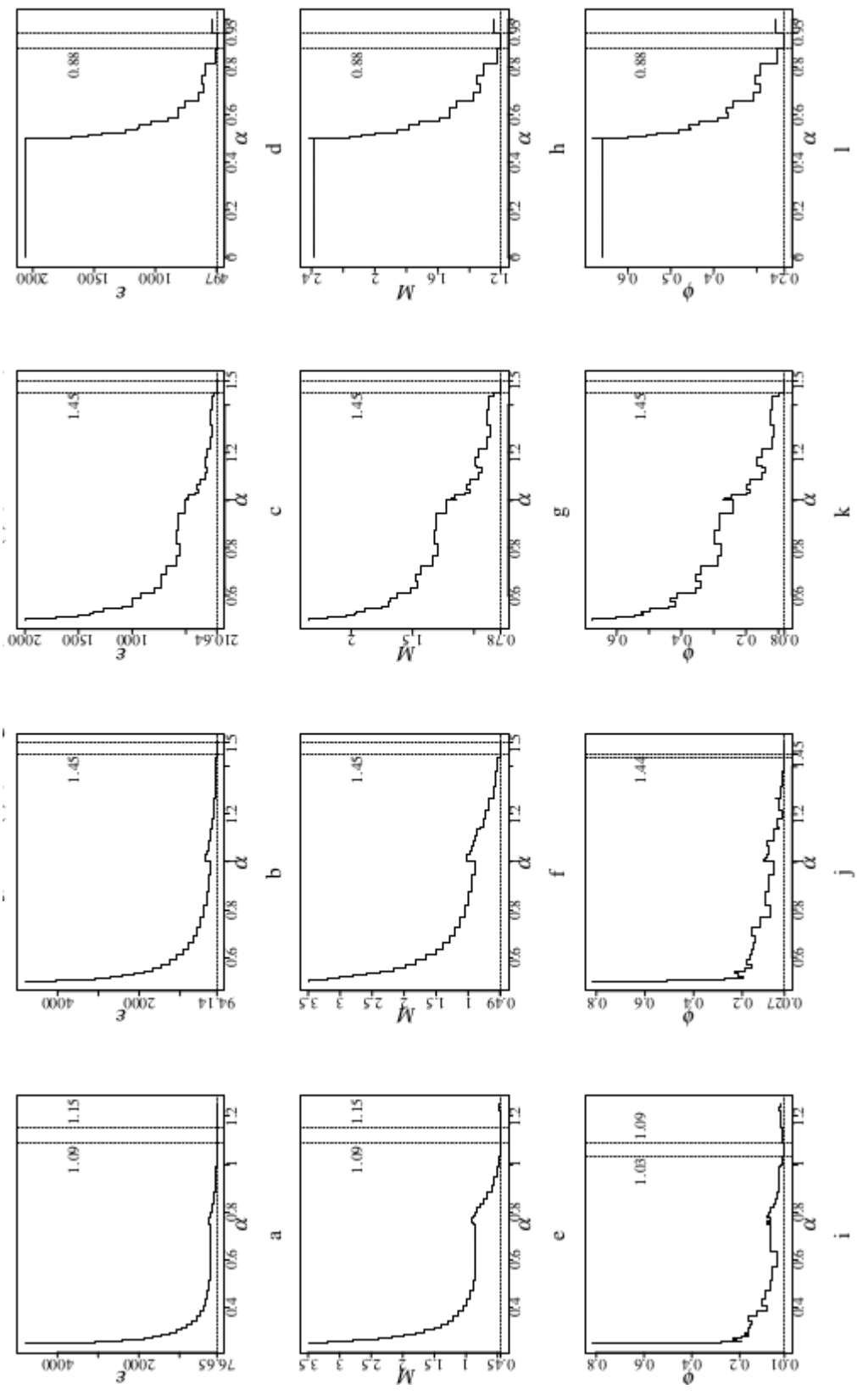


Figure 3 – Performance of the integer functions for $N = 31$ and $\mathscr{P} = \{-1, -\frac{1}{2}, 0, \frac{1}{2}, 1\}$ considering round$(\cdot)$ (3a, 3e, and 3i), trunc$(\cdot)$ (3b, 3f, and 3j), floor$(\cdot)$ (3c, 3g, and 3k), and ceil$(\cdot)$ (3d, 3h, and 3l)

Source: Author (2023).

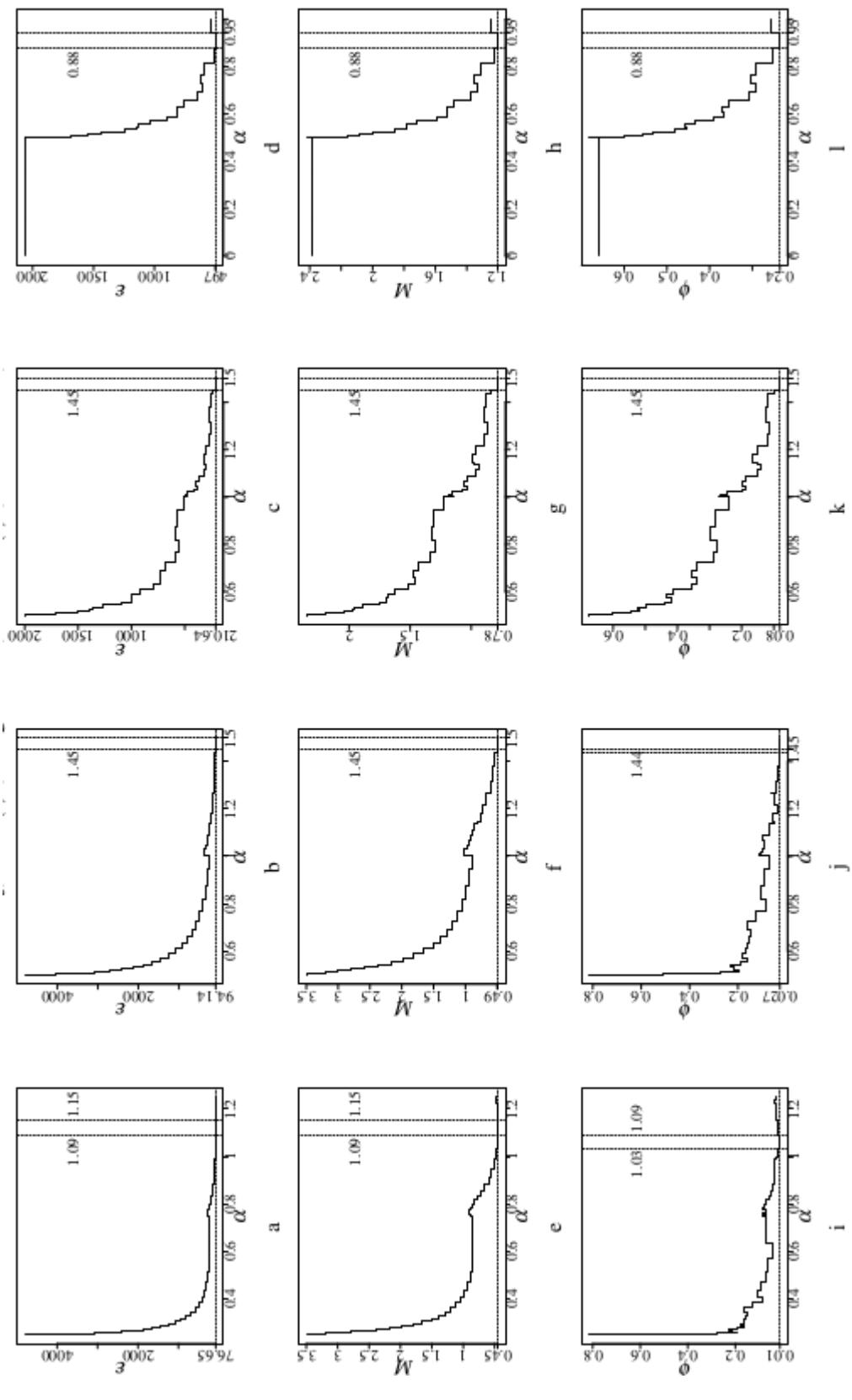
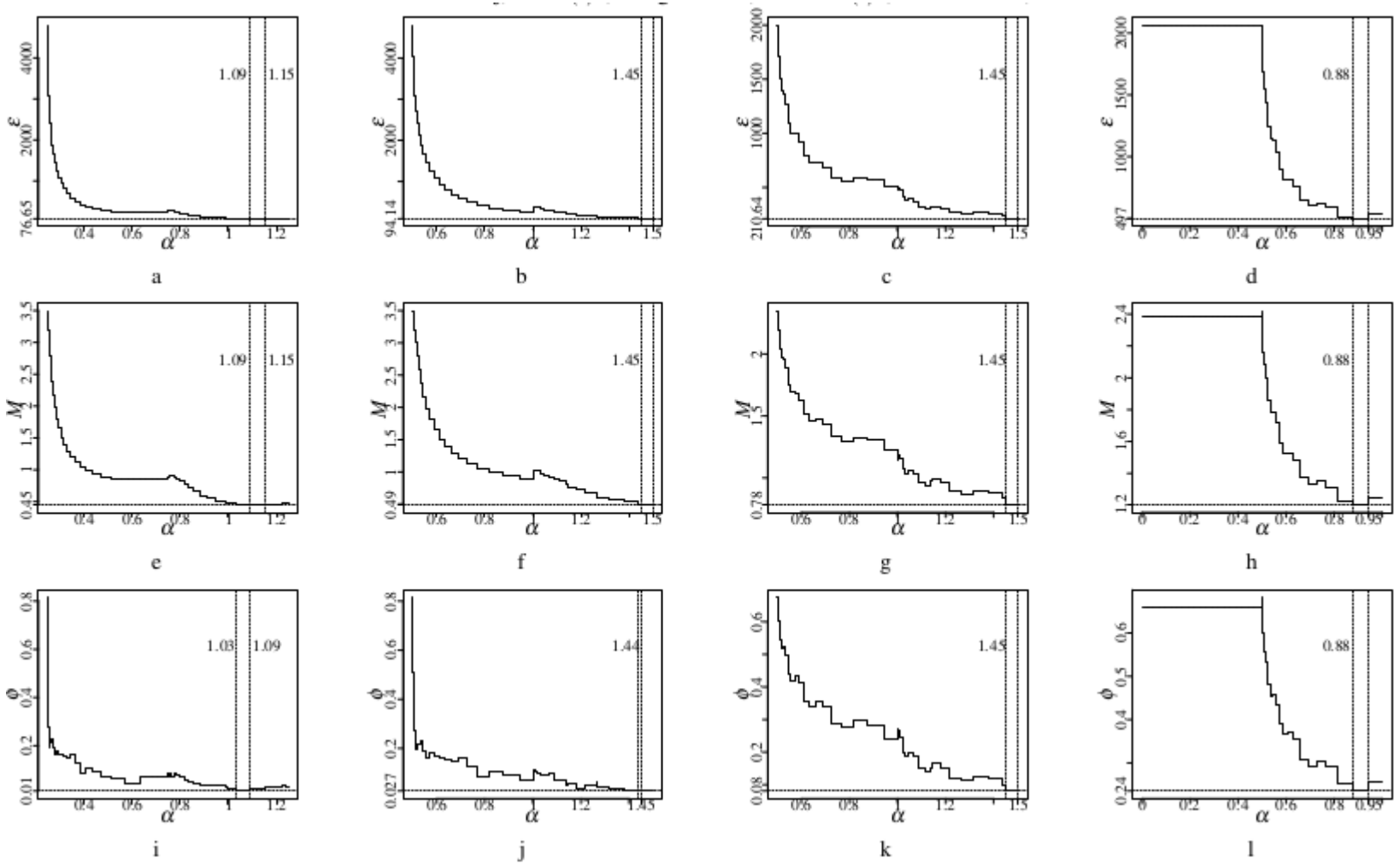


Figure 4 – Performance of the integer functions for $N = 11$ and $\mathscr{P} = \{-1, -\frac{1}{2}, 0, \frac{1}{2}, 1\}$ considering round$(\cdot)$ (4a, 4e, and 4i), trunc$(\cdot)$ (4b, 4f, and 4j), floor$(\cdot)$ (4c, 4g, and 4k), and ceil$(\cdot)$ (4d, 4h, and 4l)

Source: Author (2023).

Figure 5 – Performance of the integer functions for $N = 3$ and $\mathscr{P} = \{-1, -\frac{1}{2}, 0, \frac{1}{2}, 1\}$ considering round$(\cdot)$ (5a, 5e, and 5i), trunc$(\cdot)$ (5b, 5f, and 5j), floor$(\cdot)$ (5c, 5g, and 5k), and ceil$(\cdot)$ (5d, 5h, and 5l)



Source: Author (2023).

and

$$\mathbf{S}_3^* = \left[\begin{array}{c:c} 1 & \\ \hdashline & \sqrt{\frac{6}{7}} \cdot \mathbf{I}_2 \end{array}\right].$$

Therefore, the 31-, 11- and 3-point DFT approximations are given by

$$\hat{\mathbf{F}}_{31}^* = \mathbf{S}_{31}^* \cdot \mathbf{T}_{31}^*,$$

$$\hat{\mathbf{F}}_{11}^* = \mathbf{S}_{11}^* \cdot \mathbf{T}_{11}^*,$$

$$\hat{\mathbf{F}}_{3}^* = \mathbf{S}_{3}^* \cdot \mathbf{T}_{3}^*,$$

respectively.

4.3.4 Approximate Scale Factors

Despite the dramatic reduction in arithmetic complexity, including the absence of twiddle factors, the computation shown in (4.18) requires $2(N-1)$ real multiplications due to the elements of the diagonal $\mathbf{S}$. The diagonal values of $\mathbf{S}$, $s_i, i = 0, 1, \ldots, 1022$, are given by

$$s_i = \begin{cases} 1, & \text{if } i = 0, \\ \sqrt{\frac{6}{7}}, & \text{if } i \mod 31 = 0 \wedge i \mod 11 = 0 \wedge i \mod 3 \neq 0, \\ \sqrt{\frac{11}{13}}, & \text{if } i \mod 31 = 0 \wedge i \mod 11 \neq 0 \wedge i \mod 3 = 0, \\ \sqrt{\frac{66}{91}}, & \text{if } i \mod 31 = 0 \wedge i \mod 11 \neq 0 \wedge i \mod 3 \neq 0, \\ \sqrt{\frac{31}{38}}, & \text{if } i \mod 31 \neq 0 \wedge i \mod 11 = 0 \wedge i \mod 3 = 0, \\ \sqrt{\frac{93}{133}}, & \text{if } i \mod 31 \neq 0 \wedge i \mod 11 = 0 \wedge i \mod 3 \neq 0, \\ \sqrt{\frac{341}{494}}, & \text{if } i \mod 31 \neq 0 \wedge i \mod 11 \neq 0 \wedge i \mod 3 = 0, \\ \sqrt{\frac{1023}{1729}}, & \text{otherwise.} \end{cases}$$

Further complexity reductions can be obtained by approximating the elements of $\mathbf{S}$ according the canonical signed digit (CSD) number system [180], which satisfy the minimum adder representation criterion. The CSD representation is better detailed in

Chapter 5. The constants are irrational numbers, then we adopted the following procedure to represent in CSD. We approximated each element in $\mathbf{S}$ in a such way that its CSD representation admits at most two additions [181].

Table 2 provides a representation of the elements from $\mathbf{S}$, their occurrences, and their representation in CSD ($\bar{1}$ represents -1). Since the elements from $\mathbf{S}_{31}$, $\mathbf{S}_{11}$, and $\mathbf{S}_3$ are present in $\mathbf{S}$, multiplierless approximation for the 31- and 33-point DFT are also possible using Table 2. The approximations whose the diagonal matrices of which also approximated, follow the notation $\hat{\mathbf{F}}'_N$.

Table 2 – Approximations for the constants from $\mathbf{S}$

Constant	Approximation	Occurrences	CSD
$\sqrt{\frac{6}{7}} \approx 0.92582$	$\frac{59}{64} = 0.921875$	2	$1.000\bar{1}0\bar{1}$
$\sqrt{\frac{11}{13}} \approx 0.91987$	$\frac{59}{64} = 0.921875$	10	$1.000\bar{1}0\bar{1}$
$\sqrt{\frac{66}{91}} \approx 0.85163$	$\frac{27}{32} = 0.84375$	20	$1.00\bar{1}0\bar{1}0$
$\sqrt{\frac{31}{38}} \approx 0.90321$	$\frac{29}{32} = 0.90625$	30	$1.00\bar{1}0\bar{1}0$
$\sqrt{\frac{93}{133}} \approx 0.83621$	$\frac{27}{32} = 0.84375$	60	$1.00\bar{1}0\bar{1}0$
$\sqrt{\frac{341}{494}} \approx 0.83083$	$\frac{27}{32} = 0.84375$	300	$1.00\bar{1}0\bar{1}0$
$\sqrt{\frac{1023}{1729}} \approx 0.76920$	$\frac{49}{64} = 0.78125$	600	$1.0\bar{1}0001$

Source: Author (2023).

4.4 FAST ALGORITHMS AND ARITHMETIC COMPLEXITY

In this section, we introduce fast algorithms based on sparse matrix factorizations [21] of the proposed 31-, 11-, and 3-point approximations to reduce the number of remaining operations. The presented low-complexity matrices do not require multiplications, only additions and bit-shifting operations are needed [47, p. 221]. The arithmetic complexity of the proposed approximations is also assessed and compared with competing methods.

4.4.1 31-point Approximation

The low-complexity matrix $\mathbf{T}_{31}^{*}$ can be represented as:

$$\mathbf{T}_{31}^{*} = \mathbf{A}_1^{\top} \cdot \mathbf{M}_1 \cdot \mathbf{A}_1, \tag{4.22}$$

where

$$\mathbf{A}_1 = \left[\begin{array}{c|c} 1 & \\ \hline & \mathbf{B}_{30} \end{array}\right].$$

The matrix $\mathbf{M}_1$ is a block matrix given by

$$\mathbf{M}_1 = \left[\begin{array}{c|c} \mathbf{E}_4 & \\ \hline & j \cdot \mathbf{E}_5 \end{array}\right],$$

where

$$\mathbf{E}_4 = \left[\begin{array}{cccccccc|cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & \frac{1}{2} & \frac{1}{2} & & & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & \frac{1}{2} & & -\frac{1}{2} & -1 & -1 & -1 & -1 & -\frac{1}{2} & -\frac{1}{2} & & \frac{1}{2} & 1 & 1 \\
1 & 1 & \frac{1}{2} & -\frac{1}{2} & -1 & -1 & -1 & -\frac{1}{2} & & 1 & 1 & 1 & \frac{1}{2} & & -\frac{1}{2} & -1 \\
1 & 1 & & -1 & -1 & -\frac{1}{2} & & 1 & 1 & \frac{1}{2} & -\frac{1}{2} & -1 & -1 & -\frac{1}{2} & \frac{1}{2} & 1 \\
1 & \frac{1}{2} & -\frac{1}{2} & -1 & -\frac{1}{2} & \frac{1}{2} & 1 & 1 & -\frac{1}{2} & -1 & -1 & & 1 & 1 & & -1 \\
1 & \frac{1}{2} & -1 & -1 & & 1 & \frac{1}{2} & -\frac{1}{2} & -1 & & 1 & 1 & -\frac{1}{2} & -1 & -\frac{1}{2} & 1 \\
1 & & -1 & -\frac{1}{2} & 1 & 1 & -\frac{1}{2} & -1 & \frac{1}{2} & 1 & & -1 & -\frac{1}{2} & 1 & \frac{1}{2} & -1 \\
\hline
1 & & -1 & & 1 & -\frac{1}{2} & -1 & \frac{1}{2} & 1 & -\frac{1}{2} & -1 & \frac{1}{2} & 1 & -\frac{1}{2} & -1 & 1 \\
1 & -\frac{1}{2} & -1 & 1 & \frac{1}{2} & -1 & & 1 & -\frac{1}{2} & -1 & 1 & \frac{1}{2} & -1 & & 1 & -\frac{1}{2} \\
1 & -\frac{1}{2} & -\frac{1}{2} & 1 & -\frac{1}{2} & -1 & 1 & & -1 & 1 & & -1 & 1 & \frac{1}{2} & -1 & \frac{1}{2} \\
1 & -\frac{1}{2} & -\frac{1}{2} & 1 & -1 & & 1 & -1 & \frac{1}{2} & \frac{1}{2} & -1 & 1 & & -1 & 1 & -\frac{1}{2} \\
1 & -1 & & \frac{1}{2} & -1 & 1 & -\frac{1}{2} & -\frac{1}{2} & 1 & -1 & 1 & & -\frac{1}{2} & 1 & -1 & \frac{1}{2} \\
1 & -1 & \frac{1}{2} & & -\frac{1}{2} & 1 & -1 & 1 & -\frac{1}{2} & & \frac{1}{2} & -1 & 1 & -1 & 1 & -\frac{1}{2} \\
1 & -1 & 1 & -\frac{1}{2} & \frac{1}{2} & & -\frac{1}{2} & \frac{1}{2} & -1 & 1 & -1 & 1 & -1 & 1 & -\frac{1}{2} & \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & &
\end{array}\right],$$

$$\mathbf{E}_5 = \left[\begin{array}{cccccccc:cccccccc}
-1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & & & \\
1 & -1 & 1 & -\frac{1}{2} & & & -\frac{1}{2} & \frac{1}{2} & -1 & 1 & -1 & 1 & -1 & \frac{1}{2} & -\frac{1}{2} & \\
-1 & 1 & -\frac{1}{2} & & \frac{1}{2} & -1 & 1 & -1 & \frac{1}{2} & & -\frac{1}{2} & 1 & -1 & 1 & -\frac{1}{2} & \\
1 & -\frac{1}{2} & & 1 & -1 & 1 & & -\frac{1}{2} & 1 & -1 & \frac{1}{2} & \frac{1}{2} & -1 & 1 & -\frac{1}{2} & \\
-1 & & \frac{1}{2} & -1 & \frac{1}{2} & \frac{1}{2} & -1 & 1 & & -1 & 1 & -\frac{1}{2} & -\frac{1}{2} & 1 & -1 & \\
1 & & -1 & 1 & \frac{1}{2} & -1 & \frac{1}{2} & \frac{1}{2} & -1 & \frac{1}{2} & \frac{1}{2} & -1 & & 1 & -1 & \\
-1 & -\frac{1}{2} & 1 & & -1 & \frac{1}{2} & \frac{1}{2} & -1 & & 1 & -\frac{1}{2} & -1 & 1 & \frac{1}{2} & -1 & \\
1 & \frac{1}{2} & -1 & -\frac{1}{2} & 1 & \frac{1}{2} & -1 & -\frac{1}{2} & 1 & \frac{1}{2} & -1 & & 1 & & -1 & \\
\hdashline
-\frac{1}{2} & -1 & \frac{1}{2} & 1 & & -1 & & 1 & \frac{1}{2} & -1 & -1 & \frac{1}{2} & 1 & -\frac{1}{2} & -1 & \\
\frac{1}{2} & 1 & & -1 & -1 & \frac{1}{2} & 1 & \frac{1}{2} & -1 & -1 & & 1 & \frac{1}{2} & -\frac{1}{2} & -1 & \\
-\frac{1}{2} & -1 & -\frac{1}{2} & \frac{1}{2} & 1 & \frac{1}{2} & -\frac{1}{2} & -1 & -1 & & 1 & 1 & & -1 & -1 & \\
\frac{1}{2} & 1 & 1 & \frac{1}{2} & -\frac{1}{2} & -1 & -1 & & \frac{1}{2} & 1 & 1 & & -\frac{1}{2} & -1 & -1 & \\
-\frac{1}{2} & -1 & -1 & -1 & -\frac{1}{2} & & 1 & 1 & 1 & \frac{1}{2} & & -\frac{1}{2} & -1 & -1 & -\frac{1}{2} & \\
& \frac{1}{2} & 1 & 1 & 1 & 1 & \frac{1}{2} & & -\frac{1}{2} & -\frac{1}{2} & -1 & -1 & -1 & -1 & -\frac{1}{2} & \\
& -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -\frac{1}{2} & -\frac{1}{2} & &
\end{array}\right].$$

The matrix $\mathbf{A}_1$ requires only 60 real additions while $\mathbf{M}_1$ needs 780 real additions and 300 bit-shifting operations. Computing $\mathbf{T}_{31}^*$ directly requires 3180 additions and 1200 bit-shifting operations. However, after the factorization detailed in (4.22), the arithmetic complexity is reduced to 900 real additions and 300 bit-shifting operations. To scale the approximation maintaining the exact $\mathbf{S}_{31}^*$, 60 real multiplications are added to the previous arithmetic cost. This approximation is denoted by $\hat{\mathbf{F}}_{31}^*$. Otherwise, if the diagonal matrix is approximated then 120 additions and 120 bit-shifting operations are needed instead of the multiplications. This approximation is called $\hat{\mathbf{F}}'_{31}$.

4.4.2 11-point Approximation

For the 11-point approximation, the low-complexity matrix $\mathbf{T}_{11}^*$ is given by

$$\mathbf{T}_{11}^* = \mathbf{A}_2^\top \cdot \mathbf{M}_2 \cdot \mathbf{A}_2, \tag{4.23}$$

where

$$\mathbf{A}_2 = \left[\begin{array}{c:c} 1 & \\ \hdashline & \mathbf{B}_{10} \end{array}\right],$$

and the block matrix $\mathbf{M}_2$ is

$$\mathbf{M}_2 = \left[\begin{array}{cccccc:ccccc} 1 & 1 & 1 & 1 & 1 & 1 & & & & & \\ 1 & 1 & \frac{1}{2} & & -\frac{1}{2} & -1 & & & & & \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 & & 1 & & & & & \\ 1 & & -1 & \frac{1}{2} & 1 & -\frac{1}{2} & & & & & \\ 1 & -\frac{1}{2} & & 1 & -1 & \frac{1}{2} & & & & & \\ 1 & -1 & 1 & -\frac{1}{2} & \frac{1}{2} & & & & & & \\ \hdashline & & & & & & -j & j & -j & \frac{j}{2} & -\frac{j}{2} \\ & & & & & & j & -\frac{j}{2} & -\frac{j}{2} & j & -j \\ & & & & & & -j & -\frac{j}{2} & j & \frac{j}{2} & -j \\ & & & & & & \frac{j}{2} & j & \frac{j}{2} & -j & -j \\ & & & & & & -\frac{j}{2} & -j & -j & -j & -\frac{j}{2} \end{array}\right].$$

The matrix $\mathbf{A}_2$ requires 20 real additions and $\mathbf{M}_2$ needs 90 real additions and 40 bit-shifting operations. While the direct implementation of the $\mathbf{T}_{31}^*$ requires 380 additions and 160 bit-shifting operations, the factorization presented in (4.23) needs 130 real additions and 40 bit-shifting operations. The scaling of $\mathbf{F}_{11}^*$ requires 20 extra multiplications. On the other hand, $\mathbf{F}_{11}'$ needs 40 additions and 40 bit-shifting operations instead of 20 multiplications of the $\mathbf{F}_{11}^*$.

4.4.3 3-point Approximation

For the 3-point approximation, the low-complexity matrix $\mathbf{T}_3^*$ is given by

$$\mathbf{T}_3^* = \mathbf{A}_3^\top \cdot \mathbf{M}_3 \cdot \mathbf{A}_3, \tag{4.24}$$

where

$$\mathbf{A}_3 = \left[\begin{array}{c:c} 1 & \\ \hdashline & \mathbf{B}_2 \end{array}\right],$$

and the matrix $\mathbf{M}_3$ is

$$\mathbf{M}_3 = \left[\begin{array}{cc:c} 1 & 1 & \\ 1 & -\frac{1}{2} & \\ \hdashline & & -j \end{array}\right].$$

The matrix $\mathbf{A}_3$ requires only 4 real additions and $\mathbf{M}_3$ needs 4 real additions and 2 bit-shifting operations. The direct implementation of the $\mathbf{T}_3^*$ requires 20 additions and 8 bit-shifting operations. After applying the factorization in (4.24), the number of arithmetic operations is reduced to 12 real additions and 2 bit-shifting operations using the fast algorithm. The scaling to $\mathbf{F}_3^*$ requires 4 multiplications and the scaling to $\mathbf{F}_3'$ needs 8 additions and 8 bit-shifting operations.

4.4.4 31-, 11-, and 3-point DFT by definition

The direct computation of the DFT of a sequence of blocklength N requires complex multiplications in the order of $\mathscr{O}(N^2)$ [3, p. 750]. Considering the algorithm detailed in Section 2.4, the direct computation of the 31-, 11-, and 3-point DFT requires respectively 2700, 300, and 12 real multiplications (considering only non-trivial multiplications); and 4560, 520 and 24 real additions. However, such arithmetic complexity can be reduced by applying sparse matrix factorization in the exact DFTs.

The factorization of the $\mathbf{F}_{31}$ is obtained by:

$$\mathbf{F}_{31} = \mathbf{A}_1^\top \cdot \mathbf{M}_4 \cdot \mathbf{A}_1. \tag{4.25}$$

The block matrix $\mathbf{M}_4$ is given by

$$\mathbf{M}_4 = \left[\begin{array}{c:c} \mathbf{E}_6 & \\ \hdashline & \mathbf{E}_7 \end{array}\right],$$

where

$$\mathbf{E}_6 = \left[\begin{array}{cccccccc:cccccccc}
\delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 \\
\delta_0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 & \delta_7 & \delta_8 & \delta_9 & \delta_{10} & \delta_{11} & \delta_{12} & \delta_{13} & \delta_{14} & \delta_{15} \\
\delta_0 & \delta_2 & \delta_4 & \delta_6 & \delta_8 & \delta_{10} & \delta_{12} & \delta_{14} & \delta_{15} & \delta_{13} & \delta_{11} & \delta_9 & \delta_7 & \delta_5 & \delta_3 & \delta_1 \\
\delta_0 & \delta_3 & \delta_6 & \delta_9 & \delta_{12} & \delta_{15} & \delta_{13} & \delta_{10} & \delta_7 & \delta_4 & \delta_1 & \delta_2 & \delta_5 & \delta_8 & \delta_{11} & \delta_{14} \\
\delta_0 & \delta_4 & \delta_8 & \delta_{12} & \delta_{15} & \delta_{11} & \delta_7 & \delta_3 & \delta_1 & \delta_5 & \delta_9 & \delta_{13} & \delta_{14} & \delta_{10} & \delta_6 & \delta_2 \\
\delta_0 & \delta_5 & \delta_{10} & \delta_{15} & \delta_{11} & \delta_6 & \delta_1 & \delta_4 & \delta_9 & \delta_{14} & \delta_{12} & \delta_7 & \delta_2 & \delta_3 & \delta_8 & \delta_{13} \\
\delta_0 & \delta_6 & \delta_{12} & \delta_{13} & \delta_7 & \delta_1 & \delta_5 & \delta_{11} & \delta_{14} & \delta_8 & \delta_2 & \delta_4 & \delta_{10} & \delta_{15} & \delta_9 & \delta_3 \\
\delta_0 & \delta_7 & \delta_{14} & \delta_{10} & \delta_3 & \delta_4 & \delta_{11} & \delta_{13} & \delta_6 & \delta_1 & \delta_8 & \delta_{15} & \delta_9 & \delta_2 & \delta_5 & \delta_{12} \\
\hdashline
\delta_0 & \delta_8 & \delta_{15} & \delta_7 & \delta_1 & \delta_9 & \delta_{14} & \delta_6 & \delta_2 & \delta_{10} & \delta_{13} & \delta_5 & \delta_3 & \delta_{11} & \delta_{12} & \delta_4 \\
\delta_0 & \delta_9 & \delta_{13} & \delta_4 & \delta_5 & \delta_{14} & \delta_8 & \delta_1 & \delta_{10} & \delta_{12} & \delta_3 & \delta_6 & \delta_{15} & \delta_7 & \delta_2 & \delta_{11} \\
\delta_0 & \delta_{10} & \delta_{11} & \delta_1 & \delta_9 & \delta_{12} & \delta_2 & \delta_8 & \delta_{13} & \delta_3 & \delta_7 & \delta_{14} & \delta_4 & \delta_6 & \delta_{15} & \delta_5 \\
\delta_0 & \delta_{11} & \delta_9 & \delta_2 & \delta_{13} & \delta_7 & \delta_4 & \delta_{15} & \delta_5 & \delta_6 & \delta_{14} & \delta_3 & \delta_8 & \delta_{12} & \delta_1 & \delta_{10} \\
\delta_0 & \delta_{12} & \delta_7 & \delta_5 & \delta_{14} & \delta_2 & \delta_{10} & \delta_9 & \delta_3 & \delta_{15} & \delta_4 & \delta_8 & \delta_{11} & \delta_1 & \delta_{13} & \delta_6 \\
\delta_0 & \delta_{13} & \delta_5 & \delta_8 & \delta_{10} & \delta_3 & \delta_{15} & \delta_2 & \delta_{11} & \delta_7 & \delta_6 & \delta_{12} & \delta_1 & \delta_{14} & \delta_4 & \delta_9 \\
\delta_0 & \delta_{14} & \delta_3 & \delta_{11} & \delta_6 & \delta_8 & \delta_9 & \delta_5 & \delta_{12} & \delta_2 & \delta_{15} & \delta_1 & \delta_{13} & \delta_4 & \delta_{10} & \delta_7 \\
\delta_0 & \delta_{15} & \delta_1 & \delta_{14} & \delta_2 & \delta_{13} & \delta_3 & \delta_{12} & \delta_4 & \delta_{11} & \delta_5 & \delta_{10} & \delta_6 & \delta_9 & \delta_7 & \delta_8
\end{array}\right],$$

and

$$\mathbf{E}_7 = \left[\begin{array}{cccccccc:cccccccc}
\lambda_8 & \lambda_{24} & \lambda_9 & \lambda_{25} & \lambda_{10} & \lambda_{26} & \lambda_{11} & \lambda_{27} & \lambda_{12} & \lambda_{28} & \lambda_{13} & \lambda_{29} & \lambda_{14} & \lambda_{30} & \lambda_{15} \\
\lambda_{24} & \lambda_{10} & \lambda_{27} & \lambda_{13} & \lambda_{30} & \lambda_{16} & \lambda_2 & \lambda_{19} & \lambda_5 & \lambda_{22} & \lambda_8 & \lambda_{25} & \lambda_{11} & \lambda_{28} & \lambda_{14} \\
\lambda_9 & \lambda_{27} & \lambda_{14} & \lambda_1 & \lambda_{19} & \lambda_6 & \lambda_{24} & \lambda_{11} & \lambda_{29} & \lambda_{16} & \lambda_3 & \lambda_{21} & \lambda_8 & \lambda_{26} & \lambda_{13} \\
\lambda_{25} & \lambda_{13} & \lambda_1 & \lambda_{20} & \lambda_8 & \lambda_{27} & \lambda_{15} & \lambda_3 & \lambda_{22} & \lambda_{10} & \lambda_{29} & \lambda_{17} & \lambda_5 & \lambda_{24} & \lambda_{12} \\
\lambda_{10} & \lambda_{30} & \lambda_{19} & \lambda_8 & \lambda_{28} & \lambda_{17} & \lambda_6 & \lambda_{26} & \lambda_{15} & \lambda_4 & \lambda_{24} & \lambda_{13} & \lambda_2 & \lambda_{22} & \lambda_{11} \\
\lambda_{26} & \lambda_{16} & \lambda_6 & \lambda_{27} & \lambda_{17} & \lambda_7 & \lambda_{28} & \lambda_{18} & \lambda_8 & \lambda_{29} & \lambda_{19} & \lambda_9 & \lambda_{30} & \lambda_{20} & \lambda_{10} \\
\lambda_{11} & \lambda_2 & \lambda_{24} & \lambda_{15} & \lambda_6 & \lambda_{28} & \lambda_{19} & \lambda_{10} & \lambda_1 & \lambda_{23} & \lambda_{14} & \lambda_5 & \lambda_{27} & \lambda_{18} & \lambda_9 \\
\lambda_{27} & \lambda_{19} & \lambda_{11} & \lambda_3 & \lambda_{26} & \lambda_{18} & \lambda_{10} & \lambda_2 & \lambda_{25} & \lambda_{17} & \lambda_9 & \lambda_1 & \lambda_{24} & \lambda_{16} & \lambda_8 \\
\hdashline
\lambda_{12} & \lambda_5 & \lambda_{29} & \lambda_{22} & \lambda_{15} & \lambda_8 & \lambda_1 & \lambda_{25} & \lambda_{18} & \lambda_{11} & \lambda_4 & \lambda_{28} & \lambda_{21} & \lambda_{14} & \lambda_7 \\
\lambda_{28} & \lambda_{22} & \lambda_{16} & \lambda_{10} & \lambda_4 & \lambda_{29} & \lambda_{23} & \lambda_{17} & \lambda_{11} & \lambda_5 & \lambda_{30} & \lambda_{24} & \lambda_{18} & \lambda_{12} & \lambda_6 \\
\lambda_{13} & \lambda_8 & \lambda_3 & \lambda_{29} & \lambda_{24} & \lambda_{19} & \lambda_{14} & \lambda_9 & \lambda_4 & \lambda_{30} & \lambda_{25} & \lambda_{20} & \lambda_{15} & \lambda_{10} & \lambda_5 \\
\lambda_{29} & \lambda_{25} & \lambda_{21} & \lambda_{17} & \lambda_{13} & \lambda_9 & \lambda_5 & \lambda_1 & \lambda_{28} & \lambda_{24} & \lambda_{20} & \lambda_{16} & \lambda_{12} & \lambda_8 & \lambda_4 \\
\lambda_{14} & \lambda_{11} & \lambda_8 & \lambda_5 & \lambda_2 & \lambda_{30} & \lambda_{27} & \lambda_{24} & \lambda_{21} & \lambda_{18} & \lambda_{15} & \lambda_{12} & \lambda_9 & \lambda_6 & \lambda_3 \\
\lambda_{30} & \lambda_{28} & \lambda_{26} & \lambda_{24} & \lambda_{22} & \lambda_{20} & \lambda_{18} & \lambda_{16} & \lambda_{14} & \lambda_{12} & \lambda_{10} & \lambda_8 & \lambda_6 & \lambda_4 & \lambda_2 \\
\lambda_{15} & \lambda_{14} & \lambda_{13} & \lambda_{12} & \lambda_{11} & \lambda_{10} & \lambda_9 & \lambda_8 & \lambda_7 & \lambda_6 & \lambda_5 & \lambda_4 & \lambda_3 & \lambda_2 & \lambda_1
\end{array}\right],$$

in which $\delta_k = \cos(\frac{2\cdot\pi\cdot k}{31})$ and $\lambda_k = -j\cdot\sin(\frac{2\cdot\pi\cdot k}{31})$. The factorization detailed in (4.25) reduces the arithmetic complexity of $\mathbf{F}_{31}$ down to 900 real multiplications and $60+60+900=1020$ real additions.

Applying the factorization in $\mathbf{F}_{11}$, we have

$$\mathbf{F}_{11} = \mathbf{A}_2^{\top}\cdot\mathbf{M}_5\cdot\mathbf{A}_2, \tag{4.26}$$

where

$$\mathbf{M}_5 = \left[\begin{array}{cccccc:cccccc} \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & \delta_0 & & & & & & \\ \delta_0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & & & & & & \\ \delta_0 & \delta_2 & \delta_4 & \delta_5 & \delta_3 & \delta_1 & & & & & & \\ \delta_0 & \delta_3 & \delta_5 & \delta_2 & \delta_1 & \delta_4 & & & & & & \\ \delta_0 & \delta_4 & \delta_3 & \delta_1 & \delta_5 & \delta_2 & & & & & & \\ \delta_0 & \delta_5 & \delta_1 & \delta_4 & \delta_2 & \delta_3 & & & & & & \\ \hdashline & & & & & & \lambda_3 & \lambda_9 & \lambda_4 & \lambda_{10} & \lambda_5 \\ & & & & & & \lambda_9 & \lambda_5 & \lambda_1 & \lambda_8 & \lambda_4 \\ & & & & & & \lambda_4 & \lambda_1 & \lambda_9 & \lambda_6 & \lambda_3 \\ & & & & & & \lambda_{10} & \lambda_8 & \lambda_6 & \lambda_4 & \lambda_2 \\ & & & & & & \lambda_5 & \lambda_4 & \lambda_3 & \lambda_2 & \lambda_1 \end{array}\right].$$

The arithmetic cost of computing $\mathbf{F}_{11}$ by (4.26) is 100 real multiplications and 140 real additions.

Lastly, the factorization of $\mathbf{F}_3$ into sparse matrices is given by

$$\mathbf{F}_3 = \mathbf{A}_3^\top \cdot \mathbf{M}_6 \cdot \mathbf{A}_3, \tag{4.27}$$

where

$$\mathbf{M}_6 = \left[\begin{array}{cc:c} \delta_0 & \delta_0 & \\ \delta_0 & \delta_2 & \\ \hdashline & & \lambda_1 \end{array}\right].$$

The factorization presented in (4.27) reduces the arithmetic complexity of $\mathbf{F}_3$ down to 2 real multiplications, $4+4+4=12$ real additions, and 2 bit-shifting operations.

4.4.5 1023-point DFT Approximation

The PFA to compute the 1023-point DFT, as defined in Section 2.3, requires $(33\times 2700)+(93\times 300)+(341\times 12)=121092$ real multiplications and $(33\times 4560)+(93\times 520)+(341\times 24)=207024$ real additions using the 31-, 11-, and 3-point DFT directly from their definitions. However, if the exact DFTs are computed by the sparse

matrix factorizations detailed in Sections 4.4.1, 4.4.2, and 4.4.3, then $(33 \times 900) + (93 \times 100) + (341 \times 2) = 39682$ real multiplications, $(33 \times 1020) + (93 \times 140) + (341 \times 12) = 50772$ real additions, and $(341 \times 2) = 682$ bit-shifting operations are needed.

The unscaled algorithm to compute the 1023-point DFT approximation, referred to as $\hat{\mathbf{T}}^*_{1023}$, has null complexity of multiplications. To calculate $\hat{\mathbf{T}}^*_{1023}$, the matrix $\mathbf{T}^*_{31}$ is called 33 times contributing with $900 \times 33 = 29700$ real additions and $300 \times 33 = 9900$ bit-shifting operations. On the other hand, $\mathbf{T}^*_{11}$ is called 93 times contributing with $130 \times 93 = 12090$ real additions and $40 \times 93 = 3720$ bit-shifting operations and $\mathbf{T}^*_3$ is called 341 times which corresponds to $12 \times 341 = 4092$ real additions and $2 \times 341 = 682$ bit-shifting operations. Then, the resulting arithmetic costs of $\hat{\mathbf{T}}^*_{1023}$ are 45882 real additions and 14302 bit-shifting operations. To compute the scaled 1023-point DFT approximation with the exact $\mathbf{S}$, called $\hat{\mathbf{F}}^*_{1023}$, more 2044 multiplications are necessary. However, if $\mathbf{S}$ is approximated following Table 2, instead of multiplications, 4088 additions and 4088 bit-shifting operations are needed to achieve $\hat{\mathbf{F}}'_{1023}$.

4.4.6 Hybrid 1023-point DFT Approximation

Applying the hybrid approach detailed in Section 4.2.3 to the 1023-point DFT approximation defined in (4.18) and (4.19), we obtained 12 different approximations. The 1023-point DFT approximations consist of four elements: a diagonal matrix $\mathbf{S}$, a 31-, an 11-, and a 3-point transformation. In the hybrid approximations, these four elements are alternated between exact and approximate form. Table 3 helps to understand the approximations by providing the combinations of the four elements. All DFTs computed by the definition (exact form) use the factorization mentioned in Section 4.4.4.

The approximations $\hat{\mathbf{F}}^*_{1023\text{-I}}$ and $\hat{\mathbf{F}}'_{1023\text{-I}}$ need $(33 \times 900) + (93 \times 100) = 39000$ real multiplications, $(33 \times 1020) + (93 \times 140) + (341 \times 12) = 50772$ real additions, and $341 \times 2 = 682$ bit-shifting operations plus 1364 real multiplications if $\mathbf{S}$ is kept exact, or 2728 real additions and 2728 bit-shifting operations if $\mathbf{S}$ is approximated.

The arithmetic complexities of $\hat{\mathbf{F}}^*_{1023\text{-II}}$ and $\hat{\mathbf{F}}'_{1023\text{-II}}$ are $(33 \times 900) + (341 \times 2) = 30382$ real multiplications, $(33 \times 1020) + (93 \times 130) + (341 \times 12) = 49842$ real additions,

Table 3 – Hybrid approximations for the 1023-point DFT

		Employed transformation		
Approximation	**S**	31-point	11-point	3-point
$\hat{\mathbf{F}}^{*}_{1023\text{-I}}$	Exact	$\mathbf{F}_{31}$	$\mathbf{F}_{11}$	$\mathbf{T}^{*}_{3}$
$\hat{\mathbf{F}}'_{1023\text{-I}}$	CSD approx.	$\mathbf{F}_{31}$	$\mathbf{F}_{11}$	$\mathbf{T}^{*}_{3}$
$\hat{\mathbf{F}}^{*}_{1023\text{-II}}$	Exact	$\mathbf{F}_{31}$	$\mathbf{T}^{*}_{11}$	$\mathbf{F}_{3}$
$\hat{\mathbf{F}}'_{1023\text{-II}}$	CSD approx.	$\mathbf{F}_{31}$	$\mathbf{T}^{*}_{11}$	$\mathbf{F}_{3}$
$\hat{\mathbf{F}}^{*}_{1023\text{-III}}$	Exact	$\mathbf{T}^{*}_{31}$	$\mathbf{F}_{11}$	$\mathbf{F}_{3}$
$\hat{\mathbf{F}}'_{1023\text{-III}}$	CSD approx.	$\mathbf{T}^{*}_{31}$	$\mathbf{F}_{11}$	$\mathbf{F}_{3}$
$\hat{\mathbf{F}}^{*}_{1023\text{-IV}}$	Exact	$\mathbf{F}_{31}$	$\mathbf{T}^{*}_{11}$	$\mathbf{T}^{*}_{3}$
$\hat{\mathbf{F}}'_{1023\text{-IV}}$	CSD approx.	$\mathbf{F}_{31}$	$\mathbf{T}^{*}_{11}$	$\mathbf{T}^{*}_{3}$
$\hat{\mathbf{F}}^{*}_{1023\text{-V}}$	Exact	$\mathbf{T}^{*}_{31}$	$\mathbf{F}_{11}$	$\mathbf{T}^{*}_{3}$
$\hat{\mathbf{F}}'_{1023\text{-V}}$	CSD approx.	$\mathbf{T}^{*}_{31}$	$\mathbf{F}_{11}$	$\mathbf{T}^{*}_{3}$
$\hat{\mathbf{F}}^{*}_{1023\text{-VI}}$	Exact	$\mathbf{T}^{*}_{31}$	$\mathbf{T}^{*}_{11}$	$\mathbf{F}_{3}$
$\hat{\mathbf{F}}'_{1023\text{-VI}}$	CSD approx.	$\mathbf{T}^{*}_{31}$	$\mathbf{T}^{*}_{11}$	$\mathbf{F}_{3}$

Source: Author (2023).

and $(93 \times 40) + (341 \times 2) = 4402$ bit-shifting operations plus 1860 real multiplications keeping **S** exact or 3720 real additions and 3720 bit-shifting operations if the approximate **S** is used.

To compute $\hat{\mathbf{F}}^{*}_{1023\text{-III}}$ and $\hat{\mathbf{F}}'_{1023\text{-III}}$, we need $(93 \times 100) + (341 \times 2) = 9982$ real multiplications, $(33 \times 900) + (93 \times 140) + (341 \times 12) = 46812$ real additions, and $(33 \times 300) + (341 \times 2) = 10582$ bit-shifting operations plus 1980 real multiplications to scale the DFT with the exact **S** or 3960 real additions and 3960 bit-shifting operations with the approximate **S**.

Disregarding the scaling matrix **S**, the approximations $\hat{\mathbf{F}}^{*}_{1023\text{-IV}}$ and $\hat{\mathbf{F}}'_{1023\text{-IV}}$ require $33 \times 900 = 29700$ real multiplications, $(33 \times 1020) + (93 \times 130) + (341 \times 12) = 49842$ real additions, and $(93 \times 40) + (341 \times 2) = 4402$ bit-shifting operations. If the exact matrix **S** is used, 1984 real multiplications are added to the computation. Otherwise, if **S** is approximated by the CSD, 3968 real additions and 3968 bit-shifting operations are required.

The approximations $\hat{\mathbf{F}}^{*}_{1023\text{-V}}$ and $\hat{\mathbf{F}}'_{1023\text{-V}}$ require $93 \times 100 = 9300$ real multiplications, $(33 \times 900) + (93 \times 140) + (341 \times 12) = 46812$ real additions, and $(33 \times 300) + (341 \times 2) = 10582$ bit-shifting operations. In addition, 2024 real multiplications or 4048

real additions and 4048 bit-shifting operations are needed if the matrix $\mathbf{S}$ is kept exact or approximated, respectively.

These hybrid approximations $\hat{\mathbf{F}}^{*}_{1023\text{-VI}}$ and $\hat{\mathbf{F}}'_{1023\text{-VI}}$ demands $341 \times 2 = 682$ real multiplications, $(33 \times 900) + (93 \times 130) + (341 \times 12) = 45882$ real additions, and $(33 \times 300) + (93 \times 40) + (341 \times 2) = 14302$ bit-shifting operations. The scaling needs 2040 real multiplications if $\mathbf{S}$ is kept exact, or 4080 real additions and 4080 bit-shifting operations, if $\mathbf{S}$ is approximated.

4.4.7 Arithmetic Complexity Comparison

First, in Table 4, the arithmetic complexity of the proposed ground transformation matrices is compared with their respective definitions and the ground transformation used in [74] which is calculated by the fully optimized Cooley-Tukey Radix-2 [21] and it is denoted by $\hat{\mathbf{F}}_{32}$.

Second, in Table 5, we compare the arithmetic complexity of the proposed algorithms with the fully optimized Cooley-Tukey Radix-2 ($\mathbf{F}_{1024}$) and the three algorithms proposed in [74] ($\mathbf{F}^{\mathrm{I}}_{1024}$, $\mathbf{F}^{\mathrm{II}}_{1024}$, and $\mathbf{F}^{\mathrm{III}}_{1024}$). In the first algorithm $\mathbf{F}^{\mathrm{I}}_{1024}$ proposed in [74], both row- and column-wise 32-point DFTs are replaced by the multiplierless $\mathbf{F}_{32}$. In the second ($\mathbf{F}^{\mathrm{II}}_{1024}$) and third algorithm ($\mathbf{F}^{\mathrm{III}}_{1024}$), only one is replaced by the $\mathbf{F}_{32}$. Despite the Cooley-Tukey Radix-2 and the method proposed in [74] are used to calculate the 1024-point DFT, it is the closest comparison found in the literature.

According to Table 4, power-of-two transformation matrices benefit more from factorization than prime-length transformations as example $\hat{\mathbf{T}}^{*}_{31}$ requires 552 additions more than $\hat{\mathbf{F}}_{32}$. However, when power-of-two transformation matrices are used as a building block in the Cooley-Tukey algorithm, multiplications are required due to the mapping as we can see in Table 5. Even in $\hat{\mathbf{F}}^{\mathrm{I}}_{1024}$, where $\mathbf{F}_{32}$ is replaced twice for $\hat{\mathbf{F}}_{32}$, there is still a source of multiplication, the twiddle factors. In contrast, the mapping used in the proposed approximations does not require multiplications. The approximations $\hat{\mathbf{T}}^{*}_{1023}$ and $\hat{\mathbf{F}}'_{1023}$ are free of multiplications in all computations.

Table 4 – Arithmetic complexity of the ground transformations

Transform	Real Mult.	Real Add.	Bit-shifting
$\mathbf{F}_3$(by definition [3, p. 750])	12	24	0
$\mathbf{F}_3$(Rader prime [182] using 2-point DFT)	2	16	8
$\mathbf{F}_3$(Cyclotomic basis [183])	2	14	4
$\mathbf{F}_{11}$(by definition [3, p. 750])	300	520	0
$\mathbf{F}_{11}$(Rader prime [182] using 10-point DFT)	264	494	0
$\mathbf{F}_{31}$(by definition [3, p. 750])	2700	4560	0
$\mathbf{F}_{31}$(Rader prime [182] using 30-point DFT)	1148	2420	0
$\mathbf{F}_{32}$(Cooley-Tukey radix-2 [21, p. 76])	88	408	0
$\mathbf{F}_{32}$(Rader-Brener radix-2 [21, p. 76])	68	512	0
$\hat{\mathbf{F}}_{32}$ (proposed in [74])	0	348	0
$\mathbf{F}_3$(by the proposed FFT)	2	12	2
$\hat{\mathbf{T}}_3^*$	0	12	2
$\hat{\mathbf{F}}_3^*$	4	12	2
$\hat{\mathbf{F}}_3'$	0	20	10
$\mathbf{F}_{11}$(by the proposed FFT)	100	140	0
$\hat{\mathbf{T}}_{11}^*$	0	130	40
$\hat{\mathbf{F}}_{11}^*$	20	130	40
$\hat{\mathbf{F}}_{11}'$	0	170	80
$\mathbf{F}_{31}$(by the proposed FFT)	900	1020	0
$\hat{\mathbf{T}}_{31}^*$	0	900	300
$\hat{\mathbf{F}}_{31}^*$	60	900	300
$\hat{\mathbf{F}}_{31}'$	0	1020	420

Source: Author (2023).

4.5 ERROR ANALYSIS

The values of the proximity measures defined for the proposed approximations are compared with the approximations proposed in [74] in Table 6 and 7. Although the proposed approximations are not strictly orthogonal, their deviations from orthogonality are extremely low (all below 10^{-2}). MAPE measurements are also smaller for the proposed approximations. For example, $\hat{\mathbf{F}}'_{1023}$ which is the fully approximate transform shows a reduction of $\approx$23% when compared with $\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$ and $\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$ that are hybrid transform (more than half of the algorithm is computed by the definition).

The total error energy indicates that the proposed approximations are more refined than the approximations in [74]. For the ground approximations, it can be verified by

Table 5 – Arithmetic complexity of fast algorithms for blocklengths 1023 and 1024

Transform	Real Mult.	Real Add.	Bit-shifting
$\mathbf{F}_{1024}$(Cooley-Tukey Radix-2 [21])	10248	30728	0
$\hat{\mathbf{F}}^{\mathrm{I}}_{1024}$(proposed in [74])	2883	25155	0
$\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$(proposed in [74])	5699	27075	0
$\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$(proposed in [74])	5699	27075	0
$\mathbf{F}_{1023}$(by definition [3, p. 750])	121092	207024	0
$\mathbf{F}_{1023}$(by the proposed FFT)	39682	50772	682
$\hat{\mathbf{F}}^{*}_{1023\text{-I}}$	40364	50772	682
$\hat{\mathbf{F}}'_{1023\text{-I}}$	39000	53500	3410
$\hat{\mathbf{F}}^{*}_{1023\text{-II}}$	32242	49842	4402
$\hat{\mathbf{F}}'_{1023\text{-II}}$	30382	53562	8122
$\hat{\mathbf{F}}^{*}_{1023\text{-III}}$	11962	46812	10582
$\hat{\mathbf{F}}'_{1023\text{-III}}$	9982	50772	14542
$\hat{\mathbf{F}}^{*}_{1023\text{-IV}}$	31684	49842	4402
$\hat{\mathbf{F}}'_{1023\text{-IV}}$	29700	53810	8370
$\hat{\mathbf{F}}^{*}_{1023\text{-V}}$	11324	46812	10582
$\hat{\mathbf{F}}'_{1023\text{-V}}$	9300	50860	14630
$\hat{\mathbf{F}}^{*}_{1023\text{-VI}}$	2722	45882	14302
$\hat{\mathbf{F}}'_{1023\text{-VI}}$	682	49962	18382
$\hat{\mathbf{T}}^{*}_{1023}$	0	45882	14302
$\hat{\mathbf{F}}^{*}_{1023}$	2044	45882	14302
$\hat{\mathbf{F}}'_{1023}$	**0**	**49970**	**18390**

Source: Author (2023).

Table 6 – Error measurements of the ground transformations

Transform	ε	M	$\phi(\times 10^{-3})$
$\hat{\mathbf{F}}_{32}$	3.32×10^{2}	0.84	36.07
$\hat{\mathbf{F}}^{*}_{3}$	9.68×10^{-2}	1.59	6.73
$\hat{\mathbf{F}}'_{3}$	9.80×10^{-2}	1.60	6.77
$\hat{\mathbf{F}}^{*}_{11}$	8.88	1.19	14.12
$\hat{\mathbf{F}}'_{11}$	8.91	1.20	14.11
$\hat{\mathbf{F}}^{*}_{31}$	7.66×10^{1}	0.45	19.83
$\hat{\mathbf{F}}'_{31}$	7.69×10^{1}	0.45	19.84

Source: Author (2023).

Table 7 – Error measurements of fast algorithms

Transform	$\varepsilon(\times 10^4)$	$M(\times 10^{-3})$	$\phi(\times 10^{-3})$
$\hat{\mathbf{F}}^{\mathrm{I}}_{1024}$	93.00	44	69.42
$\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$	34.02	25.31	36.07
$\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$	34.02	25.31	36.07
$\hat{\mathbf{F}}^{*}_{1023\text{-I}}$	1.13	4.67	6.73
$\hat{\mathbf{F}}'_{1023\text{-I}}$	1.13	4.69	6.77
$\hat{\mathbf{F}}^{*}_{1023\text{-II}}$	7.68	12.83	14.12
$\hat{\mathbf{F}}'_{1023\text{-II}}$	7.70	12.86	14.11
$\hat{\mathbf{F}}^{*}_{1023\text{-III}}$	8.35	13.68	19.83
$\hat{\mathbf{F}}'_{1023\text{-III}}$	8.38	13.70	19.84
$\hat{\mathbf{F}}^{*}_{1023\text{-IV}}$	8.80	14.12	20.76
$\hat{\mathbf{F}}'_{1023\text{-IV}}$	8.88	14.18	20.79
$\hat{\mathbf{F}}^{*}_{1023\text{-V}}$	9.46	14.77	26.43
$\hat{\mathbf{F}}'_{1023\text{-V}}$	9.55	14.82	26.49
$\hat{\mathbf{F}}^{*}_{1023\text{-VI}}$	15.93	18.67	33.68
$\hat{\mathbf{F}}'_{1023\text{-VI}}$	16.66	19.86	33.78
$\hat{\mathbf{F}}^{*}_{1023}$	17.03	19.41	40.18
$\hat{\mathbf{F}}'_{1023}$	17.10	19.45	40.06

Source: Author (2023).

comparing $\hat{\mathbf{F}}_{32}$ with $\hat{\mathbf{F}}^{*}_{31}$ (reduction of $\approx$77%) or $\hat{\mathbf{F}}'_{31}$ (reduction of $\approx$76%) in Table 6 which are the fairest comparisons.

The good performance of the proposed ground approximations is transferred to the 1023-point DFT approximations. According to Table 7, the fully approximate transform $\hat{\mathbf{F}}'_{1023}$ shows reduction of approximately 50% when compared with $\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$ and $\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$ and it increases to 82% when compared with $\hat{\mathbf{F}}^{\mathrm{I}}_{1024}$ in terms of total error energy. In addition to the error measures presented, we provide in the next section an in-depth analysis of the frequency domain.

4.5.1 Frequency Response

The DFT is a time-invariant linear transformation, so it is categorized as a linear time-invariant (LTI) system [6, p. 9]. An LTI system can be completely characterized in the frequency domain by its frequency response [3, p. 288]. In this analysis, each row of

the transform matrix is considered as a finite impulse response (FIR) filter bank [3, p. 204]. The frequency response of the exact DFT matrix is given by:

$$H_m(\nu, \mathbf{F}_N) = \sum_{n=0}^{N-1} f[m,n] \cdot \exp(-jn\nu), \quad \nu \in [-\pi, \pi], \tag{4.28}$$

where $H_m(\cdot,\cdot)$ is the frequency response of the m-th row and ν is the frequency. The frequency response of exact DFT rows have lobes with specific shapes and locations that are compared with their respective approximate versions. The rows submitted to the frequency response are the ones that presented the highest values of error energy, in a worst-case scenario analysis.

We focus the analysis on $\hat{\mathbf{F}}'_3$, $\hat{\mathbf{F}}'_{11}$, $\hat{\mathbf{F}}'_{31}$, and $\hat{\mathbf{F}}'_{1023}$ because they are fully approximated, scaled and, free of multiplications. The total error energy of $\hat{\mathbf{F}}'_3$, $\hat{\mathbf{F}}'_{11}$, and $\hat{\mathbf{F}}'_{31}$ is detailed by row in Table 8. The worst measurements are in boldface. Due to the length of the 1023-point DFT approximation, the row detailing has been omitted. However, the three least performing rows are 854, 699, and 86 with 306.08, 287.1, and 286.29 of error energy, respectively, being 167.15 the mean of the error energy.

The filter bank frequency response is applied to the least three performing rows and their respective rows of the exact DFT to compare them and evaluate the worst-case scenarios. If the approximations are able to maintain the DFT characteristics on the rows with higher error energy then it is expected that the other rows also maintain. The filter banks implied by $\hat{\mathbf{F}}'_3$, $\hat{\mathbf{F}}'_{11}$, and $\hat{\mathbf{F}}'_{31}$ are presented in Figure 6, 7, 8, respectively. The frequency response is normalized and presented in terms of magnitude. Fig 6 shows that all rows of the approximation performed close to the exact DFT which is due to the fact that most of the 3-point DFT constants are already of low complexity. In Fig 7, we can see that the approximation for $N = 11$ caused a slightly displacement of the lobes. The filter bank frequency response of the rows 5 and 6 are close to their respective exact DFT row, only the secondary lobes of the row 11 have relevant performance losses. The filter bank frequency response of $\hat{\mathbf{F}}'_{31}$ plotted in Fig 8 shows us a performance close to the exact DFT even for the rows with the highest error energy.

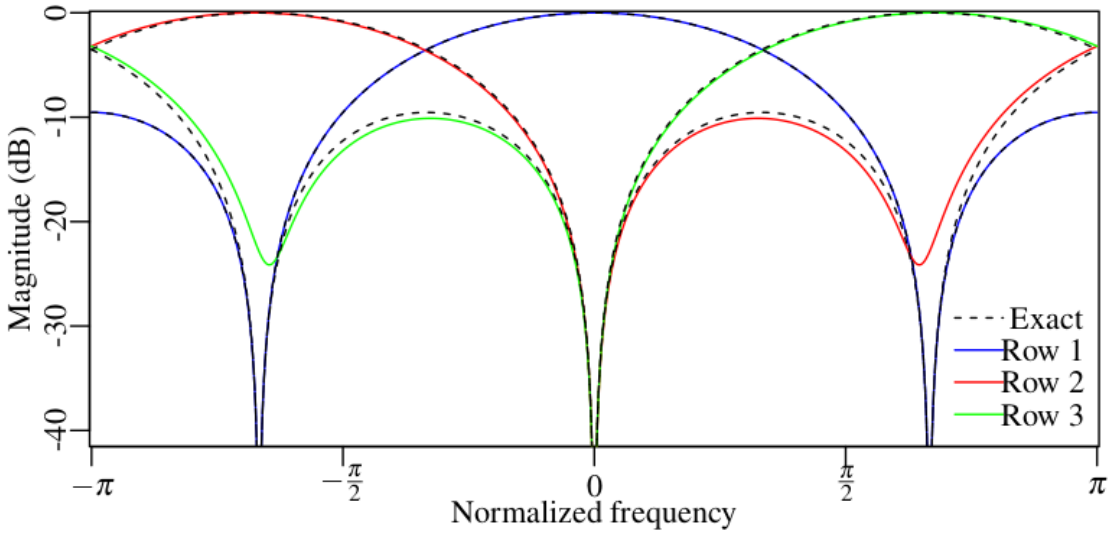
The 1023-point DFT filter bank frequency response has only one lobe as shown in Figure 9. For a better visualization, we zoom in the filter response around the lobe in

Table 8 – Error energy by rows of the ground approximations of the proposed method with the least performing rows highlighted

	Approximations		
Row	$\hat{\mathbf{F}}'_3$	$\hat{\mathbf{F}}'_{11}$	$\hat{\mathbf{F}}'_{31}$
1	**0**	0	0
2	**0.085**	0.440	2.078
3	**0.013**	1.009	2.907
4		0.928	1.564
5		**1.089**	**3.663**
6		**1.326**	1.966
7		0.456	**3.686**
8		0.692	3.264
9		0.854	0.823
10		0.773	3.357
11		**1.342**	1.558
12			3.403
13			2.557
14			1.607
15			2.745
16			1.916
17			3.213
18			2.384
19			3.522
20			2.572
21			1.726
22			3.571
23			1.773
24			**4.306**
25			1.864
26			1.442
27			3.163
28			1.466
29			3.565
30			2.222
31			3.051
Total	0.098	8.909	76.934

Source: Author (2023).

Figure 6 – Comparison between the magnitude of the filter-bank responses of the approximation $\hat{\mathbf{F}}'_3$ and the exact DFT



Source: Author (2023).

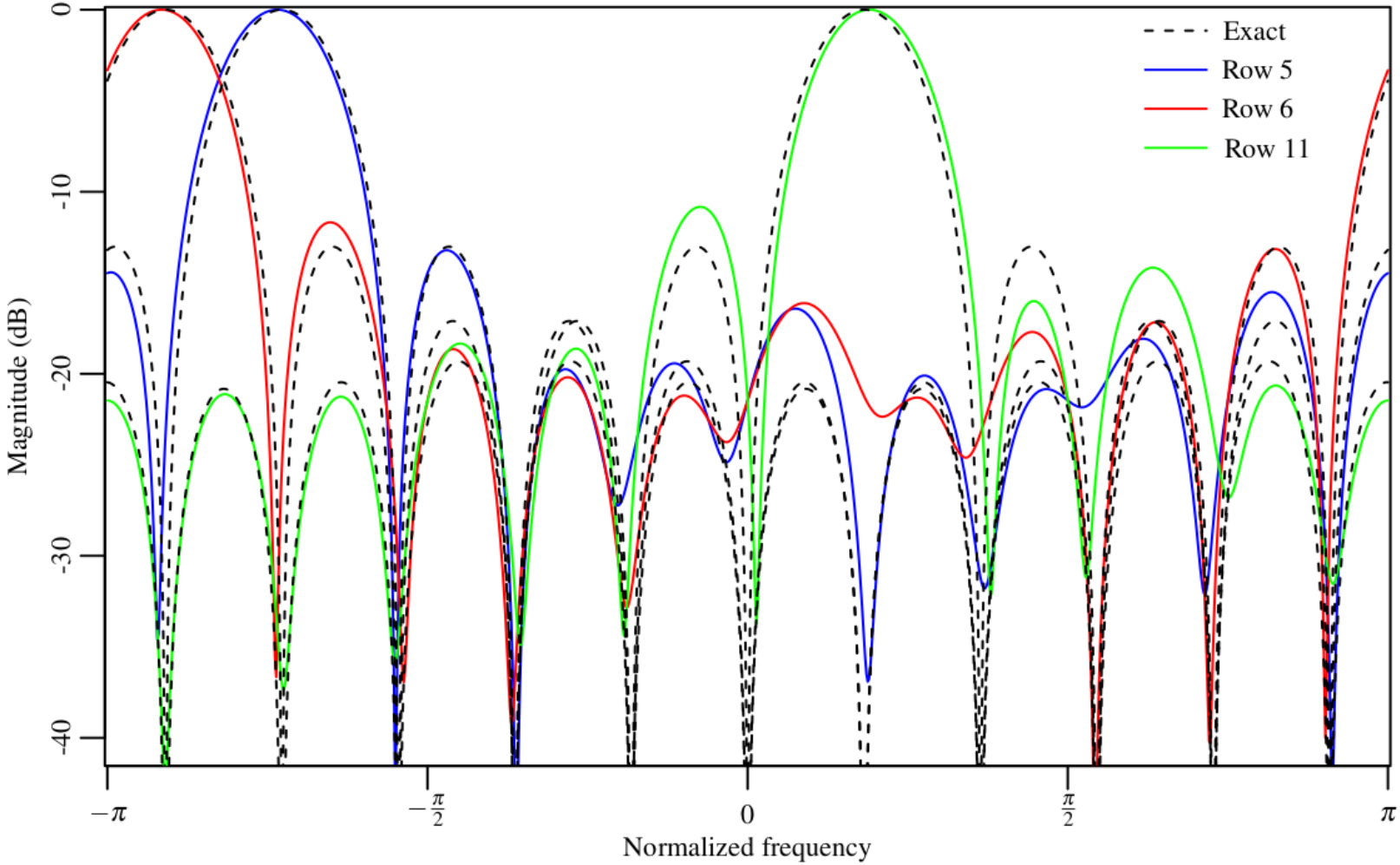
Figure 10. The three least performing rows have filter-bank response close to their exact DFT row in the lobes.

In order to understand the full performance of the approximations, we provide the error between the approximation and the exact DFT, in terms of the filter-bank response for all rows. Figures 11, 12, 13, and 14 show the error energy of all rows for $\hat{\mathbf{F}}'_3$, $\hat{\mathbf{F}}'_{11}$,$\hat{\mathbf{F}}'_{31}$, and $\hat{\mathbf{F}}'_{1023}$, respectively. Although in most of the figures it is not possible to distinguish the individual error of the row, our intention is to show that the error of all approximations are below -17 dB.

4.6 CHAPTER CONCLUSIONS

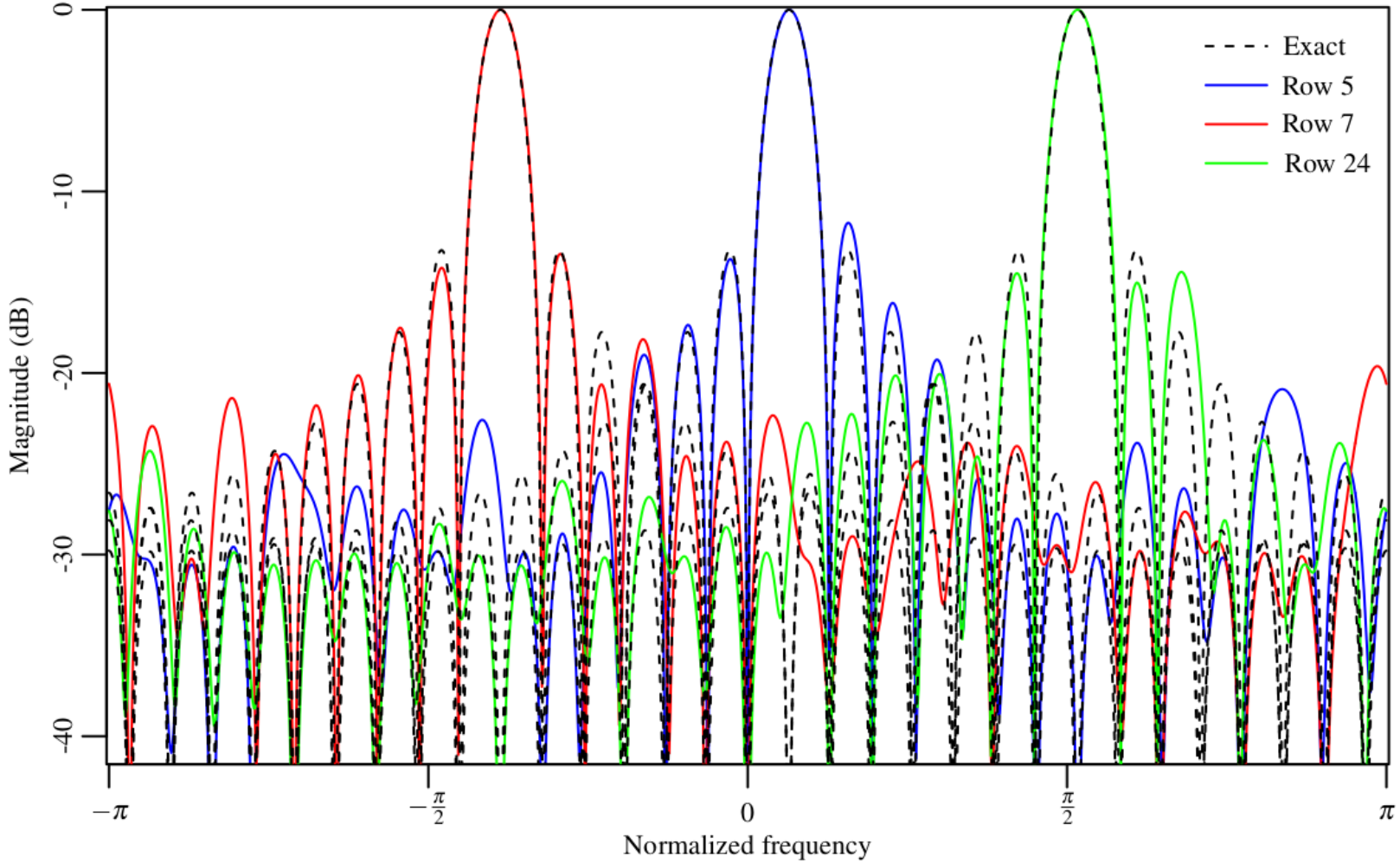
In this chapter, we proposed a method to obtain matrices with null complexity of multiplications to present an approximate version of the prime factor algorithm, the APFA. We demonstrated that if the transform length can be decomposed into relatively prime factors, then the entire computation of the algorithm can be performed without multiplications because no twiddle factors are required. Twiddle factors may emphasize the inaccuracies of approximate ground transforms used in the considered mapping. In addition, their absence contributes to group up and extract the scaling factors reducing

Figure 7 – Comparison between the magnitude of the filter-bank responses of the approximation $\hat{\mathbf{F}}'_{11}$ and the exact DFT for the least performing rows



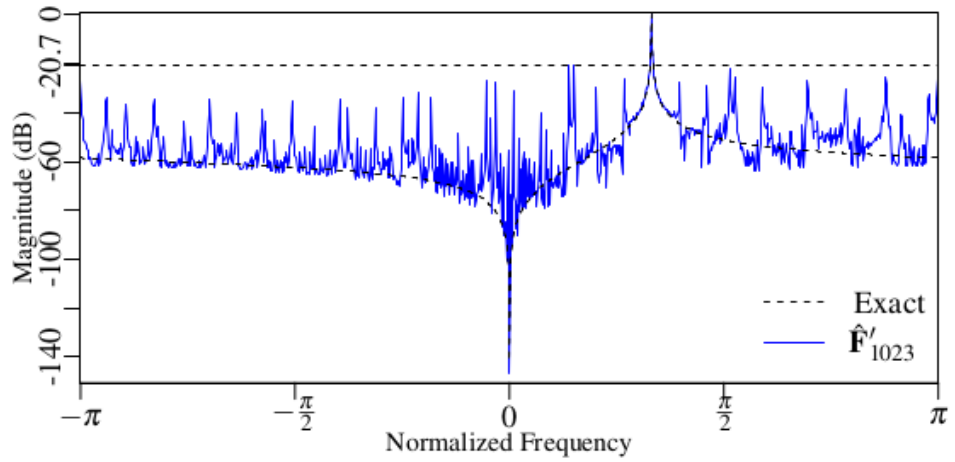
Source: Author (2023).

Figure 8 – Comparison between the magnitude of the filter-bank responses of the approximation $\hat{\mathbf{F}}'_{31}$ and the exact DFT for the least performing rows

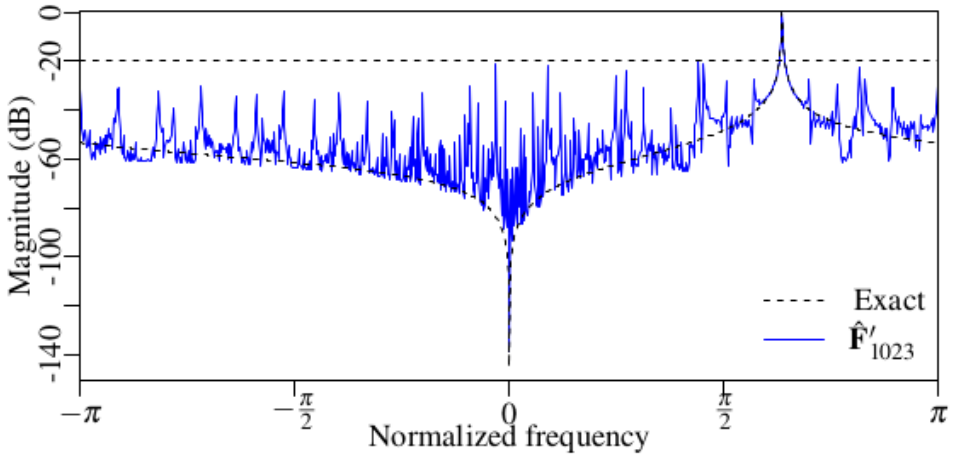


Source: Author (2023).

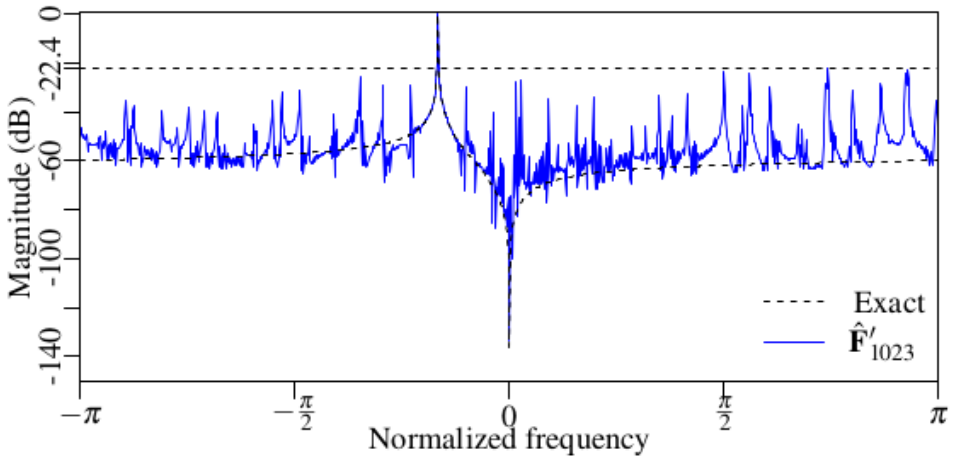
Figure 9 – Comparison between the magnitude of the filter-bank responses of the approximation $\hat{\mathbf{F}}'_{1023}$ and the exact DFT for the least performing rows



a Row 854



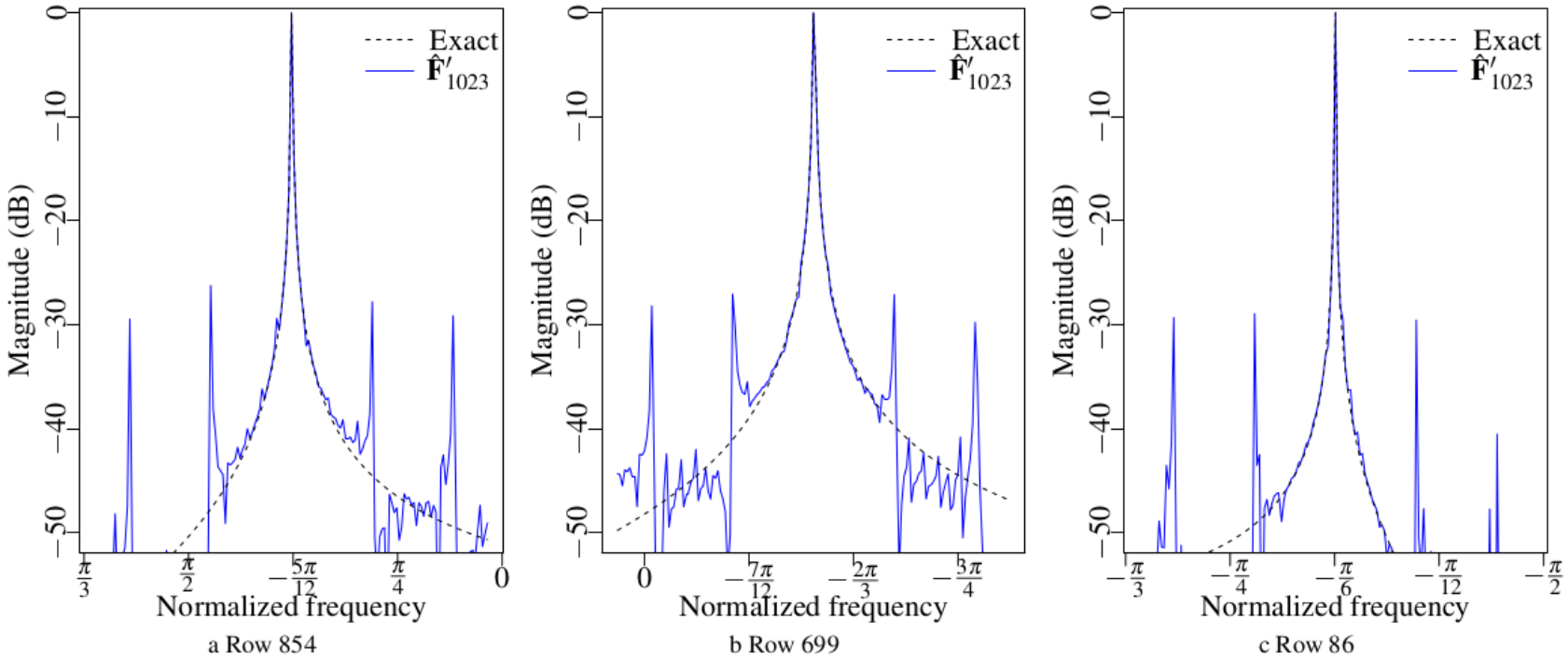
b Row 699



c Row 86

Source: Author (2023).

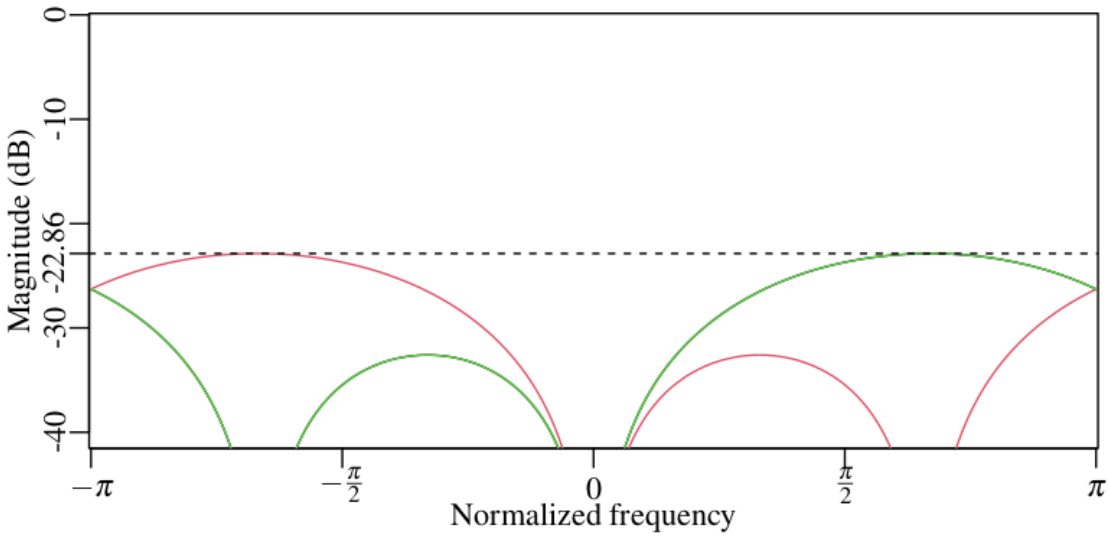
Figure 10 – Comparison between the magnitude of the filter-bank responses of the approximation $\hat{\mathbf{F}}'_{1023}$ and the exact DFT for the least performing rows focusing on the main lobe



a Row 854 b Row 699 c Row 86

Source: Author (2023).

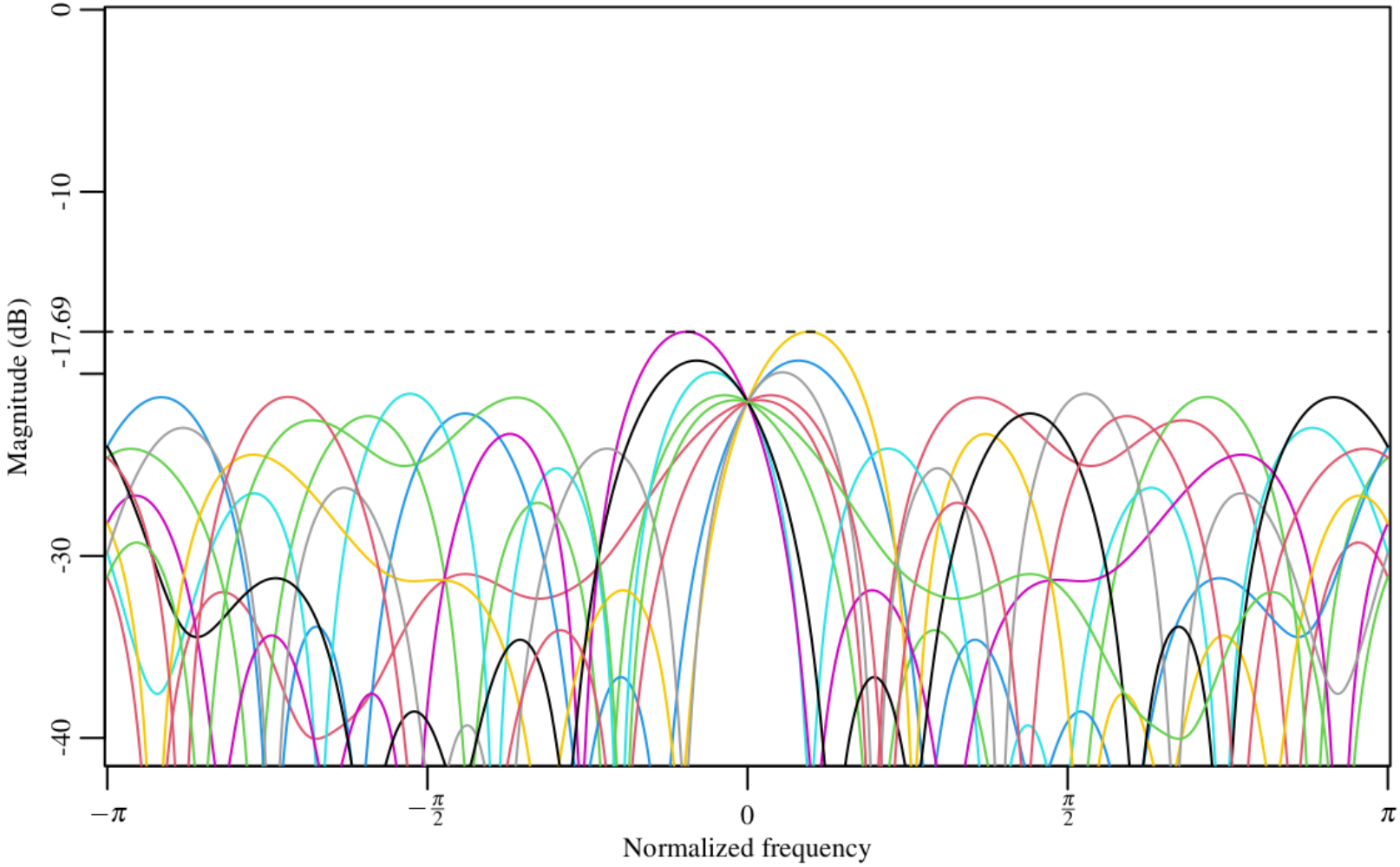
Figure 11 – Magnitude of the filter-bank responses of the error between the approximation $\hat{\mathbf{F}}'_3$ and the exact DFT for all 3 rows



Source: Author (2023).

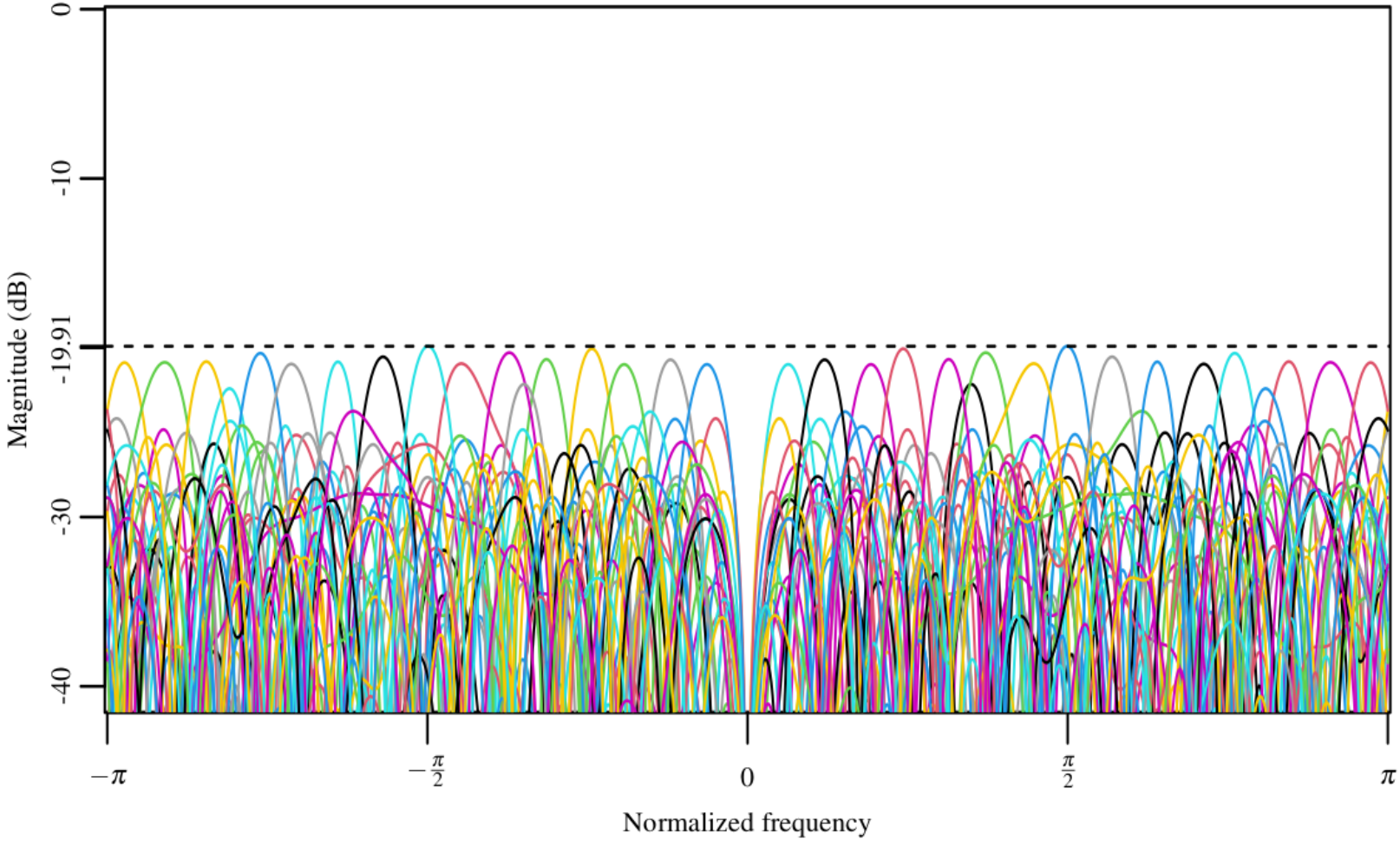
the error of approximate transforms. We applied the proposed method in the 1023-point DFT approximation presenting a collection of approximations with different levels of trade-off between accuracy and computational cost. The APFA performed better than the methods present in the literature according to the investigated figures of merit in addition to preserve the main characteristics of the DFT.

Figure 12 – Magnitude of the filter-bank responses of the error between the approximation $\hat{\mathbf{F}}'_{11}$ and the exact DFT for all 11 rows



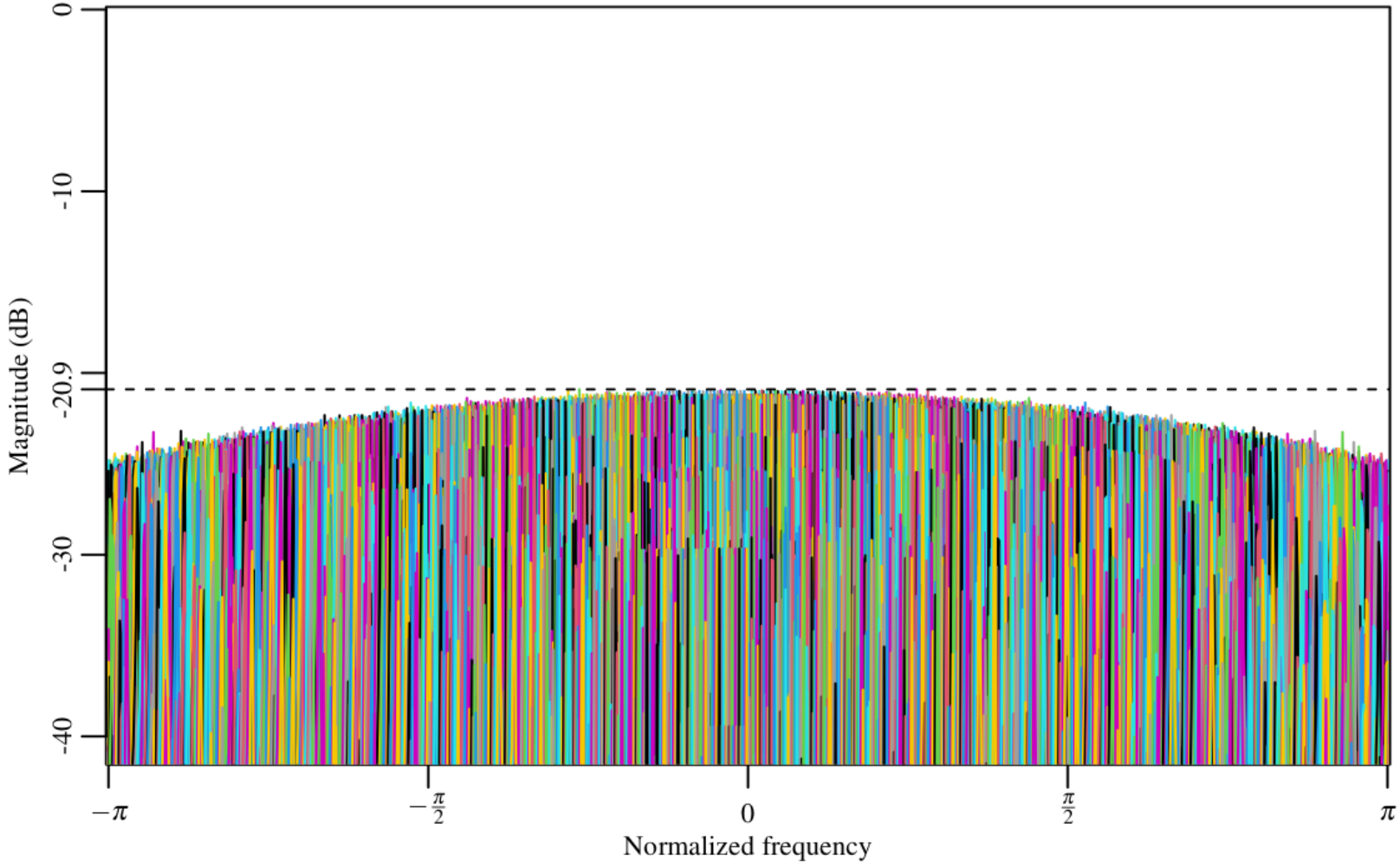
Source: Author (2023).

Figure 13 – Magnitude of the filter-bank responses of the error between the approximation $\hat{\mathbf{F}}_{31}$ and the exact DFT for all 31 rows



Source: Author (2023).

Figure 14 – Magnitude of the filter-bank responses of the error between the approximation $\hat{\mathbf{F}}'_{1023}$ and the exact DFT for all 1023 rows



Source: Author (2023).

5 MULTIPLIERLESS CSD-BASED DFT APPROXIMATIONS

Approximate transform matrices are composed of elements of low arithmetic complexity. When the blocklength and the low-complexity subspace (see Section 4.1.1) are small, all candidate matrices can be evaluated using brute force [155, p. 123]. However, as the blocklength increases, the number of candidate matrices grows rapidly even in a small low-complexity subspace. To address this issue, techniques such as integer functions [65], or make the approximation process an optimization problem by means of objective functions and error measures can be employed to reduce the search space. Regardless of the technique used, it is important to consider reducing computational cost according to the intended application because the approximation may affect the signal representation.

Managing the effects of approximation can become a difficult task when using FFT algorithms. Mapping a transformation matrix into the multiplication of smaller transformation matrices can cause the approximation error to accumulate at each level of the algorithm. Furthermore, when twiddle factors are required to adjust the mapping, the error propagation is aggravated. A set of techniques can be used together or individually to overcome these drawbacks, such as: (i) hybrid algorithms in which only part of the computation is approximated [74]; (ii) algorithms that do not require twiddle factors in the mapping as suggested in Chapter 4; (iii) correction factors (*e.g.*, the expansion factor used in Section 4.1.1). However, even using all these techniques, finding good approximations for the blocks of FFT algorithms is the key component to keep the error under control.

The canonical signed digit (CSD) [184] is a ternary class of the signed digit (SD) [181]. It has representational uniqueness because two nonzero digits are not adjacent [185], and a minimal Hamming weight [186] due to the minimal number of nonzero digits, which leads to a reduction in the number of additions in arithmetic operations [187, 188]. Although a ternary representation can be a problem in DSP as examples the demand for three distinct voltage levels on an electronic board or the need for a constellation with more than two symbols for RF transmission [189], in the context of this work, the CSD values are fixed coefficients, inherent to the algorithm and in practice are directly represented by

arithmetic operations. In this sense, we propose a multiplication-free algorithm based on the traditional Cooley-Tukey algorithm in which the ground transforms and the twiddle factor matrices are approximated by cropping their CSD representations. In particular, we applied this method to the 32-point DFT and the 32×32 twiddle factor matrix in order to obtain a multiplierless approximation for the 1024-point DFT.

5.1 CSD-BASED TRANSFORM APPROXIMATION

In this section, a brief review of the CSD representation and the methodology to obtain approximations from it are presented.

5.1.1 CSD representation

The CSD representation of a number can be obtained from its binary representation [3, p. 433]. In a binary representation, while the rightmost bit is the least significant bit (LSB), the leftmost bit is the most significant bit (MSB). The conversion of a positive decimal number to binary follows [190, p. 37]. Considering for the sake of notation that $\bar{1}$ represent -1, the conversion from binary representation to CSD representation requires that: (i) find a non-zero sequence of bits from LSB to MSB; (ii) replace the leading zero by 1; (iii) replace the last value of the sequence by -1; (iv) replace the intermediate bits by zero (*e.g.* $011 \rightarrow 10\bar{1}$ or $01111 \rightarrow 1000\bar{1}$); and (v) repeat this process until no two consecutive digits are both nonzero. If the number to be converted is infinite, then its decimal representation must be truncated or rounded to prevent the algorithm from entering an infinite loop.

As an example, we detail below how to obtain the CSD representation for 0.8515625.

1. Convert the decimal number to binary:

$$\text{bin}(0.8515625) = 0.1101101,$$

where $\text{bin}(\cdot)$ is a function that converts a decimal number to binary;

2. Find a non-zero sequence of bits from LSB to MSB:

$$0.110\underbrace{11}01;$$

3. Replace the leading zero by 1, the last value of the sequence by -1, and the intermediate bits by zero:

$$0.1110\bar{1}01;$$

4. Repeat Step 2:

$$0.\underbrace{111}0\bar{1}01;$$

5. Repeat Step 3:

$$1.00\bar{1}0\bar{1}01;$$

6. As it is not possible to repeat Step 2 because there are no more adjacent non-zero digits, the CSD representation of 0.8515625 is given by:

$$\text{csd}(\text{bin}(0.8515625)) = 1.00\bar{1}0\bar{1}01,$$

where $\text{csd}(\cdot)$ is a function that converts a binary number to CSD. Algorithm 2 details the $\text{csd}(\cdot)$ function.

5.1.2 Matrix approximation

Let $\mathbf{C}_N$ represent an $N \times N$ complex matrix. The CSD representation of $\mathbf{C}_N$ is obtained by simply computing the CSD representation of each entry of $\mathbf{C}_N$. Therefore, the CSD representation of $\mathbf{C}_N$ is given by:

$$\text{csd}(\text{bin}(\Re(\mathbf{C}_N))) + j \cdot \text{csd}(\text{bin}(\Im(\mathbf{C}_N))), \tag{5.1}$$

where $\Re(\cdot)$ and $\Im(\cdot)$ are functions that returns a matrix with the real and imaginary part of the matrix argument, respectively. The $\text{csd}(\cdot)$ function returns the CSD representation of a binary number if the argument is a single binary representation or a matrix with CSD

Algorithm 2: Pseudo-algorithm for the csd$(\cdot)$ function

Input: C_N; ▷ Enter an N-bit binary number
Output: D_M; ▷ The algorithm returns a M-trit CSD number where $M \geq N$

```
1  D_M ← C_N;
2  counter ← 1;
3  i ← (N − 1);
4  while i ≠ 0 and counter > 1 do
5      if i < 0 then
6          D_{0:(i+counter)} ← 0;
7          D_{i+counter} ← 1̄;
8          D_M ← [1|D_M];        ▷ Concatenate a extra digit when the CSD
                                   representation is larger than the binary
9          counter ← 0;
10     else
11         if C_i ≠ 0 then
12             counter ← counter + 1;
13             i ← i − 1;
14         else
15             if counter > 1 then
16                 D_{i:(i+counter)} ← 0;
17                 D_{i+counter} ← 1̄;
18                 D_i ← 1;
19                 counter ← 0;
20                 i ← N − 1;
21             else
22                 counter ← 0;
23                 i ← i − 1;
24             end
25         end
26     end
27 end
28 return D_M;
```

representations applied element-wise according to (5.1), if the argument is a matrix with binary representations.

The approximation of $\mathbf{C}_N$ is obtained by cropping the CSD representation. The crop is performed according to the number of additions that are allowed in the representation of each number. The restriction of additions refers to the number of non-zero digits in the CSD representation. Therefore, the CSD-based approximation of $\mathbf{C}_N$ is obtained by

$$\hat{\mathbf{C}}_N^{(i)} = \text{crop}(\text{csd}(\text{bin}(\Re(\mathbf{C}_N))), i) + j \cdot \text{crop}(\text{csd}(\text{bin}(\Im(\mathbf{C}_N))), i), \tag{5.2}$$

where $\text{crop}(\cdot, i)$ is a function that trims the CSD representation according to $(i+1)$-nonzero digits which implies an adder-representation [154, p. 221] of i-additions. For example, consider $\cos\left(\frac{7\pi}{16}\right)$. As this number is irrational, its binary representation is infinite, so we have:

$$\text{bin}\left(\cos\left(\frac{7\pi}{16}\right)\right) = 0.0011000111110001011100000111100 0\ldots \tag{5.3}$$

$$\text{csd}\left(\text{bin}\left(\cos\left(\frac{7\pi}{16}\right)\right)\right) = 0.010\bar{1}0010000\bar{1}0010\bar{1}00\bar{1}0000100 0\bar{1}001\ldots. \tag{5.4}$$

In Table 9, we present some approximations for $\cos(\frac{7\pi}{16})$ considering different crops in the CSD representation, their absolute error and the number of additions in each case.

5.2 LOW-COMPLEXITY DFT APPROXIMATIONS

In this section, we present approximations for the 32-point DFT and the 32×32 twiddle factor matrix that are used to obtain approximations for the 1024-point DFT by means of the Cooley-Tukey radix-32. In addition, we provide fast algorithms to reduce the arithmetic cost of the proposed approximations.

Table 9 – Approximations for $\cos\left(\frac{7\pi}{16}\right) \approx 0.19509$ considering different CSD representations

Crop	CSD	Approximation	$\approx \lvert\text{Error}\rvert$	Adds
$\text{crop}\left(\text{csd}\left(\text{bin}\left(\cos\left(\frac{7\pi}{16}\right)\right)\right), 0\right)$	0.01	0.25	0.05491	0
$\text{crop}\left(\text{csd}\left(\text{bin}\left(\cos\left(\frac{7\pi}{16}\right)\right)\right), 1\right)$	$0.010\bar{1}$	0.1875	0.00759	1
$\text{crop}\left(\text{csd}\left(\text{bin}\left(\cos\left(\frac{7\pi}{16}\right)\right)\right), 2\right)$	$0.010\bar{1}001$	0.1953125	0.00022	2
$\text{crop}\left(\text{csd}\left(\text{bin}\left(\cos\left(\frac{7\pi}{16}\right)\right)\right), 3\right)$	$0.010\bar{1}00100000\bar{1}$	0.195068359375	0.00002	3

Source: Author (2023).

Table 10 – Approximation for the DFT elements using only trivial multiplications and bit-shifting operations ($\hat{\mathbf{F}}_{32}^{(0)}$)

Constant	Approximation	|Error|	CSD
$\cos(0) = 1$	1	0	1
$\cos(\frac{\pi}{16}) \approx 0.98079$	1	0.01921	1.00
$\cos(\frac{\pi}{8}) \approx 0.92388$	1	0.07612	1.00
$\cos(\frac{3\pi}{16}) \approx 0.83147$	1	0.16853	1.00
$\cos(\frac{\pi}{4}) \approx 0.70711$	0.5	0.20711	0.10
$\cos(\frac{5\pi}{16}) \approx 0.55557$	0.5	0.05557	0.10
$\cos(\frac{3\pi}{8}) \approx 0.38268$	0.5	0.11732	0.10
$\cos(\frac{7\pi}{16}) \approx 0.19509$	0.25	0.05491	0.01
$\cos(\frac{\pi}{2}) = 0$	0	0	0.00

Source: Author (2023).

5.2.1 Proposed approximations

Based on the method detailed in Section 5.1, we propose two approximations for the 32-point DFT given by:

$$\hat{\mathbf{F}}_{32}^{(0)} = \text{crop}(\text{csd}(\text{bin}(\Re(\mathbf{F}_{32}))), 0) + j \cdot \text{crop}(\text{csd}(\text{bin}(\Im(\mathbf{F}_{32}))), 0) \tag{5.5}$$

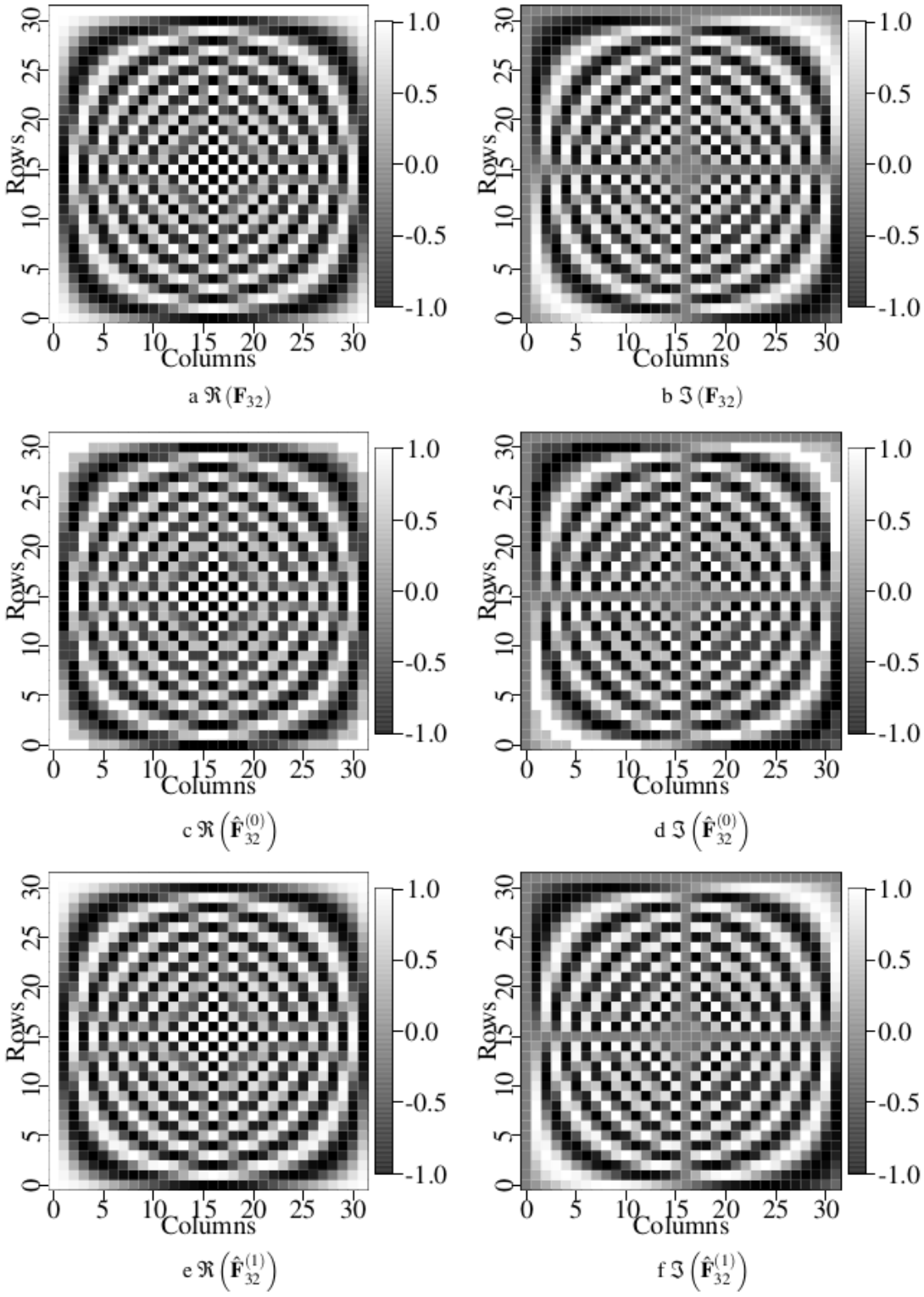
and

$$\hat{\mathbf{F}}_{32}^{(1)} = \text{crop}(\text{csd}(\text{bin}(\Re(\mathbf{F}_{32}))), 1) + j \cdot \text{crop}(\text{csd}(\text{bin}(\Im(\mathbf{F}_{32}))), 1), \tag{5.6}$$

where $\mathbf{F}_{32}$ is the exact 32-point DFT matrix. While $\hat{\mathbf{F}}_{32}^{(0)}$ requires trivial multiplications and bit-shifting operations, in $\hat{\mathbf{F}}_{32}^{(1)}$, the approximation of the constants also allows a maximum of two additions (one for the real part and one for the imaginary).

In absolute terms, the 32-point DFT consists of nine different constants. The approximations $\hat{\mathbf{F}}_{32}^{(0)}$ and $\hat{\mathbf{F}}_{32}^{(1)}$ can be obtained by substituting the DFT matrix values by their approximations according to Tables 10 and 11, respectively. Figure 15 displays a grayscale intensity diagram where each square corresponds to the values of the exact 32-point DFT and its approximations divided into real and imaginary parts. According to Figure 15, the two approximations preserved the DFT symmetries which can be useful for factoring into sparse matrices.

Figure 15 – Grayscale intensity diagram of the exact 32-point DFT and its approximations



Source: Author (2023).

Table 11 – Approximation for the DFT elements using only trivial multiplications, bit-shifting operations, and one addition ($\hat{\mathbf{F}}_{32}^{(1)}$)

Constant	Approximation	|Error|	CSD
$\cos(0) = 1$	1	0	1
$\cos(\frac{\pi}{16}) \approx 0.98079$	0.984375	0.003585	$1.00000\bar{1}$
$\cos(\frac{\pi}{8}) \approx 0.92388$	0.9375	0.01362	$1.000\bar{1}00$
$\cos(\frac{3\pi}{16}) \approx 0.83147$	0.875	0.04353	$1.00\bar{1}00$
$\cos(\frac{\pi}{4}) \approx 0.70711$	0.75	0.04289	$1.0\bar{1}00$
$\cos(\frac{5\pi}{16}) \approx 0.55557$	0.5625	0.00693	0.100100
$\cos(\frac{3\pi}{8}) \approx 0.38268$	0.375	0.00768	$0.10\bar{1}00$
$\cos(\frac{7\pi}{16}) \approx 0.19509$	0.1875	0.00759	$0.010\bar{1}00$
$\cos(\frac{\pi}{2}) = 0$	0	0	0

Source: Author (2023).

In addition, we present an approximation for the 32×32 twiddle matrix which can be obtained by:

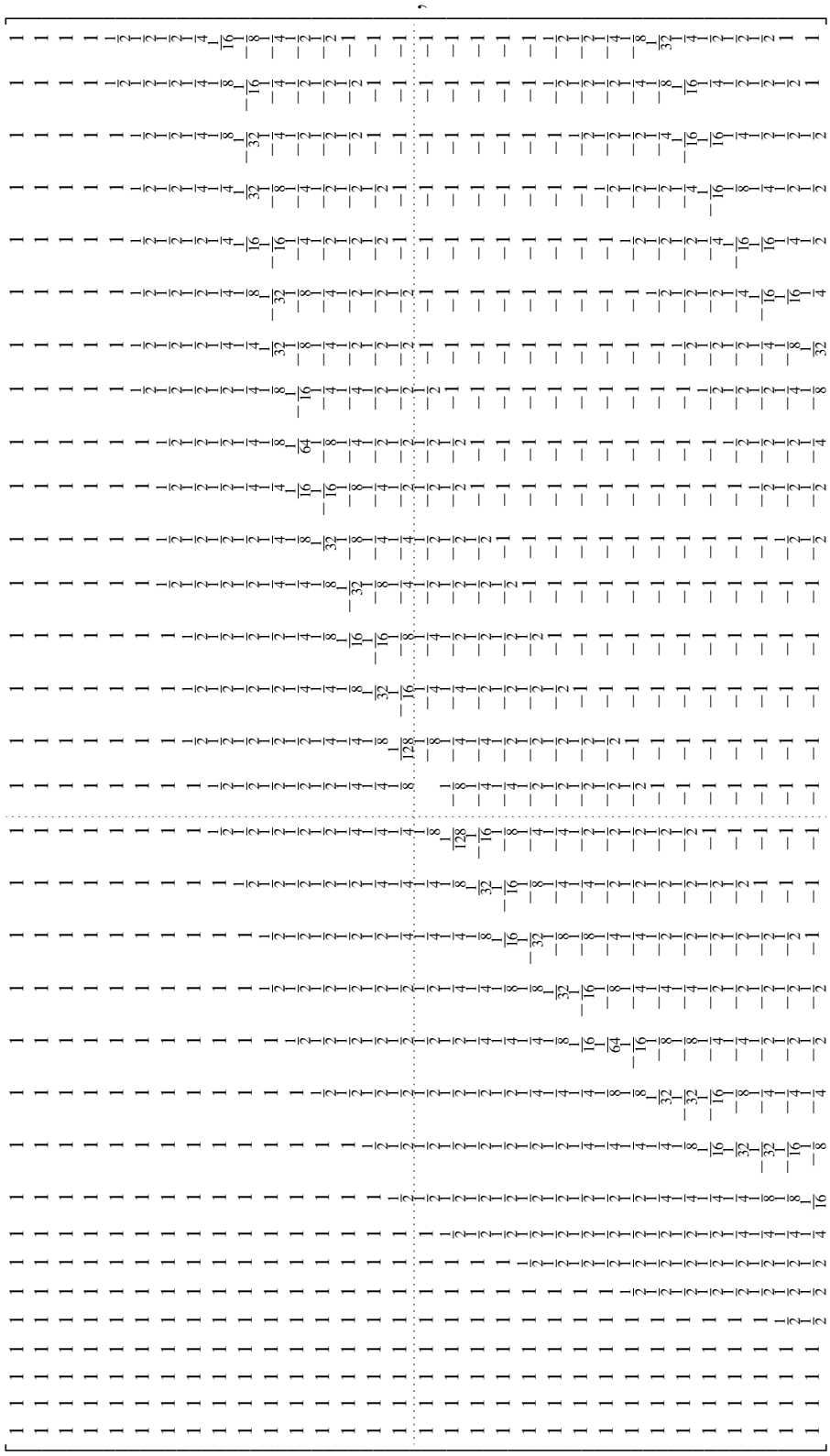
$$\hat{\mathbf{\Omega}}_{32}^{(0)} = \text{crop}(\text{csd}(\text{bin}(\Re(\mathbf{\Omega}_{32}))), 0) + j \cdot \text{crop}(\text{csd}(\text{bin}(\Im(\mathbf{\Omega}_{32}))), 0). \tag{5.7}$$

The twiddle factor matrix does not show any considerable factoring pattern which can be visually inspected in Figure 16. Therefore, we only propose the approximation $\hat{\mathbf{\Omega}}_{32}^{(0)}$ in which trivial multiplication and bit-shifting operations are allowed. The $\hat{\mathbf{\Omega}}_{32}^{(0)}$ is given by

$$\hat{\mathbf{\Omega}}_{32}^{(0)} = \mathbf{W}_1 + j \cdot \mathbf{W}_2, \tag{5.8}$$

where

$$\mathbf{W}_1 =$$



$$\mathbf{W}_2 =$$

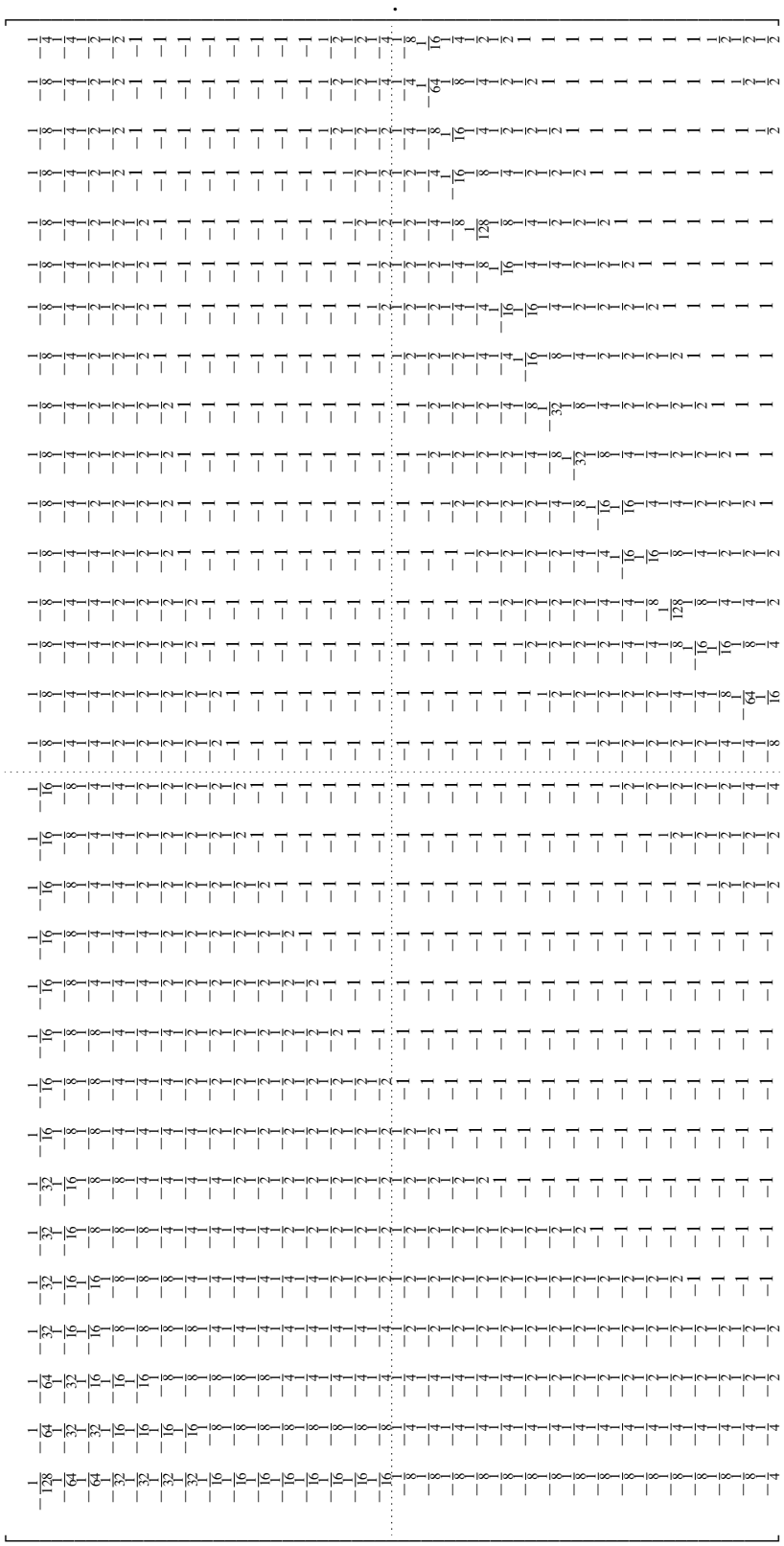
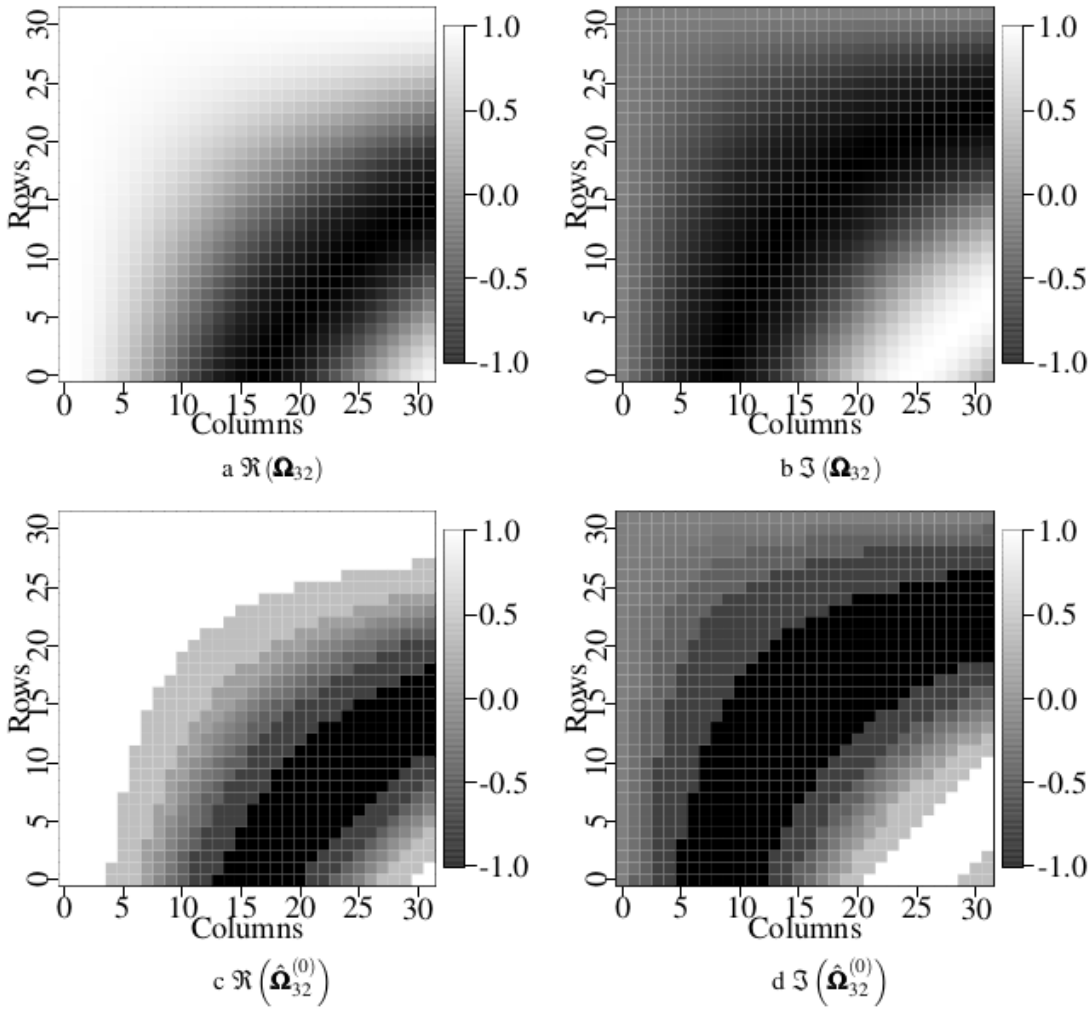


Figure 16 – Grayscale intensity diagram of the 32×32 twiddle factor matrix and its approximations



Source: Author (2023).

Based on the Cooley-Tukey Radix-32 algorithm (2.11), we propose two approximations for the 1024-point DFT as follows:

$$\hat{\mathbf{X}}_1 = \text{vec}\left(\left[\hat{\mathbf{\Omega}}_{32}^{(0)} \circ \left(\hat{\mathbf{F}}_{32}^{(0)} \cdot (\text{invvec}(\mathbf{x}))^\top\right)\right] \cdot \left(\hat{\mathbf{F}}_{32}^{(0)}\right)^\top\right), \tag{5.9}$$

and

$$\hat{\mathbf{X}}_2 = \text{vec}\left(\left[\hat{\mathbf{\Omega}}_{32}^{(0)} \circ \left(\hat{\mathbf{F}}_{32}^{(1)} \cdot (\text{invvec}(\mathbf{x}))^\top\right)\right] \cdot \left(\hat{\mathbf{F}}_{32}^{(1)}\right)^\top\right). \tag{5.10}$$

The approximations obtained by (5.9) and (5.10) are referred to as $\hat{\mathbf{F}}_{1024}^{(0)}$ and $\hat{\mathbf{F}}_{1024}^{(1)}$. Next, we present factorizations for the ground approximations to reduce the additions and bit-shifting operations.

5.2.2 Fast algorithm

The approximations were factored into sparse matrices looking for symmetries between the columns. The approximations $\hat{\mathbf{F}}_{32}^{(0)}$ and $\hat{\mathbf{F}}_{32}^{(1)}$ can be represented as:

$$\hat{\mathbf{F}}_{32}^{(0)} = \mathbf{A}_{10} \cdot \mathbf{A}_{11} \cdot \mathbf{A}_{12} \cdot \mathbf{M}_7 \cdot \mathbf{A}_9 \cdot \mathbf{A}_8 \cdot \mathbf{A}_7 \cdot \mathbf{A}_6 \cdot \mathbf{A}_5 \cdot \mathbf{A}_4,$$

and

$$\hat{\mathbf{F}}_{32}^{(1)} = \mathbf{A}_{10} \cdot \mathbf{A}_{11} \cdot \mathbf{A}_{12} \cdot \mathbf{M}_8 \cdot \mathbf{A}_9 \cdot \mathbf{A}_8 \cdot \mathbf{A}_7 \cdot \mathbf{A}_6 \cdot \mathbf{A}_5 \cdot \mathbf{A}_4,$$

where

$$\mathbf{A}_4 = \begin{bmatrix} 1 & \\ & \mathbf{B}_{31} \end{bmatrix}, \quad \mathbf{A}_5 = \begin{bmatrix} \mathbf{B}_{17} & \\ & \mathbf{B}_{15} \end{bmatrix}, \quad \mathbf{A}_6 = \begin{bmatrix} \mathbf{B}_9 & & & \\ & \mathbf{I}_8 & j \cdot \mathbf{I}_8 & \\ & \mathbf{I}_8 & j \cdot \mathbf{I}_8 & \\ & & & \mathbf{B}_7 \end{bmatrix},$$

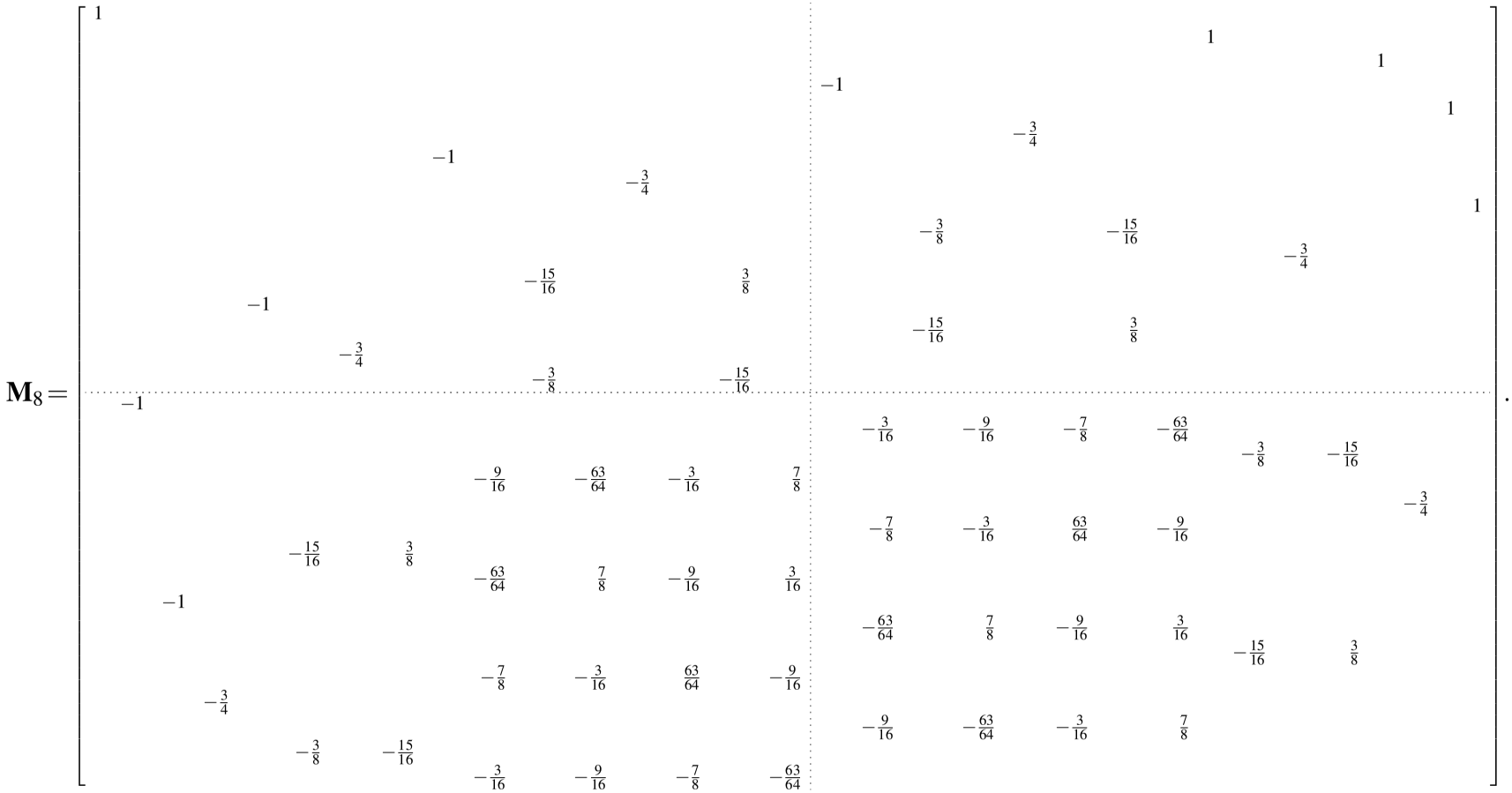
$$\mathbf{A}_7 = \begin{bmatrix} \mathbf{B}_5 & & & \\ & \mathbf{I}_4 & j \cdot \mathbf{I}_4 & \\ & & \mathbf{I}_{16} & \\ & -\mathbf{I}_4 & j \cdot \mathbf{I}_4 & \\ & & & \mathbf{B}_3 \end{bmatrix}, \quad \mathbf{A}_8 = \begin{bmatrix} \mathbf{B}_3 & & & \\ & \mathbf{I}_2 & j \cdot \mathbf{I}_2 & \\ & & \mathbf{I}_{24} & \\ & -\mathbf{I}_2 & j \cdot \mathbf{I}_2 & \\ & & & 1 \end{bmatrix},$$

$$\mathbf{A}_9 = \begin{bmatrix} \mathbf{B}_2 & & \\ & 1 & j \\ & & \mathbf{I}_{28} \\ & -1 & j \end{bmatrix},$$

$$\mathbf{A}_{10} = \begin{bmatrix} 1 & & & & & & \\ & \mathbf{I}_7 & & -\mathbf{I}_7 & & & \\ & & 1 & & & & \\ & & & \mathbf{I}_7 & & -\mathbf{I}_7 & \\ & & & & 1 & & \\ & \mathbf{I}_7 & & & \mathbf{I}_7 & & \\ & & & & & 1 & \\ & & & \mathbf{I}_7 & & & \mathbf{I}_7 \end{bmatrix}, \quad \mathbf{A}_{11} = \begin{bmatrix} 1 & & & & & & \\ & \mathbf{I}_3 & & & -\mathbf{I}_3 & & \\ & & 1 & & & & \\ & & & \mathbf{I}_3 & & & -\mathbf{I}_3 \\ & & & & 1 & & \\ & \mathbf{I}_3 & & & \mathbf{I}_3 & & \\ & & & & & 1 & \\ & & & \mathbf{I}_3 & & & \mathbf{I}_3 \\ & & & & & & \mathbf{I}_{16} \end{bmatrix},$$

$$\mathbf{A}_{12} = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & -1 & & & \\ & & 1 & & & & & \\ & & & 1 & & & -1 & \\ & & & & 1 & & & \\ & 1 & & & 1 & & & \\ & & & & & 1 & & \\ & & & 1 & & & 1 & \\ & & & & & & & \mathbf{I}_{24} \end{bmatrix},$$

$$
\mathbf{M}_7 = \left[\begin{array}{cccccccccccccccc|cccccccccccccccc}
1 &&&&& &&&&& &&&&& &&&&& &&&&& &&&&& & \\
&&&&& &&&&& &&&&& &&&&& &&&& 1 &&&&& && \\
&&&&& &&&&& &&&&& &&&&& &&&&& &&& 1 &&& \\
&&&&& &&&&& &&&&& & -1 &&&&& &&&&& &&&&& \\
&&&&& &&&&& &&&&& &&&&& &&&&& &&&&& 1 & \\
&&&&& &&&&& &&&&& &&&&& -\frac{1}{2} &&&&& &&&&& & \\
&&&&& &&& -1 &&&&& &&&&& &&&&& &&&&& &&& \\
&&&&& &&&&& && -\frac{1}{2} &&&&& &&&&& &&&&& &&&& \\
&&&&& &&&&& &&&&& &&&&& &&&&& &&&&& & 1 \\
&&&&& &&&&& &&&&& &&& -\frac{1}{2} &&&& -1 &&&&& &&&& \\
&&&&& &&&&& &&&&& &&&&& &&&&& & -\frac{1}{2} &&&&& \\
&&&&& &&&&& -1 &&&& \frac{1}{2} &&&&& &&&&& &&&&& && \\
&&&& -1 &&&&& &&&&& &&&&& &&&&& &&&&& && \\
&&&&& &&&&& &&&&& &&& -1 &&&& \frac{1}{2} &&&&& &&&& \\
&&&&& & -\frac{1}{2} &&&&& &&&&& &&&&& &&&&& &&&&& \\
&&&&& &&&&& -\frac{1}{2} &&&& -1 &&&&& &&&&& &&&&& && \\
\hline
& -1 &&&&& &&&&& &&&&& &&&&& &&&&& &&&&& \\
&&&&& &&&&& &&&&& && -\frac{1}{4} && -\frac{1}{2} && -1 && -1 &&&&& &&& \\
&&&&& &&&&& &&&&& &&&&& &&&&& -\frac{1}{2} && -1 &&&& \\
&&&&& &&&& -\frac{1}{2} && -1 && -\frac{1}{4} && 1 &&&&& &&&&& &&&&& & \\
&&&&& &&&&& &&&&& &&&&& &&&&& &&&& -\frac{1}{2} && \\
&&&&& &&&&& &&&&& && -1 && -\frac{1}{4} && 1 && -\frac{1}{2} &&&&& &&& \\
&&&&& -1 && \frac{1}{2} &&&&& &&&&& &&&&& &&&&& &&&& \\
&&&&& &&&& -1 && 1 && -\frac{1}{2} && \frac{1}{4} &&&&& &&&&& &&&&& & \\
&& -1 &&&&& &&&&& &&&&& &&&&& &&&&& &&&& \\
&&&&& &&&&& &&&&& && -1 && 1 && -\frac{1}{2} && \frac{1}{4} &&&&& &&& \\
&&&&& &&&&& &&&&& &&&&& &&&&& -1 && \frac{1}{2} &&&& \\
&&&&& &&&& -1 && -\frac{1}{4} && 1 && -\frac{1}{2} &&&&& &&&&& &&&&& & \\
&&& -\frac{1}{2} &&&&& &&&&& &&&&& &&&&& &&&&& &&& \\
&&&&& &&&&& &&&&& && -\frac{1}{2} && -1 && -\frac{1}{4} && 1 &&&&& &&& \\
&&&&& -\frac{1}{2} && -1 &&&&& &&&&& &&&&& &&&&& &&&& \\
&&&&& &&&& -\frac{1}{4} && -\frac{1}{2} && -1 && -1 &&&&& &&&&& &&&&& &
\end{array}\right],
$$


$\mathbf{M}_8 =$
1
−1
−1
−3/4
−1
−3/4
−15/16
3/8
−3/8
−15/16
−9/16
−63/64
−3/16
7/8
−15/16
3/8
−63/64
7/8
−9/16
3/16
−1
−7/8
−3/16
63/64
−9/16
−3/4
−3/8
−15/16
−3/16
−9/16
−7/8
−63/64
−1
1
1
1
−3/4
1
−3/8
−15/16
−3/4
−15/16
3/8
−3/16
−9/16
−7/8
−63/64
−3/8
−15/16
−3/4
−7/8
−3/16
63/64
−9/16
−63/64
7/8
−9/16
3/16
−15/16
3/8
−9/16
−63/64
−3/16
7/8

5.3 ARITHMETIC COMPLEXITY AND ERROR ANALYSIS

Considering complex inputs, the matrices $\mathbf{A}_4$, $\mathbf{A}_5$, and $\mathbf{A}_6$ require 60 real additions each, while $\mathbf{A}_9$ and $\mathbf{A}_{12}$ need 8 real additions. The matrices $\mathbf{A}_7$, $\mathbf{A}_8$, $\mathbf{A}_{10}$, and $\mathbf{A}_{11}$ require 28, 12, 56, and 24 real additions each, respectively. The matrix $\mathbf{M}_7$ requires 64 real additions and 60 bit-shifting operations. The matrix $\mathbf{M}_8$ requires 172 real additions and 156 bit-shifting operations. Thus, the resulting arithmetic cost of $\hat{\mathbf{F}}_{32}^{(0)}$ is 380 real additions and 60 bit-shifting operations and of $\hat{\mathbf{F}}_{32}^{(1)}$ is 488 real additions and 156 bit-shifting operations. To the best of our knowledge, only the work proposed in [74] is a direct competitor for this block length already published. Thus, in Table 12, we provide a comparison of the arithmetic costs of two proposed ground approximations with the one proposed in [69], $\hat{\mathbf{F}}_{32}$, before and after the factorization for the 32-point DFT.

The twiddle factor matrix approximation ($\hat{\mathbf{\Omega}}_{32}$) requires 4130 real additions and 2076 bit-shifting operations. Therefore, the arithmetic complexity to compute the 1024-point DFT approximation according to (5.9) is: $64 \times 380 + 4130 = 24320$ real additions and $64 \times 60 + 2076 = 5916$ bit-shifting operations. Similarly, the 1024-point DFT approximation according to (5.10) is: $64 \times 488 + 4130 = 35362$ real additions and $64 \times 156 + 2076 = 12060$ bit-shifting operations. In Table 13, the comparison between the proposed approximations and the ones proposed in [74], for the 1024-point DFT, are presented. Notice that the advantage of the proposed algorithms is mainly due to the twiddle factor matrix approximation, since the ground approximations have close amounts of arithmetic operations.

The approximations are also evaluated according to the measures presented in

Table 12 – Comparison of the arithmetic complexity between the proposed approximations with the proposed in [74], before and after the matrix factorization for the 32-point DFT

	Before factorization			After factorization		Reduction (%)	
Approximation	Mult	Adds	Bit-shifting	Adds	Bit-shifting	Adds	Bit-shifting
$\hat{\mathbf{F}}_{32}$	0	2624	0	348	0	86.74	0
$\hat{\mathbf{F}}_{32}^{(0)}$	0	1792	3392	380	60	78.79	98.23
$\hat{\mathbf{F}}_{32}^{(1)}$	0	6208	3840	488	156	92.14	95.94

Source: Author (2023).

Table 13 – Comparison of the arithmetic complexity between the proposed approximations with the proposed in [74] for the 1024-point DFT

Approximation	Arithmetic complexity		
	Mult	Adds	Bit-shifting
$\hat{\mathbf{F}}^{\mathrm{I}}_{1024}$(proposed in [74])	2883	25155	0
$\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$(proposed in [74])	5699	27075	0
$\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$(proposed in [74])	5699	27075	0
$\hat{\mathbf{F}}^{(0)}_{1024}$	0	24320	5916
$\hat{\mathbf{F}}^{(1)}_{1024}$	0	35362	12060

Source: Author (2023).

Table 14 – Comparison of the error measurements between the proposed approximations with the proposed in [74] for the 32-point DFT

Approximation	ε	M	ϕ
$\hat{\mathbf{F}}_{32}$	3.32×10^{2}	8.10×10^{-1}	3.61×10^{-2}
$\hat{\mathbf{F}}^{(0)}_{32}$	8.15×10^{1}	3.73×10^{-1}	2.79×10^{-2}
$\hat{\mathbf{F}}^{(1)}_{32}$	3.23×10^{0}	6.83×10^{-2}	8.68×10^{-4}

Source: Author (2023).

Section 4.1.2. In Table 14, the ground approximations for the 32-point DFT are compared. The proposed approximation $\hat{\mathbf{F}}^{(0)}_{32}$ requires only 32 additions and 60 bit-shifting operations more than $\hat{\mathbf{F}}_{32}$. However, it shows reductions of approximately 75%, 54%, and 23% for the total error energy, MAPE and deviation from orthogonality, respectively. On the other hand, $\hat{\mathbf{F}}^{(1)}_{32}$ requires 108 additions and 156 bit-shifting operations more than $\hat{\mathbf{F}}_{32}$, showing reductions of approximately 99%, 92%, and 98% for the total error energy, MAPE and deviation from orthogonality, respectively.

In Table 15, the approximations for the 1024-point DFT are compared. The proposed methods do not need multiplications and $\hat{\mathbf{F}}^{(0)}_{1024}$ requires even fewer additions than the three proposed methods in [74]. The approximation $\hat{\mathbf{F}}^{(0)}_{1024}$ shows reductions of approximately 30% of the total error energy when compared with $\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$ and $\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$, and increases to 74% when compared with $\hat{\mathbf{F}}^{\mathrm{I}}_{1024}$.

The error reduction is even greater in $\hat{\mathbf{F}}^{(1)}_{1024}$. While it presents reduction of approximately 73% when compared with $\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$ and $\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$, it increases to 90% when compared with $\hat{\mathbf{F}}^{\mathrm{I}}_{1024}$ in terms of total error energy.

Table 15 – Comparison of the error measurements between the proposed approximations with the proposed in [74] for the 1024-point DFT

Approximation	$\varepsilon(\times 10^4)$	$M(\times 10^{-3})$	$\phi(\times 10^{-3})$
$\hat{\mathbf{F}}^{\mathrm{I}}_{1024}$(proposed in [74])	93.00	44.00	69.42
$\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$(proposed in [74])	34.02	25.31	36.07
$\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$(proposed in [74])	34.02	25.31	36.07
$\hat{\mathbf{F}}^{(0)}_{1024}$	23.98	23.76	67.46
$\hat{\mathbf{F}}^{(1)}_{1024}$	9.05	13.61	17.71

Source: Author (2023).

5.3.1 Frequency response

Since the proposed approximations and those proposed in [74] are for the same blocklengths, it is possible to make a direct comparison in terms of the frequency response. The analysis follows the steps described in Section 4.5.1. Firstly, to compare the approximations for the ground transforms of $N = 32$, the error energy of each approximation is detailed by row in Table 16, with the least performing rows highlighted. Secondly, the rows with the highest error energy are subjected to frequency response analysis. Among the rows that had the same error energy values, the choice was made based on the location of the main lobe. The chosen rows were 20, 7, and 24 due to their performances in the approximations $\hat{\mathbf{F}}_{32}$, $\hat{\mathbf{F}}^{(0)}_{32}$, and $\hat{\mathbf{F}}^{(1)}_{32}$, respectively.

Figure 17 shows that between $\frac{\pi}{2}$ and π, $\hat{\mathbf{F}}_{32}$ presents curves far from $\mathbf{F}_{32}$ with some lobes showing magnitude close to the secondary lobes. In $\hat{\mathbf{F}}^{(0)}_{32}$, the effects of the approximation are attenuated, the remaining lobes present visibly smaller curves than the secondary lobes. In $\hat{\mathbf{F}}^{(1)}_{32}$, all lobes are close to the frequency response of the DFT computed by the definition.

In Figure 18, all approximations have primary and secondary lobes close to the $\mathbf{F}_{32}$ frequency response. However, between 0 and $\frac{\pi}{2}$, $\hat{\mathbf{F}}_{32}$ presents a lobe with magnitude greater than the secondary lobes. Such behavior is not observed in the proposed approximations.

The frequency response of the least performing row of $\hat{\mathbf{F}}^{(1)}_{32}$ is shown in Figure 19. It is possible to verify that even on the row with the highest error energy, $\hat{\mathbf{F}}^{(1)}_{32}$ performs better than the other approximations.

Table 16 – Error energy by rows of the ground approximations of the proposed method with the least performing rows highlighted

	Approximations		
Row	$\hat{\mathbf{F}}_{32}$	$\hat{\mathbf{F}}_{32}^{(0)}$	$\hat{\mathbf{F}}_{32}^{(1)}$
1	0.000	0.000	0.000
2	9.813	**4.499**	0.012
3	5.429	3.583	0.039
4	2.981	1.943	0.025
5	8.624	4.312	0.185
6	16.812	3.121	0.107
7	**21.188**	**5.970**	0.111
8	6.631	2.091	0.013
9	0.000	0.000	0.000
10	6.631	2.091	0.013
11	**21.188**	**5.970**	0.111
12	16.812	3.121	0.107
13	8.624	4.312	0.185
14	2.981	1.943	0.025
15	5.429	3.583	0.039
16	9.813	**4.499**	0.012
17	0.000	0.000	0.000
18	15.439	0.393	**0.193**
19	18.500	2.695	0.171
20	**22.271**	2.949	0.181
21	8.624	4.312	0.185
22	8.440	1.771	0.099
23	2.741	0.309	0.099
24	18.621	2.801	**0.193**
25	0.000	0.000	0.000
26	18.621	2.801	**0.193**
27	2.741	0.309	0.099
28	8.440	1.771	0.099
29	8.624	4.312	0.185
30	**22.271**	2.949	0.181
31	18.500	2.695	0.171
32	15.439	0.393	**0.193**
Total	332.228	81.498	3.226

Source: Author (2023).

Table 17 – Details of the error energy measurement for each approximation

Approximations	Total	Mean	Standard deviation	Maximum
$\hat{\mathbf{F}}^{\mathrm{I}}_{1024}$	930006	908.21	682.89	2175.69
$\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$	340205	332.23	362.06	801.66
$\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$	340205	332.23	232.88	708.02
$\hat{\mathbf{F}}^{(0)}_{1024}$	239755	234.13	111.75	551.05
$\hat{\mathbf{F}}^{(1)}_{1024}$	90549	88.43	56.89	232.74

Source: Author (2023).

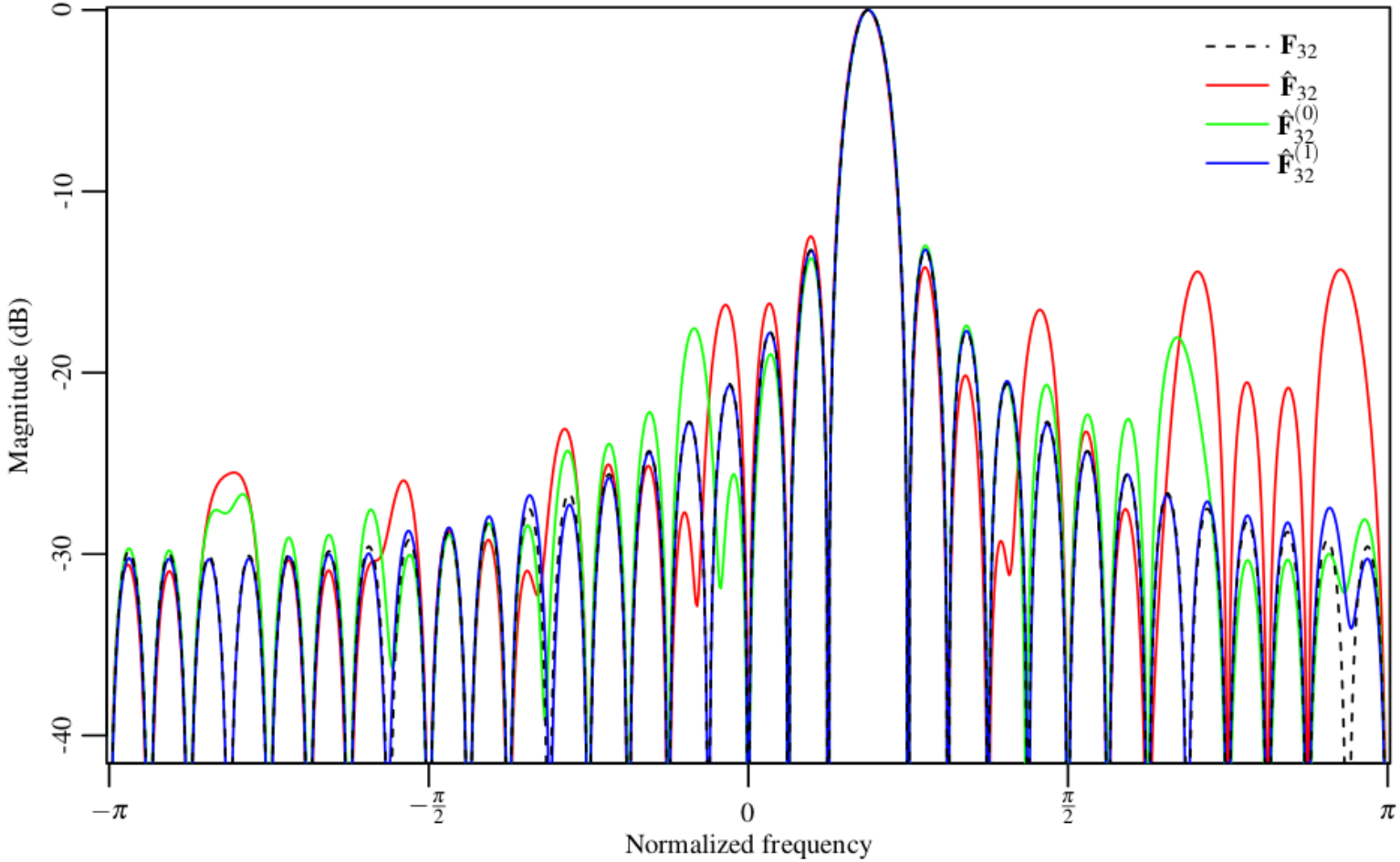
Figure 20 shows the error between the frequency response of the approximations and the DFT computed by the definition. Each color corresponds to one of the 32 rows. Although it is not possible to distinguish the rows, notice that the error is below -10 dB in $\hat{\mathbf{F}}_{32}$. In the proposed approximations, the error is below -16 dB in $\hat{\mathbf{F}}^{(0)}_{32}$ and -30 dB in $\hat{\mathbf{F}}^{(1)}_{32}$.

Due to the number of rows in the approximations for the 1024-point DFT, the error energy detailing by row is omitted. In Table 17, descriptive measures about the error energy are provided. The two proposed approximations present outperform the approximations proposed in [74] in all selected figures of merit.

The rows with the highest error energy of $\hat{\mathbf{F}}^{(0)}_{1024}$ and $\hat{\mathbf{F}}^{(1)}_{1024}$ are 645 and 806, respectively. To perform a worst-case study, the frequency response of the proposed approximation on the row with the highest error energy is computed and compared with the frequency response of the DFT by the definition (referred to as $\mathbf{F}_{1024}$) and the proposals from [74].

In Figure 21 and 22, the worst case of $\hat{\mathbf{F}}^{(0)}_{1024}$ is presented. In all approximations, the main lobe is maintained. Even on the row with the highest error energy, the proposed approximation performed competitively with the methods in the literature with lower computational cost. In Figure 23 and 24, the same experiment is performed for $\hat{\mathbf{F}}^{(1)}_{1024}$ and it has superior performance even on the hybrid approximations ($\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$ and $\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$) in which part of the computation is performed following the DFT definition. The superior performance of the proposed methods is evidenced in Figures 25 and 26 in which the frequency response error between the approximations and the DFT computed by the definition are presented. The approximations proposed in [74] keep the frequency response

Figure 17 – Comparison between the magnitude of the filter-bank responses of the exact DFT and the approximations for row 20



Source: Author (2023).

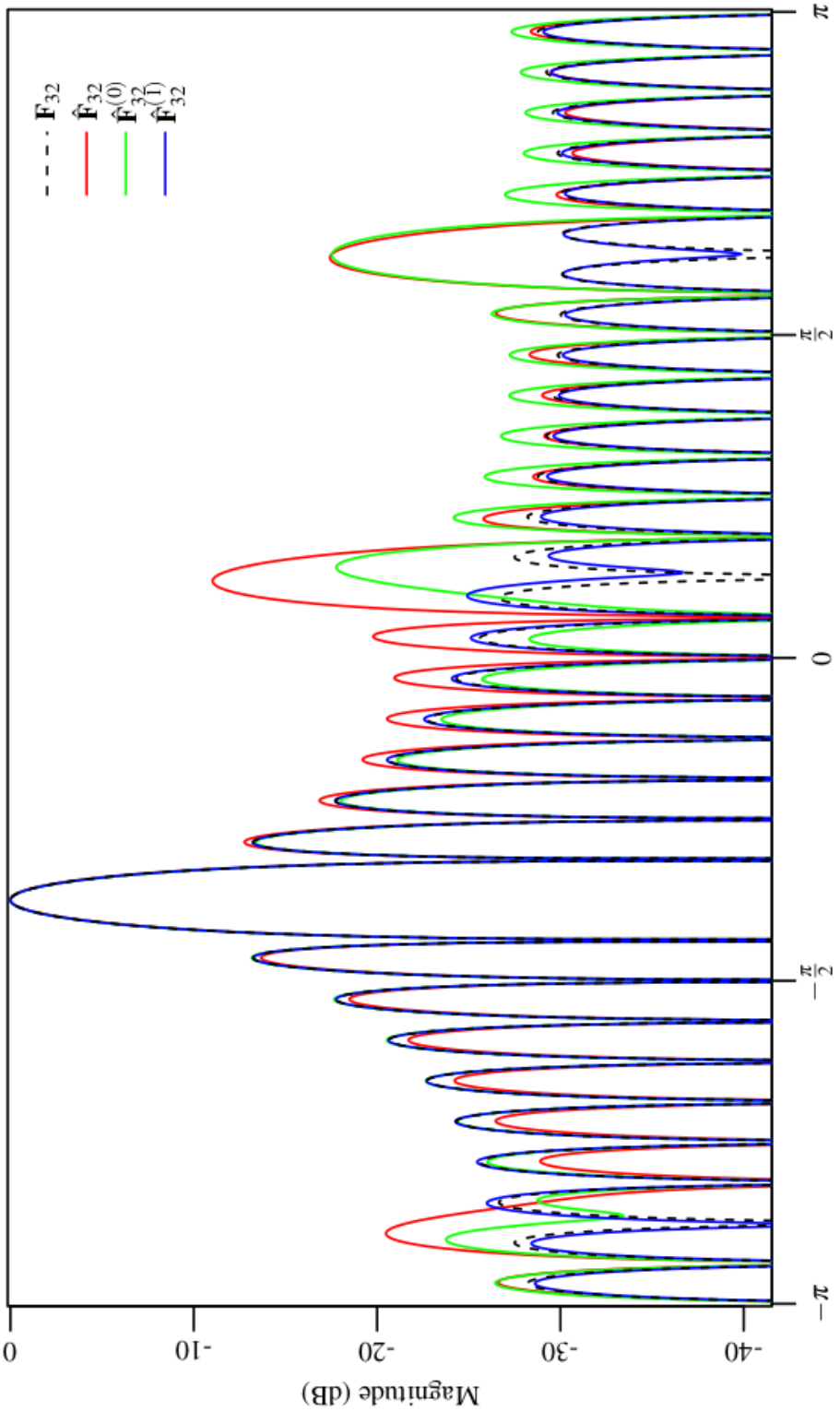
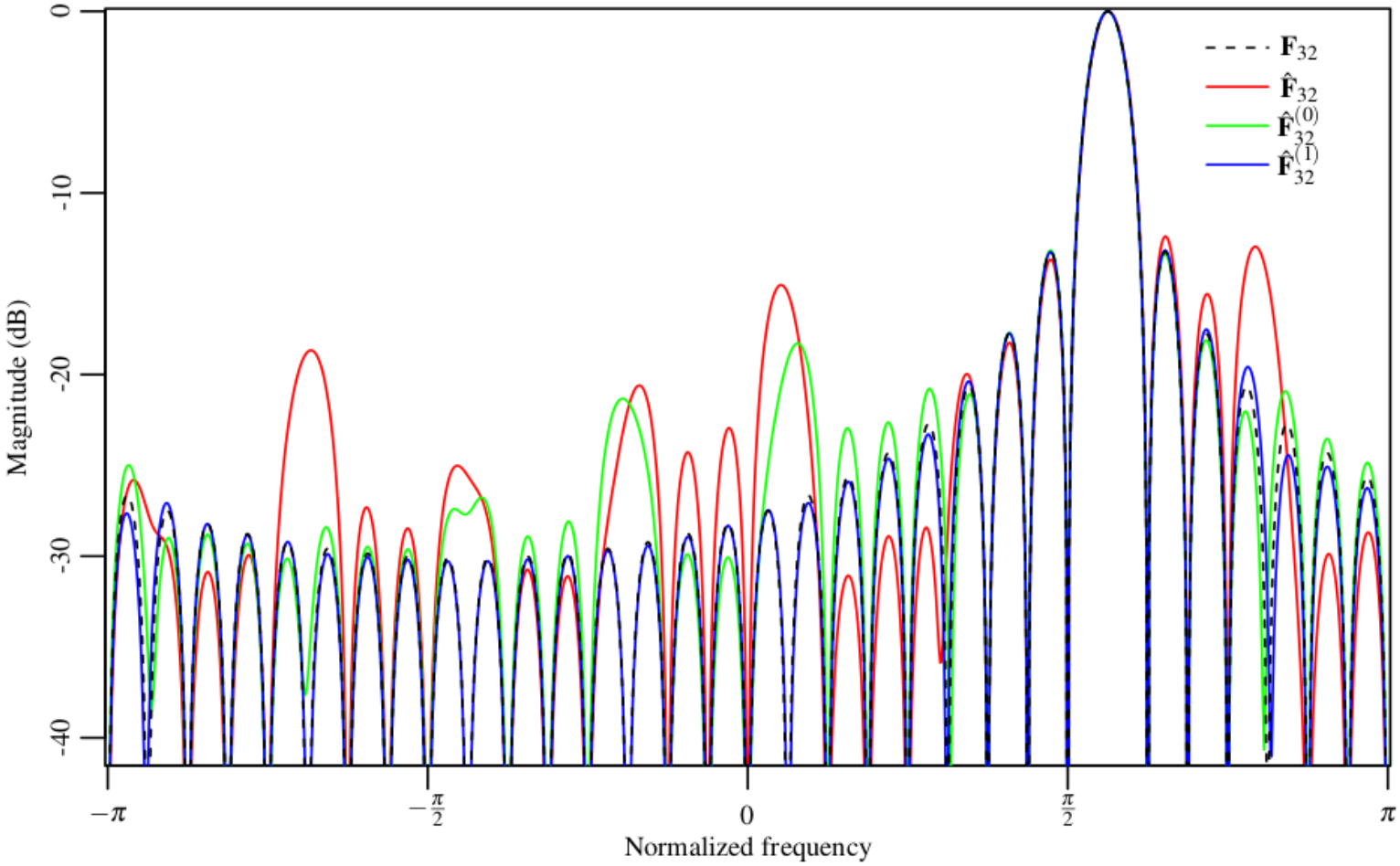


Figure 18 – Comparison between the magnitude of the filter-bank responses of the exact DFT and the approximations for row 7

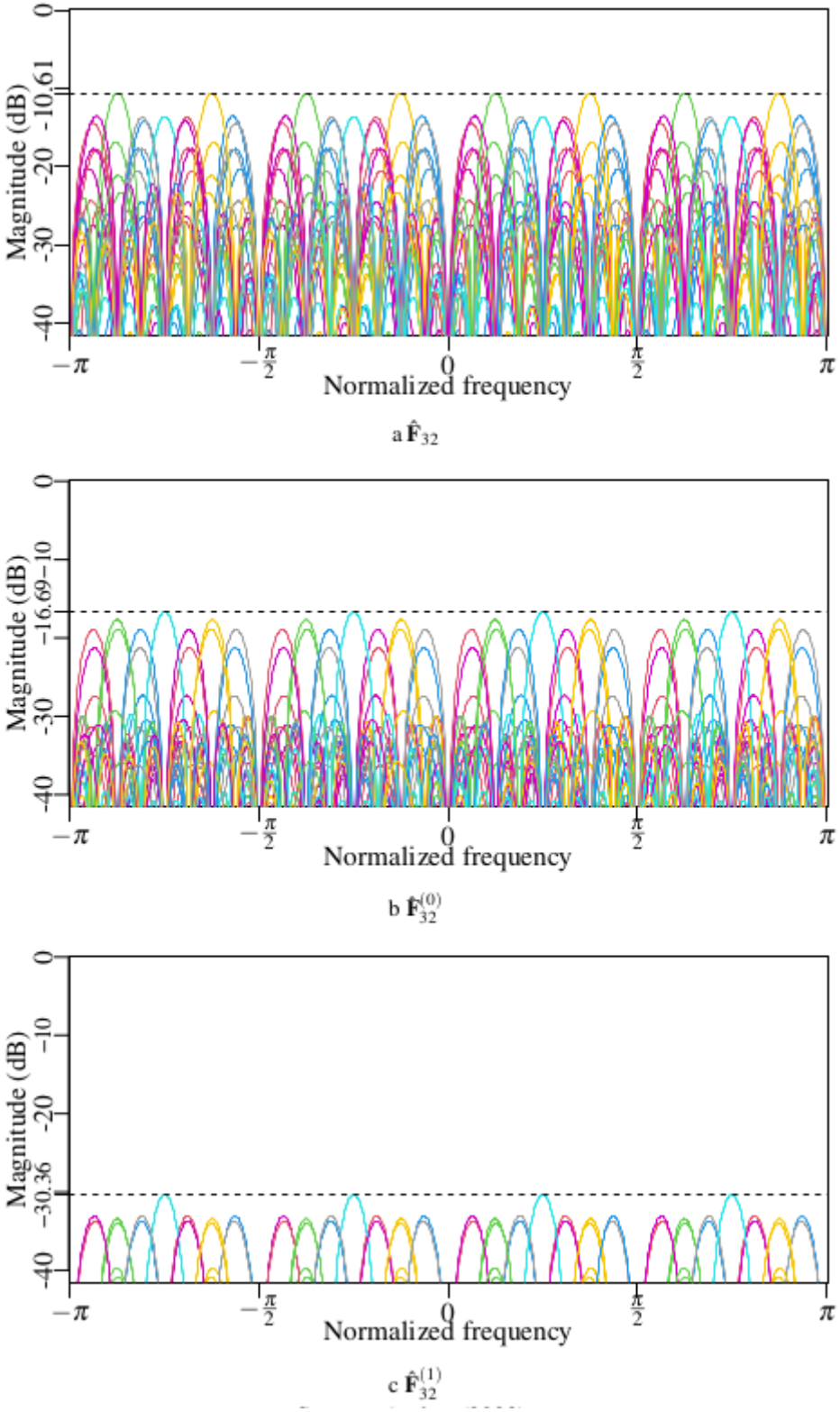
Source: Author (2023).

Figure 19 – Comparison between the magnitude of the filter-bank responses of the exact DFT and the approximations for row 24



Source: Author (2023).

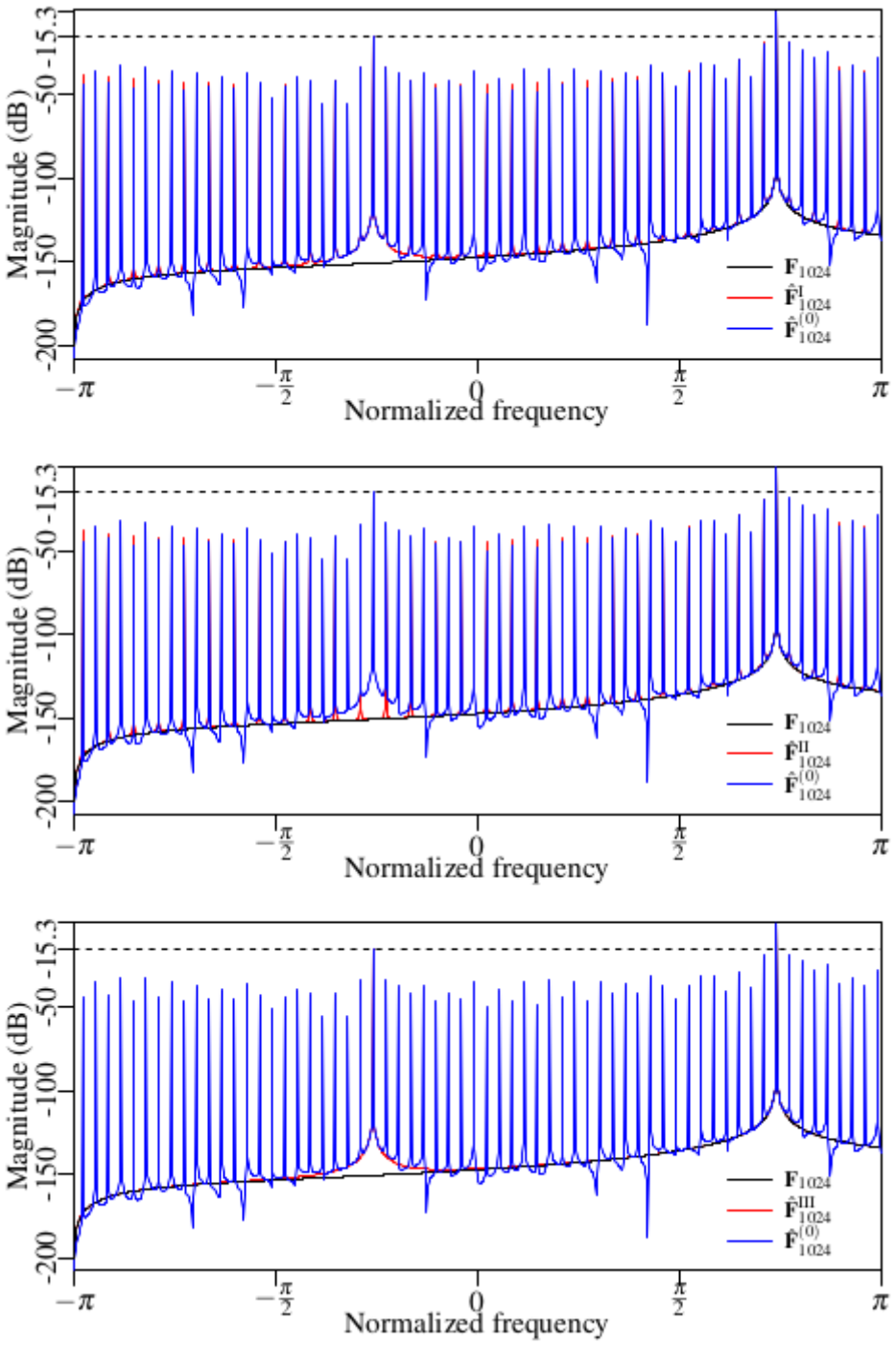
Figure 20 – Magnitude of the filter-bank responses of the error between the approximations and the exact DFT for all 32 rows



Source: Author (2023).

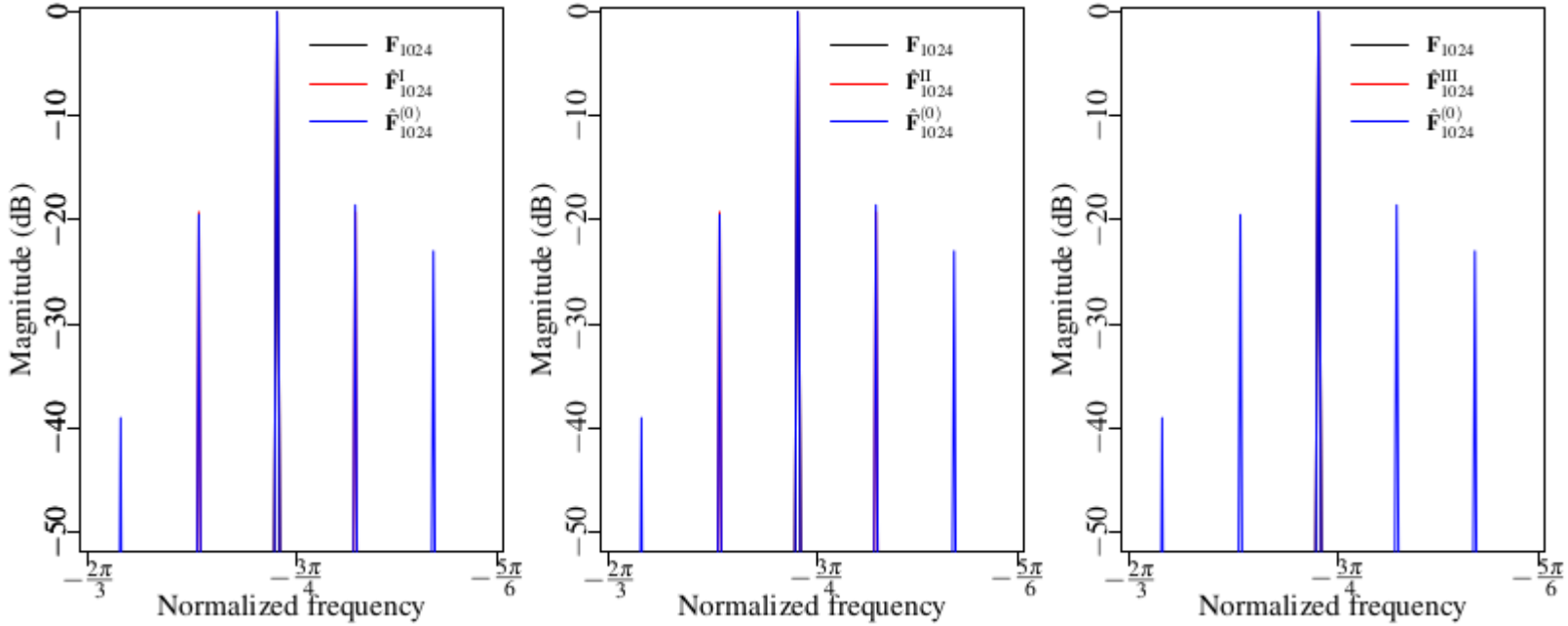
error of the rows below -6.8 dB, -11.52 dB, and -10.61 dB for $\hat{\mathbf{F}}^{\mathrm{I}}_{1024}$, $\hat{\mathbf{F}}^{\mathrm{II}}_{1024}$, and $\hat{\mathbf{F}}^{\mathrm{III}}_{1024}$, respectively. In the proposed methods, the error is reduced to at most -12.15 dB for $\hat{\mathbf{F}}^{(0)}_{1024}$ and -18.76 dB for $\hat{\mathbf{F}}^{(1)}_{1024}$.

Figure 21 – Comparison between the magnitude of the filter-bank responses of the proposed approximation $\hat{\mathbf{F}}_{1024}^{(0)}$ and the three approximations proposed in [74] for the least performing row of the proposed method (row 645)



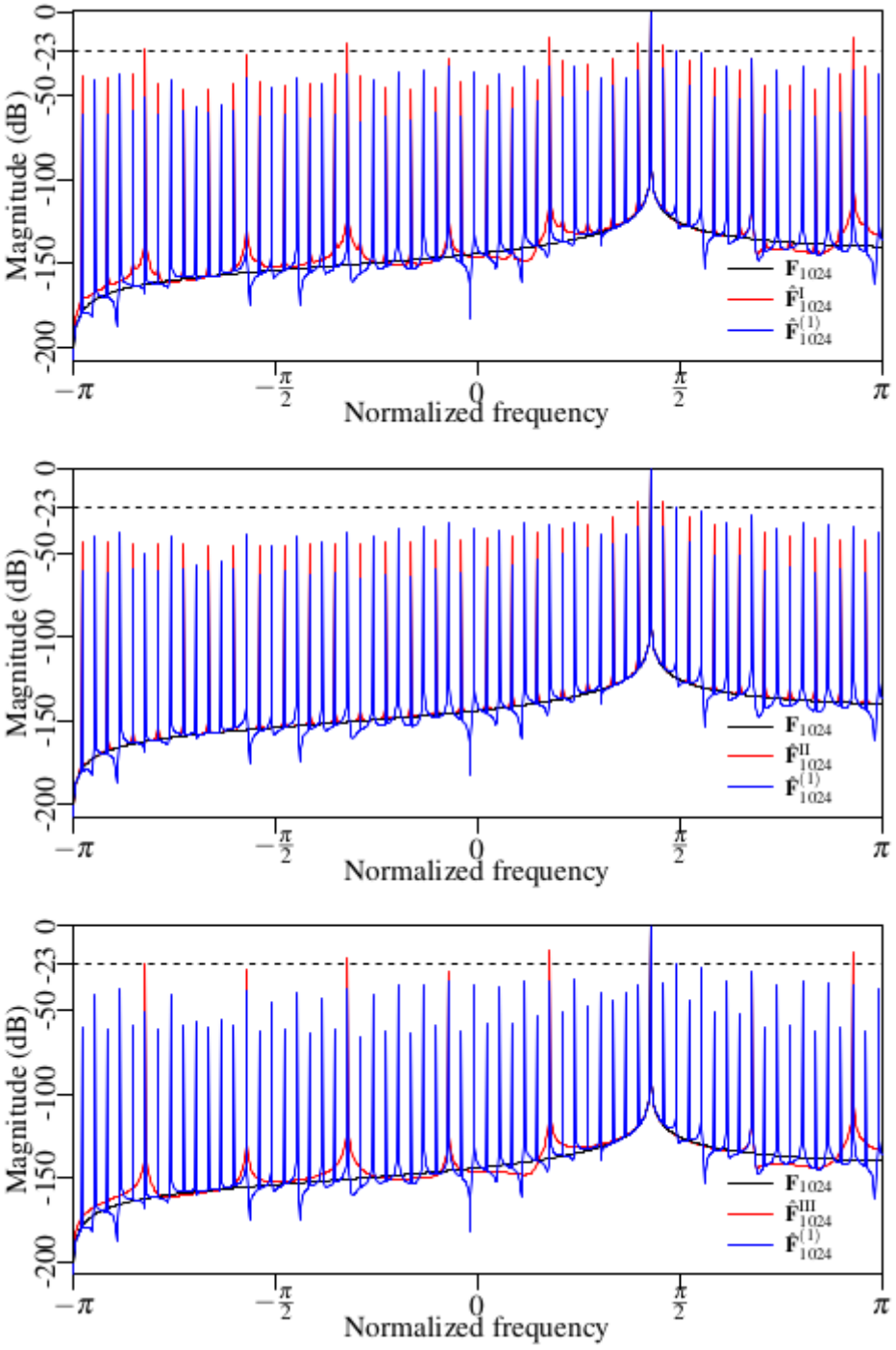
Source: Author (2023).

Figure 22 – Comparison between the magnitude of the filter-bank responses of the proposed approximation $\hat{\mathbf{F}}_{1024}^{(0)}$ and the three approximations proposed in [74] for the least performing row of the proposed method (row 645) focusing on the main lobe



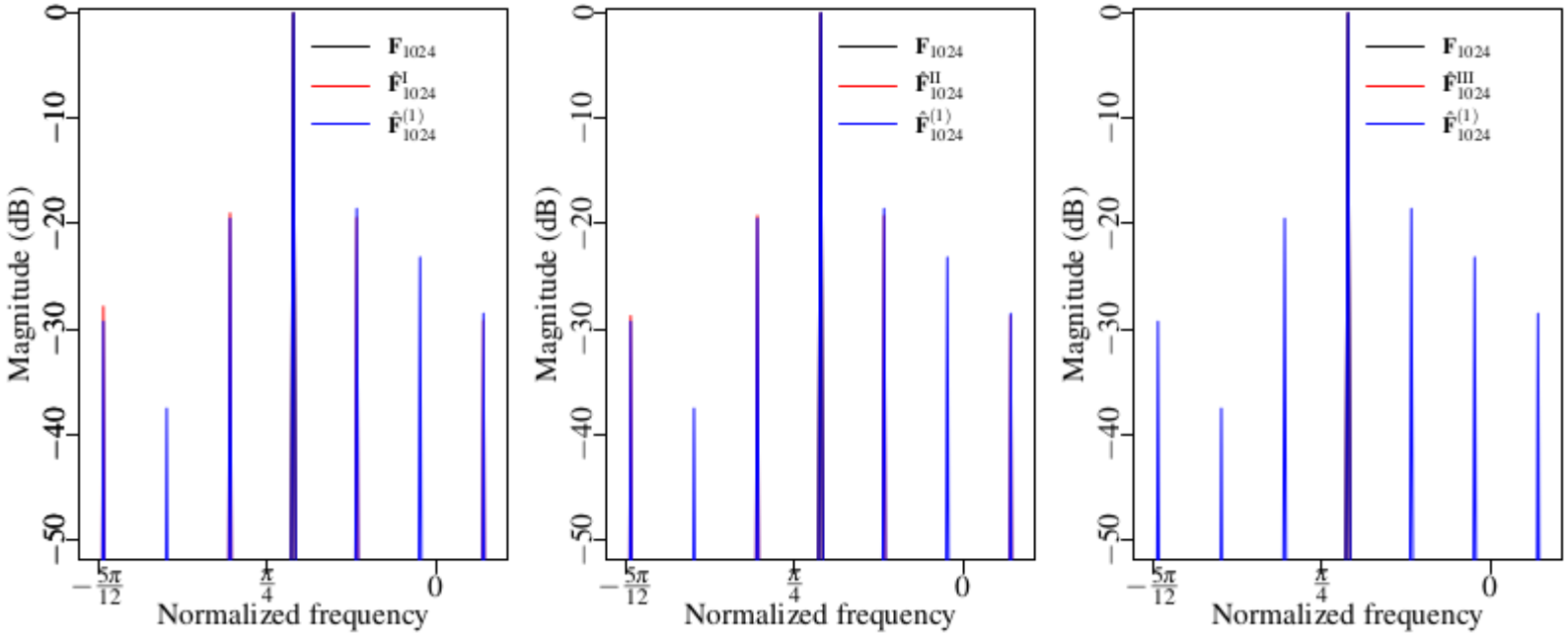
Source: Author (2023).

Figure 23 – Comparison between the magnitude of the filter-bank responses of the proposed approximation $\hat{\mathbf{F}}_{1024}^{(1)}$ and the three approximations proposed in [74] for the least performing row of the proposed method (row 806)



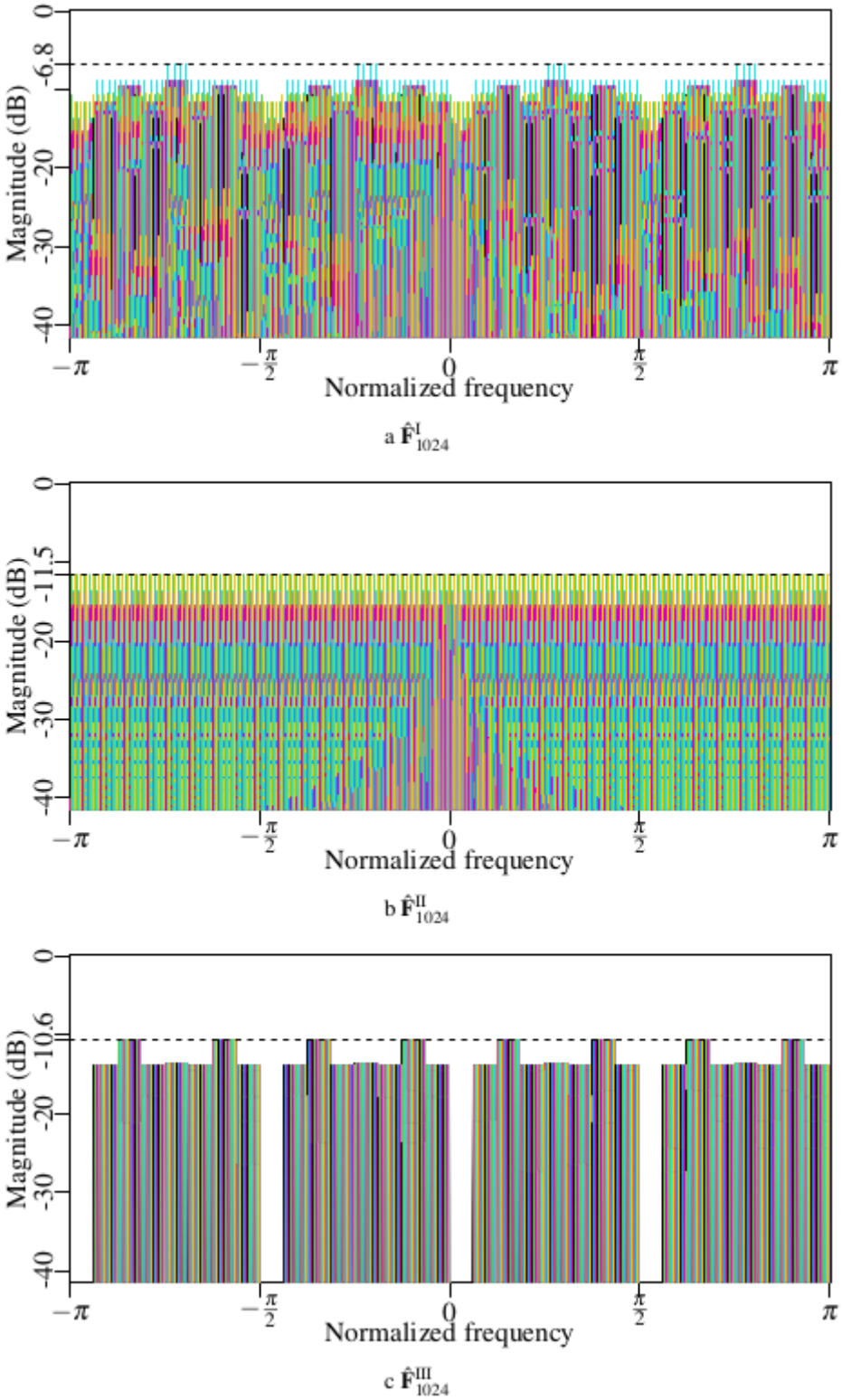
Source: Author (2023).

Figure 24 – Comparison between the magnitude of the filter-bank responses of the proposed approximation $\hat{\mathbf{F}}_{1024}^{(1)}$ and the three approximations proposed in [74] for the least performing row of the proposed method (row 806) focusing on the main lobe



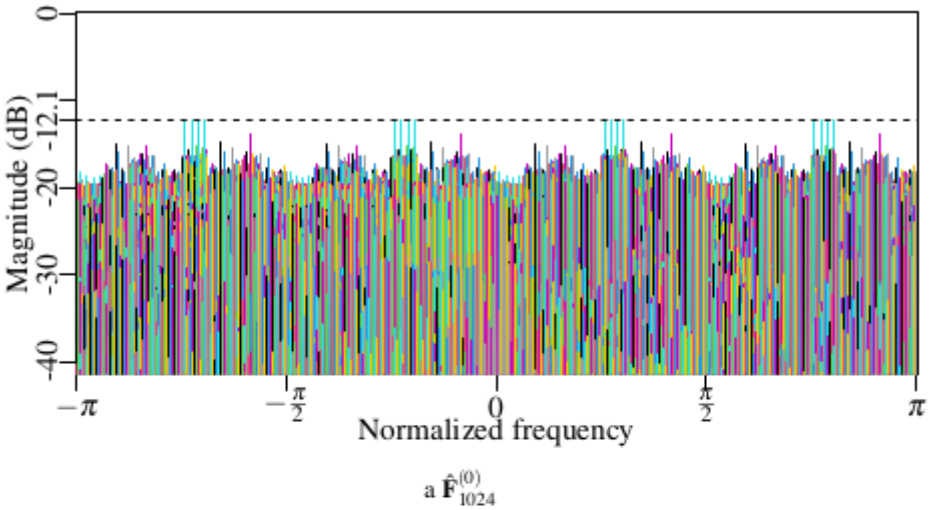
Source: Author (2023).

Figure 25 – Magnitude of the filter-bank responses of the error between the approximations proposed in [74] and the exact DFT for all 1024 rows

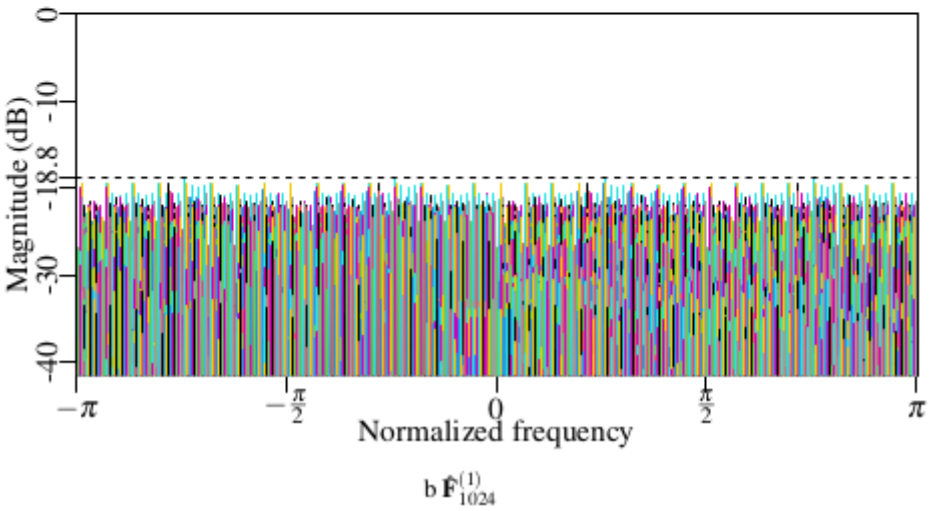


Source: Author (2023).

Figure 26 – Magnitude of the filter-bank responses of the error between the propsed approximations proposed and the exact DFT for all 1024 rows



a $\hat{\mathbf{F}}_{1024}^{(0)}$



b $\hat{\mathbf{F}}_{1024}^{(1)}$

Source: Author (2023).

5.4 CHAPTER CONCLUSIONS

In this chapter, we presented a methodology to obtain multiplierless approximation based on the crop of the CSD representation. The crop is performed according to the number of additions that each constant representation can contain. This method can be applied to both the DFT transform and the twiddle factor matrices. We applied the method to the 32-point DFT and the 32×32 twiddle factor matrix. The obtained approximations were employed to approximately compute the 1024-point DFT by means of Cooley-Tukey radix-32. Besides not requiring multiplications, the proposed algorithms present a smaller error than the approximations in the literature.

6 A LOW-COMPLEXITY AUTOCORRELATION ESTIMATOR

The autocorrelation plays a major role in the statistical signal processing. The information from an autocorrelation estimator is useful in contexts of matched filtering (communications) [191–194], signal modeling [195, 196], template matching (pattern recognition) [197], and noise reduction [198, 199]. The possibility of extracting desirable features of a given signal made the autocorrelation estimator popular in applications of radars [200–205], speech analysis [195, 206–208], biomedical signal processing [197, 209–214], among others.

Although there are numerous ways to estimate the autocorrelation, three are the most popular: (i) by convolution; (ii) by sample autocorrelation function (ACF); (iii) or by DFTs. In the first two methods, the arithmetic complexity may be prohibitively high [19]. On the other hand, according to the Wiener-Kintchine theorem [2], the autocorrelation admits a Fourier representation which can be computed by the DFT and IDFT from an observed signal. Computing the autocorrelation by the DFT and the IDFT enables the use of FFTs which represents a significant reduction of arithmetic operations. If small inaccuracies are tolerated in the application, then the remaining arithmetic complexity can be reduced using approximate computing [215–218].

Motivated by the approximate DFT methodology and based on the Wiener-Kintchine theorem, we propose a low-complexity estimator for the autocorrelatio. The performance of estimators is evaluated in autoregressive processes (AR) [219, p. 54].

.

6.1 DEFINITIONS

Let $\mathbf{x} = \{x[0], \ldots, x[N-1]\}$ represent a N-point discrete-time random process. If $\mathbf{x}$ is wide sense stationary (WSS), the autocorrelation between $x[n]$ and $x[(n+l)]$ separated by l intervals of time referred to time-lag is given by [220, p. 576] [9, p. 415]:

$$\theta[l] = \mathbb{E}\left[x[n] \cdot x[(n+l)]\right], \tag{6.1}$$

where $\mathbb{E}[\cdot]$ is the expected value operator [221, p. 76].

The autocorrelation of lag l can be estimated by [9, p. 45]:

$$\hat{\theta}[l] = \sum_{n=-(N-1)}^{N-1} x[(n+l)] \cdot x[n]. \tag{6.2}$$

Now, consider the following manipulation:

$$\hat{\theta}[l] = \sum_{n=-(N-1)}^{N-1} x[(n+l)] \cdot x[n] \tag{6.3}$$

$$= \sum_{n=-(N-1)}^{N-1} x[(-(-n)+l)] \cdot x[n] \tag{6.4}$$

$$= \sum_{n=-(N-1)}^{N-1} x[-((-n)-l)] \cdot x[n]. \tag{6.5}$$

Notice that (6.5) is the definition of the linear convolution [3, p. 689]. Therefore, the complete autocorrelation $\boldsymbol{\theta} = [\theta[-N+1], \ldots, \theta[0], \ldots, \theta[N-1]]^\top$ can also be estimated by linearly convoluting the signal with its own flipped version as follows:

$$\hat{\boldsymbol{\theta}} = \overleftarrow{\mathbf{x}} * \mathbf{x}, \tag{6.6}$$

where $\hat{\boldsymbol{\theta}} = \left[\hat{\theta}[-N+1], \ldots, \hat{\theta}[0], \ldots, \hat{\theta}[N-1]\right]^\top$, $*$ represents the linear convolution, and $\leftarrow$ provides a reversed version of the vector.

By applying the DFT to the autocorrelation estimator, the periodogram [189, p. 971] is obtained, which is given by:

$$\mathbf{P} = \mathscr{F}(\hat{\boldsymbol{\theta}}), \tag{6.7}$$

where $\mathscr{F}(\cdot)$ denotes the operation of taking the DFT of the argument. Such relationship is derived from the Wiener-Khintchine theorem [135, p. 358] which states that the Fourier transform of the autocorrelation is the power spectrum density (PSD) and, although biased, the periodogram is a consistent estimator for the PSD [2, p. 65].

From (6.6) and (6.7), the periodogram can be rewritten as:

$$\mathbf{P} = \mathscr{F}\left(\overleftarrow{\mathbf{x}} * \mathbf{x}\right). \tag{6.8}$$

The convolution theorem [222] of the DFT states that

$$\mathbf{x} \circledast \mathbf{y} \iff \mathbf{X} \cdot \mathbf{Y}, \tag{6.9}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are the output of the DFT of $\mathbf{x}$ and $\mathbf{y}$, respectively; and $\circledast$ represents the circular convolution [223, p. 515]. However, it is possible to obtain the linear convolution from the circular convolution using zero-padding [3, p. 695]. Zero-padding is a technique used in DSP to extend a signal adding zeros usually to the end of the signal. Therefore, considering an augment input $\mathbf{x}$ of length M (restrict to $M \geq 2 \cdot N - 1$) and the DFT symmetry when the input is real $\left(X[-k] = \overline{X[k]}\right)$ the peridogram can be rewritten as:

$$\mathbf{P} = \mathscr{F}\left(\overleftarrow{\mathbf{x}} * \mathbf{x}\right) \tag{6.10}$$

$$= \overline{\mathbf{X}} \cdot \mathbf{X}. \tag{6.11}$$

In this way, from the previous discussion, we can estimate the autocorrelation by applying the IDFT to the periodogram as follows:

$$\hat{\boldsymbol{\theta}} = \mathscr{F}^{-1}(\mathbf{P}) \tag{6.12}$$

$$= \mathscr{F}^{-1}(\overline{\mathbf{X}} \cdot \mathbf{X}), \tag{6.13}$$

where $\mathscr{F}^{-1}(\cdot)$ denotes the operation of taking the IDFT of the argument. For (6.12) to be accurate, it is necessary to shift the spectrum to start from the zero-component. Otherwise, the DFT symmetry is lost and this may generate an imaginary part in the transform.

Rewritten (6.13) in matrix format, the autocorrelation estimator is given by

$$\hat{\boldsymbol{\theta}} = \frac{1}{M} \cdot \overline{\mathbf{F}_M} \left[(\mathbf{F}_M \cdot \mathbf{x}) \circ (\overline{\mathbf{F}_M} \cdot \mathbf{x}) \right], \tag{6.14}$$

where $\mathbf{F}_M$ is the matrix representation of the M-point DFT and $\circ$ represents the Hadamard product [94, p. 251].

Usually, in statistics, the autocorrelation estimation is standardized to provide values between -1 and 1. This method is known as the sampled autocorrelation function (ACF) [219, p. 30] and it is given by:

$$\hat{\rho}[l] = \frac{\frac{1}{N-1} \sum_{n=0}^{N-1} (x[(n+l)] - \bar{x}) \cdot (x[n] - \bar{x})}{\frac{1}{N-1} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2}, \tag{6.15}$$

$$= \frac{\sum_{n=0}^{N-1} (x[(n+l)] - \bar{x}) \cdot (x[n] - \bar{x})}{\sum_{n=0}^{N-1} (x[n] - \bar{x})^2}, \tag{6.16}$$

where $\bar{x}$ is the sample mean of $\mathbf{x}$. Such standardization can also be obtained when the autocorrelation is computed by convolution or the DFT. However, this procedure can lead to problems in the construction of fast algorithms for (6.14) because the computation of $\hat{\boldsymbol{\theta}}$ becomes data-dependent. The mentioned problem can be easily overcome by a previous standardization of the raw data as follows:

$$z[n] = \frac{x[n] - \bar{x}}{\sqrt{\sum_{n=0}^{N-1}(x[n] - \bar{x})^2}}. \tag{6.17}$$

Thus, we have the standardized signal vector $\mathbf{z} = [z[0], \ldots, z[N-1]]^\top$. The standardization in (6.17) is a variation of the z-score transformation [224, p. 83]. Employing (6.17) in (6.16), we have:

$$\hat{\boldsymbol{\rho}}[l] = \sum_{n=0}^{N-1} z[(n+l)] \cdot z[n] \tag{6.18}$$

$$\hat{\boldsymbol{\rho}}[l] = \hat{\theta}[l]. \tag{6.19}$$

Therefore, for the standardized signal, we have

$$\hat{\boldsymbol{\rho}} = \hat{\boldsymbol{\theta}}, \tag{6.20}$$

where $\hat{\boldsymbol{\rho}} = [\hat{\rho}[-N+1], \ldots, \hat{\rho}[0], \ldots, \hat{\rho}[N-1]]^\top$. The advantage of standardizing the data is that, in addition to computing the autocorrelation, we can also directly estimate the parameters of autoregressive processes [225, p. 25]. If $\mathbf{z}$ is derived from a p-autoregressive process (AR(p)) [226, p. 58] then it satisfies

$$z[n] = \phi[1] \cdot z[n-1] + \phi[2] \cdot z[n-2] + \cdots + \phi[p] \cdot z[n-p] + \varepsilon[n], \tag{6.21}$$

where $\varepsilon[n]$ is white noise [226, p. 47]. For the autoregressive process to be stationary, the roots of the following polynomial must lie outside the unit circle:

$$1 - \phi[1] \cdot z[n-1] - \phi[2] \cdot z[n-2] - \cdots - \phi[p] \cdot z[n-p] - \varepsilon[n] = 0. \tag{6.22}$$

By means of the Yule-Walker equations [226, p. 59], we obtain the true autocorrelation of each lag l, given by:

$$\rho[l] = \phi[1] \cdot \rho[l-1] + \phi[2] \cdot \rho[l-2] + \cdots + \phi[p] \cdot \rho[l-p]. \tag{6.23}$$

Thefore, knowing the values of ϕ, we can obtain the values of ρ by (6.21) and vice versa.

6.2 PROPOSED METHOD

Based on (6.14), we propose to approximately compute the autocorrelation estimator as follows:

$$\hat{\boldsymbol{\rho}}_M^{\star} = \frac{1}{M} \cdot \overline{\hat{\mathbf{F}}_M} \left[(\hat{\mathbf{F}}_M \cdot \mathbf{z}_{\text{aug}}) \circ (\overline{\hat{\mathbf{F}}_M} \cdot \mathbf{z}_{\text{aug}}) \right], \tag{6.24}$$

where $\mathbf{z}_{\text{aug}}$ is the standardized augmented signal with $M-N$ zeros added to the end of the vector, $\overline{\hat{\mathbf{F}}_M}$ and $\hat{\mathbf{F}}_M$ are low-complexity approximations of $\overline{\mathbf{F}_M}$ and $\mathbf{F}_M$, respectively.

The kernel of the $\hat{\boldsymbol{\rho}}_M^{\star}$ computation relies on the matrices $\mathbf{F}_M$ and $\overline{\hat{\mathbf{F}}_M}$. Therefore, the performance and complexity of the estimator are dependent on the choice of an approximate DFT matrix. Once $(\hat{\mathbf{F}}_M \cdot \mathbf{z}_{\text{aug}})$ is computed, its result can be used to obtain $(\overline{\hat{\mathbf{F}}_M} \cdot \mathbf{z}_{\text{aug}})$ by simply changing the sign of the imaginary part. If M is power of two or can be decomposed into relatively prime factors, then FFT algorithms as Cooley-Tukey or Good Thomas can be used, respectively.

6.2.1 Approximate autocorrelation estimator for $N = 512$

In this section, we apply the presented methodology in real inputs of length equal to $N = 512$ from AR processes. First, the input is standardize according to (6.17). Second, the input is augmented by zero-padding it to $M = 1024$. In this way, the low-complexity autocorrelation estimator is given by:

$$\hat{\boldsymbol{\rho}}_{1024}^{\star} = \frac{1}{1024} \cdot \overline{\hat{\mathbf{F}}_{1024}} \left[(\hat{\mathbf{F}}_{1024} \cdot \mathbf{z}_{\text{aug}}) \circ (\overline{\hat{\mathbf{F}}_{1024}} \cdot \mathbf{z}_{\text{aug}}) \right]. \tag{6.25}$$

In (6.25), M is power of two, then we applied the Cooley-Tukey algorithm using the approximations proposed in Chapter 5. Applying (5.9) and (5.10) in (6.25), we have the following two low-complexity autocorrelation estimators:

$$\hat{\boldsymbol{\rho}}_{1024,\text{I}}^{\star} = \frac{1}{1024} \cdot \text{vec} \left(\left[\hat{\boldsymbol{\Omega}}_{32} \circ \left(\hat{\mathbf{F}}_{32}^{(0)} \cdot \widehat{\mathbf{P}}_1 \right) \right] \cdot \left(\hat{\mathbf{F}}_{32}^{(0)} \right)^{\top} \right), \tag{6.26}$$

and

$$\hat{\boldsymbol{\rho}}_{1024,\text{II}}^{\star} = \frac{1}{1024} \cdot \text{vec} \left(\left[\hat{\boldsymbol{\Omega}}_{32} \circ \left(\hat{\mathbf{F}}_{32}^{(1)} \cdot \widehat{\mathbf{P}}_2 \right) \right] \cdot \left(\hat{\mathbf{F}}_{32}^{(1)} \right)^{\top} \right), \tag{6.27}$$

where

$$\widehat{\mathbf{P}}_1 = \left(\left[\hat{\mathbf{\Omega}}_{32} \circ \left(\hat{\mathbf{F}}_{32}^{(0)} \cdot (\text{invvec}(\mathbf{z}_{\text{aug}}))^\top \right) \right] \cdot \left(\hat{\mathbf{F}}_{32}^{(0)} \right)^\top \right) \circ \left(\left[\hat{\mathbf{\Omega}}_{32} \circ \left(\hat{\mathbf{F}}_{32}^{(0)} \cdot (\text{invvec}(\mathbf{z}_{\text{aug}}))^\top \right) \right] \cdot \left(\hat{\mathbf{F}}_{32}^{(0)} \right)^\top \right), \tag{6.28}$$

and

$$\widehat{\mathbf{P}}_2 = \left(\left[\hat{\mathbf{\Omega}}_{32} \circ \left(\hat{\mathbf{F}}_{32}^{(1)} \cdot (\text{invvec}(\mathbf{z}_{\text{aug}}))^\top \right) \right] \cdot \left(\hat{\mathbf{F}}_{32}^{(1)} \right)^\top \right) \circ \left(\left[\hat{\mathbf{\Omega}}_{32} \circ \left(\hat{\mathbf{F}}_{32}^{(1)} \cdot (\text{invvec}(\mathbf{z}_{\text{aug}}))^\top \right) \right] \cdot \left(\hat{\mathbf{F}}_{32}^{(1)} \right)^\top \right). \tag{6.29}$$

The autocorrelation estimated by $\hat{\boldsymbol{\rho}}^{\star}_{1024,\text{I}}$ and $\hat{\boldsymbol{\rho}}^{\star}_{1024,\text{II}}$ are referred to as Method I and Method II, respectively. The approximate autocorrelation estimators may not inherit the DFT properties, and therefore may contain an imaginary part. In this case, only the real part of the approximate estimator should be considered.

6.3 ARITHMETIC COMPLEXITY

The computation of the usual autocorrelation by convolution and sampled ACF are in the order of $\mathscr{O}(N^2)$ and $\mathscr{O}((N-1)^2)$, respectively, in terms of real multiplications and real additions [21, p. 146]. The proposed methods use the factorizations presented in Chapter 5. Although the computation of the approximate DFT and IDFT in the proposed methods does not require multiplications, the estimators are not free of multiplication because of the Hadamard product. The proposed methods require M complex multiplications which translates into $3 \cdot M$ real multiplications.

Based on Table 13, for $N = 512$, the arithmetic costs of the proposed Method I are: 24320 (approximate DFT) + 24320 (approximate IDFT) + 3072 (approximate **P**) =51712 real additions; and 5916 (approximate DFT) + 6940 (approximate IDFT) =12856 bit-shifting operations. For the proposed Method II, we have a total of: 35362 (approximate DFT) + 35362 (approximate IDFT) + 3072 (approximate **P**) =73796 real additions; and 12060 (approximate DFT) + 13084 (approximate IDFT) =25144 bit-shifting operations. In Table 18, the arithmetic complexities of the estimators are summarized. Compared to convolution-based and sampled ACF estimators, the Methods I and II require

approximately 99% less real multiplications. In terms of real additions, the reduction is 80% and 72% for Method I and II, respectively.

Table 18 – Arithmetic complexity of the autocorrelation estimators for $N = 512$

Method	Multiplications	Additions	Bit-shifting
Convolution and ACF	262144	261121	0
Proposed method I	3072	51712	12856
Proposed method II	3072	73796	25144

Source: Author (2023).

6.4 ERROR ANALYSIS

The estimators were evaluated by mean of Monte Carlo simulation. We considered $R = 1000$ replicates of the AR process according to five scenarios as shown in Table 19. The white noise $\varepsilon_n \sim \mathcal{N}(0, 0.61)$ so 90% of its values are between -1 and 1 [227].

The parameters of Scenarios I, II, and III were purposely chosen to investigate the behavior of estimators in weak, strong negatively and strong positively correlated data, respectively. With regard to Scenarios IV and V, the choices were made based on the number of parameters and the position of the polynomial roots in which it is expected that the closer they are to the unit circle, the worse should be the estimator performance since the process is close to non-stationarity.

Once the sampled ACF, convolution and direct Fourier transforms provide the same autocorrelation estimates, we grouped them together and call them traditional methods. Since we have access to the parameters of the AR processes, we calculate the true autocorrelation (by Yule-Walker equations) and compare with average estimates for the 1000 replicates of each method to determine the bias of the estimators. For each scenario, we provide the average of the estimators for the first 15 lags and their biases. Furthermore, all 511 lags are presented to detail the behavior of the estimators.

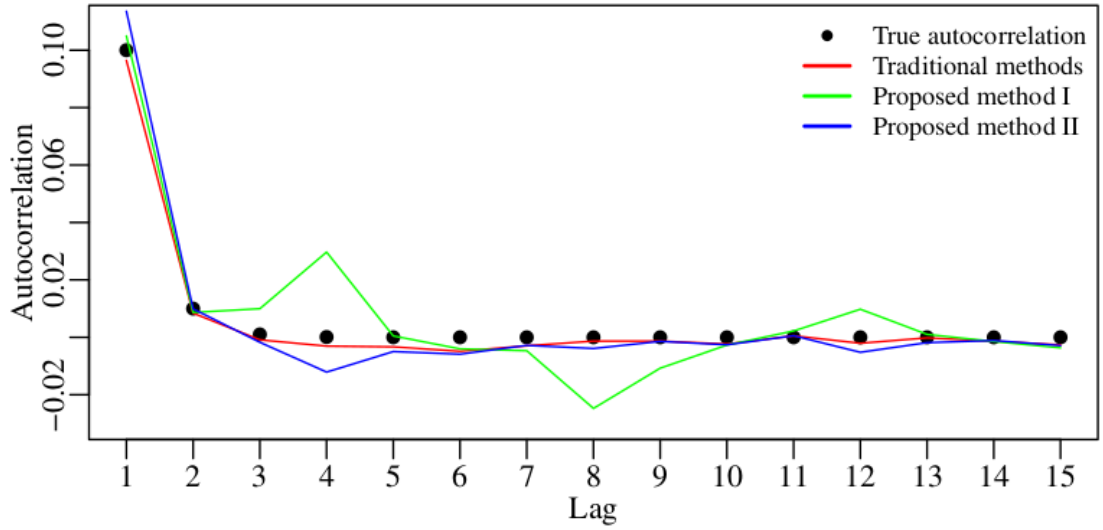
According to Figure 27, the proposed estimators have behavior close to traditional methods. The proposed Method I presents greater variability than Method II. In Figure 28, we can see that while the bias of Method I is smaller than 0.03, Method II has a bias

Table 19 – AR scenarios

Scenarios	AR parameters ϕ_1	ϕ_2	ϕ_3
I	0.1	-	-
III	-0.9	-	-
II	0.9	-	-
IV	0.4	0.5	-
V	-1.7	-1.7	-0.9

Source: Author (2023).

Figure 27 – Comparison between proposed and traditional methods considering the first 15 lags for Scenario I



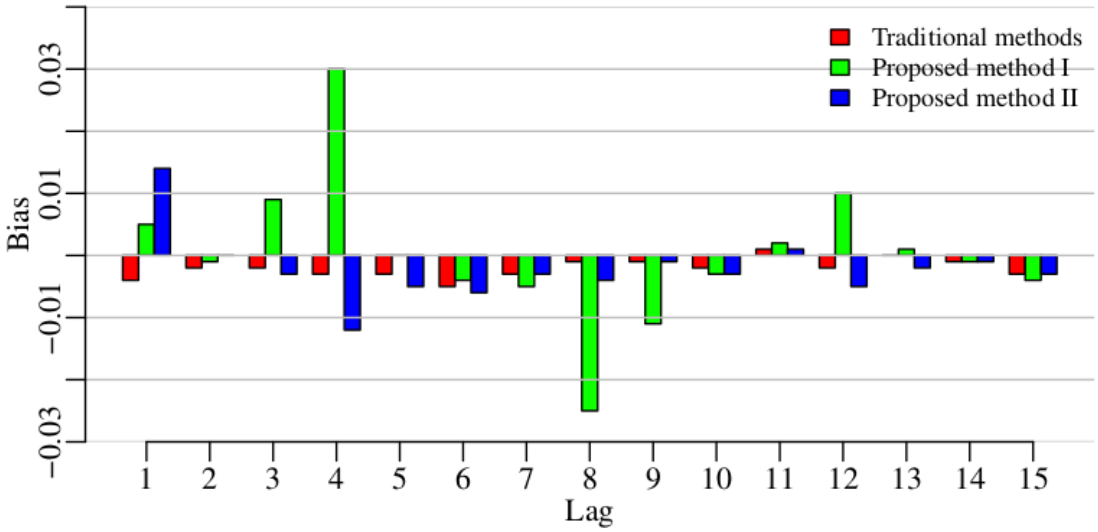
Source: Author (2023).

smaller than 0.015. The complete autocorrelation vector detailed in the Figure 29 shows that the proposed estimators fluctuate around the true autocorrelation values for all 511 lags with a few spikes every 128 lags.

In Scenario II, the negative autocorrelation causes its values to alternate between positive and negative. As we can see in Figure 30, such behavior is maintained in the approximate estimators. According to Figure 31, the proposed estimators have biases lower than 0.12 with the exception of lag 3. In Figure 32, it is possible to observe that the estimates from Method II remain close to the parameter values in all 511 lags.

In Figures 33, 34, and 35, we have a strong positive correlation close to non-stationarity. Although the proposed estimators present considerable biases in the first lags, even the traditional methods have difficulty in estimating the autocorrelation, presenting

Figure 28 – Comparison between proposed and traditional methods considering the first 15 lags for Scenario I in terms of bias



Source: Author (2023).

larger biases than the approximate estimators, as we can see in some lags from 7 onwards.

In Scenario IV, an extra parameter is added to the process, thus making it a second-order autoregressive process. In the Figure 33, it is possible to notice that Method II is a refinement of Method I. The proposed estimators tend to have a positive bias, while the traditional methods have a negative, bias as suggests Figures 34 and 35.

Scenario V is considered a worst case study. The insertion of the extra parameters causes an increase in the bias even in the traditional methods as shown in Figure 40. The proposed estimators perform close to traditional methods even in adverse scenarios, as indicated by the Figures 40 and 41.

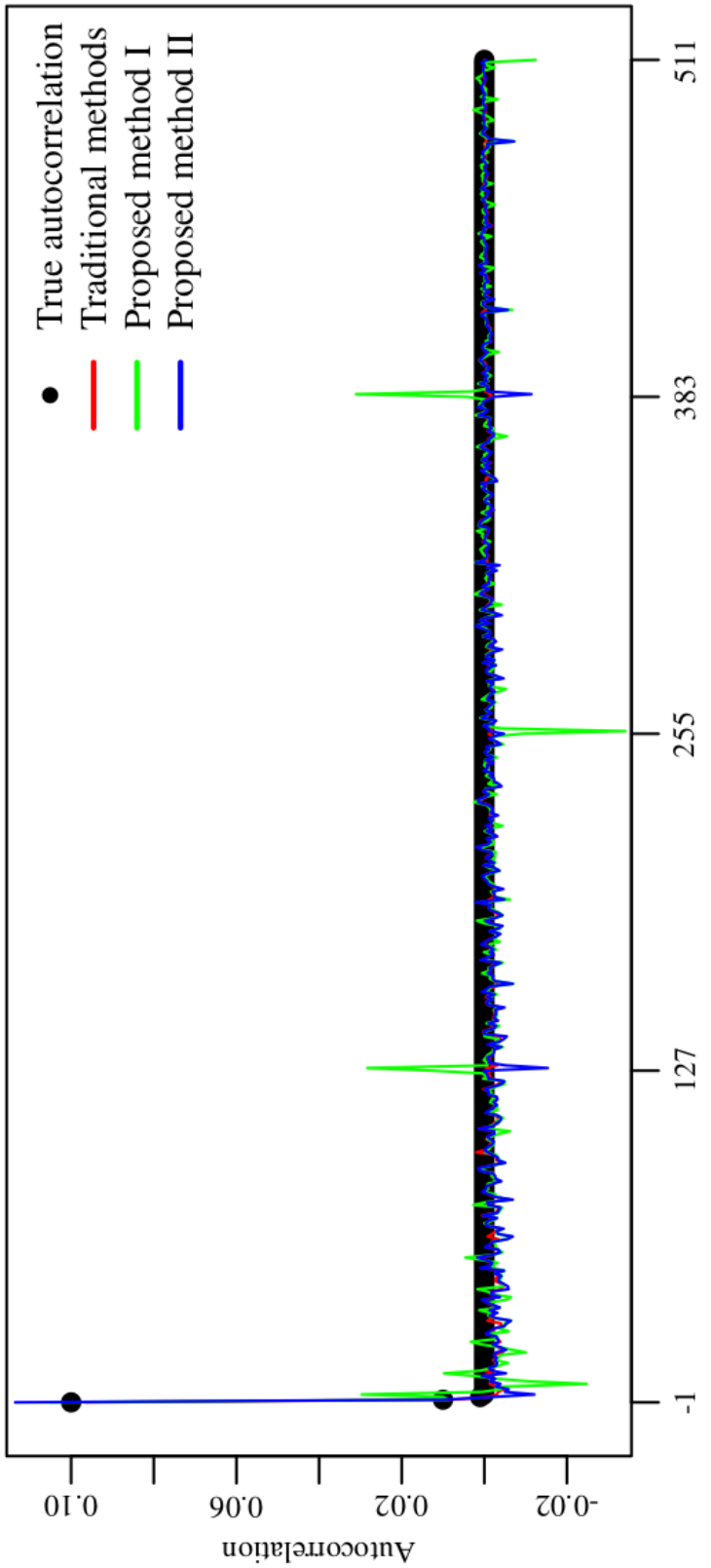
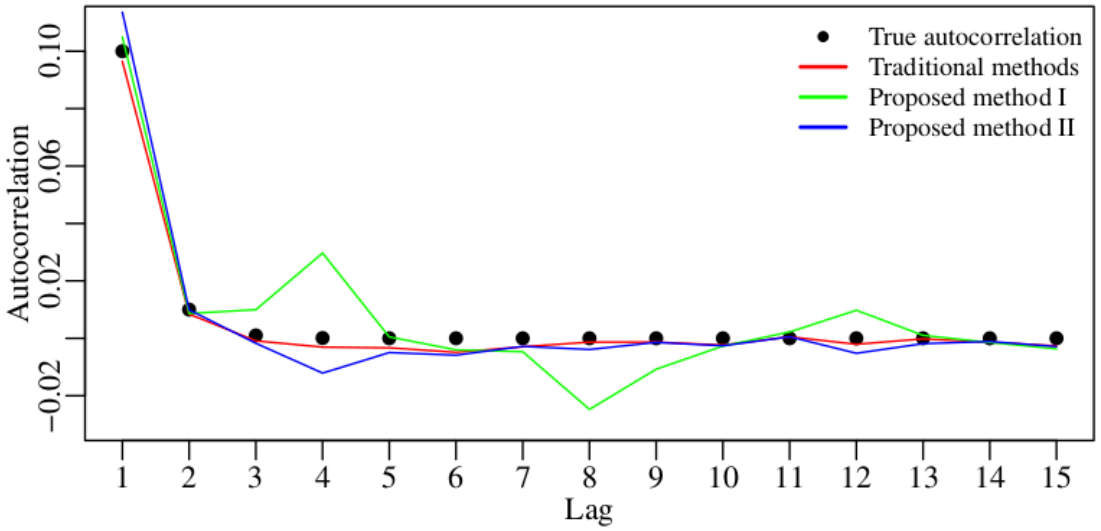


Figure 29 – Comparison between proposed and traditional methods considering all 511 lags for Scenario I

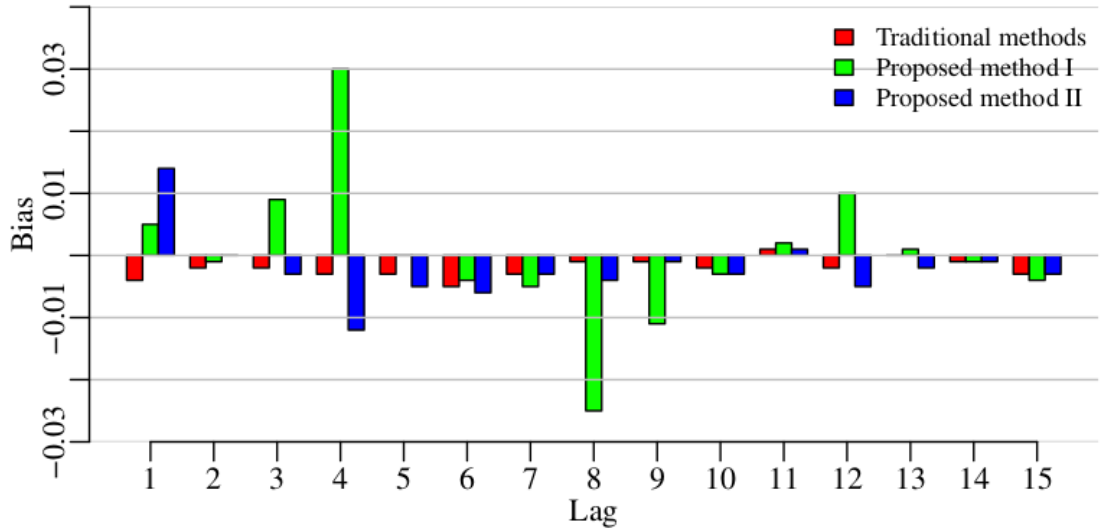
Source: Author (2023).

Figure 30 – Comparison between proposed and traditional methods considering the first 15 lags for Scenario II



Source: Author (2023).

Figure 31 – Comparison between proposed and traditional methods considering the first 15 lags for Scenario II in terms of bias



Source: Author (2023).

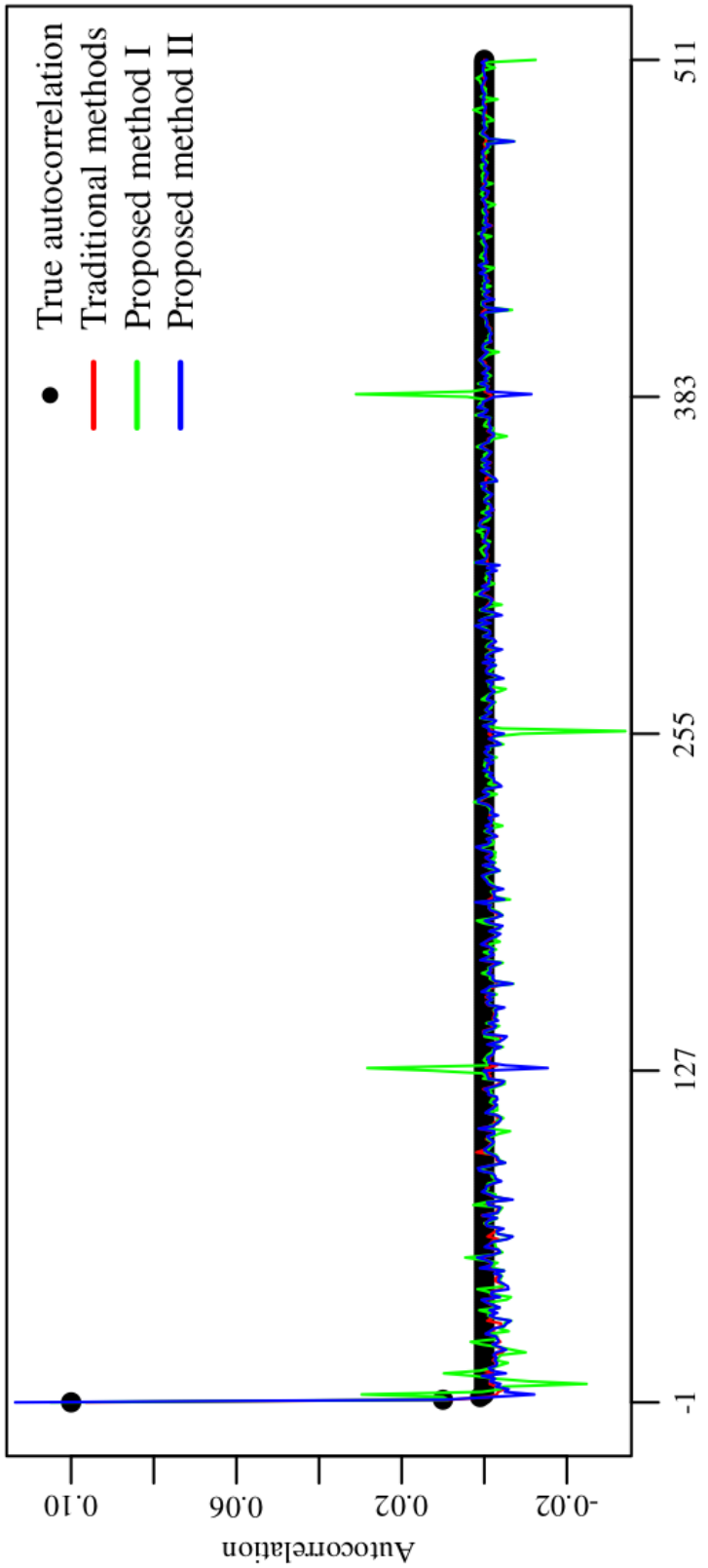
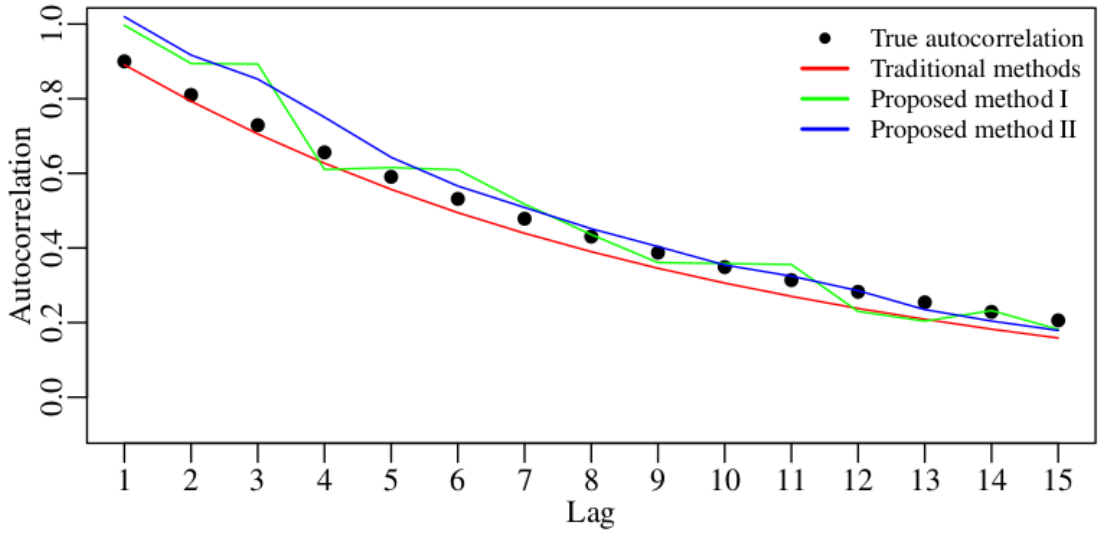


Figure 32 – Comparison between proposed and traditional methods considering all 511 lags for Scenario II

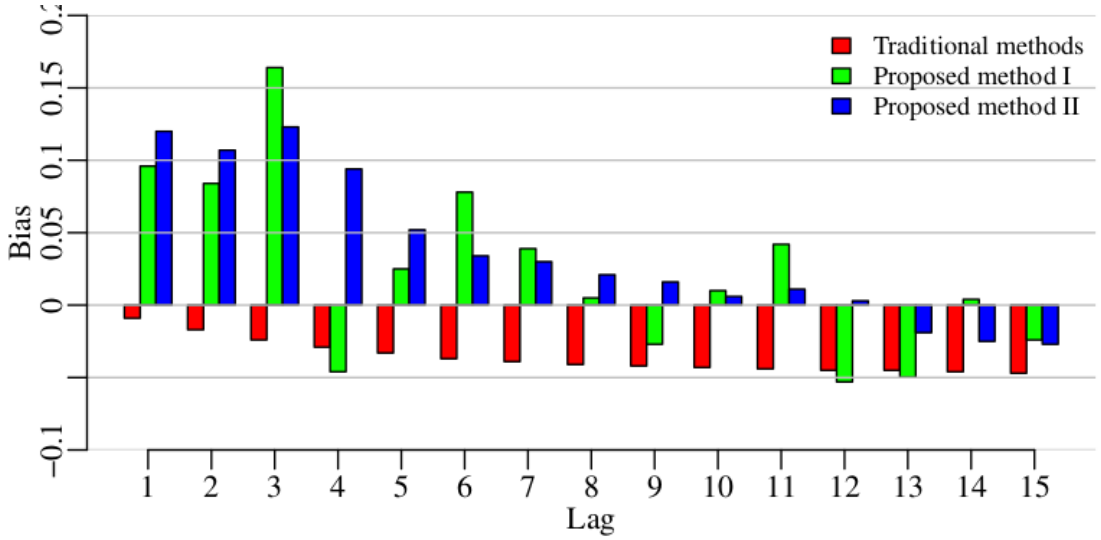
Source: Author (2023).

Figure 33 – Comparison between proposed and traditional methods considering the first 15 lags for Scenario III



Source: Author (2023).

Figure 34 – Comparison between proposed and traditional methods considering the first 15 lags for Scenario III in terms of bias



Source: Author (2023).

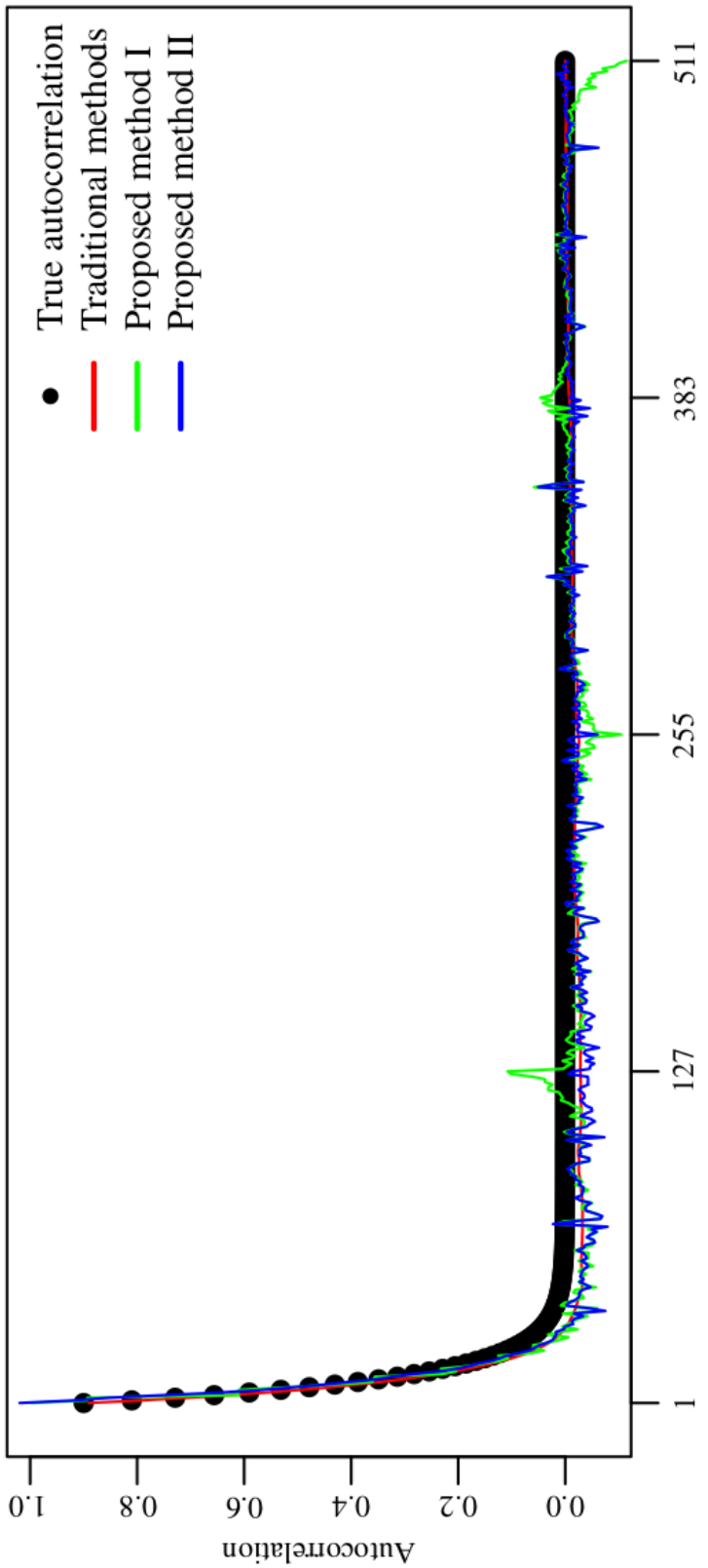
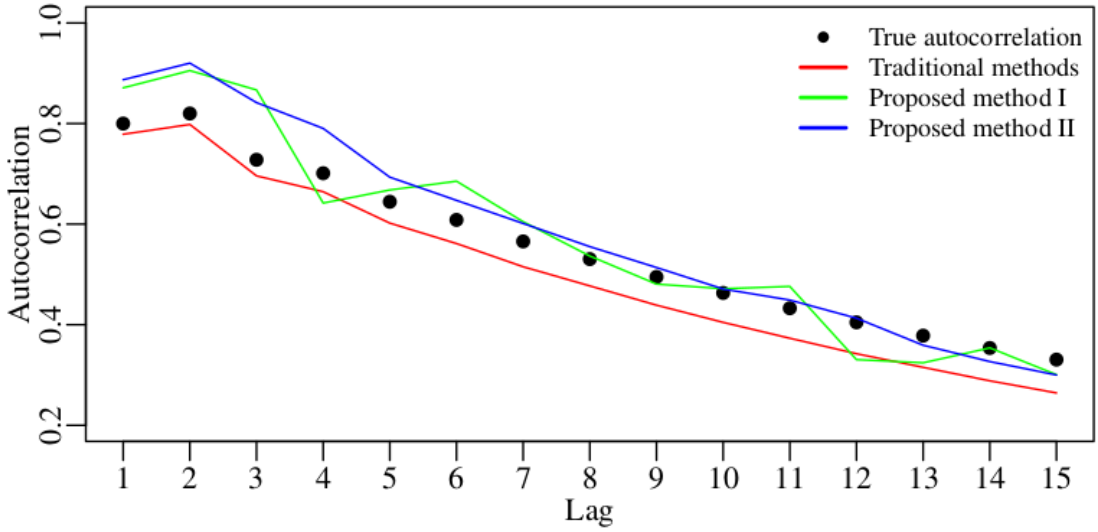


Figure 35 – Comparison between proposed and traditional methods considering all 511 lags for Scenario III

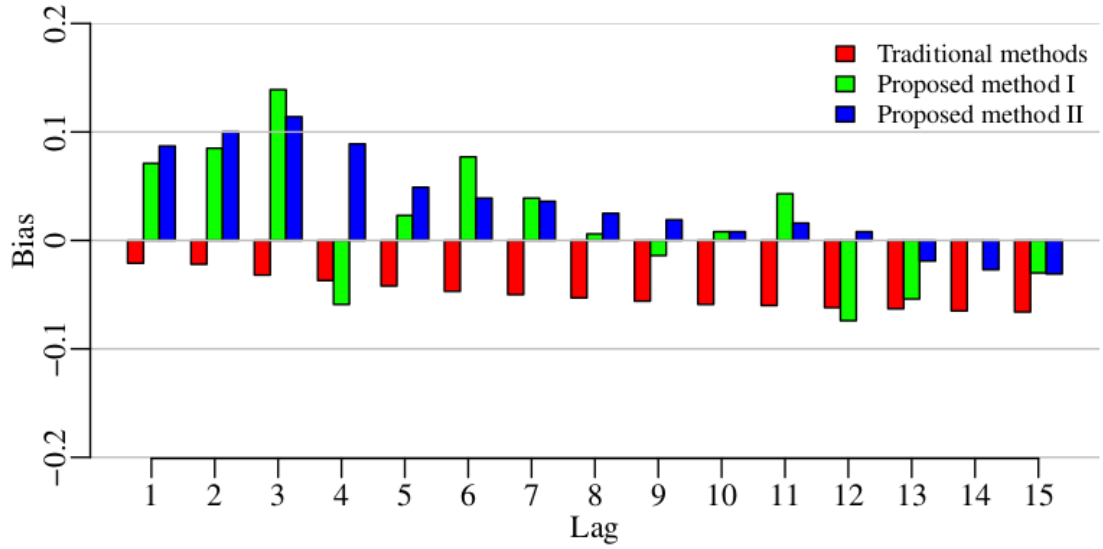
Source: Author (2023).

Figure 36 – Comparison between proposed and traditional methods considering the first 15 lags for Scenario IV



Source: Author (2023).

Figure 37 – Comparison between proposed and traditional methods considering the first 15 lags for Scenario IV in terms of bias



Source: Author (2023).

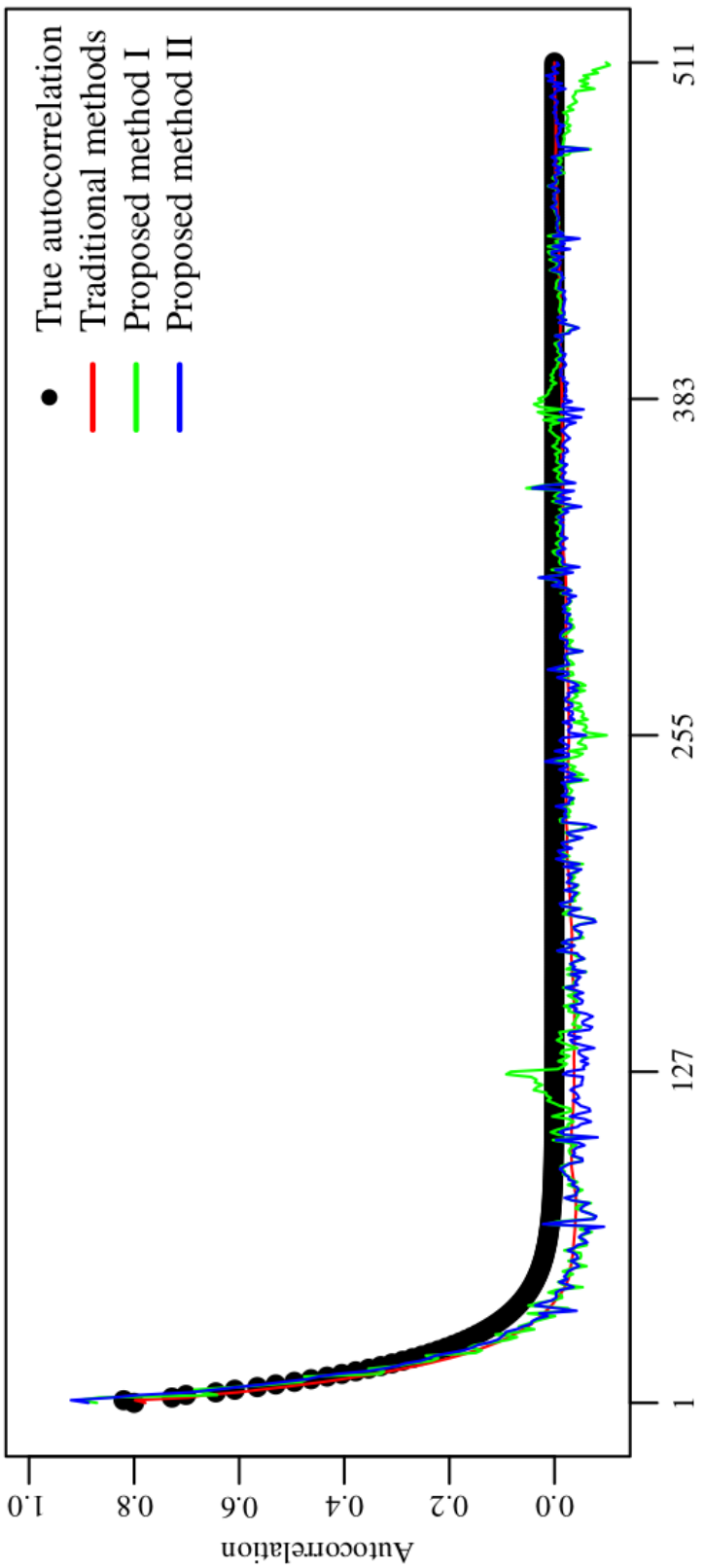
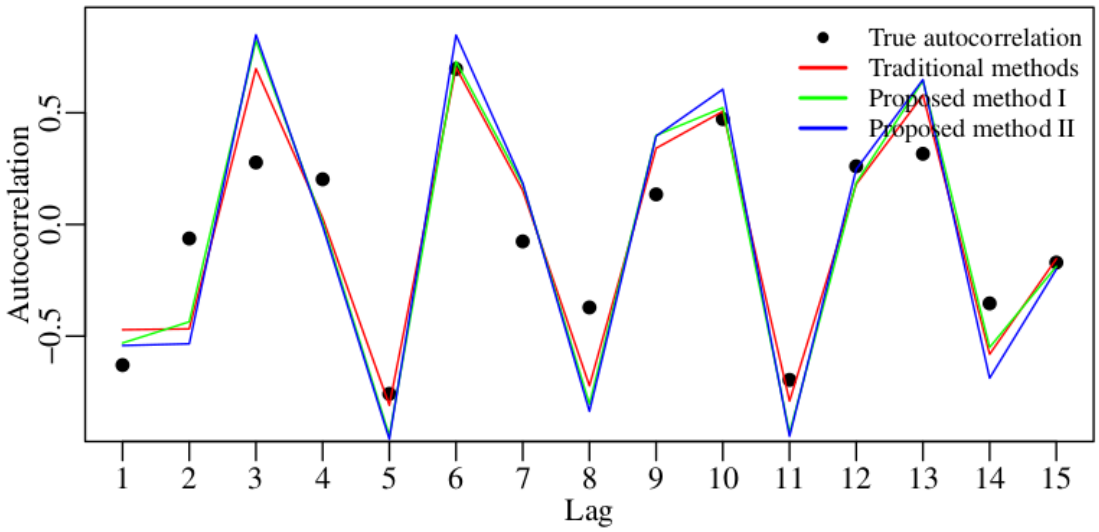


Figure 38 – Comparison between proposed and traditional methods considering all 511 lags for Scenario IV

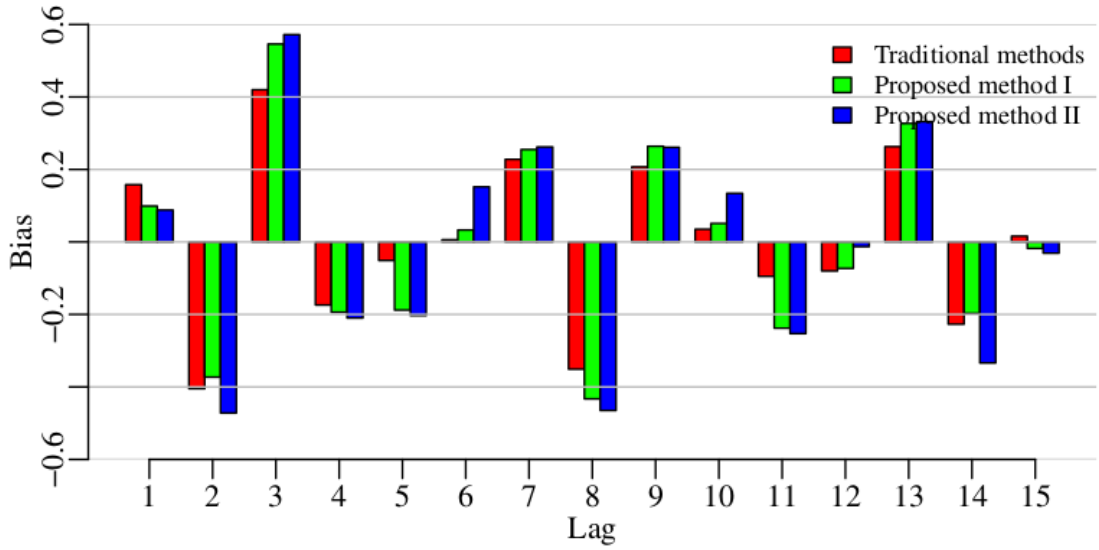
Source: Author (2023).

Figure 39 – Comparison between proposed and traditional methods considering the first 15 lags for Scenario V



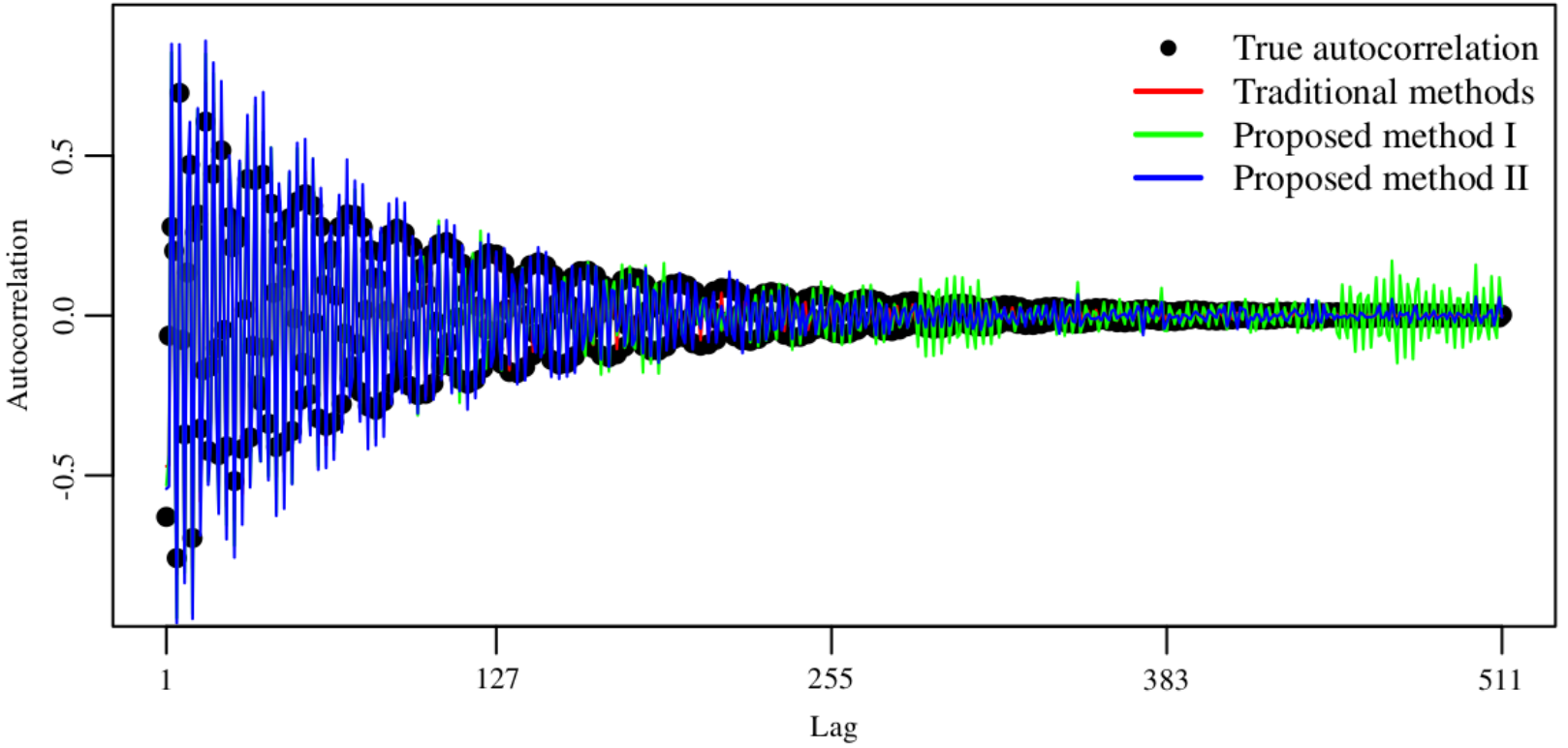
Source: Author (2023).

Figure 40 – Comparison between proposed and traditional methods considering the first 15 lags for Scenario V in terms of bias



Source: Author (2023).

Figure 41 – Comparison between proposed and traditional methods considering all 511 lags for Scenario V



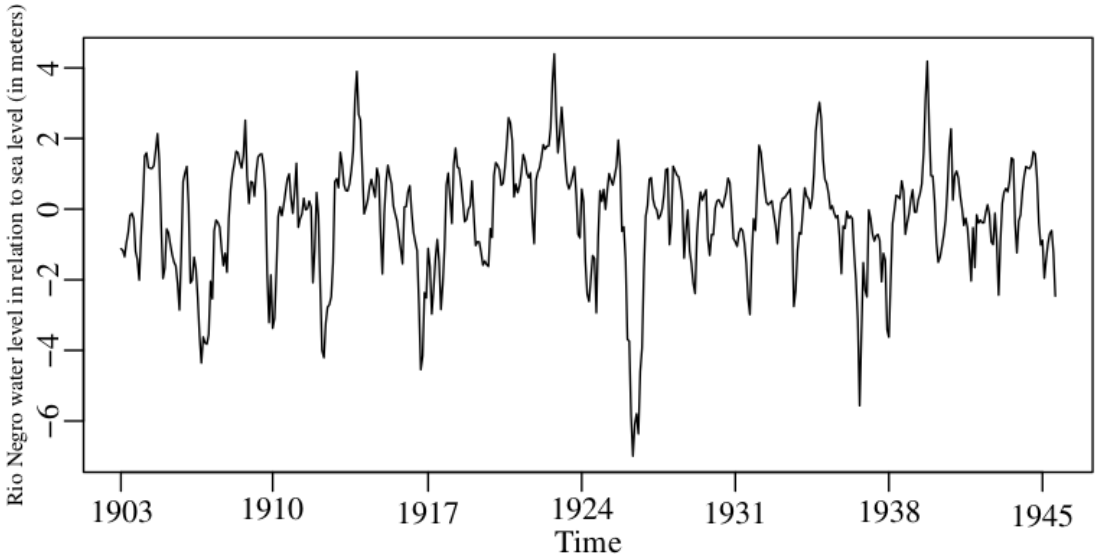
Source: Author (2023).

6.4.1 Real data application

In this section, the proposed methods are compared with traditional methods on real data. For this analysis, data from the boot package [228,229] of the R software is used. The database labeled Manaus consists of a time series referring to the monthly average of the daily stages (heights) of the Rio Negro at Manaus in relation to sea level. We extract the first 512 observations referring to the period from January 1903 to August 1945. The data is shown in Figure 42.

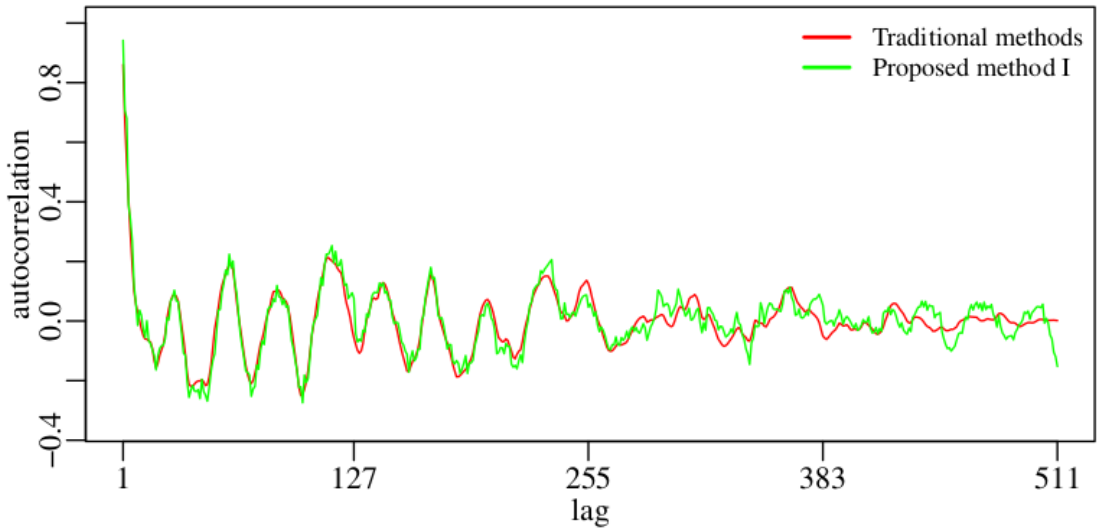
In Figures 43 and 44, the proposed methods are compared with the traditional methods for all 511 lags. Both proposed methods perform close to traditional ones, but with a reduced amount of arithmetic operations. Based on Figures 43 and 44, there is evidence that this database has a stronger autocorrelation in the first lags, thus in Figures 45 and 46, the first 15 lags are highlighted.

Figure 42 – Monthly average of the daily stages of the Rio Negro at Manaus between the years 1903 and 1945



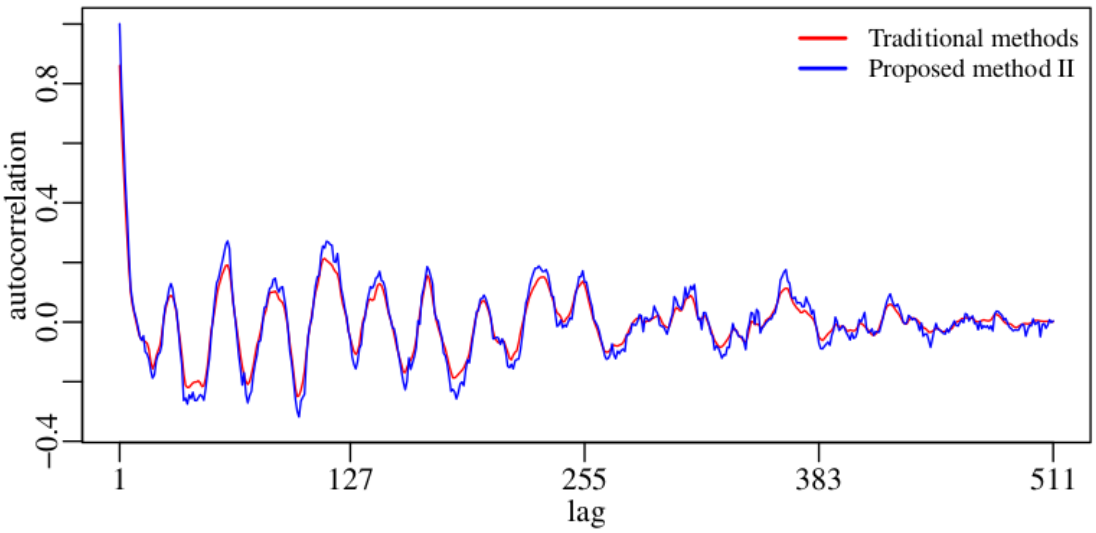
Source: Author (2023).

Figure 43 – Comparison between the proposed method I and traditional methods considering all 511 lags for the Manaus dataset



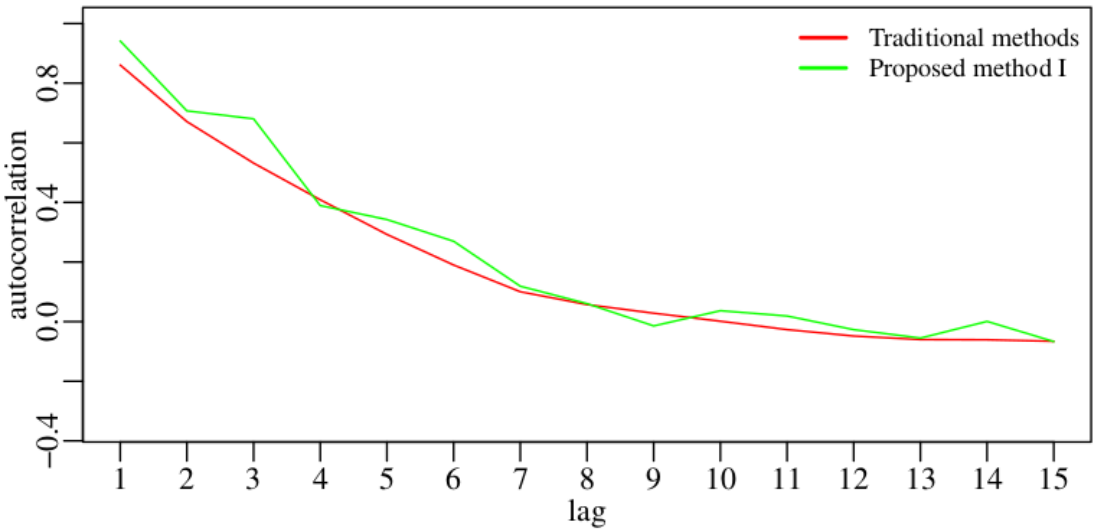
Source: Author (2023).

Figure 44 – Comparison between the proposed method II and traditional methods considering all 511 lags for the Manaus dataset



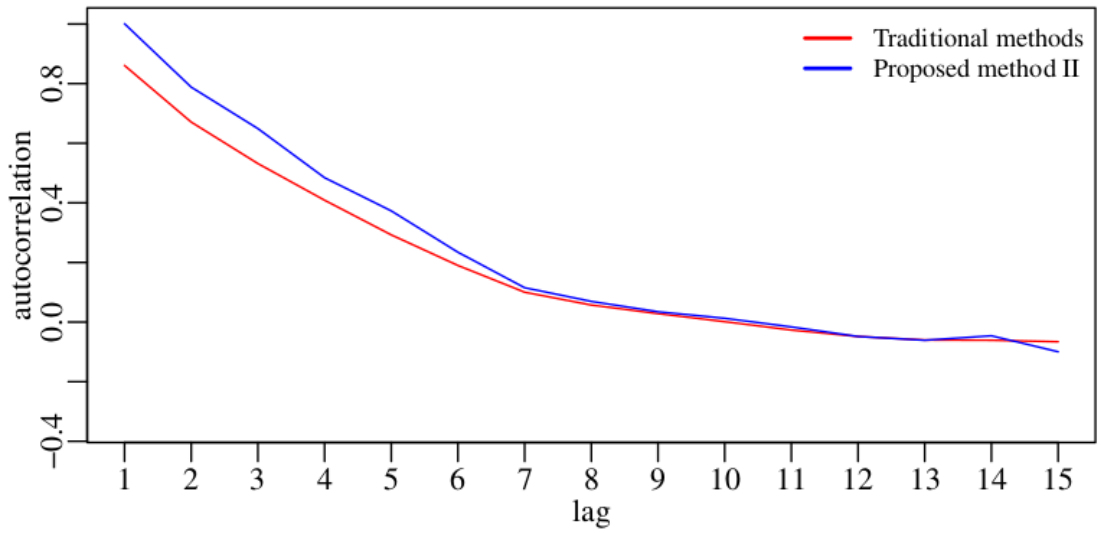
Source: Author (2023).

Figure 45 – Comparison between the proposed method I and traditional methods considering the first 15 lags for the Manaus dataset



Source: Author (2023).

Figure 46 – Comparison between the proposed method II and traditional methods considering the first 15 lags for the Manaus dataset



Source: Author (2023).

6.5 CHAPTER CONCLUSIONS

In this chapter, we proposed low-complexity autocorrelation estimators computed by means of approximate Fourier transforms. Combining fast algorithms and approximate transforms, we reduced approximately 99% of the multiplications and at least 72% of the additions that traditional methods require to estimate the autocorrelation. Specifically, we applied the methodology in scenarios derived from autoregressive processes with length of $N = 512$ and compared the proposed estimators with the usual methods by Monte Carlo simulations. The proposed low-complexity estimators performed close to traditional estimators even in adverse scenarios.

7 DOCUMENT CONCLUSIONS

In this chapter, some concluding remarks, contributions, and future works are presented.

7.1 CONCLUDING REMARKS

In Chapter 3, we explored large-sized FFTs by means of a proposed recursive scaling method that employed approximate ground transformation matrices. We analyzed the 32^4-point DFT derived after $n = 2$ iterations of the proposed methodology and ground transform of size $N = 32$, providing three different approximations. The performance of each approximation was assessed in terms of arithmetic complexity and error analysis, emphasizing the trade-off between accuracy and low-cost computation. The proposed approximations offer competing performance and can be employed as a starting point for complexity reduction in DFT-based systems. Fine tuned approximations can be obtained by changing the ground transform and the number of iterations, depending on the specifics of the given application.

In Chapter 4, we proposed a method to obtain matrices with null complexity of multiplications to present an approximate version of the prime factor algorithm, the APFA. We demonstrated that if the transform length can be decomposed into relatively prime factors than the entire computation of the algorithm can be performed without multiplications. This fact is due to the absence of twiddle factors, which prevent the propagation of the error that is common in algorithms that use them in the mapping. In particular, we applied the proposed method in the 1023-point DFT approximation presenting a collection of approximations with different levels of trade-off between accuracy and computation cost. The APFA performed better than the methods presented in the literature according to the investigated figures of merit in addition to preserve the main characteristics of the DFT.

In Chapter 5, we presented a method to achieve multiplierless approximations for the DFT obtained from the cropping of the CSD representation. The method can be applied to both ground transformations and twiddle factor matrices. Thus, we used the

Cooley-Tukey algorithm and still kept the entire approximate DFT computation free of multiplications. We introduced two new approximations for the 1024-point DFT based on the multiplierless approximations for the 32-point DFT and the 32×32 twiddle factor matrix. The proposed approximations performed better than the literature approximations with lower arithmetic cost.

Finally, in Chapter 6, low-complexity estimators for computing the autocorrelation function were presented. The proposed autocorrelation estimators are based on the properties of the DFT which allows FFTs to be used. We combined such fast algorithms with the approximation methodology. The approximations presented in Chapter 5 were used to estimate the autocorrelation in AR processes with length $N = 512$, evaluated in Monte Carlo simulations. The proposed estimators showed a performance close to the traditional method with substantial reduction in the number of arithmetic operations.

7.2 MAIN CONTRIBUTIONS

The main contributions of this work are summarized as follows:

- Chapter 3
  - A non-linear recursive scaling method to obtain DFT approximations;
  - Multiplicative complexity analysis of the proposed scaling method including hybrid cases;
  - Three new approximations for the 32^4-point DFT.
- Chapter 4
  - A method for obtaining approximations for DFTs of any blocksize;
  - A recursive algorithm to obtain multiplierless DFT approximation for blocklengths that can be decomposed into relatively prime factors;
  - Three new approximations for the 3-point DFT;
  - Three new approximations for the 11-point DFT;
  - Three new approximations for the 31-point DFT;
  - Fifteen new approximations for the 1023-point DFT (including hybrid approximations);

  - A multiplierless approximation for the 1023-point DFT;
  - All new DFT approximations have sparse matrix factorizations.
- Chapter 5
  - A new algorithm to obtain multiplierless DFT approximations based on the Cooley-Tukey algorithm;
  - Two new approximations for the 32-point DFT;
  - A new approximation for the 32×32 twiddle factor matrix;
  - Two new approximations for the 1024-point DFT.
- Chapter 6
  - A new low-complexity method to estimate the autocorrelation for AR processes.

7.3 PUBLISHED WORKS

From Chapter 3, a paper entitled "Radix-N Algorithm for Computing N^{2^n}-Point DFT Approximations" [75] was published in the international journal IEEE Signal Processing Letters <https://doi.org/10.1109/LSP.2022.3200573>.

A second paper entitled "Extensions on low-complexity DCT approximations for larger blocklengths based on minimal angle similarity" [230] was published in the international Journal of Signal Processing Systems for Signal, Image, and Video Technology <https://doi.org/10.1007/s11265-023-01848-w>. Although approximation methods for discrete transforms are also discussed in this paper, approximations for DCT were not included in this thesis.

Three manuscripts are in progress relating to Chapters 4, 5, and 6.

7.4 FUTURE WORKS

For future works, we suggest the following lines of research:

- The approximations for odd-order matrices had few terms in the factorization in sparse matrices. In this way, we intend to look for different ways of factoring

the approximations for odd lengths with the intention of reducing the number of additions remaining in the APFA algorithm;

- As approximations to the DFT occur in a case-by-case fashion, we intend to extend the approximation methods of Chapters 4 and 5 to different blocklengths;
- The approximate methods presented substantial gain in terms of arithmetic complexity. Thus, we intend to apply the methods in other discrete transforms such as the DCT, the discrete sine transform, discrete Hartley transform, and the discrete wavelet transform;
- The autocorrelation estimator is not restricted to AR processes. We intend to evaluate its performance in moving average (MA) and autoregressive moving average (ARMA) processes;
- Study the possible hardware implementation of the proposed works.

REFERENCES

1 BRACEWELL, R. N. **The Fourier transform and its applications**. 3rd. ed. New York, NY: McGraw-Hill, 2000.

2 KAY, S. M. **Modern spectral estimation: Theory and application**. 1st. ed. Englewood Cliffs, NJ: Pearson Education, 1988.

3 OPPENHEIM, A. V.; SCHAFER, R. W. **Discrete-time signal processing**. 3rd. ed. Edinburgh, UK: Pearson Education, 2014.

4 RAO, K. R.; YIP, P. C. **The transform and data compression handbook**. 1st. ed. Boca Raton, FL: CRC press, 2018.

5 PRATT, H. *et al.* FCNN: Fourier convolutional neural networks. In: SPRINGER. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. [S.l.], 2017. p. 786–798.

6 BRIGHAM, E. O. **The fast Fourier transform and its applications**. 1st. ed. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1988.

7 BRIGGS, W. L.; HENSON, V. E. **The DFT: An owners' manual for the discrete Fourier transform**. 1st. ed. Philadelphia, PA: SIAM, 1995. v. 45.

8 MYERS, D. G. **Digital signal processing: Efficient convolution and Fourier transform techniques**. 1st. ed. New York, NY: Prentice-Hall, Inc., 1990.

9 OPPENHEIM, A. V.; VERGHESE, G. C. **Signals, systems and inference**. Global. New York, NY: Pearson, 2017.

10 MATHWORKS. **MATLAB R2022b Documentation "cconv"**. 2022. Https://www.mathworks.com/help/signal/ref/cconv.html.

11 ROSSUM, G. V.; DRAKE, F. L. **Python 3 Reference Manual**. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

12 R Core Team. **R: A language and environment for statistical computing**. Vienna, Austria, 2021. Disponível em: <https://www.R-project.org/>.

13 BRUNTON, S. L.; KUTZ, J. N. **Data-driven science and engineering: Machine learning, dynamical systems, and control**. New York, NY: Cambridge University Press, 2022.

14 TSENG, C.-C.; PEI, S.-C.; HSIA, S.-C. Computation of fractional derivatives using Fourier transform and digital FIR differentiator. **Signal Processing**, Elsevier, v. 80, n. 1, p. 151–159, 2000.

15 SAPONARA, S. Real-time and low-power processing of 3D direct/inverse discrete cosine transform for low-complexity video codec. **Journal of Real-Time Image Processing**, Springer, v. 7, n. 1, p. 43–53, 2012.

16 KHAYYERI, M.; MOHAMMADI, K. Cooperative wideband spectrum sensing in cognitive radio based on sparse real-valued fast Fourier transform. **IET Communications**, IET, v. 14, n. 8, p. 1340–1348, 2020.

17 KUO, K.-C.; CHOU, C.-W. Low power and high speed multiplier design with row bypassing and parallel architecture. **Microelectronics Journal**, Elsevier, v. 41, n. 10, p. 639–650, 2010.

18 JANG, J.-W.; CHOI, S. B.; PRASANNA, V. K. Energy-and time-efficient matrix multiplication on FPGAs. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, IEEE, v. 13, n. 11, p. 1305–1319, 2005.

19 HEIDEMAN, M. T.; BURRUS, C. S. **Multiplicative complexity, convolution, and the DFT**. 1st. ed. New York, NY: Springer, 1988.

20 BURRUS, C.; PARKS, T. **DFT/FFT and convolution algorithms. Theory and Implementation**. 1st. ed. New York, NY: John Wiley and Sons, 1985.

21 BLAHUT, R. E. **Fast algorithms for signal processing**. 2nd. ed. Cambridge, UK: Cambridge University Press, 2010.

22 LIU, W. *et al.* Approximate designs for fast Fourier transform (FFT) with application to speech recognition. **IEEE Transactions on Circuits and Systems I: Regular Papers**, IEEE, v. 66, n. 12, p. 4727–4739, 2019.

23 SATHIYAMURTHI, P.; RAMAKRISHNAN, S. Speech encryption algorithm using FFT and 3D-Lorenz–logistic chaotic map. **Multimedia Tools and Applications**, Springer, v. 79, n. 25, p. 17817–17835, 2020.

24 LIU, B. *et al.* Precision adaptive MFCC based on R2SDF-FFT and approximate computing for low-power speech keywords recognition. **IEEE Circuits and Systems Magazine**, IEEE, v. 21, n. 4, p. 24–39, 2021.

25 ZHAO, H. *et al.* Implementation of discrete Fourier transform using RRAM arrays with quasi-analog mapping for high-fidelity medical image reconstruction. In: IEEE. **2021 IEEE International Electron Devices Meeting (IEDM)**. [S.l.], 2021. p. 12.4.1–12.4.4.

26 SOUZA, R.; FRAYNE, R. A hybrid frequency-domain/image-domain deep network for magnetic resonance image reconstruction. In: **SIBGRAPI Conference on Graphics, Patterns and Images**. [S.l.: s.n.], 2019. p. 257–264.

27 ZHU, B. *et al.* Image reconstruction by domain-transform manifold learning. **Nature**, Nature Publishing Group, v. 555, n. 7697, p. 487–492, 2018.

28 SHARMA, A. *et al.* Heart rate and blood pressure measurement based on photoplethysmogram signal using fast Fourier transform. **Computers and Electrical Engineering**, Elsevier, v. 101, p. 108057, 2022.

29 NOR, N. S. M. *et al.* Automated classification of eight different electroencephalogram (EEG) bands using hybrid of fast Fourier transform (FFT) with machine learning methods. **Neuroscience Research Notes**, v. 5, n. 1, p. 116–116, 2022.

30 WANG, X. *et al.* A multi-step interpolated-FFT procedure for full-field nonlinear modal testing of turbomachinery components. **Mechanical Systems and Signal Processing**, Elsevier, v. 169, p. 108771, 2022.

31 JALLOULI, K. *et al.* MIMO-OFDM LTE system based on a parallel IFFT/FFT on NoC-based FPGA. **Annals of Telecommunications**, Springer, p. 1–14, 2022.

32 ADNAN, A. *et al.* Demonstration of non-hermitian symmetry (NHS) IFFT/FFT size efficient OFDM non-orthogonal multiple access (NOMA) for visible light communication. **IEEE Photonics Journal**, IEEE, v. 12, n. 3, p. 1–5, 2020.

33 NASSIRI, M.; BAGHERSALIMI, G. Comparative performance assessment between FFT-based and FRFT-based MIMO-OFDM systems in underwater acoustic communications. **IET Communications**, Wiley Online Library, v. 12, n. 6, p. 719–726, 2018.

34 LEE, D.-U. *et al.* Energy-efficient image compression for resource-constrained platforms. **IEEE Transactions on Image Processing**, IEEE, v. 18, n. 9, p. 2100–2113, 2009.

35 MAHARATNA, K.; GRASS, E.; JAGDHOLD, U. A 64-point Fourier transform chip for high-speed wireless LAN application using OFDM. **IEEE Journal of Solid-State Circuits**, IEEE, v. 39, n. 3, p. 484–493, 2004.

36 ZHANG, Z. *et al.* Large-scale MIMO-based wireless backhaul in 5G networks. **IEEE Wireless Communications**, IEEE, v. 22, n. 5, p. 58–66, 2015.

37 KITAMURA, I.; KANAI, S.; KISHINAMI, T. Copyright protection of vector map using digital watermarking method based on discrete Fourier transform. In: IEEE. **IEEE International Geoscience and Remote Sensing Symposium**. [S.l.], 2001. v. 3, p. 1191–1193.

38 LIN, S. *et al.* FFT-based deep learning deployment in embedded systems. In: **Design, Automation Test in Europe Conference Exhibition**). [S.l.: s.n.], 2018. p. 1045–1050.

39 STANKOVIC, J. A. Research directions for the internet of things. **IEEE Internet of Things Journal**, IEEE, v. 1, n. 1, p. 3–9, 2014.

40 LU, N. *et al.* Connected vehicles: Solutions and challenges. **IEEE Internet of Things Journal**, IEEE, v. 1, n. 4, p. 289–299, 2014.

41 SHERRY, Y.; THOMPSON, N. C. How fast do algorithms improve? [point of view]. **Proceedings of the IEEE**, v. 109, n. 11, p. 1768–1777, 2021.

42 WICHERT, A. **Principles of quantum artificial intelligence: quantum problem solving and machine learning**. 2nd. ed. Hackensack, NJ: World Scientific, 2020.

43 YANNAKAKIS, G. N.; MARTÍNEZ, H. P.; JHALA, A. Towards affective camera control in games. **User Modeling and User-Adapted Interaction**, Springer, v. 20, n. 4, p. 313–340, 2010.

44 RABABAAH, A. R.; SHARMA, D. K. Integration of two different signal processing techniques with artificial neural network for stock market forecasting. **Journal of Management Information & Decision Sciences**, v. 18, n. 2, 2015.

45 XUE, Q. *et al.* Rapid driving style recognition in car-following using machine learning and vehicle trajectory data. **Journal of Advanced Transportation**, Hindawi, v. 2019, 2019.

46 ITO, I. A new pseudo-spectral method using the discrete cosine transform. **Journal of Imaging**, Multidisciplinary Digital Publishing Institute, v. 6, n. 4, p. 15, 2020.

47 BRITANAK, V.; YIP, P. C.; RAO, K. R. **Discrete cosine and sine transforms: General properties, fast algorithms and integer approximations**. 1st. ed. Oxford, UK: Elsevier, 2006.

48 HAGHIGHAT, M. B. A.; AGHAGOLZADEH, A.; SEYEDARABI, H. Multi-focus image fusion for visual sensor networks in DCT domain. **Computers & Electrical Engineering**, Elsevier, v. 37, n. 5, p. 789–797, 2011.

49 LIANG, Y. *et al.* Image encryption combining multiple generating sequences controlled fractional DCT with dependent scrambling and diffusion. **Journal of Modern Optics**, Taylor & Francis, v. 62, n. 4, p. 251–264, 2015.

50 COUTINHO, V. d. A.; CINTRA, R. J.; BAYER, F. M. Low-complexity multidimensional DCT approximations for high-order tensor data decorrelation. **IEEE Transactions on Image Processing**, IEEE, v. 26, n. 5, p. 2296–2310, 2017.

51 ZHANG, M. *et al.* Automatic detection of transmission line on UAV inspection images with the statistics approach in the DCT domain. In: IEEE. **International Conference on Sensing, Measurement & Data Analytics in the era of Artificial Intelligence**. [S.l.], 2020. p. 577–581.

52 PHAMILA Y, A. V.; AMUTHA, R. Low complexity energy efficient very low bit-rate image compression scheme for wireless sensor network. **Information Processing Letters**, Elsevier, v. 113, n. 18, p. 672–676, 2013.

53 KOUADRIA, N. *et al.* Low complexity DCT for image compression in wireless visual sensor networks. **Electronics Letters**, IET, v. 49, n. 24, p. 1531–1532, 2013.

54 KUMAR, G. G.; SAHOO, S. K.; MEHER, P. K. 50 years of FFT algorithms and applications. **Circuits, Systems, and Signal Processing**, Springer, v. 38, n. 12, p. 5665–5698, 2019.

55 QUEIROZ, S.; VILELA, J. P.; MONTEIRO, E. Is FFT fast enough for beyond 5G communications? A throughput-complexity analysis for OFDM signals. **IEEE Access**, IEEE, 2022.

56 RAPPAPORT, T. S. *et al.* Wireless communications and applications above 100 GHz: Opportunities and challenges for 6G and beyond. **IEEE Access**, p. 78729–78757, 2019.

57 GUPTA, V. *et al.* Low-power digital signal processing using approximate adders. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 32, n. 1, p. 124–137, 2013.

58 CINTRA, R. J. An integer approximation method for discrete sinusoidal transforms. **Circuits, Systems, and Signal Processing**, Springer, v. 30, n. 6, p. 1481, 2011.

59 BAYER, F. M.; CINTRA, R. J. DCT-like transform for image compression requires 14 additions only. **Electronics Letters**, IET, v. 48, n. 15, p. 919–921, 2012.

60 POTLURI, U. S. *et al.* Improved 8-point approximate DCT for image and video compression requiring only 14 additions. **IEEE Transactions on Circuits and Systems I: Regular Papers**, IEEE, v. 61, n. 6, p. 1727–1740, 2014.

61 COUTINHO, V. A. *et al.* A multiplierless pruned DCT-like transformation for image and video compression that requires ten additions only. **Journal of Real-Time Image Processing**, Springer, v. 12, n. 2, p. 247–255, 2016.

62 HAWEEL, T. I. A new square wave transform based on the DCT. **Signal processing**, Elsevier, v. 81, n. 11, p. 2309–2319, 2001.

63 BOUGUEZEL, S.; AHMAD, M. O.; SWAMY, M. Low-complexity $8\times$ 8 transform for image compression. **Electronics Letters**, IET, v. 44, n. 21, p. 1249–1250, 2008.

64 BOUGUEZEL, S.; AHMAD, M. O.; SWAMY, M. A novel transform for image compression. In: IEEE. **International Midwest Symposium on Circuits and Systems**. [S.l.], 2010. p. 509–512.

65 CINTRA, R. J.; BAYER, F. M.; TABLADA, C. Low-complexity 8-point DCT approximations based on integer functions. **Signal Processing**, Elsevier, v. 99, p. 201–214, 2014.

66 CINTRA, R. J.; BAYER, F. M. A DCT approximation for image compression. **IEEE Signal Processing Letters**, IEEE, v. 18, n. 10, p. 579–582, 2011.

67 AKYILDIZ, I. F.; MELODIA, T.; CHOWDHURY, K. R. A survey on wireless multimedia sensor networks. **Computer networks**, Elsevier, v. 51, n. 4, p. 921–960, 2007.

68 DIMITROV, V.; WAHID, K. Multiplierless DCT algorithm for image compression applications. **International Journal on Information Theories and Applications**, Institute of Information Theories and Applications, v. 11, p. 162–169, 2004.

69 VILLAGRAN, D. M. S. **Aproximações para a transformada discreta de Fourier e Aplicações em Detecção e Estimação**. Dissertação (Mestrado) — Graduate Program in Statistics, Universidade Federal de Pernambuco, Recife, Brazil, 2015. Advisors: R. J. Cintra and F. M. Bayer. Disponível em: <https://sucupira.capes.gov.br/sucupira/public/consultas/coleta/trabalhoConclusao/viewTrabalhoConclusao.jsf?popup=true&id_trabalho=2225001#>.

70 VILLAGRAN, D. M. S. *et al.* Multi-beam RF aperture using multiplierless FFT approximation. **Electronics Letters**, IET, v. 50, n. 24, p. 1788–1790, 2014.

71 KULASEKERA, S. *et al.* Multi-beam receiver apertures using multiplierless 8-point approximate DFT. p. 1244–1249, 2015.

72 ARIYARATHNA, V. *et al.* Multibeam digital array receiver using a 16-point multiplierless DFT approximation. **IEEE Transactions on Antennas and Propagation**, IEEE, v. 67, n. 2, p. 925–933, 2018.

73 MADANAYAKE, A. *et al.* Towards a low-SWaP 1024-beam digital array: A 32-beam subsystem at 5.8 GHz. **Transactions on Antennas and Propagation**, IEEE, v. 68, n. 2, p. 900–912, 2019.

74 MADANAYAKE, A. *et al.* Fast radix-32 approximate DFTs for 1024-beam digital RF beamforming. **IEEE Access**, IEEE, v. 8, p. 96613—96627, 2020.

75 PORTELLA, L. *et al.* Radix-N algorithm for computing N^{2^n}-point DFT approximations. **IEEE Signal Processing Letters**, IEEE, v. 29, p. 1838–1842, 2022.

76 GERSHMAN, A. B.; RÜBSAMEN, M.; PESAVENTO, M. One- and two-dimensional direction-of-arrival estimation: An overview of search-free techniques. **Signal Processing**, Elsevier, v. 90, n. 5, p. 1338–1349, 2010.

77 ZHU, Y.; SHASHA, D. Statstream: Statistical monitoring of thousands of data streams in real time. In: ELSEVIER. **Proceedings of the 28th International Conference on Very Large Databases**. [S.l.], 2002. p. 358–369.

78 MÖRCHEN, F. **Time series feature extraction for data mining using DWT and DFT**. University: Citeseer, 2003.

79 GOOD, I. J. The interaction algorithm and practical Fourier analysis. **Journal of the Royal Statistical Society: Series B**, Wiley Online Library, v. 20, n. 2, p. 361–372, 1958.

80 THOMAS, L. H. Using a computer to solve problems in physics. **Applications of Digital Computers**, Ginn, p. 44–45, 1963.

81 DUHAMEL, P.; VETTERLI, M. Fast Fourier transforms: A tutorial review and a state of the art. **Signal processing**, Elsevier, v. 19, n. 4, p. 259–299, 1990.

82 MILDER, P. A. *et al.* Hardware implementation of the discrete Fourier transform with non-power-of-two problem size. In: **IEEE International Conference on Acoustics, Speech and Signal Processing**. [S.l.: s.n.], 2010. p. 1546–1549.

83 VAN DE BURGWAL, M. D.; WOLKOTTE, P. T.; SMIT, G. J. Non-power-of-two FFTs: Exploring the flexibility of the montium TP. **International Journal of Reconfigurable Computing**, Hindawi, v. 2009, 2009.

84 YANG, Z.-X. *et al.* Design of a 3780-point IFFT processor for TDS-OFDM. **IEEE Transactions on Broadcasting**, v. 48, n. 1, p. 57–61, 2002.

85 DAI, J.; YIN, H. Design and realization of non-radix-2 FFT prime factor processor for 5G broadcasting in release 16. In: **International Symposium on Computational Intelligence and Design**. [S.l.: s.n.], 2020. p. 406–409.

86 STANDARD, E. Digital radio mondiale (DRM); system specification. **ETSI ES**, Citeseer, v. 201, n. 980, p. V3, 2012.

87 KRISHNEGOWDA, K. *et al.* High-speed channel equalization scheme for 100 Gbps system. In: **IEEE International Conference on Industrial Technology**. [S.l.: s.n.], 2018. p. 1430–1435.

88 BUCHERT,R; SHAHRIER, S; BECKER, P. **Optimized discrete Fourier transform method and apparatus using prime factor algorithm**. 2004. U.S. Patent 20030195911A1.

89 HSU, H. J.; SHIEH, M. D. VLSI architecture of polynomial multiplication for BGV fully homomorphic encryption. In: **IEEE International Symposium on Circuits and Systems**. [S.l.: s.n.], 2020. p. 1–4.

90 MOGHADAM, G. S.; SHIRAZI, A. A. B. DOA estimation with co-prime arrays based on multiplicative beamforming. In: **International Symposium on Telecommunications**. [S.l.: s.n.], 2018. p. 501–506.

91 BETZEL, F. *et al.* Approximate communication: Techniques for reducing communication bottlenecks in large-scale parallel systems. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 51, n. 1, p. 1–32, 2018.

92 EHRGOTT, M. **Multicriteria optimization**. Heidelberg, BW: Springer Science & Business Media, 2005. v. 491.

93 NUSSBAUMER, H. J. The fast Fourier transform. In: **Fast Fourier transform and convolution algorithms**. Heidelberg, BW: Springer, 1981. p. 80–111.

94 SEBER, G. A. **A matrix handbook for statisticians**. Hoboken, NJ: John Wiley & Sons, 2008.

95 COLLETTE, Y.; SIARRY, P. **Multiobjective optimization: Principles and case studies**. 1st. ed. Chelles, FR: Springer Science & Business Media, 2004.

96 FEIG, E.; WINOGRAD, S. Fast algorithms for the discrete cosine transform. **IEEE Transactions on Signal processing**, IEEE, v. 40, n. 9, p. 2174–2193, 1992.

97 KULASEKERA, S. *et al.* Multi-beam 8×8 RF aperture digital beamformers using multiplierless 2-D FFT approximations. In: **Moratuwa Engineering Research Conference**. [S.l.: s.n.], 2015. p. 260–264.

98 ARIYARATHNA, V. *et al.* Multi-beam 4 GHz microwave apertures using current-mode DFT approximation on 65 nm CMOS. In: IEEE. **2015 IEEE International Microwave Symposium**. [S.l.], 2015. p. 1–4.

99 ARIYARATHNA, V. *et al.* Analog approximate-FFT 8/16-beam algorithms, architectures and CMOS circuits for 5G beamforming MIMO transceivers. **IEEE Journal on Emerging and Selected Topics in Circuits and Systems**, v. 8, n. 3, p. 466–479, 2018.

100 CINTRA, R. J.; BAYER, F. M.; COELHO, D. F. G. **Sistema de processamento de sinais para a formação de feixes múltiplos para sistemas de antenas digitais de radiofrequência**. 2017. BR Patent 1020170220818.

101 MACLEOD, M. Multiplierless Winograd and prime factor FFT implementation. **IEEE Signal Processing Letters**, v. 11, n. 9, p. 740–743, 2004.

102 SILVERMAN, H. An introduction to programming the Winograd Fourier transform algorithm (WFTA). **IEEE Transactions on Acoustics, Speech, and Signal Processing**, IEEE, v. 25, n. 2, p. 152–165, 1977.

103 HAN, X. *et al.* A novel area-power efficient design for approximated small-point FFT architecture. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 39, n. 12, p. 4816–4827, 2020.

104 DING, J.; CHEN, J.; CHANG, C.-H. A new paradigm of common subexpression elimination by unification of addition and subtraction. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 35, n. 10, p. 1605–1617, 2016.

105 COUTINHO, V. *et al.* An 8-beam 2.4 GHz digital array receiver based on a fast multiplierless spatial DFT approximation. In: IEEE. **2018 IEEE International Microwave Symposium-IMS**. [S.l.], 2018. p. 1538–1541.

106 COUTINHO, V. A. *et al.* A Low-SWaP 16-beam 2.4 GHz digital phased array receiver using DFT approximation. **IEEE Transactions on Aerospace and Electronic Systems**, v. 56, n. 5, p. 3645–3654, 2020.

107 MIRFARSHBAFAN, S. H.; TANER, S.; STUDER, C. SMUL-FFT: A streaming multiplierless fast Fourier transform. **IEEE Transactions on Circuits and Systems II: Express Briefs**, IEEE, v. 68, n. 5, p. 1715–1719, 2021.

108 CHAN, S.; YIU, P. A multiplier-less 1-D and 2-D fast Fourier transform-like transformation using sum-of-powers-of-two (SOPOT) coefficients. In: **2002 IEEE International Symposium on Circuits and Systems. Proceedings**. [S.l.: s.n.], 2002. v. 4, p. 4–7.

109 KAUR, R.; SINGH, T. Design of 32-point mixed radix FFT processor using CSD multiplier. In: IEEE. **2016 Fourth International Conference on Parallel, Distributed and Grid Computing**. [S.l.], 2016. p. 538–543.

110 CHAN, S.; YIU, P. Multiplier-less discrete sinusoidal and lapped transforms using sum-of-powers-of-two (SOPOT) coefficients. In: IEEE. **The 2001 IEEE International Symposium on Circuits and Systems**. [S.l.], 2001. v. 2, p. 13–16.

111 PRESS, W. H. *et al.* **Numerical recipes: The art of scientific computing**. 3rd. ed. [S.l.]: Cambridge university press, 2007.

112 COOLEY, J. W. The re-discovery of the fast Fourier transform algorithm. **Microchimica Acta**, Springer, v. 93, n. 1, p. 33–45, 1987.

113 COOLEY, J. W.; TUKEY, J. W. An algorithm for the machine calculation of complex Fourier series. **Mathematics of Computation**, American Mathematical Society, v. 19, n. 90, p. 297–301, 1965.

114 DUONG, T. **vec, vech, invvec, invvech**. 2019. Disponível em: <https://www.rdocumentation.org/packages/ks/versions/1.10.7/topics/vector>.

115 MATHWORKS. **MATLAB R2022b Documentation "vec2mat"**. 2022. Https://www.mathworks.com/help/comm/ref/vec2mat.html.

116 MATHWORKS. **MATLAB R2022b Documentation "reshape"**. 2022. Https://www.mathworks.com/help/matlab/ref/reshape.html.

117 CHAN, C.-K.; LIN, H.-K.; LIU, C.-W. High-throughput 64K-point FFT processor for THz imaging radar system. In: **International Symposium on VLSI Design, Automation and Test**. [S.l.: s.n.], 2019. p. 1–4.

118 YUAN, L.; JIANG, R.; CHEN, Y. Gain and phase autocalibration of large uniform rectangular arrays for underwater 3-D sonar imaging systems. **IEEE Journal of Oceanic Engineering**, IEEE, v. 39, n. 3, p. 458–471, 2014.

119 HAO, Y. *et al.* Sparsity-inducing frequency-domain adaptive line enhancer for unmanned underwater vehicle sonar. **Applied Acoustics**, Elsevier, v. 173, p. 107689, 2021.

120 WANG, J.; TAO, C.; JIAO, J. Method for high frequency sonar multi-beam matched filtering based on single-chip FPGA. In: **International Conference on Electronics Technology**. [S.l.: s.n.], 2018. p. 157–161.

121 CORDA, S. *et al.* Near memory acceleration on high resolution radio astronomy imaging. In: **Mediterranean Conference on Embedded Computing**. [S.l.: s.n.], 2020. p. 1–6.

122 LIU, W. *et al.* An efficient channelization architecture and its implementation for radio astronomy. **Journal of Instrumentation**, IOP Publishing, v. 16, n. 08, 2021.

123 HE, C. *et al.* A pipelined memory-efficient architecture for ultra-long variable-size FFT processors. In: **International Conference on Computer Science and Information Technology**. [S.l.: s.n.], 2008. p. 357–361.

124 WANG, M.; LU, G. The inversion of material parameters based on FFT twofold subspace-based optimization method. In: **13th Global Symposium on Millimeter-Waves Terahertz**. [S.l.: s.n.], 2021. p. 1–3.

125 MOULDER, W. F. *et al.* Development of a high-throughput microwave imaging system for concealed weapons detection. In: **IEEE International Symposium on Phased Array Systems and Technology**. [S.l.: s.n.], 2016. p. 1–6.

126 FEAR, E. C. Microwave imaging of the breast. **Technology in Cancer Research & Treatment**, SAGE Publications Sage CA: Los Angeles, CA, v. 4, n. 1, p. 69–82, 2005.

127 KHAYYERI, M.; MOHAMMADI, K. Design and implementation of a high-performance and high-speed architecture for wideband spectrum sensing in cognitive radio networks. **Circuits, Systems, and Signal Processing**, Springer, v. 39, n. 4, p. 2151–2177, 2020.

128 DUTKIEWICZ, E. *et al.* Radio environment maps generation and spectrum sensing testbed for spectrum sharing in 5G networks. In: **IEEE Topical Conference on Antennas and Propagation in Wireless Communications**. [S.l.: s.n.], 2017. p. 33–36.

129 MERRITT, J. C.; CHISUM, J. D. High-speed cross-correlation for spectrum sensing and direction finding of time-varying signals. **IEEE Sensors Journal**, v. 18, n. 15, p. 6161–6168, 2018.

130 LEE-THORP, J. *et al.* Fnet: Mixing tokens with Fourier transforms. **arXiv preprint arXiv:2105.03824**, 2021.

131 RAO, Y. *et al.* Global filter networks for image classification. **Advances in Neural Information Processing Systems**, v. 34, p. 980–993, 2021.

132 LOU, T. *et al.* FNetAR: Mixing tokens with autoregressive Fourier transforms. **arXiv preprint arXiv:2107.10932**, 2021.

133 HUANG, X. *et al.* NUMA-aware FFT-based convolution on ARMv8 Many-core CPUs. In: **IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking**. [S.l.: s.n.], 2021. p. 1019–1026.

134 CINTRA, R. J. *et al.* Low-complexity approximate convolutional neural networks. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 29, n. 12, p. 5981–5992, 2018.

135 STEIN, J. Y. **Digital signal processing: A computer science perspective**. 2nd. ed. New York, NY: John Wiley & Sons, Inc., 2000.

136 STRANG, G. Wavelets. **American Scientist**, Sigma Xi, The Scientific Research Society, v. 82, n. 3, p. 250–255, 1994. ISSN 00030996.

137 ABARI, O. *et al.* 27.4 a 0.75-million-point Fourier-transform chip for frequency-sparse signals. In: **IEEE International Solid-State Circuits Conference Digest of Technical Papers**. [S.l.: s.n.], 2014. p. 458–459.

138 HASSANIEH, H. *et al.* Simple and practical algorithm for sparse Fourier transform. In: SIAM. **Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms**. [S.l.], 2012. p. 1183–1194.

139 RAJABY, E.; SAYEDI, S. M. A structured review of sparse fast Fourier transform algorithms. **Digital Signal Processing**, Elsevier, p. 103403, 2022.

140 MOHAPATRA, B. N.; MOHAPATRA, R. K. FFT and sparse FFT techniques and applications. In: **Fourteenth International Conference on Wireless and Optical Communications Networks**. [S.l.: s.n.], 2017. p. 1–5.

141 AGARWAL, A. *et al.* High-throughput implementation of a million-point sparse Fourier transform. In: **24th International Conference on Field Programmable Logic and Applications**. [S.l.: s.n.], 2014. p. 1–6.

142 SCHUMACHER, J. **High performance sparse fast Fourier transform**. Dissertação (Mestrado) — Computer Science, ETH Zurich, Switzerland, 2013.

143 HASSANIEH, H. *et al.* GHz-wide sensing and decoding using the sparse Fourier transform. In: **IEEE Conference on Computer Communications**. [S.l.: s.n.], 2014. p. 2256–2264.

144 KANDERS, H. *et al.* A 1 million-point FFT on a single FPGA. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v. 66, n. 10, p. 3863–3873, 2019.

145 CHEN, W.-H.; SMITH, C.; FRALICK, S. A fast computational algorithm for the discrete cosine transform. **IEEE Transactions on Communications**, v. 25, n. 9, p. 1004–1009, 1977.

146 DUHAMEL, P.; HOLLMANN, H. Split-radix FFT algorithm. **Electronics Letters**, IET Digital Library, v. 20, n. 1, p. 14–16, 1984.

147 SORENSEN, H. V.; HEIDEMAN, M.; BURRUS, C. On computing the split-radix FFT. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, IEEE, v. 34, n. 1, p. 152–156, 1986.

148 DUHAMEL, P. Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, IEEE, v. 34, n. 2, p. 285–295, 1986.

149 SUN, C.; YIP, P. Split-radix algorithms for DCT and DST. In: IEEE COMPUTER SOCIETY. **Twenty-Third Asilomar Conference on Signals, Systems and Computers, 1989.** [S.l.], 1989. v. 1, p. 508–509.

150 COELHO, D. F. *et al.* Low-complexity scaling methods for DCT-II approximations. **IEEE Transactions on Signal Processing**, IEEE, v. 69, p. 4557–4566, 2021.

151 PERERA, S. M.; LIU, J. Complexity reduction, self/completely recursive, radix-2 DCT I/IV algorithms. **Journal of Computational and Applied Mathematics**, v. 379, p. 112936, 2020.

152 ESCHENAUER, H.; KOSKI, J.; OSYCZKA, A. **Multicriteria design optimization: Procedures and applications**. 2nd. ed. Heidelberg, BW: Springer Science & Business Media, 2012.

153 M., J.; ALFALOU, A.; MEHER, P. K. A generalized algorithm and reconfigurable architecture for efficient and scalable orthogonal approximation of DCT. **IEEE Transactions on Circuits and Systems I**, v. 62, n. 2, p. 449–457, jan. 2015.

154 BRITANAK, V.; YIP, P.; RAO, K. R. **Discrete cosine and sine transforms**. 1st. ed. Boca Raton, FL: Academic Press, 2006.

155 LEVITIN, A. **Introduction to the design and analysis of algorithms**. 3rd. ed. Harlow, UK: Addison Wesley, 2011.

156 MATHWORKS. **MATLAB R2022b Documentation "round"**. 2022. Https://www.mathworks.com/help/matlab/ref/round.

157 HORN, R. A.; JOHNSON, C. R. **Matrix analysis**. 2nd. ed. New York, NY: Cambridge university press, 2012.

158 ZURAS, D. *et al.* IEEE standard for floating-point arithmetic. **IEEE Std**, v. 754, n. 2008, p. 1–70, 2008.

159 LUSTIG, M.; DONOHO, D.; PAULY, J. M. Sparse MRI: The application of compressed sensing for rapid MR imaging. **Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine**, Wiley Online Library, v. 58, n. 6, p. 1182–1195, 2007.

160 ABO-ZAHHAD, M. M. *et al.* Compressive sensing algorithms for signal processing applications: A survey. **International journal of communications, network and system sciences**, Scientific Research Publishing, v. 8, n. 06, p. 197, 2015.

161 STEIDL, G. A note on fast Fourier transforms for nonequispaced grids. **Advances in computational mathematics**, Springer, v. 9, p. 337–352, 1998.

162 YANG, S.-C.; WANG, Y.-L. A hybrid MPI-CUDA approach for nonequispaced discrete Fourier transformation. **Computer Physics Communications**, Elsevier, v. 258, p. 107513, 2021.

163 PRANDONI, P.; VETTERLI, M. **Signal processing for communications**. 1st. ed. Boca Raton, FL: EPFL press, 2008.

164 HIGHAM, N. J. Computing the polar decomposition—with applications. **SIAM Journal on Scientific and Statistical Computing**, SIAM, v. 7, n. 4, p. 1160–1174, 1986.

165 HIGHAM, N. J. Computing real square roots of a real matrix. **Linear Algebra and its applications**, Citeseer, v. 88, p. 405–430, 1987.

166 MALVAR, H. *et al.* Low-complexity transform and quantization with 16-bit arithmetic for H. 26L. In: IEEE. **Proceedings. International Conference on Image Processing**. [S.l.], 2002. v. 2, p. 2.

167 SCHOTT, J. R. **Matrix analysis for statistics**. 3rd. ed. Hoboken, NJ: John Wiley & Sons, 2016.

168 BARTLE, R. G.; SHERBERT, D. R. **Introduction to real analysis**. 4th. ed. Hoboken, NJ: Wiley New York, 2000. v. 2.

169 POTLURI, U. *et al.* Multiplier-free DCT approximations for RF multi-beam digital aperture-array space imaging and directional sensing. **Measurement Science and Technology**, IOP Publishing, v. 23, n. 11, p. 114003, 2012.

170 TABLADA, C.; BAYER, F. M.; CINTRA, R. J. A class of DCT approximations based on the Feig–Winograd algorithm. **Signal Processing**, Elsevier, v. 113, p. 38–51, 2015.

171 HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting: Principles and practice**. 2nd. ed. Wellington Rd, AU: OTexts, 2018.

172 FLURY, B. N.; GAUTSCHI, W. An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form. **SIAM Journal on Scientific and Statistical Computing**, SIAM, v. 7, n. 1, p. 169–184, 1986.

173 WATKINS, D. S. **Fundamentals of matrix computations**. 2nd. ed. New York, NY: John Wiley & Sons, 2004. v. 64.

174 BARICHARD, V.; GANDIBLEUX, X.; T'KINDT, V. **Multiobjective programming and goal programming: Theoretical results and practical applications**. Heidelberg, BW: Springer Science & Business Media, 2008. v. 618.

175 JOHNSON, S. G.; FRIGO, M. A modified split-radix FFT with fewer arithmetic operations. **IEEE Transactions on Signal Processing**, IEEE, v. 55, n. 1, p. 111–119, 2006.

176 QADEER, S. *et al.* A radix-2 DIT FFT with reduced arithmetic complexity. In: IEEE. **International Conference on Advances in Computing, Communications and Informatics**. [S.l.], 2014. p. 1892–1896.

177 FRIGO, M.; JOHNSON, S. G. **The fastest Fourier transform in the West**. [S.l.], 1997.

178 EATON, J. W.; BATEMAN, D.; HAUBERG, S. **GNU Octave Manual Version 3**. [S.l.]: Network Theory Ltd., 2008.

179 RESEARCH, W. **Round**. 2007. https://reference.wolfram.com/language/ref/Round.

180 HEWLITT, R.; SWARTZLANTLER, E. Canonical signed digit representation for FIR digital filters. In: **IEEE Workshop on Signal Processing Systems**. [S.l.: s.n.], 2000. p. 416–426.

181 AVIZIENIS, A. Signed-digit number representations for fast parallel arithmetic. **IRE Transactions on electronic computers**, IEEE, n. 3, p. 389–400, 1961.

182 RADER, C. Discrete Fourier transforms when the number of data samples is prime. **Proceedings of the IEEE**, v. 56, n. 6, p. 1107–1108, 1968.

183 SILVA, G. J.; SOUZA, R. C. Cyclotomic basis for computing the discrete Fourier transform. In: **International Telecommunications Symposium**. [S.l.: s.n.], 2010. v. 7, p. 1–5.

184 DEMPSTER, A. G.; MACLEOD, M. D. Constant integer multiplication using minimum adders. **IEE Proceedings-Circuits, Devices and Systems**, IET, v. 141, n. 5, p. 407–413, 1994.

185 RUIZ, G. A.; GRANDA, M. Efficient canonic signed digit recoding. **Microelectronics Journal**, Elsevier, v. 42, n. 9, p. 1090–1097, 2011.

186 ARNO, S.; WHEELER, F. S. Signed digit representations of minimal hamming weight. **IEEE Transactions on Computers**, IEEE, v. 42, n. 8, p. 1007–1010, 1993.

187 LIU, H.; LEE, H. A high performance four-parallel 128/64-point radix-24 FFT/IFFT processor for MIMO-OFDM systems. In: **IEEE Asia Pacific Conference on Circuits and Systems**. [S.l.: s.n.], 2008. p. 834–837.

188 TANAKA, Y. Efficient signed-digit-to-canonical-signed-digit recoding circuits. **Microelectronics Journal**, Elsevier, v. 57, p. 21–25, 2016.

189 PROAKIS, J. G. **Digital signal processing: Principles, algorithms, and applications**. 4th. ed. Upper Saddle River,NJ: Pearson Education, 2007.

190 WAKERLY, J. F. **Digital design: Principles and practices**. 4rd. ed. New York, NY: Pearson Education, 2018.

191 CHEN, N. *et al.* MIMO radar transmit waveform design for joint optimizing beampattern synthesis and spatial autocorrelation performance. In: **IEEE International Conference on Computer and Communication Systems**. [S.l.: s.n.], 2021. p. 621–625.

192 QIN, L.; VOROBYOV, S. A.; DONG, Z. Joint cancelation of autocorrelation sidelobe and cross correlation in MIMO-SAR. **IEEE Geoscience and Remote Sensing Letters**, v. 14, n. 6, p. 931–935, 2017.

193 POPOVIć, B. M. Optimum sets of interference-free sequences with zero autocorrelation zones. **IEEE Transactions on Information Theory**, v. 64, n. 4, p. 2876–2882, 2018.

194 VAN ETTEN, W. C. **Introduction to random signals and noise**. 1st. ed. Hoboken, NJ: John Wiley and Sons, 2006.

195 ALKU, P.; SAEIDI, R. The linear predictive modeling of speech from higher-lag autocorrelation coefficients applied to noise-robust speaker recognition. **IEEE/ACM Transactions on Audio, Speech, and Language Processing**, v. 25, n. 8, p. 1606–1617, 2017.

196 BROERSEN, P. M. The quality of lagged products and autoregressive Yule–Walker models as autocorrelation estimates. **IEEE Transactions on Instrumentation and Measurement**, IEEE, v. 58, n. 11, p. 3867–3873, 2009.

197 BEHERA, S.; MOHANTY, M. N. Template matching based artifact detection from the brain signals. IET, 2021.

198 APANDI, Z. F. M. *et al.* Noise reduction method based on autocorrelation for threshold-based heartbeat detection. In: **2020 International Conference on Advanced Mechatronic Systems (ICAMechS)**. [S.l.: s.n.], 2020. p. 83–88.

199 LASOTA, A.; MELLER, M. Iterative learning approach to active noise control of highly autocorrelated signals with applications to machinery noise. **IET Signal Processing**, IET, v. 14, n. 8, p. 560–568, 2020.

200 HUANG, Z.; MA, Z.; HUANG, G. Radar waveform recognition based on multiple autocorrelation images. **IEEE Access**, v. 7, p. 98653–98668, 2019.

201 NIRANJAN, R.; RAO, C. R.; SINGH, A. Improved parameters estimation of radar pulses using autocorrelation. In: **2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)**. [S.l.: s.n.], 2021. p. 1–4.

202 YANG, C. *et al.* LPI radar signal detection based on the combination of FFT and segmented autocorrelation plus PAHT. **Journal of Systems Engineering and Electronics**, v. 28, n. 5, p. 890–899, 2017.

203 TEDESSO, T. W.; ROMERO, R. Code shift keying based joint radar and communications for EMCON applications. **Digital Signal Processing**, Elsevier, v. 80, p. 48–56, 2018.

204 GAMBA, J. **Radar signal processing for autonomous driving**. 1st. ed. Conway, AR: Springer, 2020.

205 SKOLNIK, M. I. **Radar handbook**. 3rd. ed. New York, NY: McGraw-Hill Education, 2008.

206 SHIMAMURA, T.; KOBAYASHI, H. Weighted autocorrelation for pitch extraction of noisy speech. **IEEE Transactions on Speech and Audio Processing**, v. 9, n. 7, p. 727–730, 2001.

207 FARAHANI, G. Autocorrelation-based noise subtraction method with smoothing, overestimation, energy, and cepstral mean and variance normalization for noisy speech recognition. **EURASIP Journal on Audio, Speech, and Music Processing**, SpringerOpen, v. 2017, n. 1, p. 1–16, 2017.

208 KOLOKOLOV, A. S.; LYUBINSKII, I. Measuring the pitch of a speech signal using the autocorrelation function. **Automation & Remote Control**, v. 80, n. 2, 2019.

209 ZAREI, A.; ASL, B. M. Performance evaluation of the spectral autocorrelation function and autoregressive models for automated sleep apnea detection using single-lead ECG signal. **Computer Methods and Programs in Biomedicine**, Elsevier, v. 195, p. 105626, 2020.

210 MOEYERSONS, J. *et al.* Evaluation of a continuous ECG quality indicator based on the autocorrelation function. In: **2018 Computing in Cardiology Conference (CinC)**. [S.l.: s.n.], 2018. v. 45, p. 1–4.

211 ZAHIRI-AZAR, R.; SALCUDEAN, S. E. Time-delay estimation in ultrasound echo signals using individual sample tracking. **IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control**, v. 55, n. 12, p. 2640–2650, 2008.

212 BURT, J. B. *et al.* Generative modeling of brain maps with spatial autocorrelation. **NeuroImage**, Elsevier, v. 220, p. 117038, 2020.

213 ZHANG, Q.; ZHOU, D.; ZENG, X. HeartID: A multiresolution convolutional neural network for ECG-based biometric human identification in smart health applications. **IEEE Access**, v. 5, p. 11805–11816, 2017.

214 KHOSHNEVIS, S. A.; SANKAR, R. Applications of higher order statistics in electroencephalography signal processing: A comprehensive survey. **IEEE Reviews in Biomedical Engineering**, v. 13, p. 169–183, 2020.

215 HAN, J.; ORSHANSKY, M. Approximate computing: An emerging paradigm for energy-efficient design. In: **IEEE European Test Symposium (ETS)**. [S.l.: s.n.], 2013. p. 1–6.

216 XU, Q.; MYTKOWICZ, T.; KIM, N. S. Approximate computing: A survey. **IEEE Design Test**, v. 33, n. 1, p. 8–22, 2016.

217 VENKATARAMANI, S. *et al.* Approximate computing and the quest for computing efficiency. In: **2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.: s.n.], 2015. p. 1–6.

218 MITTAL, S. A survey of techniques for approximate computing. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 48, n. 4, p. 1–33, 2016.

219 BOX, G. E. *et al.* **Time series analysis: Forecasting and control**. 5th. ed. Hoboken, NJ: John Wiley & Sons, 2015.

220 KAY, S. M. **Fundamentals of statistical signal processing: Estimation theory**. 1st. ed. Upper Saddle River, NJ: Prentice-Hall, Inc., 1993.

221 BILLINGSLEY, P. **Measure and probability**. 3rd. ed. New York, NY: John Wiley & Sons, 1995.

222 SMITH, J. O. **Mathematics of the discrete Fourier transform (DFT)**. 2023. https://ccrma.stanford.edu/%7Ejos/mdft/Convolution_Theorem. Online book, 2007 edition.

223 ORFANIDIS, S. J. **Introduction to signal processing**. Us. New Jersey, NJ: Prentice-Hall, Inc., 1995.

224 MONTGOMERY, D. C. **Introduction to statistical quality control**. 6th. ed. Hoboken, NJ: John Wiley & Sons, 2009.

225 BOX, G. E.; JENKINS, G. M.; REINSEL, G. C. **Time series analysis: Forecasting and control**. 4th. ed. Hoboken, NJ: John Wiley & Sons, 2008.

226 HAMILTON, J. D. **Time series analysis**. 1st. ed. Princeton, NJ: Princeton university press, 1994.

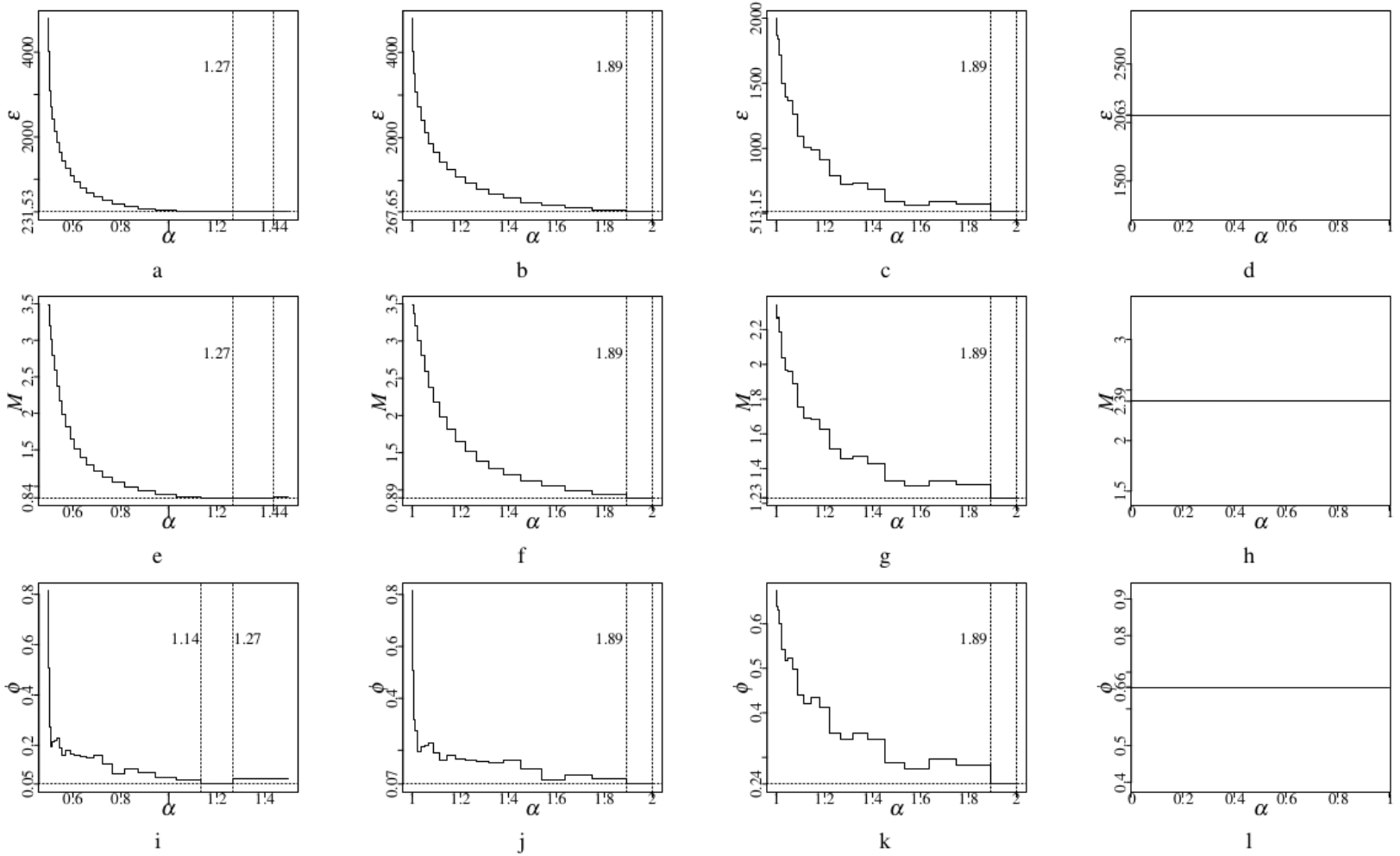
227 BORGES, A. *et al.* Low-complexity architecture for AR(1) inference. **Electronics Letters**, Wiley Online Library, v. 56, n. 14, p. 732–734, 2020.

228 CANTY, A.; RIPLEY, B. D. **Boot: Bootstrap R (S-Plus) Functions**. [S.l.], 2022. R package version 1.3-28.1.

229 DAVISON, A. C.; HINKLEY, D. V. **Bootstrap Methods and Their Applications**. 1st. ed. Cambridge,UK: Cambridge University Press, 1997.

230 RADUNZ, A. P. *et al.* Extensions on low-complexity DCT approximations for larger blocklengths based on minimal angle similarity. **Journal of Signal Processing Systems for Signal, Image, and Video Technology**, Springer, forthcoming.

Figure 47 – Performance of the integer functions for $N = 31$ and $\mathscr{P} = \{-1, 0, 1\}$ considering round$(\cdot)$ (47a, 47e, and 47i), trunc$(\cdot)$ (47b, 47f, and 47j), floor$(\cdot)$ (47c, 47g, and 47k), and ceil$(\cdot)$ (47d, 47h, and 47l)



Source: Author (2023).

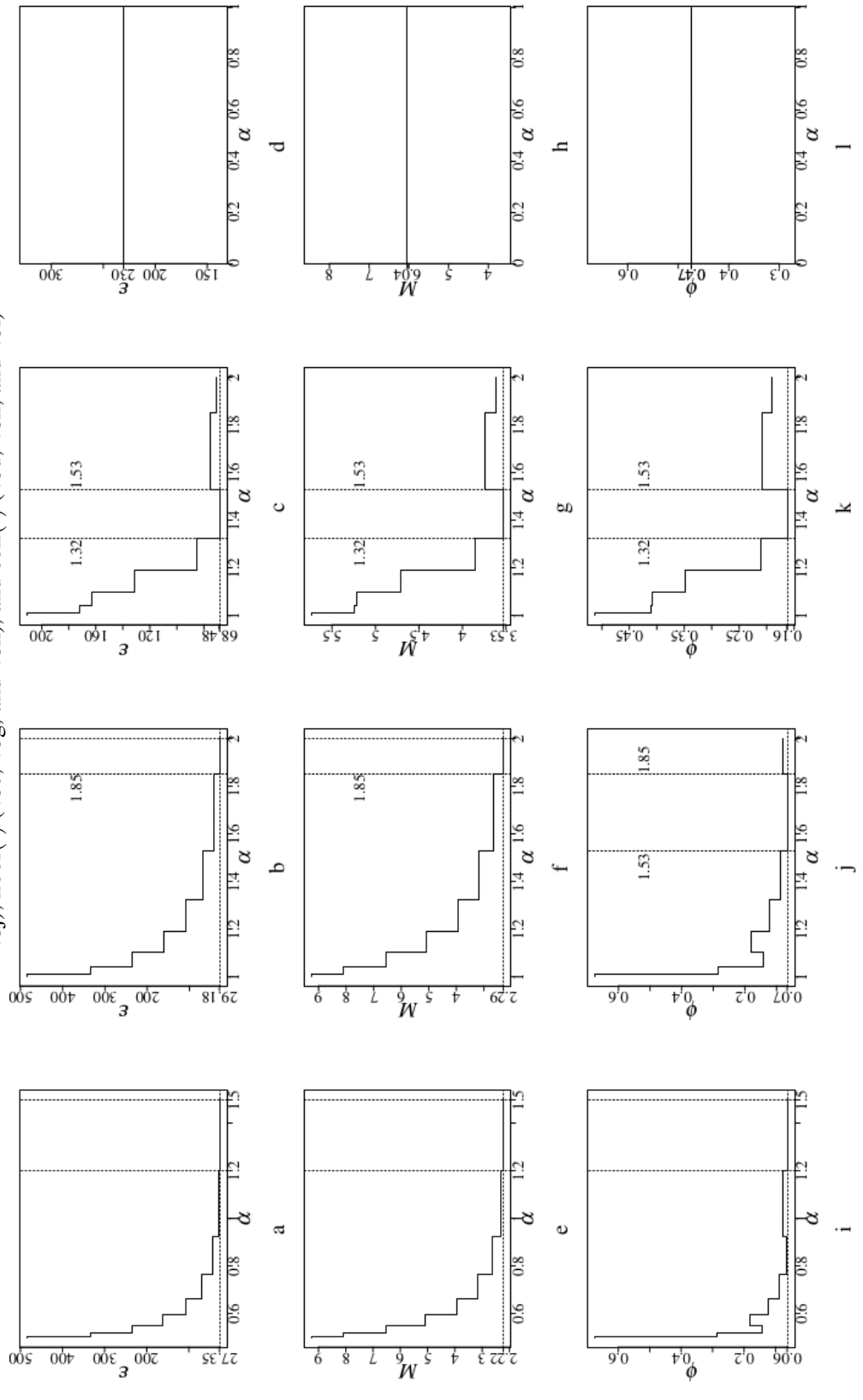


Figure 48 – Performance of the integer functions for $N = 11$ and $\mathscr{P} = \{-1, 0, 1\}$ considering round$(\cdot)$ (48a, 48e, and 48i), trunc$(\cdot)$ (48b, 48f, and 48j), floor$(\cdot)$ (48c, 48g, and 48k), and ceil$(\cdot)$ (48d, 48h, and 48l)

Source: Author (2023).

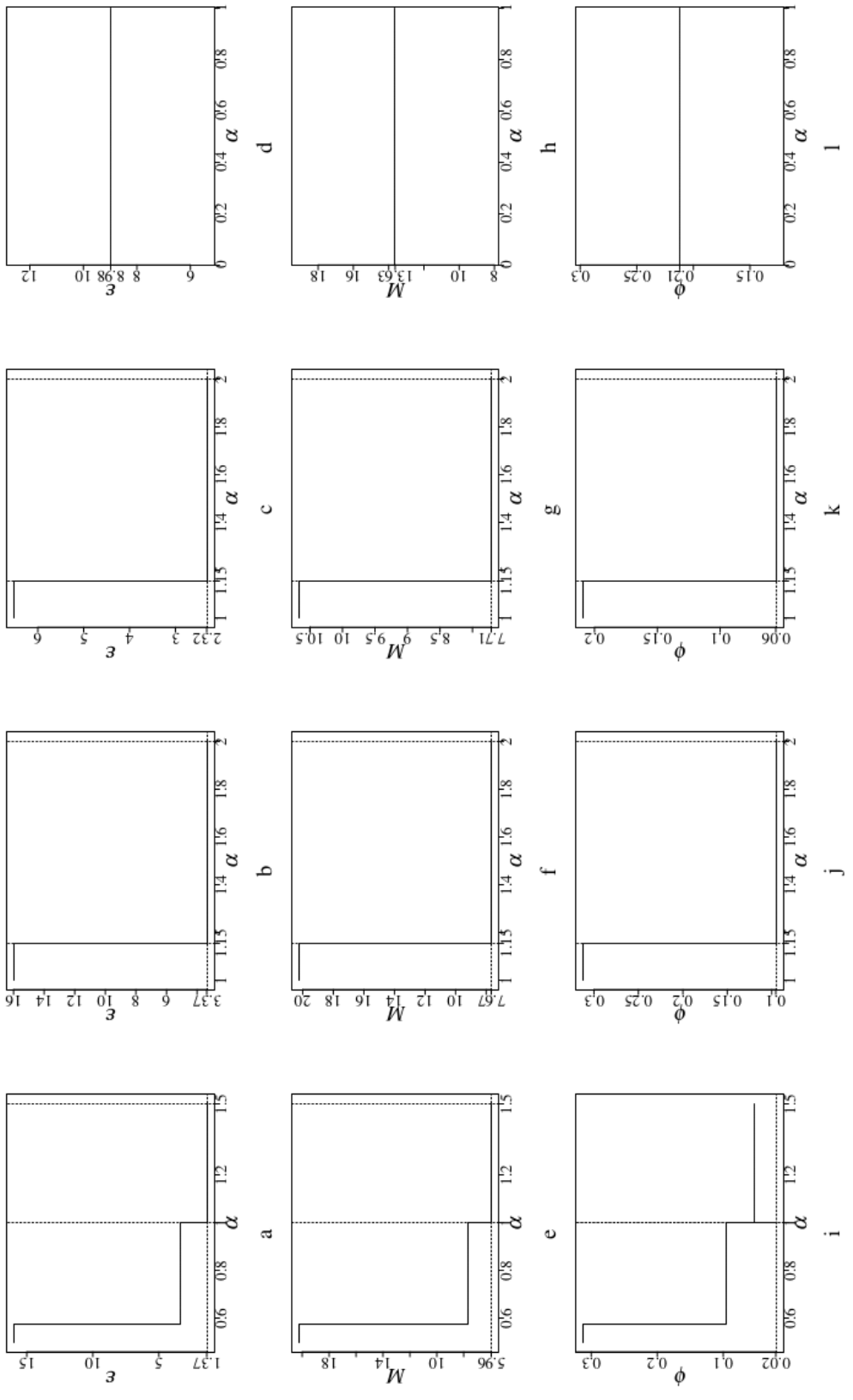


Figure 49 – Performance of the integer functions for $N = 3$ and $\mathscr{P} = \{-1, 0, 1\}$ considering round$(\cdot)$ (49a, 49e, and 49i), trunc$(\cdot)$ (49b, 49f, and 49j), floor$(\cdot)$ (49c, 49g, and 49k), and ceil$(\cdot)$ (49d, 49h, and 49l)

Source: Author (2023).