Processing algorithms for signals with time-varying frequencies.

Ju, Ying Tuo.

2009

Ju, Y. T. (2009). Processing algorithms for signals with time-varying frequencies. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/41750>

<https://doi.org/10.32657/10356/41750>

Processing Algorithms for Signals with Time-varying Frequencies

Ju Yingtuo

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2009

To my loving and supportive family.

Acknowledgements

Firstly, I would like to thank my supervisor, Dr. Guoan Bi, for his invaluable support and guidance throughout my candidature of PhD period and the course of this dissertation. I am so grateful that he agreed to supervise my PhD study when my former supervisor, Dr. Braham Barkat, resigned from the university. His continuous encouragement, enlightened direction, and extensive knowledge make my research work smoothly and positively.

Secondly, I want to thank my family who showed so much concern and care for me in the course of my studies. Without their persistent emotional support, the achievement of my thesis is impossible.

Last but not the least, I am very grateful to my university, Nanyang Technological University, for its facilities and financial supports.

Summary

In many real-life applications, such as radar, sonar, communications, and biomedical engineering, the signals under consideration are often characterized by time-varying spectral contents. For this type of signals, time analysis or frequency analysis alone cannot fully describe the time-varying characteristics of the signals.

The tools for the analysis of signals with time-varying frequencies can be generally divided into two categories: parametric and non-parametric methods. The former models the signal's instantaneous phase as a polynomial function of time and need to estimate the phase coefficients. The later represents the signal's energy over both time and frequency or over $t - \omega^0(t)$ space, where $\omega^0(t) = (\Omega(t), \Omega^1(t), \dots, \Omega^{m-1}(t))$ is the vector of the instantaneous frequency (IF) $\Omega(t)$ and its derivatives. During the studies of various signal processing tools, we have found that the polynomial time frequency transform (PTFT) is a good tool to use. Its advantages mainly come from two aspects, one is that PTFT can be used to compute Maximum Likelihood Estimation (MLE) of the phase coefficients, and the other is that it can be combined with other methods to analyze signals with time-varying frequencies, especially in the case that there is no prior information on the signals under study.

However, PTFT generally requires a huge computational complexity as it involves in multi-dimensional (MD) calculation. One of the primary objectives of this research project is to propose fast algorithms to calculate PTFTs efficiently. We have proposed a general class of fast algorithms for the computation of the PTFTs of complex-valued length- $a^p b$ input sequence, where a , b and p are integers. The proposed algorithms provide the flexibility to support various input sequence lengths by setting the parameters of a , b and p , and have regular computational structures for an easy implementation. Analysis and comparison on the computational complexity are also reported in terms of the numbers of complex additions and complex multiplications. It is verified that the computational complexity of the PTFTs is significantly reduced by using the proposed algorithms.

Next, fast algorithms for the computation of the PTFTs of real-valued sequences are presented. Similar to the fast algorithms for complex-valued input sequences, the proposed algorithms can support various input sequence lengths by setting the parameters of a , b and p , and have regular computational structures. Analysis and comparison on the computational complexity are made in terms of the numbers of real additions and real multiplications. With the conjugate symmetric property, the presented fast algorithms, with $a = 3, 4$ and 8 , effectively reduce the computational complexity to be about one-half of that needed by the reported fast algorithms for complex-valued sequences.

In many practical applications, real-time processing is required. Under these conditions, the software implementation may not be fast enough. To support high processing throughput, it is necessary to implement the fast algorithms in hardware. We propose a hardware-oriented radix-2² algorithm for PTFT of

input sequence whose length is a power of 2. In addition, a single-path pipelined hardware structure is presented, and the hardware requirement of the proposed architecture is analyzed.

Finally, we work on applying signal processing techniques to communications. In particular, we present a carrier frequency offset (CFO) estimation algorithm based on lag diversity techniques for the orthogonal frequency division multiplexing (OFDM) system. Compared with other reported estimators, such as single- h , and pilot assisted estimator, theoretical analysis and simulations show that the proposed method effectively achieves a wide estimation range with reduced mean square errors (MSEs).

Contents

Acknowledgements	i
Summary	ii
List of Figures	x
List of Tables	xiv
List of Abbreviations	xvi
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	5
1.3 Major Contributions	6
1.4 Organization of the Thesis	7
2 Background	10
2.1 Introduction	10
2.2 Notation	11
2.3 Signals with Time-varying Frequencies	12

<i>Contents</i>	vi
2.4 Parametric Methods	14
2.4.1 Polynomial-Phase Signal and Its CRLB	14
2.4.2 Maximum Likelihood (ML) Method	15
2.4.3 Suboptimal Maximum Likelihood Methods	18
2.4.4 Advantages of the ML Methods	22
2.5 Non-parametric Methods	25
2.5.1 Analytic Signal and Instantaneous Frequency (IF)	25
2.5.2 Short-Time Fourier Transform (STFT)	27
2.5.3 Quadratic and High-order Time-Frequency Distributions (TFDs)	28
2.5.4 Local Polynomial Time-Frequency Transform	31
2.6 Fast Fourier Transform Algorithms	33
3 Study of Signals with Time-varying Frequencies In Gaussian and Impulsive Noises	39
3.1 Introduction	39
3.2 Study in Gaussian Noise	40
3.3 Study in Impulsive Noise	47
3.3.1 Problem Model	48
3.3.2 M -Estimation Based Methods	50
3.3.3 L -Estimation Based Methods	53
3.3.4 Clipping Method	54
3.3.5 Numerical Study	56
3.4 Summary	65

<i>Contents</i>	vii
4 Fast Algorithms for PTFTs of Complex-Valued Sequences	67
4.1 Introduction	67
4.2 Review of the Reported Fast PTFT Algorithms	68
4.3 Properties and Decomposition	71
4.3.1 The Symmetric Property	72
4.3.2 First Step Decomposition (FSD)	74
4.4 Fast Algorithms	77
4.4.1 Radix- a Algorithm	77
4.4.2 Split-radix- $m/(ma)$ Algorithm	80
4.5 Computational Complexity	82
4.5.1 Computation Details	84
4.5.1.1 a is odd	84
4.5.1.2 a is even	87
4.5.2 Computation Complexity for the 2nd Order PTFT	90
4.5.3 Computation Complexity for the 3rd Order PTFT	92
4.6 Summary	95
5 Fast Algorithms for PTFTs of Real-Valued Sequences	96
5.1 Introduction	96
5.2 Properties and Decomposition	99
5.2.1 The Hermitian and Symmetric Properties	99
5.2.2 First Step Decomposition (FSD)	100
5.3 Radix- a Algorithms	104

<i>Contents</i>	viii
5.4 Computational Complexity	108
5.5 Split-radix Algorithms	116
5.6 Summary	121
6 Pipeline Structure for PTFT Computation	122
6.1 Introduction	122
6.2 Algorithms and Pipeline Structures for FFT	124
6.2.1 Algorithms	125
6.2.2 Reported Pipelined Structures	127
6.2.3 Proposed Structure	130
6.2.3.1 Delay and commutator unit	132
6.2.3.2 Butterfly computation	135
6.2.4 Discussions	136
6.3 Algorithms and Pipeline Structures for PTFT	137
6.3.1 Algorithms	137
6.3.2 Pipelined Structure	142
6.3.2.1 Delay and Commutator Units	142
6.3.2.2 Controllable Adder/Subtractor Units	150
6.3.3 Discussions	150
6.4 Summary	153
7 Signal Processing for OFDM	154
7.1 Introduction	154
7.2 Signal Model	156

<i>Contents</i>	ix
7.3 Data-aided CFO Estimation	158
7.4 CFO Estimation with Lag Diversity	160
7.5 Simulations	163
7.6 Summary	169
8 Conclusions and Future Work	170
8.1 Conclusions	170
8.2 Recommendations for Further Research	174
8.2.1 Properties and Practical Implementations of PTFT . . .	174
8.2.2 Application of Time-Frequency Analysis for Signals with Time-varying Frequencies	175
8.2.3 Robust Receiver Design in Wireless Communication Sys- tems	175
Appendix A - Subroutine for 2nd order PTFT based on DIF radix- a algorithm	177
Appendix	177
Author's Publications	185
Bibliography	187

List of Figures

2.1	Time domain representation (a) of the bat sonar signal and its spectrum (b).	13
2.2	PTFT of the sum of two 2nd order PPSs having the same amplitude.	18
2.3	Efficiency of the estimates, (a) $\hat{a}_1$ and (b) $\hat{a}_2$, versus SNR. . . .	23
2.4	The output SNR, normalized by N , versus input SNR for (a) 2nd order and (b) 3rd order PPS. Dashed line: PTFT, dotted line: HAF, and solid line: IGAF with (1) $N = 1024$, (2) $N = 256$, (3) $N = 64$, (4) $N = 16$	24
3.1	MSE of IF estimators for WVD and HAF.	42
3.2	IF estimation using HAF (left column) and PHAF (right column). The first row assumes 2nd-order phase modeling, and the second row assumes 3rd-order phase modeling.	43
3.3	IF estimation based on the peak of TFRs. (left) STFT, (middle) WVD and (right) MLPTFT.	44
3.4	(a) IF, (b) normalized $ \text{STFT}(t,f) $, (c) normalized $ \text{WVD}(t,f) $, and (d) normalized $ \text{MLPTFT}(t,f) $ of a multicomponent PPS. . .	45

3.5	The 3rd-order HAF of noiseless unit amplitude $x_5(n)$ is shown in the top plot. The 3rd-order HAF and 3rd-order PHAF of $x(n)$ are shown in the middle and bottom plots, respectively. The x-axis: frequency, the y-axis: magnitude.	46
3.6	(a) Observations of the signal and noise, x-axis: time, y-axis: amplitude, (b) standard and robust forms of $\text{mlHAF}[x, f, \tau]$, x-axis: scaled frequency, y-axis: normalized magnitude.	58
3.7	(a) Observations of the signal and noise, x-axis: time, y-axis: amplitude, (b) images of normalized standard and robust forms of LPP at $t = 0.2\text{s}$, x-axis: $\Omega(t)$ (IF) and y-axis: $\Omega^{(1)}(t)$ (first derivative of IF).	60
3.8	IF estimates obtained from standard and robust forms of LPP, x-axis: time t and y-axis: IF $\Omega(t)$	61
3.9	(a) Observations of the signal and noise, x-axis: time, y-axis: amplitude, (b) standard and robust modified LPTFT (MLPTFT), x-axis: time, and y-axis: frequency.	63
5.1	The normalized $ X(k_0, k_1) ^2$ of a real-valued signal with two chirp components (SNR = 3 dB).	98
5.2	Decomposition process for a 2nd order PTFT when p_1 is odd. .	101
5.3	Decomposition process for a 2nd order PTFT when p_1 is even. .	102
5.4	The ratios between the numbers of multiplications needed by the fast 2nd order PTFT (with dimensional sizes $N \times N$) algorithms for real- and complex-valued inputs.	114
5.5	The ratios between the numbers of multiplications needed by the fast 2nd order PTFT (with dimensional sizes $N \times N_1$) algorithms for real- and complex-valued inputs.	116

<i>List of Figures</i>	xii
5.6 The ratios between the numbers of additions needed by the the fast 2nd order PTFT (with dimensional sizes $N \times N_1$) algorithms of real- and complex-valued inputs.	117
6.1 Signal flow graph of (a) radix-2, (b) radix-4, (c) radix-2 ² fast algorithm ($N=16$).	128
6.2 Block diagram of the single-path pipelined structure based on the radix-2 ² algorithm ($N = 16$).	131
6.3 Block diagram of delay and commutator.	133
6.4 Block diagram of modified adder/subtractor unit.	134
6.5 Signal flow graph of the radix-2 ² fast algorithm for PTFT of length-16 input sequence.	141
6.6 Block diagram of the radix-2 ² fast algorithm for PTFT of length-16 input sequence.	143
6.7 Block diagram of the second type of delay and commutator unit (DACII).	144
6.8 Block diagram of the index reorder unit.	144
6.9 Operations of the second type of delay and commutator unit (DACII).	145
6.10 Operations of the index reorder unit after stage 2 for $N = 16$. .	146
6.11 Block diagram of the second type of modified adder/subtractor unit (MASII).	151
7.1 MSE versus h , $\nu = 0.15$ in AWGN channel.	165
7.2 MSE of single- h estimator versus ν in AWGN channel	166
7.3 MSE of the proposed estimator versus ν in AWGN channel. . .	166

7.4 $\Pr(\hat{m} \neq m)$ of PAE versus SNR in AWGN and multipath channel. 167

7.5 Performance of the proposed estimator, the single- h estimator
and the PAE in AWGN and multipath channels, $\nu = 3.25$ 167

List of Tables

3.1	Comparison of computation time in impulsive noise.	65
4.1	The number of arithmetic operations required by the FSD. . . .	77
4.2	The computational complexity required by $u_a(n, l_0, l_1, \dots, l_r)$. .	90
4.3	The number of complex multiplications for the 2nd order PTFT	92
4.4	The number of complex additions for the 2nd order PTFT . . .	93
4.5	The number of complex multiplications for the 3rd order PTFT	94
5.1	The computational overheads needed by the FSD for a 2nd order PTFT.	104
5.2	The computational complexity required by $u_a(n, l_0, l_1, \dots, l_r)$ for real-valued input.	109
5.3	The number of real multiplications for the length- N 2nd order PTFT of real-valued sequence	113
5.4	The number of real additions for the length- N 2nd order PTFT of real-valued sequence	113
5.5	The number of real multiplications and additions required by the split-radix algorithm	121

6.1	Operations of the DAC unit at stage 1 for $N=16$	132
6.2	Operations of addition and subtraction at the first stage for $N=16$	135
6.3	Comparisons among different pipelined implementations.	137
6.4	Operations of the DACII unit after stage 2 for $N=16$	149
6.5	Operations required by the controllable adder/subtractor.	150

List of Abbreviations

CRLB	Cramer-Rao low bound
CFO	carrier frequency offset
DCFT	discrete chirp Fourier transform
DFT	discrete Fourier transform
DIF	decimation in frequency
DIT	decimation in time
FFT	fast Fourier transform
FM	frequency modulated
FRFT	fractional Fourier transform
HAF	high order ambiguity function
HIM	high order instantaneous moment
IF	instantaneous frequency
i.i.d	independent and identically distributed
IGAF	integrated generalized ambiguity function
LFM	linear frequency modulated

LPA	local polynomial approximation
LPTFT	local polynomial time frequency transform
MAD	median absolute deviation about the median
MLE	maximum likelihood estimation
MSE	mean square error
MLPTFT	modified local polynomial time frequency transform
OFDM	orthogonal frequency division multiplexing
PHAF	product high-order ambiguity function
PPS	polynomial-phase signal
PPT	polynomial-phase transform
PTFT	polynomial time frequency transform
PWVD	polynomial Wigner-Ville distribution
SNR	signal to noise energy ratio
STFT	short-time Fourier transform
TFD	time-frequency distribution
WGN	white Gaussian noise
WVD	Wigner-Ville distribution

Chapter 1

Introduction

1.1 Motivation

Although most signals are generated, analyzed and processed in the time domain, their characteristics are often better understood in the frequency domain, which is generally known as frequency or spectral analysis. The time and frequency representations of signals are related by the Fourier transform. In many real-life applications, such as radar, sonar, communications, automotive signal analysis, and biomedical engineering, the signals under consideration have time-varying spectral characteristics. For this type of signals, time analysis or frequency analysis alone cannot fully describe the time-varying characteristics of the signals.

Available tools for the study of signals with time-varying frequencies generally fall into two categories: parametric and non-parametric methods. The former, is motivated by the Weierstrass approximation theorem, which implies that for a finite duration of observations, an arbitrary time varying phase can be well

approximated by a polynomial of sufficient order. Methods within this category focus on the estimation of the polynomial phase coefficients, and include the maximum likelihood (ML) estimator based on the polynomial time-frequency transform (PTFT) [1–4] and suboptimal methods such as those based on high-order ambiguity function (HAF) [5–9] and its extensions [10, 11]. The latter typically implies the time-frequency representation (TFR) to reveal the evolution of signal content as a function of both time and frequency or over $t - \omega^0(t)$ space, where $\omega^0(t) = (\Omega(t), \Omega^1(t), \dots, \Omega^{m-1}(t))$ is the vector of the instantaneous frequency (IF) $\Omega(t)$ and its derivatives. There exist many TFRs [12–16]. The choice of a TFR depends on the specific application at hand and the available properties of the TFR.

Most applications require the estimation of the coefficients which usually carry the most desirable and relevant information, as shown in the following examples:

- In conventional synthetic aperture radar (SAR) imaging, it is assumed that the received slow-time signal of a discrete scattering point in a synthetic aperture time (SAT) may be appropriately approximated as a 2nd order polynomial phase signal (PPS) [17, p.168]. More recently, there is an increasing need for high resolution at low frequencies to recognize targets moving on the ground, possibly hidden under foliage [18]. In such a case, longer observation intervals are needed to model the received signal as higher order PPS [11]. Estimation of the phase coefficients can be used to obtain the radial acceleration (and higher order derivatives, if desired), then to obtain focused and properly positioned images.
- In some communication scenarios, the received signal may suffer from

significant time-varying Doppler distribution due to the relative motion between the transmitter and the receiver [19]. This occurs, for instance, in the mobile-satellite communication systems based on a constellation of nongeostationary low-earth-orbit (LEO) satellites [20], and in millimeter-wave mobile communications for traffic control and assistance [21]. In such situations, both Doppler shift and Doppler rate should be estimated and compensated by the receiver. These two parameters can be estimated from the received signals which are modeled as the 2nd order PPSs.

- In digital video processing, motion estimation is an important topic. By using frame projections on coordinate axis and the μ -propagation [22, 23], the video sequences are mapped to frequency modulated (FM) signals. Estimations of the phase coefficients provide the estimates of velocity and acceleration of moving objects.

The ML method, which is implemented by discretising the search space, computing the polynomial time-frequency transform (PTFT), and searching for peaks of PTFT, has comparatively high output SNR and achieves a good statistical performance which asymptotically approaches the Cramer-Rao lower bound (CRLB) [2, 24]. Due to its linear property, the PTFT has the advantage of no crossterms or deterministic noise for multicomponent signals. The main disadvantage of the PTFT is that it requires a huge computational complexity to deal with the multi-dimensional computation. Therefore, the use of fast algorithms to reduce the computational load of the PTFT is critical for many real-time applications.

In practice, the signals to be analyzed are always embedded in noise. In general, the corrupting noise is assumed to be additive white Gaussian noise

(AWGN). This assumption is made mainly because of the central limit theorem and the easiness in analytic manipulations. However, in some situations, signals are disturbed by noise exhibiting impulsive characteristic. The sources of impulsive noise may be either natural (e.g. lightning, ice-cracking), or man-made. It has been observed that the performance of the standard signal transforms and TFRs deteriorates greatly in these cases. To address this problem, alternative techniques are often used, such as the M -estimation method based on the minimax Huber's theory [25–27] and the L -estimation method using linear combination of order statistics [28–31]. To the best of the author's knowledge, no performance comparison (statistical/computation) has been made between them. In addition, the realization of these approaches requires recursive or sorting procedures. For example, to obtain the robust forms of Fourier transform (FT), for each frequency bin, the M -estimation based method requires several iterations with one FT calculated in every iteration; while the L -estimation based method needs two sorting processes. Typical algorithm to sort N elements needs a computational complexity in the order from $O(N \log N)$ to $O(N^2)$. It is obvious that either method adds much computation compared to the standard FT. Similar observations can be made for the robust forms of TFRs. To minimize the computation overload, computationally efficient algorithms are desirable.

It is common that the solutions to different problems may share the same or similar basic ideas. For instance, the principle of HAF-based method for PPS's phase parameter estimation is to non-linearly transform the signal to obtain a sinusoid at a certain frequency that is directly related to the higher-order phase coefficient of that signal. The basic idea of data-aided carrier frequency offset (CFO) estimation in orthogonal frequency division multiplexing (OFDM) is to transmit at least two identical parts in the training-symbol block, and convert it

into a phase estimation problem, because the CFO produces phase shifts in these identical parts. This fact motivates us to extend some improving techniques for the HAF to the CFO problem in order to minimize its estimation error and/or to solve the aliasing problem inherent to phase/frequency estimation.

1.2 Objectives

This thesis aims to develop new signal processing algorithms for the study of signals with time-varying frequencies and their applications. The main objectives of this thesis are summarized as follows:

- To compare the performances of various approaches for the analysis of signals with time-varying frequencies in impulsive channels and to propose simple and efficient methods dealing with impulsive noise.
- Because the PTFT can provide ML estimates for the phase parameters of complex PPSs with a large computational cost, fast algorithms for the PTFT are to be developed to reduce its computational complexity.
- The performance of the PTFT for real-valued PPSs is to be studied, and the potential to further reduce its computational complexity for real-valued input is to be investigated.
- To propose a hardware-oriented algorithm and pipelined structure for the high-speed computation of the second order PTFT, which is very important for practical applications that deal with high speed sequentially presented data streams.

- To develop a signal processing algorithm with emphasis on its application to carrier frequency offset (CFO) estimation in orthogonal frequency-division multiplexing (OFDM) system.

1.3 Major Contributions

The major contributions of the thesis are as follows:

- The performance of M -estimation and L -estimation based forms of signal transforms and TFRs is evaluated in terms of statistical measures and computation time. In addition, a simple and efficient clipping method is proposed. Simulation results show that this method can provide similar performance as the M -estimation and L -estimation based methods, and reduce the computation overhead significantly. It also has the advantage of keeping the standard transform's efficient computation structure (if available).
- A general class of fast algorithms for computing the arbitrary order PTFT of complex-valued input sequence with length $N = a^p b$ is proposed, where a, p and b are integers. By assigning the values of a, p and b , various algorithms, for examples, radix- a and split-radix $2/(2a)$, can be easily obtained to provide the flexibility of supporting various input sequence length. Detailed analysis on the computational complexity required by these algorithms is also presented.
- The performance of PTFT for real-valued PPSs is studied, and it is shown that PTFT has a conjugate symmetric property, which is similar to that of the discrete Fourier transform, if the input sequence is real-valued.

Combining with this property, the fast algorithms for complex sequence are extended for real-valued input. The computational complexity needed by the resulting algorithms is analyzed, and comparisons show that the required computational complexity of some of the algorithms is about one half of that needed by the fast algorithms for complex sequence.

- Single-path pipelined algorithms and designs for the computation of FFTs and PTFTs are introduced. The proposed architectures are based on simple concepts, have regular structures, and are particularly attractive for applications dealing with sequentially presented input streams. The hardware requirements of proposed architectures are analyzed. Compared with the reported pipelined FFT structures, the proposed structure has reached the minimum requirement for both multipliers and adders.
- A CFO estimation algorithm is proposed based on lag diversity techniques for OFDM systems. Theoretical analysis and simulations show that the proposed methods effectively achieve a wide estimation range with reduced mean square errors (MSE) compared to the recently proposed single- h method. When dealing with multipath channels, simulation results show that the proposed method significantly reduces MSE compared to the pilot assisted estimator.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows.

Chapter 2 outlines the basics on the analysis of signals with time-varying frequencies. It shows, by an example, the limitations of classical methods such as

Fourier transform or time analysis in characterizing the spectral contents of signals with time-varying frequencies. Then, the study based on parametric methods is reviewed, including the background of the PPS model, the CRLB of the estimates of the phase parameters of PPSs, and several PPS-related transforms such as HAF, PHAF, and PTFT. Next, the study based on the non-parametric methods is given. In particular, it will focus on the time-frequency representations (TFRs). In the last part of this chapter, the principles of deriving fast Fourier transform (FFT) algorithms are briefly reviewed.

In Chapter 3, the performances of two classes of methods are compared, and the comparison results show the importance and effectiveness of the PTFT. Next, M -estimation and L -estimation based signal transforms and TFRs dealing with impulsive noise are also reviewed. A simple and efficient preprocessing algorithm is presented, and performance comparisons, in terms of statistical and computation, are conducted.

In Chapter 4, a general class of fast algorithms for computing the arbitrary order PTFT is presented. Then, the required computational complexity of the fast algorithms is analyzed and compared with previously reported fast algorithms to show that significant reduction of computation overload is achieved.

Chapter 5 is dedicated to the phase parameters estimation of real PPSs. The performance of applying PTFT in this case is studied, and the specific properties of the PTFT for real-valued input are explored to further reduce its computational complexity compared with the PTFT of complex input.

Chapter 6 focuses on the pipelined hardware implementation of FFT and PTFT. The reported pipelined structures are reviewed, a new pipeline structure is proposed, and the hardware requirements of different algorithms are

compared. In addition, a pipeline structure for computing the 2nd order PTFT is proposed based on radix-2² algorithm.

Chapter 7 applies a signal processing technique reviewed in the previous chapters to solve the ambiguity problem of CFO estimation in OFDM systems.

This thesis is concluded in Chapter 8. Some recommendations for future work are also given in Chapter 8.

Chapter 2

Background

2.1 Introduction

The main focus of this chapter is to review some fundamental concepts related to the study of signals with time-varying frequencies, and to provide the necessary background for understanding the subsequent chapters. The rest of this chapter is organized as follows. Section 2.2 lists the most frequently used symbols. An example is presented in Section 2.3 to show the limitations of time or frequency representation in characterizing the spectral contents of signals with time-varying frequencies. In Section 2.4, parametric methods are briefly reviewed, in particular, we will focus on the polynomial-phase signal (PPS) model, maximum-likelihood (ML) and suboptimal methods. Then, the concept of instantaneous frequency (IF) is given, and non-parametric methods including linear and nonlinear TFRs are introduced in Section 2.5. Finally, we review the fast Fourier transform (FFT) algorithms in Section 2.6 because some concepts and terminologies of the algorithms will be applied to derive new fast algorithms

for the PTFT in Chapters 4 to 6.

2.2 Notation

Throughout this thesis, the most frequently used symbols are listed as follows.

- Italic upper (lower) boldface letters: matrices (vectors)
- j : the imaginary unit
- $\Re(\cdot)$: the real part of the argument
- $\Im(\cdot)$: the imaginary part of the argument
- H superscript : Hermitian (complex conjugate transpose)
- T superscript : transpose
- * superscript : complex conjugate
- $\hat{\cdot}$ superscript: the estimator of the given parameter
- E : expectation
- var : variance
- $\mathbb{N}$: natural numbers
- $\mathbb{Z}$: integer numbers
- $\mathbb{R}$: real numbers

Integrals whose limits are not indicated will denote the limits from minus to plus infinity. Likewise, summations whose limits are not indicated will denote summations over the complete range of the indices.

The expression $x \sim \mathcal{N}(\mu, \sigma^2)$ indicates that a random variable $x \in \mathbb{R}$ is normally distributed with mean μ and variance σ^2 . The expression $z = \text{mod}(x, y)$ yields the smallest $z \geq 0$ such that $x = ny + z$ for some nonnegative integer n . The expression $c = \text{gcd}(a, b)$ is the greatest common divisor (GCD) of two positive integers a and b . The expression $y = \lfloor x \rfloor$ returns the largest integer y which is smaller than or equal to x , and $y = \lceil x \rceil$ returns the smallest integer y which is larger than or equal to x . The binomial coefficient is defined to be the natural number

$$C_n^k = \begin{cases} \frac{n!}{k!(n-k)!} & \text{if } n \geq k \geq 0 \\ 0 & \text{if } k < 0 \text{ or } k > n \end{cases}$$

where $m!$ is the factorial of m for $m \in \mathbb{N}$.

2.3 Signals with Time-varying Frequencies

As stated earlier, time domain and frequency domain constitute two alternative ways for analyzing a signal, and the Fourier transform (FT) is the bridge connecting these two domains. The spectrum or FT of a signal $s(t)$ is defined as

$$S(\omega) = \int s(t)e^{-j\omega t} dt. \quad (2.3.1)$$

Conversely, the signal can be obtained from its spectrum as

$$s(t) = \frac{1}{2\pi} \int S(\omega)e^{j\omega t} d\omega. \quad (2.3.2)$$

The importance of the FT, its properties, and its physical significance are well detailed in the literature. However, the frequency contents of most signals encountered in our everyday life change over time, such as biomedical signals, speech signals, stock indexes, and vibrations. Because the basis functions, $e^{-j\omega}$, used in the classical Fourier analysis do not associate with any particular time information, the resulting measurements, $S(\omega)$, can show which frequencies exist and their relative strength, but cannot provide easy-to-interpret information about the time localization of a spectral component [32].

To illustrate this point, let us consider a bat sonar signal, recorded with a sampling frequency of 230.4kHz¹. The time domain representation, Figure 2.1 (a) cannot say precisely what is the frequency content at each time instant. Similarly, its spectrum, Figure 2.1 (b) does not give information on what time the signal is located around a given frequency.

Figure 2.1: Time domain representation (a) of the bat sonar signal and its spectrum (b).

Two major approaches exist in the literature for the analysis of signals with time-varying frequencies.

¹This recording was part of the research program RCP 445 supported by CNRS (Centre National de la Recherche Scientifique, France), and the data is available for download from <http://tftb.nongnu.org>

2.4 Parametric Methods

For analysis tools within this category, the most commonly used model is the polynomial phase signal (PPS) model which is motivated by the Weierstrass approximation theorem. This theorem implies that for a finite duration of observations, an arbitrary time varying phase can be well approximated by a polynomial of properly chosen order. This class of methods aim to estimate the coefficients of a polynomial series for the phase.

2.4.1 Polynomial-Phase Signal and Its CRLB

The general form of analog M th order polynomial-phase signal (PPS) is expressed as $s(t) = A(t)e^{j\phi(t)}$, where the phase $\phi(t) = 2\pi \sum_{m=0}^M a_m t^m$ is a polynomial, and the amplitude $A(t)$ is a constant or a real-valued stationary process. The model of PPS has been used in many applications of signal processing, for example, radar [11, 17, 18], sonar [33] and communication systems [34, 35], image and audio watermarking [36, 37] and motion estimation in a video sequence [22, 23]. The Cramér-Rao lower bound (CRLB) expresses a lower bound on the variance of unbiased estimators of a deterministic parameter. For a constant amplitude complex PPS corrupted by additive white Gaussian noise (AWGN), the sampled signal is given by

$$x(n) = Ae^{j2\pi \sum_{m=0}^M a_m \Delta^m n^m} + w(n), \quad 0 \leq n \leq N-1 \quad (2.4.1)$$

where Δ is the sampling interval, $w(n)$ is i.i.d complex Gaussian random variables with zero mean and variance σ^2 . It is reported in [38] that, for the unbiased

m th order phase coefficient estimate, $\hat{a}_m$, the CRLB for its variance is given by

$$\text{var}\{\hat{a}_m\} \geq \frac{1}{8\pi^2 N^{2m+1} \Delta^{2m} \cdot \text{SNR}} \left[\frac{1}{2m+1} + \frac{(M+1)^2}{2N(m+1)^2} - \frac{1}{2N} + O(N^{-2}) \right] \cdot [(M+m+1)C_{M+m}^m C_M^m]^2 \quad (2.4.2)$$

where C_n^k is the binomial coefficient, and $\text{SNR} = A^2/\sigma^2$ is the input signal-to-noise ratio. Some special cases are given by

$$\begin{aligned} \text{var}\{\hat{a}_0\} &\geq \left[1 + \frac{M(M+2)}{2N} + O(N^{-2}) \right] \frac{(M+1)^2}{8\pi^2 N \cdot \text{SNR}} \\ \text{var}\{\hat{a}_1\} &\geq \left[\frac{1}{3} + \frac{(M-1)(M+3)}{8N} + O(N^{-2}) \right] \frac{[M(M+1)(M+2)]^2}{8\pi^2 N^3 \Delta^2 \cdot \text{SNR}} \\ \text{var}\{\hat{a}_M\} &\geq [1 + O(N^{-2})] \frac{(2M+1)(C_{2M}^M)^2}{8\pi^2 N^{2M+1} \Delta^{2M} \cdot \text{SNR}}. \end{aligned} \quad (2.4.3)$$

In the rest of the thesis, unless otherwise indicated, unit sampling frequency is assumed, that is to say, we discuss the discrete multicomponent PPSs, which can be written as

$$\tilde{x}(n) = \sum_{i=1}^I A_i(n) e^{j2\pi \sum_{m=0}^M \tilde{a}_{i,m} n^m} + w(n), \quad 0 \leq n \leq N-1 \quad (2.4.4)$$

where the parameter $\tilde{a}_{i,m}$ is related to the actual phase coefficient by $\tilde{a}_{i,m} = a_{i,m} \Delta^m$, and the subscript i denotes the i th component. If the subscript is omitted, we are analyzing mono component case. For simplicity of notation, the symbol $\sim$ is omitted in the following.

2.4.2 Maximum Likelihood (ML) Method

For a constant amplitude PPS embedded in AWGN, it has been shown that the maximum likelihood estimation (MLE) of the phase coefficients and the

amplitude are given by [24]

$$\begin{aligned}\{\hat{a}_M, \hat{a}_{M-1}, \dots, \hat{a}_1\} &= \arg \max_{\boldsymbol{\alpha}} \left| \sum_{n=0}^{N-1} x(n) e^{-j2\pi \sum_{m=1}^M \alpha_m n^m} \right|^2 \\ \hat{a}_0 &= \text{phase} \left(\sum_{n=0}^{N-1} x(n) e^{-j2\pi \sum_{m=1}^M \hat{a}_m n^m} \right) \\ \hat{A} &= \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) e^{-j2\pi \sum_{m=1}^M \hat{a}_m n^m} \right| \end{aligned} \quad (2.4.5)$$

where $\boldsymbol{\alpha} = \{\alpha_M, \alpha_{M-1}, \dots, \alpha_1\}$. If the amplitude $A(n)$ is assumed to be a real-valued stationary process with any structure and independent of the AWGN, the MLE method estimates the phase coefficients as follows [2]:

$$\begin{aligned}\{\hat{a}_M, \hat{a}_{M-1}, \dots, \hat{a}_1\} &= \arg \max_{\boldsymbol{\alpha}} \left| \sum_{n=0}^{N-1} x^2(n) e^{-j2\pi \sum_{m=1}^M 2\alpha_m n^m} \right|^2 \\ \hat{a}_0 &= \frac{1}{2} \text{phase} \left(\sum_{n=0}^{N-1} x^2(n) e^{-j2\pi \sum_{m=1}^M 2\hat{a}_m n^m} \right). \end{aligned} \quad (2.4.6)$$

The MLE asymptotically approaches the CRLB [2, 9, 24].

To obtain the MLE, $\boldsymbol{\alpha} = \{\alpha_M, \alpha_{M-1}, \dots, \alpha_1\}$ is discretised. Therefore, the M th order polynomial time-frequency transform (PTFT) is defined as

$$\text{PTFT}(k_0, k_1, \dots, k_{M-1}) = \sum_{n=0}^{N_0-1} y(n) W_{N_0}^{k_0 n} W_{N_1}^{k_1 n^2} \dots W_{N_{M-1}}^{k_{M-1} n^M} \quad (2.4.7)$$

where $W_N = e^{-j2\pi/N}$, $y(n) = x(n)$ for constant amplitude and $y(n) = x^2(n)$ for time-varying amplitude, $0 \leq k_r \leq N_r - 1$ for $r = 0, 1, \dots, M-1$, and k_r/N_r is the discrete form of α_{r+1} . The quantization error caused by discretion is controlled by the dimension size N_r . Define a demodulated sequence $y^{(d)}(n) = y(n) W_{N_1}^{k_1 n^2} \dots W_{N_{M-1}}^{k_{M-1} n^M}$, (2.4.7) can be expressed as

$$\text{PTFT}(k_0, k_1, \dots, k_{M-1}) = \sum_{n=0}^{N_0-1} y^{(d)}(n) W_{N_0}^{k_0 n} \quad (2.4.8)$$

i.e., PTFT of $y(n)$ is equivalent to the DFT of $y^{(d)}(n)$. If the demodulated sequence consists of sinusoidal components, due to the property of DFT, sharp peaks appear in the PTFT. This occurs when integer k_r for $r = 1, \dots, M-1$ can take values satisfying $k_r/N_r \approx a_{r+1}$. Since the parameters are estimated according to the defined grids, the performance of PTFT is influenced by the quantization errors. Let δ_r denote the deviation along the r th dimension, i.e., $\delta_r = a_{r+1} - k_r/N_r$. Since N_r points are used for quantization in the r th dimension, the maximum quantization error along this dimension is given by

$$|\delta_r| \leq \frac{1}{2N_r}. \quad (2.4.9)$$

The frequency deviation caused by the quantization errors should be much smaller than the frequency of the sinusoidal component, which can be expressed as

$$\left| \sum_{r=1}^{M-1} \delta_r n^r \right| \ll |a_1| < 0.5, \quad 0 \leq n \leq N_0 - 1, \quad (2.4.10)$$

otherwise the peak may not be easily identified due to the quantization errors.

Combining the above two equations, we have

$$\sum_{r=1}^{M-1} \frac{N_0^r}{N_r} \ll 1. \quad (2.4.11)$$

A sufficient condition for this is $N_r \gg N_0^r$ for $r = 1, \dots, M-1$.

Should the above constraint is satisfied the PTFT of a multicomponent PPS exhibits the same number of peaks as that of components in the PPS. Figure 2.2 shows the PTFT of the sum of two 2nd order PPSs having the same amplitude. The parameters of the first component are $a_{1,1} = 0.1$, $a_{1,2} = 0.0015$. In Figure 2.2(a), the parameters for the second component are $a_{2,1} = 0.2$, $a_{2,2} = 0.001$, and in Figure 2.2(b), they are $a_{2,1} = 0.05$, $a_{2,2} = 0.0015$. In spite of the presence of the -3 dB AWGN, two peaks of the PTFT are still evident.

Figure 2.2: PTFT of the sum of two 2nd order PPSs having the same amplitude.

The output SNR (SNR_{out}) is an important measurement of the estimator's sensitivity to noise. For an M th-order PPS, without noise, the maximum absolute value of $\text{PTFT}(k_0, k_1, \dots, k_{M-1})$ lies on the points of coordinates $\bar{\alpha} = (a_1 N_0, a_2 N_1, \dots, a_M N_{M-1})$. In the presence of noise, the PTFT, evaluated in $\bar{\alpha}$, is a random variable. The SNR_{out} is defined as

$$\text{SNR}_{out} = \frac{|\text{PTFT}_s(\bar{\alpha})|^2}{\text{var}[\text{PTFT}_{s+n}(\bar{\alpha})]}, \quad (2.4.12)$$

which is analog to the definition in [6, 10]. The relationship between the SNR at the input and at the output of the PTFT has been derived in [39], which is

$$\text{SNR}_{out} = N \text{SNR}_{in}. \quad (2.4.13)$$

It is clear that the ratio between the output SNR and the input SNR of PTFT is a function of the number of samples and independent of the order M .

2.4.3 Suboptimal Maximum Likelihood Methods

The polynomial-phase transform (PPT), which was later called the high-order ambiguity function (HAF) was specifically introduced to estimate the param-

ters of PPS's [5–8]. Among the suboptimal ML methods, HAF-based estimation is the most popular candidate due to its computational efficiency.

The generalized M th order multi-lag high-order instantaneous moment (mlHIM) [10] is defined recursively as

$$\begin{aligned}
 \text{mlHIM}_1[x(n)] &= x(n), \\
 \text{mlHIM}_2[x(n); \tau_1] &= x(n + \tau_1)x^*(n - \tau_1), \\
 &\dots \\
 \text{mlHIM}_k[x(n); \tau_{k-1}] &= \text{mlHIM}_{k-1}[x(n + \tau_{k-1}); \tau_{k-2}] \\
 &\quad \times \text{mlHIM}_{k-1}^*[x(n - \tau_{k-1}); \tau_{k-2}], \quad 3 \leq k \leq M
 \end{aligned} \tag{2.4.14}$$

where $\tau_{k-1} = [\tau_1, \tau_2, \dots, \tau_{k-1}]$ is the vector containing all the lags for k th order mlHIM. Applying the Fourier transform of the mlHIM, we obtain the multi-lag HAF (mlHAF) of the signal $x(n)$,

$$\text{mlHAF}_M[x, f, \tau_{M-1}] = \sum_{n=0}^{N-1} \text{mlHIM}_M[x(n); \tau_{M-1}] e^{-j2\pi f n}. \tag{2.4.15}$$

In particular, the HIM and HAF given in [5–8] correspond to the case of $\tau_1 = \tau_2 = \dots = \tau_{M-1} = \tau$ and nonsymmetric definitions (for example, $\text{HIM}_2[x(n); \tau] = x(n)x^*(n - \tau)$). It has been proved in [40] that the symmetric definition yields the same variance in the estimation of the highest order coefficient but smaller variances for the estimates of all other coefficients compared with the nonsymmetric definition. Therefore, the symmetric definition is preferred.

One remarkable property, which makes mlHIM an attractive tool in the analysis of PPS, is that the M th order mlHIM reduces a sampled PPS of degree M into a single tone with frequency $f = 2^{M-1}M! \prod_{k=1}^{M-1} \tau_k a_M$. Thus, finding

the peak of the magnitude of (2.4.15) enables us to estimate the coefficient a_M , i.e.,

$$\hat{a}_M = \frac{1}{2^{M-1} M! \prod_{k=1}^{M-1} \tau_k} \arg \max_f |\text{mlHAF}_M[x, f, \tau_{M-1}]|. \quad (2.4.16)$$

Removing the M th order phase contribution from the observed signal by multiplying $x(n)$ with $e^{-j2\pi\hat{a}_M n^M}$, we obtain a PPS of order $M - 1$, which can be expressed as $x^1(n) = x(n)e^{-j2\pi\hat{a}_M n^M}$, where $\hat{a}_{M-1}$ can be obtained from the peak location of the magnitude of $\text{mlHAF}_{M-1}[x^1, f, \tau_{M-2}]$. The above procedures are repeated until all the coefficients are estimated.

The parameter a_M can be estimated unambiguously only if

$$|a_M| < \frac{1}{2^M M! \prod_{k=1}^{M-1} \tau_k}. \quad (2.4.17)$$

Clearly large lags limit the dynamic range of the estimated parameters [41, 42]. However, the bigger the lags, the better is the resolution of a_M [11]. Hence, the choice of lags is facing conflicting requirements. Two types of solutions have been proposed. One is the iterative approach [41] that uses a small lag to obtain a rough estimate followed by fine estimations with the increased values of lags. The other is to compute two HAFs with two large co-prime lags and to solve the ambiguity problem via liner Diophantine equation [42]. Note that the second approach motivated us to use lag diversity to avoid aliasing in carrier frequency offset (CFO) estimation, which will be presented in Chapter 7.

Due to its nonlinear nature, mlHAF suffers from noise-masking effects at low SNR and crossterms when dealing with multicomponent signals [8]. Moreover, it has been shown in [11, 40] that HAF exhibits spurious peaks if multicomponent signals have the same highest order phase coefficients, from order P to M , with $P > 1$. To avoid identifiability problem and improve noise rejection capability,

a transformation called product high-order ambiguity function (PHAF) was proposed. Given L different lag sets, the PHAF is obtained by multiplying the mlHAFs computed for some lag sets, i.e.,

$$\text{PHAF}_M^{(L)}[x, f, \boldsymbol{\tau}^{(L)}] = \prod_{l=1}^L \text{mlHAF}_M \left[x, \frac{\prod \tau_{M-1}^{(l)}}{\prod \tau_{M-1}^{(1)}} f, \boldsymbol{\tau}^{(l)} \right] \quad (2.4.18)$$

where $\boldsymbol{\tau}_{M-1}^{(l)} = [\tau_1^{(l)}, \tau_2^{(l)}, \dots, \tau_{M-1}^{(l)}]$, and $\prod \tau_{M-1}^{(1)}$ returns the product of all the elements of the vector. The PHAF is based on the observation that the useful sinusoids remain at the same positions, thus the frequency scaling operation ensures that the desirable peak locations are aligned and hence amplified in the product. In contrast, the crossterms with different lags fall in different positions related with the lag parameters, thus spurious sinusoids and higher order crossterms are weakened in the product.

Several algorithms have been proposed to improve the statistical performance including SNR_{out} and asymptotic variance. For instance, the observation that the number of crossterms increases drastically with the order of HAF (PHAF) motivated the authors to propose a “bottom-up” PPS parameter estimation algorithm, which estimates the parameters from the lowest order [43]. Another algorithm is the integrated generalized ambiguity function (IGAF) [10], which is defined by coherent integrations along each lag in the mlHIM defined in (2.4.14). Two phase parameters can be estimated at each iteration. Because the integrations effectively suppress the crossterms, the IGAF provides higher output SNR, which leads to a lower SNR threshold and a better accuracy to approach to the CRLB.

It should be mentioned that the improvements on the output SNR and estimation variances of these reported methods are limited, especially for PPSs with high order and/or low input SNR. This is because these methods do not change

the nonlinearity and sequential estimation structure, which are the fundamental reasons for the performance deterioration.

2.4.4 Advantages of the ML Methods

A significant advantage of the ML method based on the PTFT is that it is a linear transform with respect to the signal $x(n)$. This linearity makes the PTFT work well even at low SNRs and/or higher orders. In general, the PTFT holds advantages in the following three aspects.

First, it is proved in [2] that the PTFT provides ML estimation of the PPS's phase parameters and asymptotically achieves the CRLB. While statistical analysis in [7] demonstrates that the asymptotic variances of estimates based on HAF exhibit a consistent departure from the CRLB and the deviation increases with the phase polynomial order. The IGAF is a sequential approach that estimates two phase parameters at each iterations. It is asymptotically efficient at the second order but loses the efficiency when the degree of the polynomial increases [10]. As shown in [40], although the PHAF improves the performance of the HAF in the presence of noise, the variances of the estimates obtained from PHAF cannot reach the CRLB, that is due to the noise masking effects caused by the nonlinearity structure of the estimator. Therefore, the estimation based on the HAF or PHAF is comparable to that based on PTFT only when SNR is high and the phase polynomial order is small. The following experiment was conducted to compare their performances. A 2nd order PPS corrupted by AWGN is considered, with the parameter $a_1 = 0.1$, $a_2 = 0.0015$ and $N = 128$ samples. The numbers of the grids for computing HAF and PTFT are selected large enough to keep the maximum quantization errors smaller than the root

of the CRLB. At each SNR, the variances were obtained from 200 independent trials. Figure 2.3 shows the ratios between the variance of estimates $\hat{a}_1$ and $\hat{a}_2$ and their respective CRLB's [38]. It is observed that the results acquired from PTFT are always closer to the CRLB than the HAF-based methods. The results achieved from the HAF-based methods become close to the CRLB only when the SNR is greater than a threshold. There is a rapid performance degradation when SNR is smaller than the threshold, which means the HAF-based methods cannot work under this condition. The threshold SNR is about 0 dB for HAF and -2 dB for PHAF (with $L = 3$ sets).

Figure 2.3: Efficiency of the estimates, (a) $\hat{a}_1$ and (b) $\hat{a}_2$, versus SNR.

Secondly, compared with HAF-based methods, the PTFT has a higher output SNR (SNR_{out}), which is an important measurement of the operating range [6] and sensitivity [10, 44] of the algorithm. In Figure 2.4, we plot the output SNR normalized with respect to the number of samples N as a function of input SNR, i.e., SNR_{out}/N vs. SNR_{in} . The dashed, solid and dotted lines refer to the PTFT, IGAF and HAF respectively, and the curves are plotted according to equations (2.4.13) and (25)-(28) in [10]. Both PTFT and HAF have only one line because the normalized output SNRs do not depend on the number of

samples N . From this figure, it is observed that:

- Both IGAF and HAF exhibit threshold effect, that is to say, below a certain SNR value (the threshold), the output SNRs decrease rapidly. In addition, the threshold effect becomes more evident with the increase of polynomial orders.
- The IGAF has a decreasing threshold as the number of samples increase; while the HAF has a threshold independent of the number of samples.
- For a given input SNR, the PTFT has the highest output SNR without threshold effect.

Figure 2.4: The output SNR, normalized by N , versus input SNR for (a) 2nd order and (b) 3rd order PPS. Dashed line: PTFT, dotted line: HAF, and solid line: IGAF with (1) $N = 1024$, (2) $N = 256$, (3) $N = 64$, (4) $N = 16$.

For PHAF, there is generally no closed form expression of the SNR_{out} . For monocomponent PPS, it was shown in [11, 40] that the main advantage of PHAF over HAF is the reduction of the SNR threshold. Thus, we can say the SNR_{out} of the PHAF cannot exceed the PTFT.

Finally, the PTFT estimates all the phase coefficients simultaneously, so it does not suffer from the error propagations in the sequential structures of the HAF, PHAF and IGAF algorithms. In addition, for multicomponent PPSs, the PTFT has no crossterms which generally exist in HAF, PHAF and IGAF algorithms due to their nonlinear characteristics.

2.5 Non-parametric Methods

Analysis tools within this category employ time-frequency representations (TFRs) to describe the time-varying phases for which the pure frequency or Fourier representation is not sufficient.

There exist many TFRs. The dependence of TFR upon the signal may be linear or otherwise nonlinear. The choice of the "best" TFR depends on the nature of the signal to be analyzed and also on issues such as the mathematical properties required, limitations in computation and storage, etc [12, 32].

In the following section, we will review some fundamental concepts related to the popular TFRs.

2.5.1 Analytic Signal and Instantaneous Frequency (IF)

Gabor introduced the concept of uniquely constructing an analytic signal [12], which is complex, from a real one. If the real signal, $s(t)$, has the spectrum $S(f)$, the complex signal, $z(t)$, resulted from Gabor's method has a spectrum containing only the positive frequencies in $S(f)$. Mathematically, $z(t)$ is defined

as

$$z(t) = \frac{2}{\sqrt{2\pi}} \int_0^\infty S(f) e^{j2\pi ft} df. \quad (2.5.1)$$

The factor of 2 is needed so that the real part of the complex signal will be $s(t)$.

Applying the definition of $S(f)$ into the above expression, and using

$$\int_0^\infty e^{j2\pi fx} df = \frac{1}{2} \left(\delta(x) + \frac{j}{\pi x} \right), \quad (2.5.2)$$

we obtain

$$\begin{aligned} z(t) &= s(t) + j H[s(t)] \\ &= a_z(t) e^{j\phi_z(t)}, \end{aligned} \quad (2.5.3)$$

where $H[\cdot]$ means the Hilbert transform (HT).

The importance in using analytic signal is that it allows us to obtain the phase and amplitude of a real signal unambiguously. Therefore, an expression of instantaneous frequency (IF) [12] can be used as an indication of the frequency components that exist at a particular time,

$$f_i(t) = \frac{1}{2\pi} \frac{d\phi_z(t)}{dt}. \quad (2.5.4)$$

The definition of IF loses its meaning when considering a multicomponent signal. In this case, the signal can be viewed as a weighted sum of monocomponent signals, each one with its own IF [33].

Much research has been carried out to estimate the IF of signals and various approaches have been proposed. One of them is the differentiation of the phase [45]. It meets the CRLB for estimating frequency of a stationary signal at high SNR but fails when SNR is low or the signal variation is large. The other class of estimators are filter-based techniques [24] including the phase-locked

loop (LPP), least mean squares (LMS) and recursive least square (RLS) algorithm, etc. This class of methods have difficulties in tracking rapid variations in IF. The effectiveness of TFDs to provide accurate estimates of the IF has been extensively investigated in the literature over the past decades [12, 24, 32, 46–48].

2.5.2 Short-Time Fourier Transform (STFT)

One of the simplest TFRs is the short-time Fourier transform (STFT). Given a signal $x(t)$, its STFT is defined as

$$\text{STFT}(t, \omega) = \int h(\tau)x(t + \tau)e^{-j\omega\tau}d\tau, \quad (2.5.5)$$

where $h(t)$ is window function of finite sequence length. The technique of multiplying signal by a window function is known as windowing, and it results in segmented signals. The basic idea of the STFT is to divide the signal into small time segments and perform Fourier analysis on each time segment to ascertain the frequencies in that segment. The totality of such spectra indicates how the spectrum of the signal is varying in time.

Note that the choice of the window $h(t)$ significantly influences the resolution of the STFT. There is an uncertainty relationship between the time and frequency resolutions of an STFT. In general, a wide window gives better frequency resolution but poor time resolution, while a narrow window gives good time resolution but poor frequency resolution.

Despite this limitation, STFT is still one of the most popular tools for TF signal analysis. For instance, it has been used in many applications including inverse synthetic aperture radar (ISAR) imaging [49], interference excision in

direct sequence spread spectrum (DSSS) communications [50], onset detection in music signals [51] and fingerprint enhancement [52]. This is mainly due to two desirable characteristics of STFT. One is its simplicity and ease of implementation, and the other is the linearity property which is important to avoid the undesirable crossterms that appear in many quadratic TFTs.

2.5.3 Quadratic and High-order Time-Frequency Distributions (TFDs)

In the last two decades, intensive efforts have been made to study time-frequency distributions (TFDs) in order to characterize and analyze signals with time-varying frequencies. Many TFDs of current interest are members of the quadratic (or Cohen's) class.

All quadratic TFDs can be obtained from

$$C(t, \omega) = \frac{1}{4\pi^2} \int \int \int x(\mu + \frac{1}{2}\tau) x^*(\mu - \frac{1}{2}\tau) \phi(\theta, \tau) e^{-j\theta t - j\tau\omega + j\theta\mu} d\mu d\tau d\theta \quad (2.5.6)$$

where $\phi(\theta, \tau)$ is a two dimensional function called the kernel, which determines the distribution and the properties of the TFTs. $C(t, \omega)$ can also be expressed as [12]

$$C(t, \omega) = \frac{1}{4\pi^2} \int \int M(\theta, \tau) e^{-j\theta t - j\tau\omega} d\theta d\tau \quad (2.5.7)$$

with

$$M(\theta, \tau) = \phi(\theta, \tau) \int x(\mu + \frac{1}{2}\tau) x^*(\mu - \frac{1}{2}\tau) e^{j\theta\mu} d\mu \quad (2.5.8)$$

$$= \phi(\theta, \tau) A(\theta, \tau) \quad (2.5.9)$$

where $A(\theta, \tau)$ is the symmetrical ambiguity function. This indicates that a quadratic TFD is obtained from a two-dimensional filtering in the ambiguity domain.

There are many members of the quadratic class, and the Wigner-Ville distribution (WVD) is one of the most popular TFDs. For the analysis of monocomponent linear frequency modulated (LFM) signal, the WVD has been shown to be optimal [53] in that it yields delta function around the IF for an LFM signal in the time-frequency domain. Its importance also stems from the fact that many other time-frequency analysis tools can be expressed in terms of WVD [12].

The kernel of WVD is equal to unity, i.e., the WVD of a signal $x(t)$ can be defined as

$$\text{WVD}(t, \omega) = \int x(t + \tau/2)x^*(t - \tau/2)e^{-j\omega\tau} d\tau \quad (2.5.10)$$

For the analysis of multicomponent signals, the WVD presents undesirable high amplitude crossterms. That is because WVD kernel is an allpass filter in the ambiguity domain, and therefore, the crossterms are neither reduced nor removed.

Therefore, one main deficiency of the WVD is the so-called crossterm interference, which has substantially limited the application of WVD. Reducing the crossterm interference without destroying the useful properties of the TFDs has been a very important topic in time-frequency analysis. Various TFDs have been proposed to reduce or suppress the crossterms for multicomponent signals, such as the Choi-Williams distribution [54], the Cone-Shape distribution [55] and the B-distribution [56]. In general, these solutions are to introduce various kernels to suppress crossterms. For example, the kernel of B-distribution is given by [56]

$$\phi(\theta, \tau) = |\tau|^\alpha \frac{2^{2\alpha-1}}{\Gamma(2\alpha)} \Gamma(\alpha + j\pi\theta) \Gamma(\alpha - j\pi\theta) \quad (2.5.11)$$

where $\Gamma(x)$ is the gamma function given by $\Gamma(x) = \int_0^\infty u^{x-1} e^{-u} du$, and α is

a real parameter ($0 < \alpha \leq 1$) which controls the sharpness of cutoff of the 2-D filter $\phi(\theta, \tau)$ in the ambiguity domain. Another method introduced for the suppression of the crossterms is the S-method proposed in [57]. This approach performs well if the components are well separated, but its performance deteriorates when the signal components overlap in the time-frequency domain [58].

To represent high-order PPS as delta functions along the IF of the signal in the time-frequency plane, the polynomial Wigner-Ville distribution (PWVD) was introduced in [59] as

$$\text{PWVD}(t, \omega) = \int \prod_{i=1}^{q/2} x(t + d_i \tau) x^*(t + d_{-i} \tau) e^{-j\omega \tau} d\tau \quad (2.5.12)$$

where q is an even integer indicating the order of the PWVD, and d_i and d_{-i} are real coefficients. The realness of the PWVD results in $d_i = -d_{-i}$. The design of PWVD and the selection criteria for the optimal set of kernel coefficients are discussed in [60].

In generally, it is not possible to have a particular TFD that can be used effectively in all possible applications. The choice of a particular TFD depends on the specific application at hand [12, 32]. For a monocomponent signal, performance of the TFDs is usually measured by the energy concentration that the TFD attains about the signal IF. For a multicomponent signal, performance of the TFR involves not only the energy concentration of each component but the resolution as well. Resolution measure criteria was introduced in [61] to quantify the quality of TFDs instead of relying solely on visual inspection of their plots.

2.5.4 Local Polynomial Time-Frequency Transform

The local polynomial time-frequency transform (LPTFT) [16] was developed as a high-order generalization of the STFT. It is in the form of

$$\text{LPTFT}(t, \boldsymbol{\omega}) = \int h(\tau) x(t + \tau) e^{-j\theta(\tau, \boldsymbol{\omega})} d\tau \quad (2.5.13)$$

where

$$\begin{aligned} \theta(\tau, \boldsymbol{\omega}) &= \sum_{i=1}^m \omega_i \tau^i / i! \\ \boldsymbol{\omega} &= (\omega_1, \omega_2, \dots, \omega_m)^T \in \mathbb{R} \end{aligned} \quad (2.5.14)$$

here m is the order of local polynomial approximation (LPA), and t is the center of LPA. The local polynomial periodogram (LPP) is defined as

$$\text{LPP}(t, \boldsymbol{\omega}) = |\text{LPTFT}(t, \boldsymbol{\omega})|^2. \quad (2.5.15)$$

The STFT and spectrogram correspond to the particular case of LPTFT and LPP with $m = 1$.

The principal difference between the LPTFT and the quadratic and high-order TFDs is that the LPTFT is linear with respect to the signal and uses the polynomial function, $\theta(\tau, \boldsymbol{\omega})$, in the complex exponent (or the transform kernel). On the other hand, quadratic and higher degree polynomials of the signals are used with the exponential function $e^{-j\omega\tau}$ in TFDs, wherein the argument of the exponent is linear with respect to variable τ .

The LPTFT can be interpreted as a time-frequency energy distribution over the $t - \boldsymbol{\omega}^0(t)$ space, where $\boldsymbol{\omega}^0(t)$ is a vector of the true values of the IF $\Omega(t)$ and the derivatives of IF $\Omega(t)$, mathematically, it is given by $\boldsymbol{\omega}^0(t) = [\Omega(t), \Omega^{(1)}(t), \dots, \Omega^{(m-1)}(t)]^T$. The energy concentration of LPTFT($t, \boldsymbol{\omega}$) in

$\omega^0(t)$ for the time instant t is illustrated in [14, 15]. This concentration motivates the use of the LPTFT as an IF estimator. Asymptotic covariance matrix and bias of the estimates are studied in [14, 15]. It is also shown that the bandwidth of the window controls the tradeoff of bias and variance. A data-driven approach based on the intersection of confidence intervals (ICI) was proposed in [62] to solve the problem of bandwidth selection. Note that when nonsymmetric rectangular window is used and the signal is corrupted by the AWGN, the LPP estimator coincides with the ML estimator mentioned in Section 2.4.2.

The LPP estimator is computationally demanding because it is based on pointwise estimation, which involves calculating a multi-dimensional (MD) function and finding the maximum of the MD function. To decrease calculation burden, the 2nd order LPTFT with pointwise chirp-rate parameter estimation was proposed in [30, 63], which can be expressed as

$$\begin{aligned} \text{LPTFT}_{\alpha(t)}(t, \omega) &= \int h(\tau) x(t + \tau) e^{-j\alpha(t)\tau^2/2} e^{-j\omega\tau} d\tau \\ \hat{\alpha}(t) &= \arg \max_{\alpha \in \Lambda} H(\alpha, t) \end{aligned} \quad (2.5.16)$$

where Λ is a set of values of considered chirp-rate parameters, and $H(\alpha, t)$ is concentration measure discussed in [64, 65]. Another efficient method, named modified LPTFT (MLPTFT), was introduced in [66]. It is computed through the following steps:

- Divide the signal into a number of segments, model each segment as an M th order PPS and estimate the phase parameters using PTFT
- Decide the window length according to the difference of the polynomial phase function on consecutive segments
- Compute LPTFT with the obtained estimates and selected window length

The length of overlap between two consecutive segments controls the computation load as well as the smoothness of the spectrum.

2.6 Fast Fourier Transform Algorithms

As stated in earlier sections, the PTFT can provide MLE for phase coefficients of PPS signals and can also be combined with LPTFT to analyze signals with time-varying frequencies. Because the PTFT generally requires a huge computation complexity, fast algorithms for the PTFT are very important for practical applications. This will be addressed in the following two chapters. Some basic concepts and terminologies used to derive fast PTFT algorithms are similar to those utilized to derive fast Fourier transforms (FFT) algorithms. Thus, in this section, we will review the FFT algorithms briefly.

The DFT of a discrete-time signal $x(n)$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk}, \quad 0 \leq k \leq N-1 \quad (2.6.1)$$

where $W_N = e^{-j2\pi/N}$. Direct calculation of (2.6.1) requires about N^2 complex multiplications and $N(N-1)$ complex additions.

To compute DFT efficiently, many FFT algorithms are reported in the literature. They can be generally classified into two categories: the radix-type and the non-radix type. The radix-type category particularly supports the input sequence length $N = q^m$, where q and m are positive integers. Based on a particular radix number q or a blend of radices, the radix-type algorithms recursively divide the entire computation task into DFTs of small lengths. This category includes the well-known radix-2, radix-4 and split-radix algorithms [67, 68] for a DFT of length 2^m , and algorithms in [69, 70] for DFT of length 3^m and 5^m . The

non-radix type algorithms include the Rader's fast algorithm [71], the prime factor algorithm [72], the Good-Thomas fast algorithm [73], which usually require an index mapping to rearrange the sequence of the input/output data. The fast algorithms used for a particular transform size cannot be easily extended for other transform sizes and/or different number of dimensions. Here, we focus on the radix-type algorithms as they have a regular computational structure due to the recursive decomposition processes. The regularity is desirable for implementation, testing, maintenance and components reuse.

The main concepts and steps involved in the derivation of FFT algorithms, as shown in examples of the well-known radix-2 and split-radix algorithms, will be illustrated below. The sequence length N is assumed to be a power of two. In general, the algorithms use the strategy of "divide and conquer" to decompose the DFT computation into DFTs of smaller sizes recursively, and explore

$$\begin{aligned} \text{Symmetry property: } W_N^{k+N/2} &= -W_N^k \\ \text{Periodicity property: } W_N^{k+N} &= W_N^k \end{aligned} \quad (2.6.2)$$

of the twiddle factor W_N to reduce the computational complexity greatly. The input sequence $x(n)$ can be split into two subsequences $x_1(n) = x(2n)$ and $x_2(n) = x(2n+1)$, and the fact $W_N^{mnk} = W_{N/m}^{nk}$ is used. Thus the DFT is calculated by

$$\begin{aligned} X(k) &= \sum_{n \text{ even}} x(n)W_N^{nk} + \sum_{n \text{ odd}} x(n)W_N^{nk} \\ &= \sum_{n=0}^{N/2-1} x_1(n)W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x_2(n)W_{N/2}^{nk} \\ &= X_1(k) + W_N^k X_2(k) \end{aligned} \quad (2.6.3)$$

where $k = 0, 1, \dots, N/2 - 1$. Due to the periodicity property shown in (2.6.2), we have $X_i(k + N/2) = X_i(k)$ for $i = 1, 2$. Applying the symmetry property,

we obtain

$$X(k + N/2) = X_1(k) - W_N^k X_2(k). \quad (2.6.4)$$

It can be seen from (2.6.3) and (2.6.4) that the length- N DFT can be obtained by computing two length- $N/2$ DFTs with input being $x_1(n)$ and $x_2(n)$ then combining those results to produce the DFT of the whole sequence. Similarly, the length- $N/2$ DFTs can be further divided into length- $N/4$ DFTs in the same way. For $N = 2^q$, this decimation process can repeat q times until length-2 DFTs remain. Since the input sequence of DFT is traditionally considered to be in the time domain, the algorithm developed in such a way is known as the *decimation-in-time* (DIT) algorithm. The multiplication factor W_N^k in the above two equations is known as a *twiddle factor*, which may become trivial, i.e., $W_N^k = \pm 1, \pm j$. Obviously multiplying a complex number with a trivial twiddle factor can be easily implemented.

The decimation process can also be performed by splitting the transformed outputs, which is traditionally considered to be in the frequency domain, into even and odd subsequences. This type of approach is known as the *decimation-in-frequency* (DIF) algorithm. To derive the algorithm, we first split the DFT into two summations, the first one involves the sum over the first $N/2$ data points and the second one involves the last $N/2$ data points. Thus we obtain

$$\begin{aligned} X(k) &= \sum_{n=0}^{N/2-1} x(n)W_N^{nk} + \sum_{n=N/2}^{N-1} x(n)W_N^{nk} \\ &= \sum_{n=0}^{N/2-1} [x(n) + (-1)^k x(n + N/2)]W_N^{nk} \end{aligned} \quad (2.6.5)$$

Then $X(k)$ is split into even and odd indexed samples, which can be expressed

as

$$X(2k) = \sum_{n=0}^{N/2-1} [x(n) + x(n + N/2)] W_{N/2}^{nk}, \quad (2.6.6)$$

$$X(2k + 1) = \sum_{n=0}^{N/2-1} \{[x(n) - x(n + N/2)] W_N^n\} W_{N/2}^{nk}, \quad (2.6.7)$$

where $\alpha = 0, 1$ and $0 \leq k \leq N/2 - 1$. This is to say that the length- N DFT is divided into two length- $N/2$ DFTs with the input sequence being $[x(n) + x(n + N/2)]$ and $[x(n) - x(n + N/2)] W_N^n$, respectively. Similar to the DIT decomposition process, this process can be performed recursively until length-2 DFTs remain.

As stated above, the radix-2 FFT algorithms decompose the length- N DFT in terms of two length- $N/2$ DFTs recursively. To improve the computation efficiency, the split-radix FFT is proposed, which uses a blend of radices 2 and 4, i.e., it recursively decomposes a length- N DFT into one length- $N/2$ DFT and two length- $N/4$ DFTs. The DIF version of split-radix algorithms is illustrated here. The underlying idea is that the even indexed transform outputs $X(2k)$ are given by (2.6.6), while the odd indexed outputs are further decomposed into

$$X(4k + 1) = \sum_{n=0}^{N/4-1} \{[t_1(n) - jt_2(n)] W_N^n\} W_{N/4}^{kn} \quad (2.6.8)$$

$$X(4k + 3) = \sum_{n=0}^{N/4-1} \{[t_1(n) + jt_2(n)] W_N^{3n}\} W_{N/4}^{kn}, \quad (2.6.9)$$

where $0 \leq k \leq N/4 - 1$, $t_1(n) = x(n) - x(n + N/2)$ and $t_2(n) = x(n + N/4) - x(n + 3N/4)$. In a similar manner, the DIT version of the split-radix algorithms can be derived.

The radix-type algorithm has the advantage that the same recursive decomposition process can be used for different number of dimensions. Instead of

providing a general description, let us consider the two-dimensional (2D) DFT, which is defined as

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2} \quad (2.6.10)$$

where $x(n_1, n_2)$ is the input sequence, and $k_i = 0, 1, \dots, N_i - 1$ for $i = 1, 2$. $X(k_1, k_2)$ can be computed by first transform along the n_1 dimension, then along the n_2 dimension, this is known as the *row-column* algorithm.

The DIT radix- (2×2) FFT algorithm [74] splits the samples $x(n_1, n_2)$ into four parts $x(2n_1 + \alpha, 2n_2 + \beta)$, $\alpha, \beta = 0, 1$, then computes the 2D DFT in (2.6.10) by combining four terms. Each one of them, $S_{\alpha\beta}(k_1, k_2)$, is an $(N_1/2 \times N_2/2)$ -point DFT over the samples $x(2n_1 + \alpha, 2n_2 + \beta)$, and can be expressed as

$$S_{\alpha\beta}(k_1, k_2) = \sum_{n_1=0}^{N_1/2-1} \sum_{n_2=0}^{N_2/2-1} x(2n_1 + \alpha, 2n_2 + \beta) W_{N_1/2}^{n_1 k_1} W_{N_2/2}^{n_2 k_2}. \quad (2.6.11)$$

By exploring the symmetry property, $X(k_1, k_2)$, $X(k_1, k_2 + N_2/2)$, $X(k_1 + N_1/2, k_2)$ and $X(k_1 + N_1/2, k_2 + N_2/2)$ can be simultaneously obtained from the combinations of $S_{\alpha\beta}(k_1, k_2)$ as follows:

$$\begin{aligned} X(k_1, k_2) &= \sum_{\alpha} \sum_{\beta} W_{N_1}^{\alpha k_1} W_{N_2}^{\beta k_2} S_{\alpha\beta}(k_1, k_2) \\ X(k_1, k_2 + N_2/2) &= \sum_{\alpha} \sum_{\beta} (-1)^{\beta} W_{N_1}^{\alpha k_1} W_{N_2}^{\beta k_2} S_{\alpha\beta}(k_1, k_2) \\ X(k_1 + N_1/2, k_2) &= \sum_{\alpha} \sum_{\beta} (-1)^{\alpha} W_{N_1}^{\alpha k_1} W_{N_2}^{\beta k_2} S_{\alpha\beta}(k_1, k_2) \\ X(k_1 + N_1/2, k_2 + N_2/2) &= \sum_{\alpha} \sum_{\beta} (-1)^{\alpha+\beta} W_{N_1}^{\alpha k_1} W_{N_2}^{\beta k_2} S_{\alpha\beta}(k_1, k_2). \end{aligned} \quad (2.6.12)$$

Similarly, the DIF radix- (2×2) FFT [75] breaks the 2D DFT in (2.6.10) into four sums according to the parities of output indices, and uses the symmetry

and periodicity properties. This leads to

$$\begin{aligned}
 X(2k_1 + \alpha, 2k_2 + \beta) &= \sum_{n_1=0}^{N_1/2-1} \sum_{n_2=0}^{N_2/2-1} [A_{\alpha\beta}(n_1, n_2) W_{N_1}^{n_1\alpha} W_{N_2}^{n_2\beta}] W_{N_1/2}^{n_1k_1} W_{N_2/2}^{n_2k_2} \\
 &\quad k_i = 0, 1, \dots, N_i/2 - 1 \text{ for } i = 1, 2 \\
 &\quad \alpha, \beta = 0, 1
 \end{aligned} \tag{2.6.13}$$

where

$$\begin{aligned}
 A_{\alpha\beta}(n_1, n_2) &= x(n_1, n_2) + (-1)^\alpha x(n_1 + N_1/2, n_2) + (-1)^\beta x(n_1, n_2 + N_2/2) \\
 &\quad + (-1)^{\alpha+\beta} x(n_1 + N_1/2, n_2 + N_2/2).
 \end{aligned} \tag{2.6.14}$$

Equation (2.6.13) can be identified as an $(N_1/2 \times N_2/2)$ -point DFT. The DIT and DIF decomposition processes repeat until only DFTs of small sizes remain. Split-radix algorithm for 2D FFT can be obtained by further applying radix-4 decomposition to the non-even-even indexed terms in (2.6.11) and (2.6.13). Detailed derivations are given in [75].

Chapter 3

Study of Signals with Time-varying Frequencies In Gaussian and Impulsive Noises

3.1 Introduction

Most real-life signals, such as radar, sonar, speech and biomedical signals, have time-varying frequencies. As stated in Chapter 2, two classes of methods are introduced to analyze this kind of signals. Parametric methods model the signal's instantaneous phase as a polynomial function of time and estimate the phase coefficients using methods like PTFT [2], HAF [5] and their variations. Non-parametric methods depict the signal energy over both time and frequency (for example, see [12, 13]) or over $t - \omega^0(t)$ space [14], where $\omega^0(t) = (\Omega(t), \Omega^1(t), \dots, \Omega^{m-1}(t))$ is the vector of the instantaneous frequency (IF) $\Omega(t)$ and the derivatives of IF $\Omega(t)$. Thus, they provide temporal and spec-

tral information simultaneously. This chapter presents performance evaluations of these methods in Gaussian and impulsive noise environments.

In Section 3.2, Gaussian noise model is assumed. Numerical examples are presented to compare the performance of the two classes of methods.

In some situations the noise deviates from the Gaussian model. Section 3.3 presents the study of signals with time-varying frequencies in these environments. In particular, it generalizes the robust forms of signal transforms and time-frequency representations (TFRs) based on M -estimation, reviews the robust forms based on L -estimation, and proposes a new approach based on clipping with self-determined clipping threshold value. A comparative investigation on various methods for studying signals corrupted by impulsive noise is presented and analyzed in terms of computational complexity and mean square error (MSE).

Finally, Section 3.4 concludes this chapter.

3.2 Study in Gaussian Noise

To study signals with time-varying frequencies, parametric methods assume the signals of interest follow the PPS model, then determine the model coefficients using various approaches including HAF, PHAF, PTFT, etc. Non-parametric methods employ TFRs, such as STFT, WVD, and LPTFT, to track or estimate the unknown time-varying phase of the signals.

Compared to non-parametric methods, parametric methods have a good noise rejection capability if the number of parameters to be estimated is small relative to the number of data samples [76]. Their main drawbacks can be seen

from the following two aspects. One is that it is often difficult to choose a model which describes well the signal under analysis. The other is that parametric approaches are sensitive to deviations of signal from the assumed model.

Now, let us present some examples to compare the performance of different methods.

Example 1. The first signal considered is an LFM signal embedded in AWGN, the signal has $N = 128$ samples and its IF is confined in the range of $0.05 - 0.45\text{Hz}$. It is known that if the signal follows a linear frequency law, the WVD will reduce to delta functions on the IF law. This suggests the use of the WVD peaks as an IF estimator. For parametric methods, we employ HAF to estimate the phase coefficients and then construct the estimate of IF. The performance is measured by the MSE of IF estimate, which is given by

$$\text{MSE}(n) = 10 \log 10 E\{[\hat{f}_i(n) - f_i(n)]^2\}. \quad (3.2.1)$$

Simulations of 1000 realizations were run for each SNR value, and the evaluated MSEs at $\text{SNR} = 3\text{dB}$ are plotted in Figure 3.1. It can be observed from this figure that the statistical performance of the IF estimator based on WVD deteriorates as the time instants move away from the middle of the time interval, which is in accordance with the analysis performed in [60]. The average MSEs over the whole time interval for WVD and HAF are -25.5 and -62.3 dB, and the average MSEs around the middle of the time interval (for example, between the 40th and 80th time instants) are -63.9 and -64.8 dB respectively. The superiority of HAF over WVD is confirmed. Similar observations can be obtained for other SNR values. This example illustrates the advantage of the parametric methods over non-parametric methods in the estimation accuracy.

Example 2. Consider a synthetic signal which has a quadratic phase for the

Figure 3.1: MSE of IF estimators for WVD and HAF.

first 105 time instants and a cubic phase in the succeeding 195 time instants embedded in $\text{SNR} = 3\text{dB}$ AWGN.

Let us first assume a 2nd-order PPS model, HAF and PHAF with 3 sets of lags used to estimate the phase coefficients. Then IF estimations are obtained with the estimated phase parameters. The same procedures are repeated for a 3rd-order PPS model assumption. The obtained IF estimates (the 'x' symbols) are plotted in Figure 3.2 along with the theoretical IF (the solid lines). It is obvious that the evaluated IFs do not match the correct behavior of the signal. This illustrates that parametric methods methods are sensitive to deviations from the assumed model.

Now, let us employ non-parametric methods, STFT and WVD, to analyze the same signal. MLPTFT, which is based on LPTFT and PTFT, is used too. The TFR peaks are used as IF estimators. From Figure 3.3 which depicts the IF

Figure 3.2: IF estimation using HAF (left column) and PHAF (right column). The first row assumes 2nd-order phase modeling, and the second row assumes 3rd-order phase modeling.

estimates as well as the exact IFs, it can be seen that the MLPTFT estimates the IF accurately with respect to the real one, the WVD provides relative accurate estimates as well, and even the STFT captures some TF characteristics of the analyzed signal.

Example 3. The signal under study, $x(n)$, is a multicomponent PPS with mixed orders and different amplitudes. It consists of one harmonic $x_1(n)$, three LFMs $x_2(n)$, $x_3(n)$, $x_4(n)$, and one QFM $x_5(n)$. The IF of each component is plotted in Figure 3.4(a), and the amplitude of each component is given by $[A_1, A_2, A_3, A_4, A_5] = [10.5, 2.5, 5.5, 4, 8.5]$. The data length is $N = 1024$, and the AWGN with a variance of 0.5 is also added.

To study this signal, let us first apply non-parametric methods including STFT and WVD. Similarly, MLPTFT is used. The obtained results are shown in Figure 3.4(b)-(d). It is clear that the WVD provides a poor description

Figure 3.3: IF estimation based on the peak of TFRs. (left) STFT, (middle) WVD and (right) MLPTFT.

of the signal due to the presence of the crossterms, which detrimentally affect signal interpretation and limit the practical applications of WVD. The STFT and MLPTFT present satisfactory behaviors as each component of the signal is revealed. However, the MLPTFT has a better frequency resolution and a higher component concentration. For example, it can separate the three closely spaced parallel LFM, while the STFT shows blurring between them.

Next, we consider parametric methods. For reference, let us compute the 3rd-order HAF of noiseless unit amplitude $x_5(n)$, and show its magnitude in the top plot of Figure 3.5. Now, we consider the 3rd-order HAF and PHAF of $x(n)$, there should be one peak at the location corresponding to $x_5(n)$, and another peak at zero frequency since $x(n)$ consists of harmonic and LFM components. The middle plot of Figure 3.5 displays the HAF, which has two significant

Figure 3.4: (a) IF, (b) normalized $|\text{STFT}(t,f)|$, (c) normalized $|\text{WVD}(t,f)|$, and (d) normalized $|\text{MLPTFT}(t,f)|$ of a multicomponent PPS.

peaks. However, the peak on the right side is due to the spurious harmonics. The PHAF, as shown in the bottom plot of Figure 3.5, somehow suppresses the crossterms and reveals a distinct peak at zero frequency. But the desired peak corresponding to the highest phase coefficients of $x_5(n)$ is not distinguishable.

The problem of phase parameter estimation of PPS signals with uneven amplitudes and/or mixed order has been addressed in [43], where a bottom-up recursive approach starting from the lowest order $m = 1$ was proposed, and a method was also presented to ascertain whether the maximum of HAF (PHAF) magnitude at each step can be called a peak, whose location corresponds to the

phase coefficient of a signal component. The problem is on the decision between the standard HAF (PHAF) method and the bottom-up HAF (PHAF) method. In this situation, non-parametric method can be used as an initial guess. As an illustration, let us look at the MLPTFT representation in Figure 3.4(d), from which we can find that the signal consists of multi components, has three different types of phase models, and has different amplitudes. The appropriate decision can be made based on these observations accordingly.

Figure 3.5: The 3rd-order HAF of noiseless unit amplitude $x_5(n)$ is shown in the top plot. The 3rd-order HAF and 3rd-order PHAF of $x(n)$ are shown in the middle and bottom plots, respectively. The x-axis: frequency, the y-axis: magnitude.

In summary, for the analysis of signals with time-varying frequencies, the choice of the appropriate methods is signal-dependent, that is to say, it should be made according to the problem at hand. If the analyzed signal can be well modeled by the polynomial approximation, parametric methods can be used to

describe the signals' behavior correctly. If the signal is not the subject of any physical assumption, non-parametric methods should be employed to study the signals and/or act as the initial try before other postprocessing techniques are applied.

3.3 Study in Impulsive Noise

The effects of AWGN to the above mentioned transforms have been well studied in [5, 13, 15, 47, 77, 78]. In some situations, however, the Gaussian assumption does not hold and the noise processes that are characterized by infrequent appearances and high amplitude/energy over short periods. This kind of noise, sometimes referred to as spikes noise or impulsive noise, exists in many applications. For instance, impulsive noise is frequently encountered in digital image transmission as a consequence of man-made noise sources or decoding errors [79]. It is encountered in extremely low frequency (ELF) electromagnetic field due to thunderstorms or in under-ice acoustic environment due to iceberg breakup [78]. In wireless data communication systems, it originates from co-existing electromagnetic and electronic equipment [80]. It is also a serious problem in digital video broadcasting (DVB) systems [81].

When the noise deviates from the Gaussian assumption, the standard signal transforms and TFRs produce poor performance, therefore, alternative analysis tools are required. The purpose of this section is to study different impulsive noise correction mechanisms.

3.3.1 Problem Model

In general, the discrete-time signal transforms can be written as

$$X(k) = \sum_{n=0}^{N-1} x(nT)\varphi_k(nT) \quad (3.3.1)$$

where $x(nT)$ is the length- N signal samples, T is the sampling interval, and $\varphi_k(nT)$, $0 \leq k \leq N-1$, is the basis functions. With the appropriate basis function, numerous discrete transforms can be defined, including DFT, Hadamard, discrete Walsh, discrete cosine, and discrete sine transforms.

Similarly, if we denote the TFR kernel as $K_x(t, nT)$, and the exponential function as $e^{-jg(\boldsymbol{\omega}, nT)}$, where $\boldsymbol{\omega} = (\omega_1, \omega_2, \dots, \omega_m)^T$, $0 \leq \omega_i < 2\pi(i-1)!$, a general form of TFRs can be expressed as

$$X(t, \boldsymbol{\omega}) = \sum_n w_h(nT) K_x(t, nT) e^{-jg(\boldsymbol{\omega}, nT)} \quad (3.3.2)$$

where $w_h(nT) = T/h \cdot w(nT/h)$, with h denoting the window width and $w(t)$ being a real-valued symmetric window, i.e. $w(t) = w(-t)$. A specific TFR is obtained by substituting the appropriate $K_x(t, nT)$ and $g(\boldsymbol{\omega}, nT)$ into (3.3.2). For example, the STFT can be obtained with

$$K_x(t, nT) = x(t + nT), \quad g(\boldsymbol{\omega}, nT) = \boldsymbol{\omega} \cdot nT, \quad (3.3.3)$$

the WVD with

$$K_x(t, nT) = 2x(t + nT)x^*(t - nT), \quad g(\boldsymbol{\omega}, nT) = 2\boldsymbol{\omega} \cdot nT, \quad (3.3.4)$$

and the LPTFT with

$$K_x(t, nT) = x(t + nT), \quad g(\boldsymbol{\omega}, nT) = \sum_{i=1}^m \omega_i \frac{(nT)^i}{i!}. \quad (3.3.5)$$

Note that the LPTFT with nonsymmetric rectangular window coincides with the maximum likelihood estimation (MLE) of PPS phase coefficients based on the PTFT (see Section 2.5.4). Therefore, the robust PTFTs can be obtained from the generalized robust TFR with LPTFT's kernel and exponential functions, as well nonsymmetric rectangular window function.

Consider the noisy sampled signal $x(n)$ given by $x(n) = s(n) + v(n)$, where $v(n)$ are i.i.d. random observation errors with the probability density function (pdf) $f_v(x)$. The goal here is to estimate the signal transform or TFR of the noiseless signal, i.e., $\hat{S}(k)$ or $\hat{S}(t, \omega)$, using signal samples $x(nT)$. The solutions are obtained from the optimization problem given by [82]

$$\begin{aligned}\hat{S}(k) &= \arg \min_m J(m, k) \\ &= \arg \min_m \sum_n F(e(nT, k))\end{aligned}\quad (3.3.6)$$

or

$$\begin{aligned}\hat{S}(t, \omega) &= \arg \min_C J_h(t, \omega; C) \\ &= \arg \min_C \sum_n w_h(nT) F(e(t, nT, \omega))\end{aligned}\quad (3.3.7)$$

where $F(e)$ is the loss function, and

$$e(nT, k) = x(nT)\varphi_k(nT) - m \quad (3.3.8)$$

$$e(t, nT, \omega) = K_x(t, nT) - Ce^{ig(\omega, nT)} \quad (3.3.9)$$

are the error functions.

The loss function $F(e) = -\log f_v(e)$ produces the maximum likelihood estimation (MLE) of $\hat{S}(k)$ or $\hat{S}(t, \omega)$ [82]. That is to say, the MLE depends on the distribution of the noise. When $v(n)$ is Gaussian noise, the standard signal transforms and TFRs given in (3.3.1) and (3.3.2) coincide with the MLE of

$\hat{S}(k)$ and $\hat{S}(t, \omega)$. When the noise pdf $f_v(x)$ deviates from the Gaussian assumption, the standard transforms produce poor performance. In this case, the implementation of MLE requires to approximate the noise distribution function. For some types of noise, such as Middleton Class A noise [83] and the symmetric α -stable ($S\alpha S$) distribution [84], algorithms like these in [85–87] are available to perform this task at the cost of complex estimators. Furthermore, the MLE is quite sensitive with respect to deviations from the hypothetical distribution [25]. For instance, an MLE designed for the Cauchy distributed noise is tested under the $S\alpha S$ noise environments in [88]. Cauchy distribution is a special case of the $S\alpha S$ distribution with $\alpha = 1$. As $|\alpha - 1|$ increases, the degradation in the performance of the estimator can be observed. To address the above problems, alternative analysis tools are to be introduced.

3.3.2 M -Estimation Based Methods

M -estimator, where M stand for maximum likelihood-type, is the solution which optimizes certain function of the data. It is one type of robust estimators based on the Huber's estimation theory. In this section, we propose a general form of robust M -estimation based TFRs for the analysis of signals with time-varying frequencies influenced by impulsive noise. With the appropriate kernel and exponential functions, the corresponding robust M -TFR can be defined. Then, an iterative procedure for calculating the robust TFR is given, and the uniqueness and convergence of the recursive procedure are discussed.

To analyze signals with time-varying frequencies embedded in impulsive noise, the robust M -periodogram [25], the robust Winger-Ville distribution (M -WVD) [26] and the robust polynomial WVD (M -PWVD) [27] were introduced.

As discussed in Chapter 2, the selection of the optimum TFR is a signal-oriented problem and some TFRs outperform others for certain class of signals [13, 61]. For instance, comparative study in [89] shows that Bessel distribution performs the best for the Doppler ultrasound blood flow signal, while investigation of real data in [90] recommends the Cone-Shaped distribution (CSD) as the most appropriate TFR for the analysis of spinal somatosensory evoked potential (SSEP) signals. Therefore, it is important to have alternative tools to deal with different classes of signals with time-varying frequencies corrupted by impulsive noise.

The general form of signal transforms, obtained from using the loss function $F(e) = |e|^p$ in (3.3.6), are presented in Appendix A of [29], which is given by

$$\hat{S}(k) = \frac{\sum_n |e(nT, k)|^{p-2} x(nT) \varphi_k(nT)}{\sum_n |e(nT, k)|^{p-2}}. \quad (3.3.10)$$

The robust and standard signal transforms are obtained with $p = 1$ and $p = 2$ respectively. Consider the same loss function, the robust M -estimation based TFR can be derived from the optimization problem

$$\hat{S}(t, \omega) = \arg \min_C \sum_n w_h(nT) |e(t, nT, \omega)|^p. \quad (3.3.11)$$

It can be easily seen that

$$\begin{aligned} & \frac{\partial J_h(t, \omega; C)}{\partial C^*} \\ &= \frac{p}{2} \sum_n w_h(nT) |e(t, nT, \omega)|^{p-2} \cdot [C - K_x(t, nT) e^{-jg(\omega, nT)}]. \end{aligned} \quad (3.3.12)$$

According to the minimum condition $\frac{\partial J_h(t, \omega; C)}{\partial C^*} = 0$, we get

$$\hat{S}(t, \omega) = \frac{\sum_n w_h(nT) |e(t, nT, \omega)|^{p-2} K_x(t, nT) e^{-jg(\omega, nT)}}{\sum_n w_h(nT) |e(t, n, \omega)|^{p-2}}. \quad (3.3.13)$$

When $p = 2$, the standard TFRs are found to be

$$\hat{S}_S(t, \omega) = \sum_n \frac{w_h(nT)}{\sum_n w_h(nT)} K_x(t, nT) e^{-jg(\omega, nT)} \quad (3.3.14)$$

while the robust TFRs are obtained from $p = 1$,

$$\hat{S}_R(t, \omega) = \frac{\sum_n \frac{w_h(nT)}{|e(t, nT, \omega)|} K_x(t, nT) e^{-jg(\omega, nT)}}{\sum_n \frac{w_h(nT)}{|e(t, nT, \omega)|}}. \quad (3.3.15)$$

Note that the error function $e(t, nT, \omega)$ on the right-hand side of the solution (3.3.15) needs to be calculated using $\hat{S}_R(t, \omega)$. Since $\hat{S}_R(t, \omega)$ appears on both sides of (3.3.15), we need to solve an equation of the form $x = f(x)$. Here, we will use an iterative procedure, similar to the calculation in [25–27], to calculate the robust TFR. It is stated as follows.

- With iteration index $k = 0$, calculate the standard TFR, and set $C^0(t, \omega) = \hat{S}_S(t, \omega)$.
- Set $k = k + 1$, compute $e(t, nT, \omega) = K_x(t, nT) - C^{(k-1)}(t, \omega)e^{jg(\omega, nT)}$, and evaluate $C^{(k)}(t, \omega)$ using (3.3.15). If the relative absolute difference between two iterations is smaller than a fixed threshold η , i.e.,

$$\frac{|C^{(k)}(t, \omega) - C^{(k-1)}(t, \omega)|}{|C^{(k-1)}(t, \omega)|} \leq \eta$$

or the maximal number of iterations (K) has been reached, i.e., $k = K$, then stop the iteration. Otherwise repeat this step.

- Obtain $\hat{S}_R(t, \omega) = C^{(k)}(t, \omega)$.

Uniqueness and convergence of the iterative procedure for robust WVD are given in [26], and the analysis can be easily extended to the general case. The solution (3.3.15) can be expressed in the form $m^{(k+1)} = \sum_n h_n(m^{(k)})\bar{x}_n / \sum_n h_n(m^{(k)})$, where $h_n(m^{(k)}) = w_h(nT)/e(t, nT, \omega)$, and $\bar{x}_n = K_x(t, nT)e^{-jg(\omega, nT)}$. If the initial value is within $C^0 \in [\min(\bar{x}_n), \max(\bar{x}_n)]$, the fix-point iterative algorithm

will converge to a local minimum within the same range [26]. In this case, the initial value (3.3.14) is a mean value, so it satisfies $C^0 \in [\min(\bar{x}_n), \max(\bar{x}_n)]$. Because $J_h(t, \omega; C)$ has a single minimum, the iterative procedure will converge to that single minimum. Note that the robust TFRs introduced in [25–27] can be obtained from (3.3.15) and calculated through the above iterative procedures by substituting the corresponding kernel function $K_x(t, nT)$ and exponential function $e^{-jg(\omega, nT)}$.

3.3.3 L -Estimation Based Methods

L -estimator, where L stand for linear combination of order statistics, is another type of robust estimator utilized when the noise has impulsive characters. In this section, we review the main concepts and the reported algorithms for parameter selection in L -estimators.

It has been shown in [29] that the optimization problems in (3.3.6) can be solved using the marginal median (MM) approach. The MM approach treats the real and imaginary parts of the error function, i.e., $\Re\{e\}$ and $\Im\{e\}$, as statistically independent processes which are separately minimized. With the loss function $F(e) = |\Re\{e\}| + |\Im\{e\}|$, the solution is of the form [29]

$$\hat{S}_{MM}(k) = \text{median}\{\Re\{\mathbf{E}_k\}\} + j \text{median}\{\Im\{\mathbf{E}_k\}\}, \quad (3.3.16)$$

where $\mathbf{E}_k = \{x(nT)\varphi_k(nT), 0 \leq n \leq N-1\}$. If the signal and the basis functions are real-valued, the above solution is reduced to a median filter,

$$\hat{S}_{MM}(k) = \text{median}\{x(nT)\varphi_k(nT), 0 \leq n \leq N-1\}. \quad (3.3.17)$$

The optimization problem in (3.3.7) can be solved using MM approach similarly,

and the solution is given by [28, 29]

$$\hat{S}_{MM}(t, \omega) = \text{median}\{\Re\{\mathbf{E}_{t, \omega}\}\} + j \text{median}\{\Im\{\mathbf{E}_{t, \omega}\}\}, \quad (3.3.18)$$

where $\mathbf{E}_{t, \omega} = \{w_h(nT)K_x(t, nT)e^{-jg(\omega, nT)}, -h/(2T) < n < h/(2T)\}$. Some TFRs have the property of realness, i.e. $X^*(t, \omega) = X(t, \omega)$, then the MM based TFRs are reduced to

$$\begin{aligned} \hat{S}_{MM}(t, \omega) = \text{median}\{\Re\{w_h(nT)K_x(t, nT)e^{-jg(\omega, nT)}\}, \\ -h/(2T) < n < h/(2T)\}. \end{aligned} \quad (3.3.19)$$

Note that the optimization problems in (3.3.6) and (3.3.7) can also be solved using vector median (VM) method [91]. As illustrated in [28], the optimization problems with the loss function $F(e) = |e|$ can be understood as filtering of the vector-based variables $x(nT)\varphi_k(nT)$ or $w_h(nT)K_x(t, nT)e^{-jg(\omega, nT)}$, where the real and imaginary parts of the variables are the vector coordinates. The experiments in [28] show that the robust forms using VM and MM produce similar results. The statistical comparisons in [92] also confirm that the VM filter and MM filter have similar performances. The latter is preferable in most practical systems because the former requires a higher computational complexity.

3.3.4 Clipping Method

A standard tool dealing with impulsive noise is clipper [84] due to its natural ability to eliminate outliers. Passing a signal $x(n)$ through a clipper, the output is given by

$$y(n) = \begin{cases} x(n) & |x(n)| \leq \varepsilon \\ \varepsilon e^{j \arg(x(n))} & \text{otherwise} \end{cases} \quad (3.3.20)$$

where the $\arg$ function extracts the angular component of a complex number or function, and ε is the threshold value of the clipper. The design problem of clipper is the selection of threshold value ε . In this section, we propose a method to determine the threshold value according to the characteristic of the available observation samples, and then apply the standard signal transform or TFR to the clipped signal.

Assume for a given vector $\mathbf{y}$, $\bar{\mathbf{y}}$ and $\tilde{\mathbf{y}}$ return the mean and median values of the vector, the function $\text{mad}(\mathbf{y})$ returns the median absolute deviation about the median (MAD) of the vector, i.e.,

$$\text{mad}(\mathbf{y}) = \text{median}\{|y(n) - \tilde{\mathbf{y}}|, 0 \leq n \leq N - 1\}. \quad (3.3.21)$$

Let $x_r(n)$ and $x_i(n)$ denote the real and imaginary parts of $x(n)$, i.e., $\mathbf{x}_r = \{|\Re\{x(n)\}|, 0 \leq n \leq N - 1\}$ and $\mathbf{x}_i = \{|\Im\{x(n)\}|, 0 \leq n \leq N - 1\}$. The threshold value is determined by

$$\varepsilon = \begin{cases} 2 \sum_{n=0}^{N-1} |x(n)| / N & p < p_0 \\ (\text{mad}(\mathbf{x}_r) + \text{mad}(\mathbf{x}_i)) / 8 & \text{otherwise} \end{cases} \quad (3.3.22)$$

where

$$\begin{aligned} p &= (p_r + p_i) / 2 \\ x_{ar}(n) &= |x_r(n)|, \quad 0 \leq n \leq N - 1 \\ x_{ai}(n) &= |x_i(n)|, \quad 0 \leq n \leq N - 1 \\ p_r &= \Pr\{x_{ar}(n) \geq 2\bar{x}_{ar}, 0 \leq n \leq N - 1\} \\ p_i &= \Pr\{x_{ai}(n) \geq 2\bar{x}_{ai}, 0 \leq n \leq N - 1\}. \end{aligned} \quad (3.3.23)$$

Clipping may introduce distortion to signal samples. If we choose relatively large value of p_0 , more signal samples may not be corrupted. However, more

effect of impulsive noise remains. The parameter p_0 was set as 5% in all the numerical examples in this chapter. This setting of p_0 was based on intensive experiments.

Note that it is generally hard to determine the optimal clipping value, especially when the impulsive noise model and parameters are unknown. The basic idea here is to use characteristics of the signal samples to decide threshold values, in particular, the characteristics used include the average of sample amplitude, the MADs of the real and imaginary part of samples, the percentage of the absolute values of real and imaginary part of samples larger than twice of their averages. If the average percentage is small, we set a relatively large threshold value, which is twice of the average of sample amplitude, otherwise we choose a relatively small value.

Compared with the standard transforms, the major computation overload introduced by the proposed method comes from the two MAD calculations in (3.3.22). The MAD can be computed with two median operations. The median of n items can be obtained by sorting the list and then extracting the desired element. This method typically requires a computation complexity in the order from $O(n \lg n)$ to $O(n^2)$. It should be pointed out that the median can be computed using a “divide and conquer” algorithm [93] as well, which requires only $O(n)$ operations.

3.3.5 Numerical Study

This section presents several numerical examples to justify the proposed approach, and compares the performances of different methods dealing with impulsive noise. Error performance and computational complexity are used as

comparison measures.

Experiment 1: Consider the HAF of a noisy signal $x(n)$ with $N = 256$ samples. The noiseless signal is a linear FM given by

$$s(n) = 4e^{j2\pi(a_1n+a_2n^2)+j\phi}, \quad 0 \leq n \leq N-1 \quad (3.3.24)$$

where $a_1 = 0.2$, $a_2 = 0.001$, ϕ is uniformly distributed in $(0, 2\pi)$, and it is influenced by standard Cauchy distributed complex noise, i.e.,

$$v(n) = v_1(n) + jv_2(n) \quad (3.3.25)$$

where $v_1(n)$ and $v_2(n)$ are mutually independent with the standard Cauchy distribution $f_v(x) = \frac{1}{\pi(1+x^2)}$. Figure 3.6(a) shows the signal and noise waveforms. As mentioned in Chapter 2, the $\text{mlHIM}_2[x(n); \tau]$ reduces the signal into a harmonic at frequency $f_0 = 4\tau a_2$, thus, the estimate of a_2 can be obtained from the peak location of $\text{mlHAF}[x, f, \tau]$. Figure 3.6(b) displays the standard, median-based, M -estimation based $\text{mlHAF}[x, f, \tau]$, and the standard $\text{mlHAF}[x, f, \tau]$ with clipping pre-processing. For convenience, all the plots are shown on a scaled frequency axis, i.e., $f/(4\tau)$ instead of f , so that the phase coefficient estimate can be read directly from the peak locations in these plots. It can be easily seen that the impulsive noise destroys the standard $\text{mlHAF}[x, f, \tau]$, as the peak of the harmonic at scaled frequency a_2 is not distinguished. At the same time, the peak at a_2 can be clearly seen on the other three figures. Besides the typical realizations shown in Figure 3.6(b), the MSEs of the estimates $\hat{a}_2$ obtained from the standard, median, M -estimation, and clipping methods are calculated from 100 independent realizations, which are 1.9501×10^{-6} , 5.6833×10^{-11} , 5.5492×10^{-11} , and 5.566×10^{-11} , respectively. It is obvious that the robust forms provide orders of magnitude improvements in MSE.

Figure 3.6: (a) Observations of the signal and noise, x-axis: time, y-axis: amplitude, (b) standard and robust forms of $mIHAF[x, f, \tau]$, x-axis: scaled frequency, y-axis: normalized magnitude.

Experiment 2: Consider a signal with a linear IF corrupted by heavy-tailed noise. The phase model considered in [14] is used here, which is $\varphi(t) = 2\pi t + 8\pi t^2/2$, and the sampling interval $T = 0.1$ s. The impulsive noise is assumed to have the same model as in [31, 94], that is to say, positive and negative impulses with the same probability $p/2 = 5\%$ and amplitude $A = 10$ appear in both real and imaginary parts. Figure 3.7(a) shows the signal and noise waveforms. The TFR used here is the second-order LPP with a window width $h = 10.1$ s. For Gaussian noise, the standard LPP has the energy concentration in the points $\omega^0(t) = [\Omega(t), \Omega^{(1)}(t)]$, thus IF can be estimated from the peaks of Max-LPP. In impulsive noise environment, however, the standard LPP yields poor performance, as shown in Figure 3.7(b), where the normalized standard and robust forms of $\text{LPP}(t, \omega_1, \omega_2)$ at the time instant $t = 0.2$ s are depicted. It is obvious that the robust forms of the LPP show concentrations around IF $\Omega(0.2) = 11.31$ and its first derivative $\Omega^{(1)}(0.2) = 25.13$. Figure 3.8 displays the IF estimates (plus sign “+”) obtained from standard and robust forms of LPP at four time instants $t = (0.2, 0.4, 0.6, 0.8)$ s. It also plots the exact IFs in solid lines. It can be clearly seen that the robust forms produce accurate estimates while the standard one becomes useless due to the effect of impulses. Generating 100 independent realizations, the MSEs of IF estimate $\hat{\Omega}(t)$, obtained from the standard, median, M -estimation, and clipping methods, are found to be 1.4267, 0.0196, 0.0002 and 0.0003, respectively. The superiority of the robust forms over the standard one is obvious.

Experiment 3: Consider a sinusoidal FM (SFM) signal with $N = 520$ samples embedded in impulsive noise. The signal’s IF is within the range $[0.05, 0.45]$, the period of SFM is 380, the IF at time instant $t_0 = 200$ is 0.3 and the IF is in the increasing direction around t_0 . The noise model used is the cubic of

Figure 3.7: (a) Observations of the signal and noise, x-axis: time, y-axis: amplitude, (b) images of normalized standard and robust forms of LPP at $t = 0.2s$, x-axis: $\Omega(t)$ (IF) and y-axis: $\Omega^{(1)}(t)$ (first derivative of IF).

Figure 3.8: IF estimates obtained from standard and robust forms of LPP, x-axis: time t and y-axis: IF $\Omega(t)$.

Gaussian noise [25], i.e.,

$$v(n) = a(v_1^3(n) + jv_2^3(n)), \quad (3.3.26)$$

where $v_1(n)$ and $v_2(n)$ are mutually independent Gaussian noise with zero mean and unitary variance, i.e., $v_i \sim \mathcal{N}(0, 1)$ for $i = 1, 2$, and the variance of the kind of noise is $\sigma_v^2 = 30a^2$. Figure 3.9(a) shows the signal and noise waveforms. The TFR adopted is the modified LPTFT (MLPTFT) proposed in [66], which first divides the signal into a number of segments, then locally estimates the parameters needed by LPTFT using 2nd order PTFT within each segment, and finally computes the LPTFT at every time instant with the obtained estimates. In order to obtain the robust forms to deal with the impulsive noise, we need to apply robust PTFT estimations within each segment, and then calculate robust LPTFT at each time instant. In the simulations, the noise parameter

is set as $a = 0.75$, and the MLPTFT divides the given signal into non-overlap segments with length 65. Figure 3.9(b) presents the standard and robust forms of the MLPTFT, and clearly shows that good performance is achieved by using robust forms instead of the standard one, for the robust forms can achieve concentration along the signal's IF while the standard one cannot. The MSEs of the IF estimates, calculated from 100 independent realizations, are 12.6134, 0.5854, 0.5713, and 0.1563, respectively.

From these examples we can observe a significant improvement of the robust forms over the standard ones in the case of impulsive noise. This observation verifies our proposed robust forms in Sections 3.3.2 and 3.3.4.

Computational complexity analysis: Now, let us analyze the computational complexity of robust forms of transforms. The M -estimation based forms and the L -estimation based forms require iterative procedures and sorting procedures, respectively. In general, the transforms map 1-D signal to 1-D or multi-dimensional space to obtain the robust forms, and the iterative or sorting procedure must be performed for each point in the space. For instance, the DFT maps 1-D time domain signal to 1-D frequency domain to obtain the robust forms of the DFT, and the iterative or sorting is performed in each frequency bin. In contrast, the TFR (such as the WVD) maps 1-D time domain signal to 2-D time-frequency space, and the iterative or sorting is required by all the points in the 2-D space to get the robust forms of the TFR. For the proposed clipping method, as discussed earlier, its main computation overhead comes from the two MAD calculations.

Another factor affecting the computational complexity is that standard signal transforms and TFRs may utilize efficient computational algorithms, such as

Figure 3.9: (a) Observations of the signal and noise, x-axis: time, y-axis: amplitude, (b) standard and robust modified LPTFT (MLPTFT), x-axis: time, and y-axis: frequency.

FFT algorithms for DFT and fast algorithms for PTFT. Because the proposed clipping method processes the signal before applying the standard transform, it can still use the efficient computation methods (if available). On the contrary, the M -estimation based and L -estimation based forms change the computation structures, thus they cannot use the existing fast algorithms.

To compare the computational complexity required by the different robust forms, we compute the ratio between the computation time needed by the corresponding robust form transform and that required by the standard transform. The speed comparisons are performed on a MATLAB 6.5 platform to measure the computation time by using the variable ‘clock’ and function `etime` provided by MATLAB. In experiment 1 and experiment 3, FFT and fast PTFT algorithms (which will be described in the following chapters) are utilized in the implementations of standard transforms, while in experiment 2, the LPP is computed directly according to its definition. The results based on the average of computation time for 100 simulations are listed in Table 3.1. Compared with the standard form, both M -estimation and L -estimation based robust forms are computationally consuming, especially when the standard form can be implemented via efficient algorithms. For the proposed clipping method, if the computational complexity required by the MAD calculations is trivial compared to that needed by the standard transform, it requires almost the same computation time as the standard form. Otherwise its computation complexity is several times of that of standard transform. Among all the robust forms, the M -estimation based forms requires the longest computation time, while the clipping method is the most computational efficient one.

It should be mentioned that the ratios shown in Table 3.1 can be used

only as the general tendency of the comparison between the computational complexities required by different algorithms because the measured computing time may be affected by a number of factors that may not directly related to the computational complexity.

Table 3.1: Comparison of computation time in impulsive noise.

	ratio-median	ratio-M	ratio-clipping
experiment 1	112.05	439.04	3.86
experiment 2	2.76	6.02	1.05
experiment 3	80.97	260.02	1.06

3.4 Summary

In this chapter, comparisons were made between different methods for the analysis of signals with time-varying frequencies. In general, there is no single method that is effective in all situations. If the signal under study is well modeled by the polynomial approximation, parametric methods should be employed for the analysis; otherwise other methods may be a appropriate choice. Among parametric methods, the PTFT has the advantages including asymptotically achieving the CRLB, linearity and higher output SNR. Its major drawback is that the PTFT requires a high computation complexity. MLPTFT, which is based on PTFT and non-parametric method LPTFT, is a good choice if the signal under study consists of multicomponents or there is no prior information about the signal’s behavior. However, the MLPTFT relies on the use of PTFT to provide the needed parameter estimations. Thus it suffers from the huge computation requirement of PTFT too.

Study of the non-stationary signals in impulsive noise is also performed in

this chapter. A general form of robust M -estimation based signal transform and TFR is proposed for analyzing signals in the impulsive noise environments. The robust form can be calculated from the standard one through a recursive process. L -estimation based forms are another class of algorithms for use when the noise deviates from the Gaussian model, and they are briefly reviewed. Clipper is a standard tool to deal with impulsive noise. A method is introduced to decide the clipping value according to the observation samples. Experiments are performed to justify the proposed methods. Comparisons among the M -estimation based form, L -estimation based form and the proposed clipping method are made according to the criteria including statistical error measure and the required computation time. Among all the robust forms, the proposed clipping method requires the smallest computational cost with a performance similar to that obtained by other methods, especially when the standard transforms can be computed effectively using fast algorithms.

Chapter 4

Fast Algorithms for PTFTs of Complex-Valued Sequences

4.1 Introduction

Polynomial-phase signals (PPSs) are often used in many applications, such as pulse compression radar systems [5], synthetic aperture radar imaging [95] and mobile communications [34, 96]. Most applications require the estimation of the coefficients a_m from the received signal $x(n)$ [1, 2, 11, 43, 60, 97], because these coefficients carry desirable information (see Section 1.1 for some examples).

As stated in Chapter 2, for estimation of PPS phase coefficients, there mainly exist ML estimation methods using the PTFTs [1–4] and suboptimal ML methods based on HAF [5] and its extensions such as IGAF [10] and PHAF [11]. The advantages of PTFT have been introduced in Section 2.4.4 from the aspects of estimation variances and output SNRs with details. Examples of combining the PTFT with LPTFT for the analysis of signals with time-varying frequencies

can also be found in [66] and the comparative investigations in Section 2.6.

In [2], minimizing the nonlinear least-squares (NLS) criterion results in the PTFT for a complex PPS. The analysis in [1, 4] uses complex signal and complex PPS model is adopted in [3, 5–11] as well. Thus, in this chapter, the signal is assumed to be complex-valued. This chapter is organized as follows. In Section 4.2, we review the reported fast algorithms to compute PTFT. Section 4.3 presents a symmetric property of the PTFT to reduce the number of transform points that are to be computed. Procedures are also proposed to convert a PTFT, whose dimensional sizes are different, into a number of smaller PTFTs whose dimensional sizes are the same. Section 4.4 describes the derivation details of the fast algorithm. Section 4.5 provides the analysis of the required computational complexity for the length- a PTFT, and examples are given for the 2nd and 3rd order PTFTs. Finally, Section 4.6 presents the conclusions.

4.2 Review of the Reported Fast PTFT Algorithms

In general, the PTFT requires a huge computation complexity to deal with multi-dimensional (MD) calculation. To reduce the computational complexity, some algorithms proposed in the literature are to be reviewed in this section.

The definition of the PTFT was mentioned in Chapter 2 and is rewritten here for the convenience of the readers to have a better understanding on the remaining part of this chapter. The $(r + 1)$ th order PTFT is generally defined

as:

$$X(k_0, k_1, \dots, k_r) = \sum_{n=0}^{N_0-1} x(n) W_{N_0}^{k_0 n} W_{N_1}^{k_1 n^2} \dots W_{N_r}^{k_r n^{r+1}}, \quad (4.2.1)$$

$$0 \leq k_i \leq N_i - 1, \quad 0 \leq i \leq r$$

where $W_N = e^{-j2\pi/N}$, $x(n)$ is the one-dimensional (1D) input sequence of N_0 points and N_i is the size of the i th dimension of $X(k_0, k_1, \dots, k_r)$. As discussed in section 2.4.2, it is required that $N_j > N_i > N_0$ for $j > i > 0$ to achieve a satisfactory accuracy for parameter estimation.

Defining demodulated sequences of $x(n)$ as

$$x^{(k_1, k_2, \dots, k_r)}(n) = x(n) W_{N_1}^{k_1 n^2} \dots W_{N_r}^{k_r n^{r+1}}, \quad (4.2.2)$$

the PTFT of $x(n)$ can be expressed as

$$X(k_0, k_1, \dots, k_r) = \text{DFT}_n[x^{(k_1, k_2, \dots, k_r)}(n)], \quad (4.2.3)$$

where the subscript n indicates that the DFT is performed in terms of index n . Thus, the computation task defined in (4.2.1) can be performed using 1D DFT calculations. This is similar to the row-column method for the MD DFTs. For an $(r+1)$ th order PTFT of a length- N_0 input sequence, $(\prod_{i=1}^r N_i)$ 1D length- N_0 DFTs are required. Although the computational cost can be reduced with the use of 1D fast Fourier transforms (FFTs) [98], the PTFT still needs a huge computational complexity. For example, assuming the input sequence length N is a power of two, it is well known that the computational complexity of 1D FFT is $O(N \log_2 N)$. Thus the computational complexity of an $(r+1)$ th order PTFT is in the order of $(\prod_{i=1}^r N_i) N \log_2 N$.

To achieve a higher computational efficiency, fast quadratic phase transform (FQPT) [3] was proposed for the second order PTFT, which can reduce the

computational complexity by a factor of $\log_2 N$ compared with that needed by directly using the 1D FFTs, where N is the input sequence length. The work in [3] was extended for the third order PTFT and was reported in [99]. It was further generalized to support arbitrary order PTFT in [100] which exploits the “quasi-periodic property” of the PTFT to reduce the overall computational complexity. The algorithm uses the concept of DIT decomposition along each dimension of k_i for $i = 1, 2, \dots, r$. For a given $(k_1, k_2, \dots, k_r)$, the output at q th stage of the decomposition can be defined as

$$x_q^{(k_1, k_2, \dots, k_r)}(k_0) = \text{DFT}_m [x^{(k_1, k_2, \dots, k_r)}(2^q m + s)] \quad (4.2.4)$$

where q is from $\log_2 N - 1$ to 0, $m = 0, 1, \dots, N_0/2^q - 1$, and $s = 0, 1, \dots, 2^r - 1$.

A “key property” was found, which can be stated as [100, (16)]

$$\begin{aligned} & x_q^{(k_1 + a_1 N_1/2^{q+1}, k_2 + a_2 N_2/2^{q+1}, \dots, k_r + a_r N_r/2^{q+1})}(k_0) \\ &= \begin{cases} \Phi(q, s) \left[x_{q+1}^{(k_1, k_2, \dots, k_r)}(k_0) + W^h x_{q+1}^{(k_1, k_2, \dots, k_r)}(k_0 + 2^r) \right] & \gamma \text{ is even} \\ \Phi(q, s) \left[x_{q+1}^{(k_1, k_2, \dots, k_r)}(k_0) - W^h x_{q+1}^{(k_1, k_2, \dots, k_r)}(k_0 + 2^r) \right] & \gamma \text{ is odd} \end{cases} \end{aligned} \quad (4.2.5)$$

where $a_i = 0$ or 1 for $i = 1, 2, \dots, r$, h is an integer obtained by shifting and bit-reversing the binary expression of k_0 , $\gamma = \sum_{i=1}^r a_i(i+1)s^i$, and $\Phi(q, s) = e^{-j2\pi \sum_{i=1}^r a_i s^{i+1}/2^{r+1}}$. This property suggests that the outputs with parameters $k_i + a_i N_i/2^{q+1}$ at stage- q can be obtained by combining the outputs with parameters k_i at stage- $(q+1)$. Thus, the redundancy along $(k_1, k_2, \dots, k_r)$ is removed. In general, it reduces the numbers of complex multiplications and additions by a factor of $2^r \log_2 N$ for the $(r+1)$ -th order PTFT of length- N input sequence, compared with those needed by only using the 1D FFTs.

However, further reduction on computational complexity can be achieved because some properties of the PTFT are not fully utilized. Recent work [101,

102] was reported to achieve a better computational efficiency for an arbitrary order of the PTFT.

It is noted that these reported fast algorithms for PTFT [3, 99–102] support sequence length being a power of two only. Fast algorithms for other sequence lengths are also desired for applications that need sequence lengths rather than a power of two. It is well known that when the sequence length is not supported by the available fast algorithms, zero padding techniques have to be used to augment the input sequence to the next available size supported by the fast algorithm. This mismatch obviously wastes the computational resources and decreases the computational efficiency of the system. The availability of fast algorithms based on various radix numbers is very useful to minimize the possibility and the degree of the mismatch between the input sequence length and the transform sizes supported by the fast algorithm. This motivates us to propose a general method to derive fast algorithms for the PTFTs of various sequence lengths and any order of the PTFTs.

4.3 Properties and Decomposition

The $(r + 1)$ th order PTFT is defined in (4.2.1). To support various sequence lengths, we assume that $N_0 = a^p b$, where a , b and p are positive integers and $a \neq b$. In practice, it is often $N_j > N_i > N_0$ for $j > i > 0$ to achieve a satisfactory accuracy of the parameter estimation. Thus, it is assumed that $p_i = N_i/N_0$ is a positive integer for any $i > 0$.

4.3.1 The Symmetric Property

Let us first minimize the number of PTFT outputs to be computed in (4.2.1) by using a symmetric property of the PTFT described as [100]:

$$\begin{aligned}
 & X(k_0, k_1 + \alpha_1 \frac{N_1}{2}, \dots, k_r + \alpha_r \frac{N_r}{2}) \\
 &= \begin{cases} X(k_0, k_1, \dots, k_r) & \text{if } \sum_{i=1}^r \alpha_i \text{ is even} \\ X(k_0 + \frac{N_0}{2}, k_1, \dots, k_r) & \text{if } \sum_{i=1}^r \alpha_i \text{ is odd} \end{cases}
 \end{aligned} \tag{4.3.1}$$

where $0 \leq k_0 \leq N_0 - 1$, $0 \leq k_i \leq N_i/2 - 1$, and $\alpha_i = 0, 1$ for $i = 1, 2, \dots, r$. This symmetric property indicates that the computation for those PTFT points with indices $k_i = N_i/2, \dots, N_i - 1$ is not necessary because they can be replaced by those with indices $k_i = 0, \dots, N_i/2 - 1$. For an $(r + 1)$ th-order PTFT, this property means that the space for each k_i is reduced by half. Thus, the entire PTFT computation space is reduced by 2^l times if there are l dimensions whose sizes are even. With the symmetric property, for example, the 3rd-order PTFT can be expressed as:

$$\begin{aligned}
 X(k_0, k_1 + N_1/2, k_2 + N_2/2) &= X(k_0, k_1, k_2), \\
 X(k_0, k_1 + N_1/2, k_2) &= X(k_0 + N_0/2, k_1, k_2), \\
 X(k_0, k_1, k_2 + N_2/2) &= X(k_0 + N_0/2, k_1, k_2),
 \end{aligned} \tag{4.3.2}$$

where $k_i = 0, \dots, N_i/2 - 1$. It can be verified that the total number of PTFT outputs is $(N_0 N_1 N_2)/4$, instead of $N_0 N_1 N_2$.

In general, it is difficult to apply the decomposition techniques based on a radix value for PTFTs that have different dimensional sizes. This can be illustrated by the following 2nd-order PTFT, which is given by

$$X(k_0, k_1) = \sum_{n=0}^{N_0-1} x(n) W_{N_0}^{k_0 n} W_{N_1}^{k_1 n^2} \tag{4.3.3}$$

where $0 \leq k_0 \leq N_0 - 1$, $0 \leq k_1 \leq N_1 - 1$ and N_0 is a power of two. If we use the DIT decomposition to split the input sequence into two subsequences $x(2n)$ and $x(2n + 1)$, the $X(k_0, k_1)$ is written as

$$\begin{aligned} X(k_0, k_1) &= \sum_{n=0}^{N_0/2-1} x(2n) W_{N_0/2}^{k_0 n} W_{N_1/2}^{2k_1 n^2} \\ &+ \sum_{n=0}^{N_0/2-1} x(2n+1) W_{N_0}^{k_0} W_{N_1}^{k_1(4n+1)} W_{N_0/2}^{k_0 n} W_{N_1/2}^{2k_1 n^2} . \end{aligned} \quad (4.3.4)$$

Unlike the 2D FFT, the PTFT with size $(N_0 \times N_1)$ cannot be decomposed into two PTFTs with size $(N_0/2 \times N_1/2)$. If DIF decomposition is used, we have

$$\begin{aligned} &X(2k_0 + \alpha, 2k_1 + \beta) \\ &= \sum_{n=0}^{N_0/2-1} \left\{ \left[x(n) + (-1)^\alpha e^{-j2\pi(2k_1+\beta)\left(\frac{nN_0}{N_1} + \frac{N_0^2}{4N_1}\right)} x\left(n + \frac{N_0}{2}\right) \right] W_{N_0}^{\alpha n} W_{N_1}^{\beta n^2} \right\} \\ &\quad \cdot W_{N_0/2}^{k_0 n} W_{N_1/2}^{k_1 n^2} . \end{aligned} \quad (4.3.5)$$

It is easy to see that the PTFT can be decomposed into four sums only when the twiddle factor inside the square brackets $e^{-j2\pi k_1\left(\frac{2nN_0}{N_1} + \frac{N_0^2}{2N_1}\right)} = 1$, in which $N_0 = N_1$ becomes a sufficient condition. Therefore, the computational task defined in (4.2.1) is dealt with by two steps. The first one divides the entire computational task into many smaller ones that have equal dimensional sizes, which is to be described as follows in this section. The second step further decomposes these smaller PTFTs based on the periodic and symmetric properties of the PTFTs, which is similar to the decomposition process used for deriving FFT algorithms and is to be presented in the next section.

4.3.2 First Step Decomposition (FSD)

It is shown in [101, 102] that the $(r + 1)$ th order PTFT defined in (4.2.1) can be expressed as

$$\begin{aligned} X(k_0, p_1 k_1 + m_1, \dots, p_r k_r + m_r) \\ = \sum_{n=0}^{N_0-1} \left\{ x(n) W_{N_1}^{m_1 n^2} \dots W_{N_r}^{m_r n^{r+1}} \right\} W_{N_0}^{\theta_r(n, \mathbf{k})} \end{aligned} \quad (4.3.6)$$

where $\theta_r(n, \mathbf{k}) = k_0 n + \dots + k_r n^{r+1}$, $0 \leq k_i \leq N_0 - 1$ and $0 \leq m_i \leq p_i - 1$ for $i = 1, \dots, r$. If the symmetric property in (4.3.1) is used, the range of k_i should be reduced by one half. Equation (4.3.6) defines $p_1 p_2 \dots p_r$ $(r + 1)$ th order PTFTs whose sizes of all dimensions are the same, i.e., N_0 . The decomposition cost is $(p_1 p_2 \dots p_r - 1)(N_0 - 1)$ complex multiplications. It was pointed out in [101, 102] that when p_i , $1 \leq i \leq r$, is odd, the multiplicative overhead could be further minimized.

We now consider the general case in which the values p_i can be even or odd. The decomposition process is similar to that reported in [102]. In general, the PTFT defined in (4.2.1) is converted into $p_1 p_2 \dots p_r$ PTFTs whose sizes of all dimensions are N_0 . When $m_i = 0$ for $i = 1, \dots, r$, we have

$$X(k_0, p_1 k_1 \dots, p_r k_r) = \sum_{n=0}^{N_0-1} x(n) W_{N_0}^{\theta_r(n, \mathbf{k})}. \quad (4.3.7)$$

Define

$$\begin{aligned} X_1(k_0, \dots, k_r) &= \sum_{n=0}^{N_0-1} [x(n) \cos(\gamma)] W_{N_0}^{\theta_r(n, \mathbf{k})} \\ X_2(k_0, \dots, k_r) &= \sum_{n=0}^{N_0-1} [x(n) \sin(\gamma)] W_{N_0}^{\theta_r(n, \mathbf{k})} \end{aligned} \quad (4.3.8)$$

where $\gamma = 2\pi\theta_r(n, \mathbf{m}')/N_0$ and $m'_i = s_i m_i / p_i$ for $s_i \in \{1, -1\}$. It is noted that the sinusoidal factors in (4.3.8) are real and each requires two real multiplica-

tions. The final PTFT outputs can be easily obtained from

$$\begin{aligned} X(k_0, p_1 k_1 \pm s_1 m_1, \dots, p_r k_r \pm s_r m_r) \\ = X_1(k_0, \dots, k_r) \mp j X_2(k_0, \dots, k_r). \end{aligned} \quad (4.3.9)$$

where $0 \leq k_i \leq N_0 - 1$ for $i = 0, 1, \dots, r$. The above decomposition can be used for any values of p_i . When p_i is even, the range of m_j is from zero to $p_i/2 - 1$ and when p_i is odd, m_i is from zero to $(p_j - 1)/2$. Special attention must be given when $m_i = p_i/2$. In addition, when $p_i = 0$, i.e., the index of the transform output is negative, the periodicity property of W_N needs to be applied. For illustration, let us consider an example for $r = 2$. Define

$$\begin{aligned} A(k_0, k_1, k_2) &= \sum_{n=0}^{N_0-1} [x(n) \cos(2\pi\alpha)] W_{N_0}^{\theta_2(n,k)} \\ B(k_0, k_1, k_2) &= \sum_{n=0}^{N_0-1} [x(n) \sin(2\pi\alpha)] W_{N_0}^{\theta_2(n,k)} \\ C(k_0, k_1, k_2) &= \sum_{n=0}^{N_0-1} [x(n) \cos(2\pi\beta)] W_{N_0}^{\theta_2(n,k)} \\ D(k_0, k_1, k_2) &= \sum_{n=0}^{N_0-1} [x(n) \sin(2\pi\beta)] W_{N_0}^{\theta_2(n,k)} \end{aligned} \quad (4.3.10)$$

where $\alpha = (\frac{m_1}{p_1} n^2 + \frac{m_2}{p_2} n^3)/N_0$, and $\beta = (\frac{m_1}{p_1} n^2 - \frac{m_2}{p_2} n^3)/N_0$. The 3rd order PTFT outputs can be easily obtained from

$$X(k_0, p_1 k_1, p_2 k_2) = \sum_{n=0}^{N_0-1} x(n) W_{N_0}^{\theta_2(n,k)} \quad (4.3.11)$$

and the PTFTs defined in (4.3.10), such as

$$\begin{aligned} X(k_0, p_1 k_1 \mp m_1, p_2 k_2 \mp m_2) &= A(k_0, k_1, k_2) \pm j B(k_0, k_1, k_2) \\ X(k_0, p_1 k_1 \mp m_1, p_2 k_2 \pm m_2) &= C(k_0, k_1, k_2) \pm j D(k_0, k_1, k_2) \\ X(k_0, -m_1, p_2 k_2 - m_2) &= X(k_0, N_1 - m_1, p_2 k_2 - m_2) \\ X(k_0, p_1 k_1 - m_1, -m_2) &= X(k_0, p_1 k_1 - m_1, N_2 - m_2) \\ X(k_0, -m_1, -m_2) &= X(k_0, N_1 - m_1, N_2 - m_2). \end{aligned} \quad (4.3.12)$$

To obtain all the transform outputs when p_1 and p_2 are odd numbers, the overall additive overhead is $(p_1 p_2 - 1)N_0^3$ complex additions/subtractions and the multiplicative overhead is $2(p_1 p_2 - 1)(N_0 - 1)$ real multiplications. When p_i is even, special attention must be given when $m_i = p_i/2$. For example, if p_1 is even, we have

$$E(k_0, k_1, k_2) = \sum_{n=0}^{N_0-1} x(n) \cos(\phi) W_{N_0}^{\theta_2(n,k)} \quad (4.3.13)$$

$$F(k_0, k_1, k_2) = \sum_{n=0}^{N_0-1} x(n) \sin(\phi) W_{N_0}^{\theta_2(n,k)}. \quad (4.3.14)$$

where $\phi = 2\pi(0.5p_2 n^2 + m_2 n^3)/N_2$. Then we have

$$X(k_0, p_1(k_1 + 0.5), p_2 k_2 \pm m_2) = E(k_0, k_1, k_2) \mp jF(k_0, k_1, k_2) \quad (4.3.15)$$

$$X(k_0, p_1(k_1 + 0.5), p_2 k_2) = \sum_{n=0}^{N_0-1} x(n) W_{N_0}^{\frac{n^2}{2}} W_{N_0}^{\theta_2(n,k)}. \quad (4.3.16)$$

Similarly, when p_2 is even, we have

$$G(k_0, k_1, k_2) = \sum_{n=0}^{N_0-1} x(n) \cos(\varphi) W_{N_0}^{\theta_2(n,k)} \quad (4.3.17)$$

$$H(k_0, k_1, k_2) = \sum_{n=0}^{N_0-1} x(n) \sin(\varphi) W_{N_0}^{\theta_2(n,k)} \quad (4.3.18)$$

where $\varphi = 2\pi(m_1 n^2 + 0.5p_1 n^3)/N_1$. From (4.3.17) and (4.3.18), we have

$$X(k_0, p_1 k_1 \pm m_1, p_2(k_2 + 0.5)) = G(k_0, k_1, k_2) \mp jH(k_0, k_1, k_2) \quad (4.3.19)$$

$$X(k_0, p_1 k_1, p_2(k_2 + 0.5)) = \sum_{n=0}^{N_0-1} x(n) W_{N_0}^{\frac{n^3}{2}} W_{N_0}^{\theta_2(n,k)}. \quad (4.3.20)$$

When both p_1 and p_2 are even, special arrangement has to be made for $m_1 = p_1/2$ and $m_2 = p_2/2$.

$$X(k_0, p_1(k_1 + 0.5), p_2(k_2 + 0.5)) = \sum_{n=0}^{N_0-1} x(n) W_{N_0}^{\frac{n^2+n^3}{2}} W_{N_0}^{\theta_2(n,k)}. \quad (4.3.21)$$

It is seen that the sinusoidal factors in (4.3.13), (4.3.14), (4.3.17) and (4.3.18) are real and each requires two real multiplications, while the twiddle factors in (4.3.16), (4.3.20) and (4.3.21) need complex multiplications. The total computational overhead needed by the above decomposition process is shown in Table 4.1.

Table 4.1: The number of arithmetic operations required by the FSD.

p_1	p_2	Number of real multiplications	Number of complex additions
odd	odd	$2(p_1p_2 - 1)(N_0 - 1)$	$(p_1p_2 - 1)N_0^3$
odd	even	$2p_1p_2(N_0 - 1)$	$(p_1p_2 - 2)N_0^3 + (N_0 - 1)$
even	odd	$2p_1p_2(N_0 - 1)$	$(p_1p_2 - 2)N_0^3 + (N_0 - 1)$
even	even	$2(p_1p_2 + 2)(N_0 - 1)$	$(p_1p_2 - 4)N_0^3 + 3(N_0 - 1)$

4.4 Fast Algorithms

This section presents the second step decomposition that is a general method used to derive various fast algorithms. It is assumed that the sizes of all dimensions are the same, i.e., N . We first consider the radix- a algorithm, where a is an integer greater than 1. Then the radix-2/(2a) algorithm is to be derived from the extension of the radix- a algorithm.

4.4.1 Radix- a Algorithm

The $(r + 1)$ th order PTFT is defined as

$$X(k_0, k_1, \dots, k_r) = \sum_{n=0}^{N-1} x(n) W_N^{\theta_r(n, k)}. \quad (4.4.1)$$

By using the periodic property of W_N , (4.4.1) is expressed as

$$\begin{aligned}
 & X(ak_0 + l_0, ak_1 + l_1, \dots, ak_r + l_r) \\
 &= \sum_{n=0}^{N-1} x(n) W_N^{(ak_0 + l_0)n + (ak_1 + l_1)n^2 + \dots + (ak_r + l_r)n^{r+1}} \\
 &= \sum_{n=0}^{N-1} x(n) W_N^{\theta_r(n, l)} W_{N/a}^{\theta_r(n, k)} \quad (4.4.2)
 \end{aligned}$$

where $0 \leq k_i \leq N/a - 1$ and $0 \leq l_i \leq a - 1$ for $i = 0, 1, \dots, r$. The above equation can be rewritten as

$$\begin{aligned}
 & X(ak_0 + l_0, \dots, ak_r + l_r) \\
 &= \sum_{n=0}^{N/a-1} \left\{ \sum_{c=0}^{a-1} x(n + c\frac{N}{a}) W_N^{\theta_r((n + \frac{cN}{a}), l)} \right\} W_{N/a}^{\theta_r(n, k)}. \quad (4.4.3)
 \end{aligned}$$

When $p \geq 2$ and by using

$$(n + c\frac{N}{a})^i = \sum_{j=0}^i C_i^j n^{i-j} (c\frac{N}{a})^j$$

and the periodic property of W_N , we have

$$W_N^{\theta_r((n + \frac{cN}{a}), l)} = W_N^{\theta_r(n, l)} e^{-j2\pi \frac{c \sum_{i=0}^r (i+1) l_i n^i}{a}}.$$

Thus, (4.4.2) can be simplified into

$$\begin{aligned}
 & X(ak_0 + l_0, ak_1 + l_1, \dots, ak_r + l_r) \\
 &= \sum_{n=0}^{N/a-1} \left[u_a(n, l_0, l_1, \dots, l_r) W_N^{\theta_r(n, l)} \right] W_{N/a}^{\theta_r(n, k)} \quad (4.4.4)
 \end{aligned}$$

where

$$\begin{aligned}
 & u_a(n, l_0, l_1, \dots, l_r) \\
 &= \sum_{c=0}^{a-1} x(n + \frac{cN}{a}) \exp \left\{ -j2\pi \frac{c \sum_{i=0}^r (i+1) l_i n^i}{a} \right\}. \quad (4.4.5)
 \end{aligned}$$

With the radix- a decomposition described above, (4.4.4) defines the $(r+1)$ th order PTFT whose dimensional size is N/a . With all the combinations of l_i

values, an $(r+1)$ th order PTFT of length- N is decomposed into a^{r+1} $(r+1)$ th order PTFTs of length- (N/a) .

To study (4.4.5), let us define

$$\begin{aligned} P &= \sum_{i=0}^r (i+1)l_i n^i = P_1 + 2P_2 \\ P_1 &= \sum_{t=0}^{\lfloor r/2 \rfloor} l_{2t} n^{2t} \\ P_2 &= \sum_{t=1}^{\lfloor r/2 \rfloor} l_{2t} t n^{2t} + \sum_{t=0}^{\lfloor (r-1)/2 \rfloor} l_{2t+1} (t+1) n^{2t+1} \end{aligned} \quad (4.4.6)$$

where $\lfloor x \rfloor$ returns the largest integer that is smaller than or equal to x . It is clear that the parity of P is the same as that of P_1 .

When a is odd, using $\exp[-j2\pi(a-c)P/a] = \exp[j2\pi cP/a]$, (4.4.5) is rewritten into

$$u_a(n, l_0, l_1, \dots, l_r) = x(n) + \sum_{c=1}^{(a-1)/2} [s_{1,c}(n) \cos(\beta) - j s_{2,c}(n) \sin(\beta)]. \quad (4.4.7)$$

When a is even and $\text{mod}(a, 4) \neq 0$, using $\exp\{-j2\pi(a/2-c)P/a\} = (-1)^{P_1} \exp\{j2\pi cP/a\}$, we have

$$\begin{aligned} &u_a(n, l_0, l_1, \dots, l_r) \\ &= \left[x(n) + (-1)^{P_1} x\left(n + \frac{N}{2}\right) \right] \\ &\quad + \sum_{c=1}^{\lfloor a/4 \rfloor} \{ [s_{1,c}(n) + s_{2,c}(n)] \cos(\beta) - j [s_{1,c}(n) - s_{2,c}(n)] \sin(\beta) \}, \end{aligned} \quad (4.4.8)$$

when $\text{mod}(a, 4) = 0$,

$$\begin{aligned} &u_a(n, l_0, l_1, \dots, l_r) \\ &= \left[x(n) + (-1)^{P_1} x\left(n + \frac{N}{2}\right) \right] + (-j)^P \left[x\left(n + \frac{N}{4}\right) + (-1)^{P_1} x\left(n + \frac{3N}{4}\right) \right] \\ &\quad + \sum_{c=1}^{a/4-1} \{ [s_{1,c}(n) + s_{2,c}(n)] \cos(\beta) - j [s_{1,c}(n) - s_{2,c}(n)] \sin(\beta) \}. \end{aligned} \quad (4.4.9)$$

In (4.4.7) to (4.4.9), the following definitions are used

$$\begin{aligned} s_{1,c}(n) &= x(n + \frac{cN}{a}) + (-1)^{P_1}(x + \frac{N}{2} + \frac{cN}{a}) \\ s_{2,c}(n) &= x(n + N - \frac{cN}{a}) + (-1)^{P_1}(x + \frac{N}{2} - \frac{cN}{a}) \\ \beta &= 2\pi cP/a. \end{aligned} \quad (4.4.10)$$

4.4.2 Split-radix- $m/(ma)$ Algorithm

A further extension of the above decomposition is to derive split-radix type algorithms. Here, we derive split-radix- $m/(ma)$ type algorithms, where m is a positive divisor of the integer a , which is usually denoted as $m|a$. Note that for $m = 1$, split-radix- $1/a$ is equivalent to radix- a algorithm, which has already been analyzed in the last section. Thus, we exclude the case of $m = 1$.

The outputs of an $(r + 1)$ th order PTFT, whose indices are divisible by m , for $i = 0, 1, \dots, r$, are computed by

$$X(mk_0, mk_1, \dots, mk_r) = \sum_{n=0}^{\frac{N}{m}-1} u_m(n, 0, \dots, 0) W_{N/m}^{\theta_r(n, \bar{k})} \quad (4.4.11)$$

where $0 \leq k_i \leq N/m - 1$, and

$$u_m(n, 0, \dots, 0) = \sum_{c=0}^{m-1} x(n + c\frac{N}{m}). \quad (4.4.12)$$

It is easily seen that the sequence $u_m(n, 0, \dots, 0)$ is formulated from the input sequence $x(n)$ with operations of real additions only.

The other PTFT outputs can be computed according to

$$\begin{aligned} &X(mak_0 + l_0, mak_1 + l_1, \dots, mak_r + l_r) \\ &= \sum_{n=0}^{\frac{N}{ma}-1} \sum_{c=0}^{ma-1} x(n + c\frac{N}{ma}) \prod_{i=0}^r W_N^{(mak_i + l_i)(n + c\frac{N}{ma})^{i+1}} \end{aligned} \quad (4.4.13)$$

where $0 \leq k_i \leq N/(ma) - 1$, $0 \leq l_i \leq ma - 1$, and at least one l_i is not a multiple of m . Note that to use this decomposition process, $N/(ma) = a^{p-2}b(a/m)$ must be an integer. Since m is a factor of a , the condition can be guaranteed by $p \geq 2$. Now, let us study the twiddle factor in (4.4.13), when $i = 0$,

$$W_N^{(mak_0+l_0)(n+c\frac{N}{ma})} = W_N^{l_0n} W_{N/(ma)}^{k_0n} e^{-j2\pi c\frac{l_0}{ma}}. \quad (4.4.14)$$

For other values of i , using $(n+b)^j = \sum_{i=0}^j C_j^i n^{j-i} b^i$, we have

$$\begin{aligned} W_N^{(mak_i+l_i)(n+c\frac{N}{ma})^{i+1}} \\ = W_N^{l_i n^{i+1}} W_{N/(ma)}^{k_i n^{i+1}} e^{-j2\pi c\frac{l_i(i+1)n^i}{ma}} e^{-j2\pi k_i \sum_{j=2}^{i+1} C_{i+1}^j n^{i+1-j} c^j (\frac{N}{ma})^{j-1}} \\ e^{-j2\pi l_i \sum_{j=2}^{i+1} C_{i+1}^j n^{i+1-j} c^j \frac{N^{j-1}}{(ma)^j}}. \end{aligned} \quad (4.4.15)$$

Since $(\frac{N}{ma})^{j-1}$, $j \geq 2$, is an integer, the fourth term in (4.4.15) equals 1. Additionally, $\frac{N^{j-1}}{(ma)^j} = a^{p(j-1)-2j} b^{j-1} (a/m)^j$, $j \geq 2$, is an integer when $p \geq \frac{2j}{j-1}$. It is easily seen that the maximum value of $\frac{2j}{j-1}$ for an integer $j \geq 2$ is 4. If $p \geq 4$, the fifth term in (4.4.15) becomes 1 too. Therefore, (4.4.13) is simplified into

$$\begin{aligned} X(mak_0 + l_0, mak_1 + l_1, \dots, mak_r + l_r) \\ = \sum_{n=0}^{\frac{N}{ma}-1} u_{ma}(n, l_0, \dots, l_r) W_N^{\theta_r(n, \bar{l})} W_{N/(ma)}^{\theta_r(n, \bar{k})}. \end{aligned} \quad (4.4.16)$$

To obtain the outputs of length- N $(r+1)$ th order PTFT, we need to compute one length- (N/m) $(r+1)$ th order PTFT and $a^{r+1}(m^{r+1} - 1)$ length- $(N/(ma))$ $(r+1)$ th order PTFTs. The above decomposition process can be further continued until the decomposed PTFTs become sufficiently small to be easily computed.

Let us consider some special cases. For any integer a , it must have a divisor of $m = a$. Substituting $m = a$ into (4.4.11) and (4.4.16), split-radix- a/a^2

algorithm is obtained, which decomposes the $(r + 1)$ th order PTFT into one PTFT of length- (N/a) and $[a^{2(r+1)} - a^{r+1}]$ length- $(N/(a^2))$ PTFTs.

When a is an even number, it must have a divisor of $m = 2$. Thus, the even indexed outputs and the other outputs of the PTFT can be computed according to (4.4.11) and (4.4.16), respectively. Therefore, the $(r + 1)$ th order PTFT is decomposed into one PTFT of length- $(N/2)$ and $[(2a)^{r+1} - a^{r+1}]$ length- $(N/(2a))$ PTFTs. When $a = 2$ and 4, for example, split-radix-2/4 and -2/8 algorithms are obtained.

4.5 Computational Complexity

We now look at the required computational complexity. It is known that arithmetic operations are needed for computing $u_a(n, l_0, l_1, \dots, l_r)$ and the twiddle factors $W_N^{\theta_r(n, \vec{l})}$. Let $C_{a,add} \frac{N}{a}$ and $C_{a,mul} \frac{N}{a}$ represent, respectively, the numbers of complex additions and complex multiplications needed by the computation of $u_a(n, l_0, l_1, \dots, l_r)$, for $n = 0, 1, \dots, N/a - 1$. When calculating the number of complex multiplications required by the twiddle factors $W_N^{\theta_r(n, \vec{l})}$, the operations for trivial twiddle factors when $n = 0$ and/or $l_0 = l_1 = \dots = l_r = 0$ are not counted in our analysis.

Based on the radix- a decomposition procedures presented in the last section, the numbers of complex additions and multiplications required by the radix- a algorithm are:

$$A_{r+1}(N) = a^{r+1} A_{r+1}\left(\frac{N}{a}\right) + C_{a,add} \frac{N}{a} \quad (4.5.1)$$

$$M_{r+1}(N) = a^{r+1} M_{r+1}\left(\frac{N}{a}\right) + C_{a,mul} \frac{N}{a} + (a^{r+1} - 1) \left(\frac{N}{a} - 1\right) \quad (4.5.2)$$

where $N = a^p b$ and the last term in (4.5.2) is the number of complex multipli-

cations needed by the twiddle factors. At the m th step of the decomposition process, the length- N PTFT is decomposed into length- (N/a^m) PTFTs. After necessary manipulation, $A_{r+1}(N)$ and $M_{r+1}(N)$ can be expressed as

$$A_{r+1}(a^p b) = A_{r+1}(a^{p-m} b) a^{m(r+1)} + C_{a,add} Q(a^{mr} - 1) \quad (4.5.3)$$

$$\begin{aligned} M_{r+1}(a^p b) &= (M_{r+1}(a^{p-m} b) - 1) a^{m(r+1)} \\ &\quad + (C_{a,mul} + a^{r+1} - 1) Q(a^{mr} - 1) + 1. \end{aligned} \quad (4.5.4)$$

where $Q = \frac{a^{p-1}b}{a^r-1}$. By setting $b = 1$, let us consider the case that uses initial values of $A_{r+1}(a^{q_0})$ and $M_{r+1}(a^{q_0})$ to obtain

$$A_{r+1}(a^p) = A_{r+1}(a^{q_0}) a^{(r+1)(p-q_0)} + C_{a,add} (a^{r(p-q_0)} - 1) Q \quad (4.5.5)$$

$$\begin{aligned} M_{r+1}(a^p) &= (M_{r+1}(a^{q_0}) - 1) a^{(r+1)(p-q_0)} \\ &\quad + (C_{a,mul} + a^{r+1} - 1) Q (a^{r(p-q_0)} - 1) + 1. \end{aligned} \quad (4.5.6)$$

In order to minimize the total computational complexity, the initial values used in the above equations are the numbers of multiplications and additions required by the optimized program subroutines to implement the length- (a^{q_0}) PTFTs.

We now consider some special cases. When a is even, the split-radix algorithms can be obtained as shown in (4.4.11) and (4.4.16). The required numbers of complex additions and complex multiplications are

$$A_{r+1}(N) = A_{r+1}\left(\frac{N}{2}\right) + (2^{r+1} - 1) a^{r+1} A_{r+1}\left(\frac{N}{2a}\right) + C_{2a,add} \frac{N}{2a} \quad (4.5.7)$$

$$\begin{aligned} M_{r+1}(N) &= M_{r+1}\left(\frac{N}{2}\right) + (2^{r+1} - 1) a^{r+1} [M_{r+1}\left(\frac{N}{2a}\right) + \left(\frac{N}{2a} - 1\right)] \\ &\quad + C_{2a,mul} \frac{N}{2a} \end{aligned} \quad (4.5.8)$$

In particular, when $N = 2^p$, with the initial values of $A_{r+1}(8)$, $A_{r+1}(16)$, $M_{r+1}(8)$ and $M_{r+1}(16)$, the closed form solutions to the above recursive equations are

$$A_{r+1}(2^p) = b_1(2^{r+1})^p + b_2(1 - 2^{r+1})^p + b_3 2^p \quad (4.5.9)$$

where

$$\begin{aligned} b_1 &= \frac{(2^r - 1)A_{r+1}(16) + (2^r - 1)(2^{r+1} - 1)A_{r+1}(8) + 32}{8^{r+1}(2^r - 1)(2^{r+2} - 1)} \\ b_2 &= -\frac{2^{r+1}(2^{r+1} + 1)A_{r+1}(8) - (2^{r+1} + 1)A_{r+1}(16) + 64}{(2^{r+1} - 1)^3(2^{r+2} - 1)(2^{r+1} + 1)} \\ b_3 &= -\frac{4}{(2^r - 1)(2^{r+1} + 1)}, \end{aligned}$$

and

$$M_{r+1}(2^p) = c_1(2^{r+1})^p + c_2(1 - 2^{r+1})^p + c_32^p + 1 \quad (4.5.10)$$

where

$$\begin{aligned} c_1 &= \frac{M_{r+1}(16) - 2^{r+1}}{8^{r+1}(2^{r+2} - 1)} + \frac{[(2^r - 1)M_{r+1}(8) + 2^{r+3}](2^{r+1} - 1)}{8^{r+1}(2^r - 1)(2^{r+2} - 1)} \\ c_2 &= -\frac{2^{r+1}M_{r+1}(8) - M_{r+1}(16)}{(2^{r+1} - 1)^3(2^{r+2} - 1)} - \frac{2^{r+4} - 2^{r+1} - 1}{(2^{r+1} - 1)^2(2^{r+2} - 1)(2^{r+1} + 1)} \\ c_3 &= -\frac{2^r(2^{r+1} - 1)}{(2^r - 1)(2^{r+1} + 1)}. \end{aligned}$$

4.5.1 Computation Details

In this section, we consider the computational complexity needed by $u_a(n, l_0, l_1, \dots, l_r)$, where $0 \leq n \leq \frac{N}{a} - 1$, $0 \leq l_i \leq a - 1$ for $i = 1, 2, \dots, r$. One desirable property obtained from the analysis (which will be shown in the sequel) is that the computational complexity for $u_a(n, l_0, l_1, \dots, l_r)$ is independent of the order of the PTFT computation.

4.5.1.1 a is odd

By representing the greatest common divisor of a and c as $\gcd(a, c)$, we define

$$a' = \frac{a}{\gcd(a, c)} \quad \text{and} \quad c' = \frac{c}{\gcd(a, c)}.$$

The use of a' and c' helps to discount the multiplications for the trivial twiddle factors from the required computational complexity. It can be easily proven that a' should be odd when a is odd. In this case, $u_a(n, l_0, l_1, \dots, l_r)$ can be rewritten as

$$u_a(n, l_0, l_1, \dots, l_r) = x(n) + \sum_{c=1}^{(a-1)/2} v_{1,c,a}(n, l_0, l_1, \dots, l_r) - j \sum_{c=1}^{(a-1)/2} v_{2,c,a}(n, l_0, l_1, \dots, l_r) \quad (4.5.11)$$

where

$$\begin{aligned} v_{1,c,a}(n, l_0, l_1, \dots, l_r) &= \left[x(n + c\frac{N}{a}) + x(n + N - c\frac{N}{a}) \right] \cos\left(2\pi c' \frac{P}{a'}\right) \\ v_{2,c,a}(n, l_0, l_1, \dots, l_r) &= \left[x(n + c\frac{N}{a}) - x(n + N - c\frac{N}{a}) \right] \sin\left(2\pi c' \frac{P}{a'}\right). \end{aligned} \quad (4.5.12)$$

Let us analyze the possible values of the sinusoidal terms $(\cos(2\pi c' P/a'), \sin(2\pi c' P/a'))$, which are in the form

$$(1, 0), \underbrace{(y_1, \pm z_1), (y_2, \pm z_2), \dots}_{(a'-1)/2 \text{ terms}}$$

Based on the above observations, let $t_1(n) = x(n + cN/a) + x(n + N - cN/a)$, the possible values of $v_{1,c,a}(n, l_0, l_1, \dots, l_r)$ are in the form of $t_1(n)$ and $y_i t_1(n)$ for $i = 1, 2, \dots, (a' - 1)/2$. Thus, the computation required is

- N/a complex additions are needed by $t_1(n)$
- $(a' - 1)N/a$ real multiplications are needed by $y_i t_1(n)$ for $i = 1, 2, \dots, (a' - 1)/2$

Because when $a' = 3$, the cosine terms become 1 or -0.5 , shift operations rather than multiplications are needed when the cosine terms equal 0.5. In this case,

$v_{1,c,a}(n, l_0, l_1, \dots, l_r)$ needs only N/a complex additions. For the computation of $t_2(n) = x(n+cN/a) - x(n+N-cN/a)$, the possible values of $v_{2,c,a}(n, l_0, l_1, \dots, l_r)$ are 0 and $\pm z_i t_2(n)$ for $i = 1, 2, \dots, (a' - 1)/2$. Thus, the computation required by $v_{2,c,a}(n, l_0, l_1, \dots, l_r)$ is also N/a complex additions and $(a' - 1)N/a$ real multiplications. Therefore, the total required numbers of complex additions and the real multiplications to form $v_{1,c,a}(n, l_0, l_1, \dots, l_r)$ and $v_{2,c,a}(n, l_0, l_1, \dots, l_r)$ for $c = 1, 2, \dots, (a - 1)/2$ are, respectively, $\sum_{c=1}^{(a-1)/2} 2 \frac{N}{a} = \frac{a-1}{a} N$ and

$$\sum_{c=1, a' \neq 3}^{(a-1)/2} (a' - 1) \frac{N}{a} + \sum_{c=1}^{(a-1)/2} (a' - 1) \frac{N}{a} = 2 \sum_{c=1}^{(a-1)/2} (a' - 1) \frac{N}{a} - \sum_{c=1, a'=3}^{(a-1)/2} \frac{N}{a} \quad (4.5.13)$$

The number of complex additions for summing up the three terms in (4.5.11) is $\sum_{c=1}^{(a-1)/2} [1 + (a' - 1)/2 + (a' - 1)] N/a$.

In total, the numbers of complex additions and real multiplications for the computation of $u_a(n, l_0, l_1, \dots, l_r)$ are

$$A_u(N) = \frac{a-1}{a} N + \sum_{c=1}^{\frac{a-1}{2}} [1 + \frac{3}{2}(a' - 1)] \frac{N}{a} \quad (4.5.14)$$

$$M_u(N) = 2 \sum_{c=1}^{(a-1)/2} (a' - 1) \frac{N}{a} - 2 \sum_{c=1, a'=3}^{(a-1)/2} \frac{N}{a}. \quad (4.5.15)$$

In particular, when $a = 3$, the numbers of complex additions and real multiplications needed by $u_3(n, l_0, l_1, \dots, l_r)$ are

$$A_u(N) = \frac{2}{3} N + \frac{4}{3} N = 2N \quad (4.5.16)$$

$$\begin{aligned} M_u(N) &= 2 \sum_{c=1}^{(3-1)/2} \left(\frac{3}{\gcd(3, c)} - 1 \right) \frac{N}{3} - 2 \sum_{c=1, \frac{3}{\gcd(3, c)}=3}^{(3-1)/2} \frac{N}{3} \\ &= \frac{2}{3} N \end{aligned} \quad (4.5.17)$$

thus, $C_{3,add} = 11/2$ and $C_{3,mul} = 1/2$.

4.5.1.2 a is even

When a is even, it is difficult to obtain a general formula of the computation complexity needed by $u_a(n, l_0, l_2, \dots, l_r)$ because there exist many trivial twiddle factors that are useful to reduce the number of arithmetic operations. Let us consider a few widely used cases as follows.

Case 1. When $a = 2$ for $0 \leq n \leq N/2 - 1$, we have

$$u_2(n, l_0, l_1, \dots, l_r) = x(n) + (-1)^{P_1} x(n + N/2) \quad (4.5.18)$$

where P_1 is defined in (4.4.6). Because only N complex additions/subtractions are needed, $C_{2,add} = 2$ and $C_{2,mul} = 0$.

Case 2. When $a = 4$ for $0 \leq n \leq N/4 - 1$,

$$\begin{aligned} u_4(n, l_0, l_1, \dots, l_r) = & [x(n) + (-1)^{P_1} x(n + N/2)] \\ & + (-j)^P [x(n + N/4) + (-1)^{P_1} x(n + 3N/4)]. \end{aligned} \quad (4.5.19)$$

Because P_1 and P have the same parity, as seen in (4.4.6), when P_1 is even we have $(-1)^{P_1} = 1$ and $(-j)^P = \pm 1$. The computational complexity for $n = 0, 1, \dots, N/4 - 1$ is N complex additions only.

When P_1 is odd, P is odd, $(-1)^{P_1} = -1$ and $(-j)^P = \pm j$. The computational complexity is the same as that when P_1 is even. Thus, $C_{4,add} = 8$, and $C_{4,mul} = 0$.

Case 3. When $a = 6$ for $0 \leq n \leq N/6 - 1$,

$$\begin{aligned} u_6(n, l_0, l_1, \dots, l_r) = & [x(n) + (-1)^{P_1} x(n + N/2)] \\ & + [x_1(n) + x_2(n)] \cos(P\pi/3) - j[x_1(n) - x_2(n)] \sin(P\pi/3) \end{aligned} \quad (4.5.20)$$

where

$$\begin{aligned} x_1(n) &= x(n + N/6) + (-1)^{P_1} x(n + 2N/3) \\ x_2(n) &= (-1)^{P_1} x(n + N/3) + x(n + 5N/6) \end{aligned}$$

If P_1 and P are both even, $(-1)^{P_1} = 1$, $\cos(P\pi/3) = 1$ or -0.5 , and $\sin(P\pi/3) = 0$ or ± 0.866 . The computational complexity for $n = 0, 1, \dots, N/6 - 1$ is summarized as follows:

- $N/2$ complex additions are needed by $t_0(n) = x(n) + x(n + N/2)$, $t_1(n) = x(n + N/6) + x(n + N/3)$ and $t_2(n) = x(n + N/3) + x(n + 5N/6)$;
- $2N/3$ complex additions are needed by $t_3(n) = t_1(n) + t_2(n)$, $t_4(n) = t_1(n) - t_2(n)$, $t_5(n) = t_0(n) + t_3(n)$ and $t_6(n) = t_0(n) - 0.5t_3(n)$;
- $N/3$ real multiplications are needed to obtain $\pm 0.866t_4(n)$;
- $N/3$ complex additions are needed to obtain $\pm 0.866t_4(n) + t_6(n)$

In total, $3N/2$ complex additions and $N/3$ real multiplications are needed.

When P_1 and P are odd, $(-1)^{P_1} = -1$, $\cos(P\pi/3) = -1$ or 0.5 , and the values of $\sin(P\pi/3) = 0$ or ± 0.866 . Therefore, the required computation is the same as that needed for the case that P_1 is even. Thus, the required computational complexity for $u_6(n, l_0, l_1, \dots, l_r)$ is $3N$ complex additions and $2N/3$ real multiplications, which leads to $C_{6,add} = 17$, and $C_{6,mul} = 1$.

Case 4. When $a = 8$ for $0 \leq n \leq \frac{N}{8} - 1$,

$$\begin{aligned}
 & u_8(n, l_0, l_1, \dots, l_r) \tag{4.5.21} \\
 &= [x(n) + (-1)^{P_1}x(n + N/2)] + (-j)^P [x(n + N/4) + (-1)^{P_1}x(n + 3N/4)] \\
 &+ [x_1(n) + x_2(n)] \cos(P\pi/4) - j[x_1(n) - x_2(n)] \sin(P\pi/4)
 \end{aligned}$$

where

$$\begin{aligned}
 x_1(n) &= x(n + N/8) + (-1)^{P_1}x(n + 5N/8) \\
 x_2(n) &= (-1)^{P_1}x(n + 3N/8) + x(n + 7N/8)
 \end{aligned}$$

With P_1 and P being even, $(-1)^{P_1} = 1$ and $(-j)^P = \pm 1$. Therefore,

$$(\cos(P\pi/4), \sin(P\pi/4)) = (\pm 1, 0) \text{ and } (0, \pm 1).$$

The computational complexity for $n = 0, 1, \dots, N/8 - 1$ is summarized as follows:

- $N/2$ complex additions are needed by $t_0(n) = x(n) + x(n + \frac{N}{2})$, $t_1(n) = x(n + N/4) + (x + 3N/4)$, $t_2(n) = x(n + N/8) + (x + 5N/8)$, and $t_3(n) = x(n + 3N/8) + (x + 7N/8)$;
- $N/2$ complex additions are needed by $t_4(n) = t_2(n) + t_3(n)$, $t_5(n) = t_2(n) - t_3(n)$, $t_6(n) = t_0(n) + t_1(n)$ and $t_7(n) = t_0(n) - t_1(n)$;
- $N/2$ complex additions are needed by $\pm t_4(n) + t_6(n)$ and $\pm jt_5(n) + t_7(n)$

In total, $3N/2$ complex additions are needed.

With P_1 and P being odd, $(-1)^{P_1} = -1$, $(-j)^P = \pm j$ and $(\cos(P\pi/4), \sin(P\pi/4)) = (\pm 1/\sqrt{2}, \pm 1/\sqrt{2})$. For this case, similar analysis of computational complexity leads to $7N/4$ complex additions and $N/2$ real multiplications. In summary, the computation of $u_8(n, l_0, l_1, \dots, l_r)$ requires $13N/4$ complex additions and $N/2$ real multiplications, or equivalently, $C_{8,add} = 25$, and $C_{8,mul} = 1$.

Table 4.2 lists the numbers of complex multiplications and complex additions needed for $u_a(n, l_0, l_1, \dots, l_r)$ where $0 \leq n \leq \frac{N}{a} - 1$, $0 \leq l_i \leq a - 1$ for $i = 1, 2, \dots, r$, and $a = 2, 4, 6$ and 8 . One desirable property observed from the above analysis is that the computational complexity for $u_a(n, l_0, l_1, \dots, l_r)$ is independent of the order of the PTFT computation.

Table 4.2: The computational complexity required by $u_a(n, l_0, l_1, \dots, l_r)$

$a =$	Complex additions	Complex multiplications
2	N	0
3	$11N/6$	$N/6$
4	$2N$	0
6	$17N/6$	$N/6$
8	$25N/8$	$N/8$

4.5.2 Computation Complexity for the 2nd Order PTFT

Based on the presented analysis of the required computational complexity, let us consider the example for the 2nd Order PTFT. Substituting $r = 1$ into (4.5.3) and (4.5.4), at the m th step of the decomposition process, we have

$$A_2(a^p b) = A_2(a^{p-m} b) a^{2m} + C_{a,add}(a^m - 1)Q \quad (4.5.22)$$

$$\begin{aligned} M_2(a^p b) &= (M_2(a^{p-m} b) - 1) a^{2m} \\ &\quad + (C_{a,mul} + a^2 - 1)Q(a^m - 1) + 1. \end{aligned} \quad (4.5.23)$$

With $b = 1$ and initial values of $A_2(a^{q_0})$ and $M_2(a^{q_0})$, we have

$$A_2(a^p) = A_2(a^{q_0}) a^{2(p-q_0)} + C_{a,add}(a^{p-q_0} - 1)Q \quad (4.5.24)$$

$$\begin{aligned} M_2(a^p) &= (M_2(a^{q_0}) - 1) a^{2(p-q_0)} \\ &\quad + (C_{a,mul} + a^2 - 1)(a^{p-q_0} - 1)Q + 1. \end{aligned} \quad (4.5.25)$$

It is also possible to use split-radix algorithm to further reduce the required number of arithmetic operations. With (4.4.11) and (4.4.16), split-radix-2/4 and -2/8 can be obtained by setting $a = 2$ and 4, respectively. The numbers of complex multiplications and additions required by the split-radix-2/4 algorithm

are

$$A_2(N) = A_2\left(\frac{N}{2}\right) + 3a^2 A_2\left(\frac{N}{2a}\right) + C_{2a,add} \frac{N}{2a} \quad (4.5.26)$$

$$M_2(N) = M_2\left(\frac{N}{2}\right) + 3a^2 \left[M_2\left(\frac{N}{2a}\right) + \frac{N}{2a} - 1 \right] + C_{2a,mul} \frac{N}{2a} . \quad (4.5.27)$$

When $N = 2^p$, for example, with the initial values of $A_2(8)$, $A_2(16)$, $M_2(8)$ and $M_2(16)$, the solutions to the above equations are

$$A_2(2^p) = b_1 4^p + b_2 (-3)^p + b_3 2^p , \quad (4.5.28)$$

where

$$\begin{aligned} b_1 &= \frac{A_2(16) + 3A_2(8) + 32}{64 \times 7} \\ b_2 &= -\frac{20A_2(8) - 5A_2(16) + 64}{27 \times 7 \times 5} \\ b_3 &= -\frac{4}{5} , \end{aligned}$$

and

$$M_2(2^p) = c_1 4^p + c_2 (-3)^p + c_3 2^p + 1 , \quad (4.5.29)$$

where

$$\begin{aligned} c_1 &= \frac{M_2(16) + 3M_2(8) + 44}{64 \times 7} \\ c_2 &= -\frac{20M_2(8) - 5M_2(16) + 81}{27 \times 7 \times 5} \\ c_3 &= -\frac{6}{5} . \end{aligned}$$

The PTFT can be also decomposed into one length- $(N/2)$ PTFT and 48 length- $(N/8)$ PTFTs, which is the split-radix-2/8 algorithm. Due to the space limit, the analysis of the required computational complexity is not presented. Table 4.3 shows comparison on the numbers of complex multiplications needed by

various proposed fast algorithms for the 2nd order PTFT. For the radix- a algorithms, initial values, shown in *Italic font*, are used. It is seen that in general, the number of complex multiplications is reduced as the value of a is increased, and the split-radix-2/8 uses the lowest number of complex multiplications. The additive complexity needed by the presented algorithms can be analyzed similarly. Table 4.4 shows the number of complex additions for the 2nd order PTFT. In general, it is observed that similar comparisons can be made among the additive complexities needed by the algorithms previously discussed. For example, the split-radix algorithms generally require smaller additive complexities than other proposed algorithms. Compared to other reported fast algorithms, our split-radix-2/4 algorithm uses the same computational complexity as that in [101] and achieves significant savings on computational complexity compared to that in [3].

Table 4.3: The number of complex multiplications for the 2nd order PTFT

N	radix-2	split-radix-2/4	radix-4	split-radix-2/8	radix-8
8	<i>2</i>	<i>2</i>	<i>2</i>	<i>2</i>	<i>2</i>
16	29	<i>24</i>	<i>24</i>	<i>24</i>	<i>24</i>
32	161	132	137	<i>132</i>	<i>132</i>
64	737	600	609	572	577
128	3137	2556	2657	2460	2497
256	12929	10512	10689	10316	10433
512	52481	42708	44417	40860	40961

4.5.3 Computation Complexity for the 3rd Order PTFT

Let us consider the example for the 3rd Order PTFT. For $r = 2$, at the m th step decomposition process, the numbers of complex additions and multiplications

Table 4.4: The number of complex additions for the 2nd order PTFT

N	radix-2	split-radix-2/4	radix-4	split-radix-2/8	radix-8
8	58	58	58	58	58
16	248	234	234	234	234
32	1024	946	992	946	946
64	4160	3786	3872	3780	3762
128	16768	15202	16128	15112	15076
256	67328	60762	62464	60720	60744
512	269824	243442	259072	242560	241168

are

$$A_3(a^p b) = A_3(a^{p-m} b) a^{3m} + C_{a,add}(a^{2m} - 1)Q \quad (4.5.30)$$

$$\begin{aligned} M_3(a^p b) &= (M_3(a^{p-m} b) - 1) a^{3m} \\ &\quad + (C_{a,mul} + a^3 - 1)(a^{2m} - 1)Q + 1 . \end{aligned} \quad (4.5.31)$$

With $b = 1$, and the initial values of $A_3(a^{q_0})$ and $M_3(a^{q_0})$, we have

$$A_3(a^p) = A_3(a^{q_0}) a^{3(p-q_0)} + C_{a,add}(a^{2(p-q_0)} - 1)Q \quad (4.5.32)$$

$$\begin{aligned} M_3(a^p) &= (M_3(a^{q_0}) - 1) a^{3(p-q_0)} \\ &\quad + (C_{a,mul} + a^3 - 1)(a^{2(p-q_0)} - 1)Q + 1 . \end{aligned} \quad (4.5.33)$$

For the split-radix-2/4 algorithm, when $N = 2^p$, the solutions are

$$A_3(2^p) = b_1 8^p + b_2 (-7)^p + b_3 2^q , \quad (4.5.34)$$

where

$$\begin{aligned} b_1 &= \frac{3A_3(16) + 21A_3(8) + 32}{8^3 \times 3 \times 15} \\ b_2 &= -\frac{72A_3(8) - 9A_3(16) + 64}{7^3 \times 15 \times 9} \\ b_3 &= -\frac{4}{27} , \end{aligned}$$

and

$$M_3(2^p) = c_1 8^p + c_2 (-7)^p + c_3 2^p + 1 , \quad (4.5.35)$$

where

$$\begin{aligned} c_1 &= \frac{3M_3(16) + 21M_3(8) + 200}{8^3 \times 3 \times 15} \\ c_2 &= -\frac{72M_3(8) - 9M_3(16) + 385}{7^3 \times 15 \times 9} \\ c_3 &= -\frac{28}{27} . \end{aligned}$$

Table 4.5 lists the number of complex multiplications required by various proposed fast algorithms for the 3rd order PTFT. It is seen that the split-radix algorithms generally need less computational complexity than the radix- a algorithms, and the algorithms with higher radix also use less computational complexity. However, the saving on the computational complexity by using the higher radix for the 3rd order PTFT is not substantial as that for the 2nd order PTFT. One possible reason is that as the order of the PTFT increases, the number of complex multiplications needed for the twiddle factors become more dominating so that the saving achieved by the decomposition process becomes less effective.

Table 4.5: The number of complex multiplications for the 3rd order PTFT

N	radix-2	split-radix-2/4	radix-4	split-radix-2/8	radix-8
8	2	2	2	2	2
16	65	65	65	65	65
32	625	569	569	569	569
64	5217	5049	5105	4609	4609
128	42177	38649	38369	40465	40961
256	338305	324921	330689	309297	307201
512	2708225	2496377	2463617	2402417	2392065

4.6 Summary

A general class of fast algorithms for the computation of the PTFTs is presented. Analysis and comparison on the computational complexity are also reported in terms of the numbers of complex additions and complex multiplications. The proposed algorithms provide the flexibility to support various input sequence lengths by setting the parameters of a , b and p , and generally have regular computational structures for an easy implementation. With the use of symmetric property and the presented fast algorithms, computational complexity for the applications of the PTFTs is significantly reduced to support applications that deal with the polynomial phase signals.

Chapter 5

Fast Algorithms for PTFTs of Real-Valued Sequences

5.1 Introduction

As stated in Chapter 4, most applications require the estimation of the phase coefficients from the received polynomial-phase signal (PPS), because these coefficients carry desirable information. For complex-valued PPSs, the PTFT serves as the MLE of the phase coefficients, and can be efficiently computed using the generalized class of fast algorithms presented in Chapter 4, which supports arbitrary order of PTFT with the dimension size being $a^p b$, where a, p and b are positive integers.

In practice, many applications inherently deal with real-valued signals. The natural signals such as speech, marine mammal sounds, heart rate etc. are real-valued. In image and audio watermarking schemes [36, 37], the real-valued chirp signals are embedded as watermark signals, where each chirp rate corresponds

to a unique message. The general form of $(r + 1)$ th order real-valued PPSs is expressed as

$$x(n) = A(n) \cos[\phi(n)] + w(n) \quad (5.1.1)$$

where $\phi(n) = 2\pi \sum_{m=0}^r a_m n^m$, $n = 0, 1, \dots, N - 1$, and $w(n)$ is white Gaussian noise. The amplitude $A(n)$ is a constant or a real-valued stationary Gaussian process, and a_m , $0 \leq m \leq r$, are the coefficients associated with the polynomial phase.

There has not been much research reported in the literature for processing real-valued PPSs. But since one real-valued PPS is equivalent to 2 complex-valued PPSs,

$$\begin{aligned} s(n) &= A(n) \cos[\phi(n)] \\ &= \frac{1}{2} A(n) e^{j2\pi \sum_{m=0}^r a_m n^m} + \frac{1}{2} A(n) e^{-j2\pi \sum_{m=0}^r a_m n^m} \\ &= \frac{1}{2} A(n) e^{j2\pi \sum_{m=0}^r a_m n^m} + \frac{1}{2} A(n) e^{j2\pi \sum_{m=0}^r (1-a_m) n^m} \end{aligned} \quad (5.1.2)$$

the $(r + 1)$ th order PTFT of $s(n)$ will show peaks when k_i/N_i matches a_i and $1 - a_i$ for $i = 0, 1, \dots, r$. As a simple illustration, Figure 5.1 shows the 2nd order PTFT of a real-valued signal that has 2 chirp components. It is seen that for each chirp component, the values of a_0 and a_1 can be easily obtained from the peak locations of the PTFT.

To calculate the PTFT of real-valued sequences, it is possible to convert the real-valued signal to its analytic version (see Section 2.5.1 for the conversion procedure) on which the fast algorithms reported in Chapter 4 can be applied. The other approach is to develop fast algorithms to direct compute the PTFT of real-valued sequence. It is well known that for discrete Fourier transform, the fast algorithms for real-valued sequence generally require about one half of

Figure 5.1: The normalized $|X(k_0, k_1)|^2$ of a real-valued signal with two chirp components (SNR = 3 dB).

the computational complexity needed by the fast algorithms of complex-valued sequence (for example, see [103, 104]). By using some interesting properties, the fast algorithms for the PTFT of real-valued sequence can also achieve further savings on computational complexity compared to the fast algorithm for the PTFTs of complex-valued sequences. This chapter proposes such fast algorithms that also have good computational structures.

This chapter is organized as follows. Section 5.2 describes the symmetric and Hermitian properties of the PTFT that are used to reduce the number of transform points to be calculated. Procedures are also presented to convert a PTFT, whose dimensional sizes are different, into a number of smaller PTFTs whose all dimensional sizes are the same. Section 5.3 describes the derivations of the fast algorithm for an arbitrary order of the PTFT of length- $a^p b$, where a, b and p are positive integers, and presents an example for the 2nd order PTFT to show the details of the proposed fast algorithm. In Section 5.4, the required computation complexity is analyzed and compared for different values

of a . Section 5.5 presents the split-radix algorithms and their computation complexity. Finally, Section 5.6 presents the conclusions.

5.2 Properties and Decomposition

To support various sequence lengths, as in Chapter 4, we assume that $N_0 = a^p b$, where a , b and p are positive integers and $a \neq b$. In practice, it is often required that $N_j \geq N_i \geq N_0$ for $j > i > 0$ to achieve a satisfactory accuracy of the parameter estimation [3]. Thus, we assume that $p_i = N_i/N_0$ is a positive integer.

5.2.1 The Hermitian and Symmetric Properties

In general, for an $(r + 1)$ th order PTFT, the total number of PTFT outputs is $\prod_{i=0}^r N_i$. For real-valued sequences, the PTFT has properties that can be explored to reduce the number of transform outputs to be calculated. Let us first consider the Hermitian property of the PTFTs for real-valued sequence as follows:

$$X(N_0 - k_0, \dots, N_r - k_r) = X^*(k_0, \dots, k_r). \quad (5.2.1)$$

This property, as illustrated in Figure 5.1, indicates that for an $(r + 1)$ th order PTFT, it is not necessary to calculate these transform points with $k_i = \lceil N_i/2 \rceil, \dots, N_i - 1$, where i is any one and only one value in the range between 0 and r , because they can be obtained from those with indices $k_i = 1, \dots, \lfloor N_i/2 \rfloor$. Thus, the number of transform outputs to be calculated is decreased to $(\lfloor N_i/2 \rfloor + 1) \prod_{j=0, j \neq i}^r N_j$, where $\lfloor x \rfloor$ returns an integer smaller than or equal to x and $\lceil x \rceil$ returns an integer larger than or equal to x .

When N_i is even, the PTFT has a symmetric property, which is described as [100]

$$\begin{aligned} & X(k_0, k_1 + \alpha_1 \frac{N_1}{2}, \dots, k_r + \alpha_r \frac{N_r}{2}) \\ &= \begin{cases} X(k_0, k_1, \dots, k_r) & \text{if } \sum_{i=1}^r \alpha_i \text{ is even} \\ X(k_0 + \frac{N_0}{2}, k_1, \dots, k_r) & \text{if } \sum_{i=1}^r \alpha_i \text{ is odd} \end{cases} \end{aligned} \quad (5.2.2)$$

where $0 \leq k_0 \leq N_0 - 1$, $0 \leq k_i \leq N_i/2 - 1$, and $\alpha_i = 0, 1$ for $i = 1, 2, \dots, r$. This property indicates that the range of k_i is reduced by half, which leads to a reduction of the computational complexity by a factor of 2^r . For real-valued input sequence, this symmetric property is also valid. Furthermore, applying the Hermitian property (5.2.1) to (5.2.2), we obtain the following property

$$\begin{aligned} & X(N_0 - k_0, \beta_1 N_1 - k_1, \dots, \beta_r N_r - k_r) \\ &= \begin{cases} X^*(k_0, k_1, \dots, k_r) & \text{if } \sum_{i=1}^r \beta_i \text{ is integer} \\ X^*(k_0 + \frac{N_0}{2}, k_1, \dots, k_r) & \text{otherwise} \end{cases} \end{aligned} \quad (5.2.3)$$

where $0 \leq k_0 \leq N_0 - 1$, $0 \leq k_i \leq N_i/2 - 1$ for $i = 1, \dots, r-1$, $0 \leq k_r \leq \lfloor N_r/4 \rfloor$, and $\beta_i = 1$ or $1/2$ for $i = 1, \dots, r$. With this property, the range of k_i can be further reduced by roughly another half. Using the properties in (5.2.2) and (5.2.3), the total number of transform outputs to be calculated is reduced to $N_0(\prod_{i=1}^{r-1} N_i)(\lfloor N_r/4 \rfloor + 1)/2^{r-1}$.

5.2.2 First Step Decomposition (FSD)

It has been illustrated in Chapter 4 that the radix-type decomposition is only valid when the sizes of all dimensions of the PTFT are the same. Otherwise, the original PTFT has to be decomposed into a number of PTFTs that have the same but smaller dimensional sizes. For real-valued input sequences, the

Figure 5.2: Decomposition process for a 2nd order PTFT when p_1 is odd.

Figure 5.3: Decomposition process for a 2nd order PTFT when p_1 is even.

first step decomposition (FSD) process is similar to that for complex-valued input sequences presented in Chapter 4. However, the decomposition cost can be reduced. Because the input sequences are real, one half of the multiplicative cost needed by (4.3.8) is saved compared to that required by complex-valued input sequences. In addition, for complex-valued input sequences, the dimension range in (4.3.9) is $0 \leq k_i \leq N_0 - 1$ for $i = 1, \dots, r$. While for real-valued input sequences, assume k_i being the dimension used for the Hermitian property in (5.2.1), k_i is in the range of $0, \dots, \lfloor N_0/2 \rfloor$, and $k_j = 0, \dots, N_0 - 1$ for $1 \leq j \leq r$ and $j \neq i$.

Instead of providing a general analysis of the decomposition overhead, let us consider an example for the 2nd order PTFT, which is defined as

$$X(k_0, k_1) = \sum_{n=0}^{N_0-1} x(n) W_{N_0}^{k_0 n} W_{N_1}^{k_1 n^2}. \quad (5.2.4)$$

Figures 5.2 and 5.3 show the FSD for a 2nd order PTFT with p_1 being odd and even, respectively. In these figures, cPTFT denotes the PTFT of complex-valued sequence and r/cPTFT denotes the PTFT of real or complex sequence depending on the input.

Based on the depicted FSD process, it can be easily seen that for complex-valued sequence, (5.2.4) is converted into p_1 2nd order PTFTs of complex-valued sequence, and for real-valued sequence, (5.2.4) is converted into p_1 2nd order PTFTs, where p_1 is odd, of real-valued sequence with a dimensional size $N_0 \times N_0$, or $(p_1 - 1)$ 2nd order PTFTs, where p_1 is even, of real-valued sequence and one 2nd order PTFT of complex-valued sequence. The computational overhead needed by the FSD is analyzed and summarized in Table 5.1. It is clear that the decomposition cost for the real-valued sequence is about one half of that for complex-valued sequence.

Table 5.1: The computational overheads needed by the FSD for a 2nd order PTFT.

Input	p_1 is odd	
	No. of real multiplications	No. of real additions
Real	$(p_1 - 1)(N_0 - 1)$	$(p_1 - 1)N_0^2$
Complex	$2(p_1 - 1)(N_0 - 1)$	$2(p_1 - 1)N_0^2$
	p_1 is even	
	No. of real multiplications	No. of real additions
Real	$p_1(N_0 - 1)$	$(p_1 - 2)N_0^2$
Complex	$2p_1(N_0 - 1)$	$2(p_1 - 2)N_0^2 + 2(N_0 - 1)$

5.3 Radix- a Algorithms

This section derives fast algorithms based on the radix- a decompositions for the $(r + 1)$ th order PTFT whose sizes of all dimensional are the same, i.e., N . In the following, this kind of PTFT will be called length- N PTFT. In addition, an example for $r = 1$ is to be presented,

First, let us briefly review the radix- a algorithm derived in section 4.4.1, where complex-valued input sequence is assumed. Based on that fast algorithm, an $(r + 1)$ th order length- N PTFT is decomposed into a^{r+1} $(r + 1)$ th order length- (N/a) PTFTs. The mathematical expressions for the radix- a algorithm are given by (4.4.4)- (4.4.9).

To study (4.4.4) for real-valued input, let us define

$$\begin{aligned}
 l'_i &= \text{mod}(a - l_i, a) \\
 &= \begin{cases} 0 & l_i = 0 \\ a - l_i & \text{otherwise} \end{cases}
 \end{aligned} \tag{5.3.1}$$

$$\alpha_i = \begin{cases} 0 & l_i = 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.3.2}$$

$$k'_i = \frac{N}{a} - \alpha_i - k_i. \tag{5.3.3}$$

From (5.3.1) and (5.3.2), it can be easily verified that

$$a\alpha_i - l'_i = l_i. \quad (5.3.4)$$

From (4.4.6), we have

$$\begin{aligned} P'_1 &= \sum_{t=0}^{\lfloor r/2 \rfloor} l'_{2t} n^{2t} = \sum_{t=0, l_{2t} \neq 0}^{\lfloor r/2 \rfloor} (a - l_{2t}) n^{2t} \\ &= a \sum_{t=0, l_{2t} \neq 0}^{\lfloor r/2 \rfloor} n^{2t} - P_1 \end{aligned} \quad (5.3.5)$$

and

$$\begin{aligned} P' &= \sum_{i=0}^r (i+1) l'_i n^i = \sum_{i=0, l_i \neq 0}^r (i+1) (a - l_i) n^i \\ &= a \sum_{i=0, l_i \neq 0}^r (i+1) n^i - P. \end{aligned} \quad (5.3.6)$$

By defining

$$\begin{aligned} \beta' &= 2\pi c P' / a \\ &= 2\pi c \sum_{i=0, l_i \neq 0}^r (i+1) n^i - 2\pi c P / a, \end{aligned} \quad (5.3.7)$$

we have

$$\cos(\beta') = \cos(\beta), \sin(\beta') = -\sin(\beta). \quad (5.3.8)$$

From (5.3.1) to (5.3.6), the twiddle factors can be expressed as

$$\begin{aligned} W_N^{\theta_r(n, \bar{l}')} W_{N/a}^{\theta_r(n, \bar{k}')} &= W_N^{\theta_r(n, \bar{l}')} W_{N/a}^{\theta_r(n, \overline{N/a - \alpha - k})} \\ &= W_N^{\theta_r(n, \bar{l}')} W_{N/a}^{-\theta_r(n, \alpha + k)} \\ &= W_N^{\theta_r(n, \bar{l}')} W_N^{-\theta_r(n, \bar{\alpha} \alpha)} W_{N/a}^{-\theta(n, \bar{k})} \\ &= W_N^{-\theta_r(n, \overline{a\alpha - l'})} W_{N/a}^{-\theta_r(n, \bar{k})} = W_N^{-\theta_r(n, \bar{l})} W_{N/a}^{-\theta_r(n, \bar{k})}. \end{aligned} \quad (5.3.9)$$

With the above derived results, let us now consider $u_a(n, l_0, l_1, \dots, l_r)$ for real-valued sequence according to the parity of a as follows.

(i). **a is odd.** In this case, following the expression (4.4.7) and applying the property (5.3.8), we have

$$u_a(n, l'_0, l'_1, \dots, l'_r) = u_a^*(n, l_0, l_1, \dots, l_r) \quad (5.3.10)$$

which leads to the conjugate symmetric property of the PTFT of real-valued sequence as shown latter.

(ii). **a is even.** In this case, we have

$$(-1)^{P_i} = (-1)^{P_1}, \quad (5.3.11)$$

When $\text{mod}(a, 4) \neq 0$, with the expression (4.4.8) and the properties (5.3.8) and (5.3.11), the equality in (5.3.10) is also satisfied. When $\text{mod}(a, 4) = 0$, we have

$$(-j)^{P'} = -(-j)^P. \quad (5.3.12)$$

With (4.4.9), (5.3.8), (5.3.11) and (5.3.12), it is straightforward to prove the relationship in (5.3.10).

Using (4.4.4), (5.3.9), (5.3.10) and (5.3.10), we have

$$\begin{aligned} & X(ak'_0 + l'_0, ak'_1 + l'_1, \dots, ak'_r + l'_r) \\ &= \sum_{n=0}^{N/a-1} \left[u_a(n, l'_0, l'_1, \dots, l'_r) W_N^{\theta_r(n, \bar{l}')} \right] W_{N/a}^{\theta_r(n, \bar{k}')} \\ &= \sum_{n=0}^{N/a-1} u_a^*(n, l_0, l_1, \dots, l_r) W_N^{-\theta_r(n, \bar{l})} W_{N/a}^{-\theta_r(n, \bar{k})} \\ &= X^*(ak_0 + l_0, ak_1 + l_1, \dots, ak_r + l_r). \end{aligned} \quad (5.3.13)$$

When the index of the output $ak'_0 + l'_0 = N$, it is replaced by 0 due to the property $W_N^{kN^i} = W_N^0$. Equation (5.3.13) shows the conjugate symmetric property of

the PTFT of real-valued sequence, which leads to a further reduction of the computational complexity compared to that needed by the fast algorithms for complex-valued sequences. For simplicity, denote $X(ak_0 + l_0, ak_1 + l_1, \dots, ak_r + l_r)$ as $X_{l_0, l_1, \dots, l_r}$. Therefore, (5.3.13) can be written as

$$X_{l'_0, l'_1, \dots, l'_r} \Rightarrow X_{l_0, l_1, \dots, l_r} \quad (5.3.14)$$

where $y \Rightarrow x$ means that y can be obtained from x .

Based on (5.3.14), when a is odd, the outputs of length- N $(r+1)$ th order PTFT of real-valued sequence can be obtained from one length- (N/a) $(r+1)$ th order PTFT of real-valued sequence for $l_i = 0, i = 0, 1, \dots, r$, and $(a^{r+1} - 1)/2$ length- (N/a) $(r+1)$ th order PTFTs of complex-valued sequence. When a is even, the outputs of length- N $(r+1)$ th order PTFT real-valued sequence can be obtained from one length- (N/a) $(r+1)$ th order PTFT of real-valued sequence for $l_i = 0, i = 0, 1, \dots, r$, and $(a^{r+1}/2 + 2^r - 1)$ length- (N/a) $(r+1)$ th order PTFTs of complex-valued sequence.

The entire computation can be accomplished by jointly using the above presented algorithms for real-valued sequences and those for complex-valued sequence reported in section 4.4.1.

Let us consider an example for $r = 1$ to illustrate the details of the derived fast algorithms.

When a is odd, (5.3.14) is elaborated into

$$\begin{aligned} X_{0, l_1} &\Rightarrow X_{0, a-l_1}, \quad 1 \leq l_1 \leq (a-1)/2 \\ X_{l_0, 0} &\Rightarrow X_{a-l_0, 0}, \quad 1 \leq l_0 \leq (a-1)/2 \\ X_{l_0, l_1} &\Rightarrow X_{a-l_0, a-l_1}, \quad 1 \leq l_0 \leq (a-1), 1 \leq l_1 \leq (a-1)/2. \end{aligned} \quad (5.3.15)$$

Therefore, to obtain all the PTFT outputs, we need to compute one length-

(N/a) 2nd order PTFT, $X_{0,0}$, of real-valued sequence and $(a^2 - 1)/2$ length- (N/a) 2nd order PTFTs of complex-valued sequence.

When a is even, (5.3.14) is elaborated into

$$\begin{aligned}
 X_{0,l_1} &\Rightarrow X_{0,a-l_1}, 1 \leq l_1 \leq a/2 - 1 \\
 X_{l_0,0} &\Rightarrow X_{a-l_0,0}, 1 \leq l_0 \leq a/2 - 1 \\
 X_{l_0,a/2} &\Rightarrow X_{a-l_0,a/2}, 1 \leq l_0 \leq a/2 - 1 \\
 X_{l_0,l_1} &\Rightarrow X_{a-l_0,a-l_1}, 1 \leq l_0 \leq (a-1), 1 \leq l_1 \leq a/2 - 1. \quad (5.3.16)
 \end{aligned}$$

and $X_{0,0}$, $X_{0,a/2}$, $X_{a/2,0}$ and $X_{a/2,a/2}$ are also to be computed. For all the PTFT outputs, we need to compute one length- (N/a) 2nd order PTFT, $X_{0,0}$, of real-valued sequence and $(a^2/2 + 1)$ length- (N/a) 2nd order PTFTs of complex-valued sequence. It should be pointed out that when $a = 2$, the conjugate symmetric property in (5.3.16) is not valid. Appendix A gives the Matlab implementation of the radix- a algorithm for the length- N 2nd order PTFTs of real-valued sequence.

5.4 Computational Complexity

This section considers the required computational complexity of the fast algorithms presented in the last two sections in terms of the number of arithmetic operations on real numbers. It is assumed that a complex multiplication uses four real multiplications and two real additions. Firstly, we consider the computational complexity of length- N PTFT, then the analysis is extended to PTFT of arbitrary dimensional sizes.

In general, arithmetic operations are needed for $u_a(n, l_0, l_1, \dots, l_r)$ and the

twiddle factors $W_N^{\theta_r(n, \vec{l})}$. Table 5.2 lists the numbers of real multiplications and real additions needed for $u_a(n, l_0, l_1, \dots, l_r)$ for real-valued input, where $0 \leq n \leq N/a - 1$, $0 \leq l_i \leq a - 1$ for $i = 1, 2, \dots, r$, and $a = 2, 3, 4, 6$ and 8 . One desirable property observed from the above analysis is that the computational complexity for $u_a(n, l_0, l_1, \dots, l_r)$ is independent of the order of the PTFT. When calculating the number of multiplications required by the

Table 5.2: The computational complexity required by $u_a(n, l_0, l_1, \dots, l_r)$ for real-valued input.

a	2	3	4	6	8
real add.	N	$4N/3$	$3N/2$	$7N/3$	$5N/2$
real mul.	0	$N/3$	0	$N/3$	$N/4$

twiddle factors, we discount the operations that are needed by the trivial twiddle factors when $n = 0$ and/or $l_i = 0$ for $i = 0, 1, \dots, r$. Because $u_a(n, l_0, l_1, \dots, l_r)$ is complex, complex multiplications are generally needed by the twiddle factors. In the case of $a = 2$ and 4 (for some combinations of l_i), $u_a(n, l_0, l_1, \dots, l_r)$ becomes real and only real multiplications are needed.

Let $C_{a,add,R} \frac{N}{a}$ and $C_{a,mul,R} \frac{N}{a}$ represent the numbers of real additions and real multiplications, as shown in Table 5.2, needed by the computation of $u_a(n, l_0, l_1, \dots, l_r)$ for real-valued input, where $n = 0, 1, \dots, N/a - 1$, and $D_{a,add}(\frac{N}{a} - 1)$ and $D_{a,mul}(\frac{N}{a} - 1)$ represent the numbers of additions and multiplications required by the twiddle factors. The total numbers of additions and multiplications needed by the proposed radix- a algorithms are derived as follows.

Based on the radix- a algorithm presented in the last section, the required numbers of real additions and real multiplications for the $(r+1)$ th order PTFT

of real-valued sequence can be calculated by

$$A_R(N) = A_R\left(\frac{N}{a}\right) + K_a A_C\left(\frac{N}{a}\right) + C_{a,add,R} \frac{N}{a} + D_{a,add}\left(\frac{N}{a} - 1\right) \quad (5.4.1)$$

$$M_R(N) = M_R\left(\frac{N}{a}\right) + K_a M_C\left(\frac{N}{a}\right) + C_{a,mul,R} \frac{N}{a} + D_{a,mul}\left(\frac{N}{a} - 1\right) \quad (5.4.2)$$

where the subscript, R or C , indicates that the quantity is for PTFTs of real- or complex-valued sequence,

$$K_a = \begin{cases} (a^{r+1} - 1)/2 & a \text{ is odd} \\ a^{r+1}/2 + 2^r - 1 & a \text{ is even} \end{cases} \quad (5.4.3)$$

and

$$D_{a,add} = \begin{cases} 0 & a = 2 \\ K_a - 1 & a = 4 \\ 2K_a & \text{other values} \end{cases}, \quad D_{a,mul} = \begin{cases} 2K_a & a = 2 \\ 3K_a - 1 & a = 4 \\ 4K_a & \text{other values} \end{cases} \quad (5.4.4)$$

At the m th step of the decomposition process, the length- N PTFT of real-valued sequence is decomposed into length- (N/a^m) PTFTs of real-valued and complex-valued sequences. After necessary manipulation, $A_R(N)$ and $M_R(N)$, where $N = a^p b$, can be expressed as

$$A_R(N) = A_R\left(\frac{N}{a^m}\right) + K_a \sum_{i=1}^m A_C\left(\frac{N}{a^i}\right) + [C_{a,add,R} + D_{a,add}] \frac{N(1 - a^{-m})}{a - 1} - m D_{a,add} \quad (5.4.5)$$

$$M_R(N) = M_R\left(\frac{N}{a^m}\right) + K_a \sum_{i=1}^m M_C\left(\frac{N}{a^i}\right) + [C_{a,mul,R} + D_{a,mul}] \frac{N(1 - a^{-m})}{a - 1} - m D_{a,mul}. \quad (5.4.6)$$

Based on the radix- a algorithm for complex-valued sequence proposed in Chapter 4, the numbers of additions and multiplications needed for the $(r + 1)$ th

order PTFT of complex-valued sequence can be computed from

$$A_C(N) = a^{r+1}A_C\left(\frac{N}{a}\right) + C_{a,add,C}\frac{N}{a} + 2(a^{r+1} - 1)\left(\frac{N}{a} - 1\right) \quad (5.4.7)$$

$$M_C(N) = a^{r+1}M_C\left(\frac{N}{a}\right) + C_{a,mul,C}\frac{N}{a} + 4(a^{r+1} - 1)\left(\frac{N}{a} - 1\right). \quad (5.4.8)$$

Therefore,

$$\begin{aligned} \sum_{i=1}^m A_C\left(\frac{N}{a^i}\right) &= \left[A_C\left(\frac{N}{a^m}\right) - 2\right] E_1(a, m, r) + 2m \\ &\quad + [C_{a,add,C} + 2(a^{r+1} - 1)]NE_2(a, m, r) \end{aligned} \quad (5.4.9)$$

$$\begin{aligned} \sum_{i=1}^m M_C\left(\frac{N}{a^i}\right) &= \left[M_C\left(\frac{N}{a^m}\right) - 4\right] E_1(a, m, r) + 4m \\ &\quad + [C_{a,mul,C} + 4(a^{r+1} - 1)]NE_2(a, m, r) \end{aligned} \quad (5.4.10)$$

where

$$\begin{aligned} E_1(a, m, r) &= \frac{a^{m(r+1)-1}}{a^{r+1} - 1} \\ E_2(a, m, r) &= \frac{1}{a^{m+1}(a^r - 1)} \left[E_1(a, m, r) - \frac{a^m - 1}{a - 1} \right]. \end{aligned} \quad (5.4.11)$$

Substituting (5.4.9) and (5.4.10) into (5.4.1) and (5.4.2), we obtain

$$\begin{aligned} A_R(N) &= A_R\left(\frac{N}{a^m}\right) - mD_{a,add} + [C_{a,add,R} + D_{a,add}]\frac{N(1 - a^{-m})}{a - 1} \\ &\quad + K_a \left\{ \left[A_C\left(\frac{N}{a^m}\right) - 2 \right] E_1(a, m, r) \right. \\ &\quad \left. + [C_{a,add,C} + 2(a^{r+1} - 1)]NE_2(a, m, r) + 2m \right\} \end{aligned} \quad (5.4.12)$$

$$\begin{aligned} M_R(N) &= M_R\left(\frac{N}{a^m}\right) - mD_{a,mul} + [C_{a,mul,R} + D_{a,mul}]\frac{N(1 - a^{-m})}{a - 1} \\ &\quad + K_a \left\{ \left[M_C\left(\frac{N}{a^m}\right) - 4 \right] E_1(a, m, r) \right. \\ &\quad \left. + [C_{a,mul,C} + 4(a^{r+1} - 1)]NE_2(a, m, r) + 4m \right\}. \end{aligned} \quad (5.4.13)$$

Let us consider the case that uses initial values of $A_R(a^{q_0}b)$, $M_R(a^{q_0}b)$, $A_C(a^{q_0}b)$

and $M_C(a^{q_0}b)$,

$$\begin{aligned} A_R(N) &= A_R(a^{q_0}b) - (p - q_0)D_{a,add} + [C_{a,add,R} + D_{a,add}](N - a^{q_0}b)/(a - 1) \\ &\quad + K_a \{ [A_C(a^{q_0}b) - 2] E_1(a, p - q_0, r) + 2(p - q_0) \\ &\quad + [C_{a,add,C} + 2(a^{r+1} - 1)]NE_2(a, p - q_0, r)/a \} \end{aligned} \quad (5.4.14)$$

$$\begin{aligned} M_R(N) &= M_R(a^{q_0}b) - (p - q_0)D_{a,mul} + [C_{a,mul,R} + D_{a,mul}](N - a^{q_0}b)/(a - 1) \\ &\quad + K_a \{ [M_C(a^{q_0}b) - 4] E_1(a, p - q_0, r) + 4(p - q_0) \\ &\quad + [C_{a,mul,C} + 4(a^{r+1} - 1)]NE_2(a, p - q_0, r)/a \}. \end{aligned} \quad (5.4.15)$$

For the 2nd order PTFT, let $I_0 = 2^{q_0}b$ for $a = 2$, the numbers of real additions and real multiplications are

$$\begin{aligned} A_R^2(N) &= \frac{A_C^2(I_0) - 2 + 5I_0}{I_0^2}N^2 - 13N + A_R^2(I_0) - A_C^2(I_0) \\ &\quad + 8I_0 + 2 + 6(p - q_0) \end{aligned} \quad (5.4.16)$$

$$\begin{aligned} M_R^2(N) &= \frac{M_C^2(I_0) - 4 + 6I_0}{I_0^2}N^2 - 12N + M_R^2(I_0) - M_C^2(I_0) \\ &\quad + 6I_0 + 4 + 6(p - q_0). \end{aligned} \quad (5.4.17)$$

Similarly, let $I_0 = 4^{q_0}b$ for $a = 4$, we have

$$\begin{aligned} A_R^2(N) &= \frac{3}{5} \frac{A_C^2(I_0) - 2 + 23I_0/6}{I_0^2}N^2 - \frac{41}{6}N + A_R^2(I_0) - \frac{3}{5}A_C^2(I_0) \\ &\quad + \frac{6}{5} + \frac{68}{15}I_0 + 10(p - q_0) \end{aligned} \quad (5.4.18)$$

$$\begin{aligned} M_R^2(N) &= \frac{3}{5} \frac{M_C^2(I_0) - 4 + 5I_0}{I_0^2}N^2 - \frac{19}{3}N + M_R^2(I_0) - \frac{3}{5}M_C^2(I_0) \\ &\quad + \frac{12}{5} + \frac{10}{3}I_0 + 10(p - q_0). \end{aligned} \quad (5.4.19)$$

Table 5.3 shows comparisons on the numbers of real multiplications needed by various proposed fast algorithms for the 2nd order PTFT. For the radix- a

algorithms, initial values, shown in *Italic font*, are used. It is seen that in general, the number of real multiplications decreases as the value of a is increased. The additive complexity needed by the presented algorithms can be analyzed

Table 5.3: The number of real multiplications for the length- N 2nd order PTFT of real-valued sequence

N_1	radix-2	radix-4	radix-8	N_2	radix-3
8	<i>4</i>	<i>4</i>	<i>4</i>	3	<i>2</i>
16	70	<i>70</i>	<i>70</i>	9	53
32	508	258	<i>258</i>	27	614
64	2626	1504	1208	81	5969
128	11848	5996	5910	243	55082
256	50254	27946	22498	729	499853
512	206932	104950	85816	2187	4511054

similarly. Table 5.4 shows the number of real additions for the 2nd order PTFT. Similar observations to that from Table 5.3 can be made among the additive complexities needed by the proposed fast algorithms. For example, the algorithms using larger values of a generally require smaller additive complexities. When $a = 3$, however, the proposed algorithm needs more real additions and multiplications than the algorithms for other values of a .

Table 5.4: The number of real additions for the length- N 2nd order PTFT of real-valued sequence

N_1	radix-2	radix-4	radix-8	N_2	radix-3
8	<i>88</i>	<i>88</i>	<i>88</i>	3	<i>9</i>
16	464	<i>464</i>	<i>464</i>	9	173
32	2158	1272	<i>1272</i>	27	1769
64	9332	5666	4670	81	16493
128	38842	22034	20056	243	150089
256	158528	92684	78472	729	1355693
512	640582	361948	306382	2187	12215849

Now, we compare the required computational complexities of the proposed fast algorithms, as listed in Tables 5.3 and 5.4, with those needed by the fast

algorithms for complex-valued sequences. Figure 5.4 shows the comparisons on the required numbers of real multiplications for $a = 2, 3, 4$ and 8. For $a = 2$, the proposed algorithm does not achieve much reduction of the computational complexity because the conjugate symmetric property is not valid. When $a = 3, 4$ or 8, the proposed algorithms achieve about 40% to 50% savings on the computational complexity. Similar observations can be made about the ratios between the number of real additions needed by the proposed algorithms for real-valued input and the algorithms for complex-valued input.

Figure 5.4: The ratios between the numbers of multiplications needed by the fast 2nd order PTFT (with dimensional sizes $N \times N$) algorithms for real- and complex-valued inputs.

Next, let us analyze the computational complexity of 2nd order PTFT with dimensional sizes $N \times N_1$, where $N \neq N_1$ and $N_1 = p_1 N$. Based on the FSD, the required numbers of real additions and real multiplications can be obtained

from:

$$A_R(N \times N_1) = p_1 A_R(N) + (p_1 - 1)N^2$$

$$M_R(N \times N_1) = p_1 M_R(N) + (p_1 - 1)(N - 1)$$

for odd p_1 , and

$$A_R(N \times N_1) = (p_1 - 1)A_R(N) + A_C(N) + (p_1 - 2)N^2$$

$$M_R(N \times N_1) = (p_1 - 1)M_R(N) + M_C(N) + p_1(N - 1)$$

for even p_1 . Thus, the ratio of the number of multiplications between the fast algorithms for real and complex inputs can be written as

$$\begin{aligned} & \frac{M_R(N \times N_1)}{M_C(N \times N_1)} \\ &= \begin{cases} \frac{p_1 M_R(N) + (p_1 - 1)(N - 1)}{p_1 M_C(N) + 2(p_1 - 1)(N - 1)}, & p_1 \text{ is odd} \\ \frac{(p_1 - 1)M_R(N) + M_C(N) + p_1(N - 1)}{p_1 M_C(N) + 2(p_1 - 1)(N - 1)}, & p_1 \text{ is even} \end{cases} \end{aligned} \quad (5.4.20)$$

Since $M_R(N) \gg N$ and $M_C(N) \gg N$, this ratio can be approximated by

$$\frac{M_R(N \times N_1)}{M_C(N \times N_1)} \approx \begin{cases} \frac{M_R(N)}{M_C(N)}, & p_1 \text{ is odd} \\ \frac{1}{p_1} + (1 - \frac{1}{p_1}) \frac{M_R(N)}{M_C(N)}, & p_1 \text{ is even} \end{cases} \quad (5.4.21)$$

The ratio of the numbers of additions between the fast algorithms for real- and complex-valued sequences is given by

$$\begin{aligned} & \frac{A_R(N \times N_1)}{A_C(N \times N_1)} \\ &= \begin{cases} \frac{p_1 A_R(N) + (p_1 - 1)N^2}{p_1 A_C(N) + 2(p_1 - 1)N^2}, & p_1 \text{ is odd} \\ \frac{(p_1 - 1)A_R(N) + A_C(N) + (p_1 - 2)N^2}{p_1 A_C(N) + 2(p_1 - 2)N^2 + 2(N - 1)}, & p_1 \text{ is even} \end{cases} \end{aligned} \quad (5.4.22)$$

Because $A_R(N) < A_C(N)$, we can easily prove $A_R(N \times N_1) < A_C(N \times N_1)$.

Figure 5.5 and Figure 5.6, respectively, show the ratios between the numbers of real multiplications and additions needed by the fast algorithms for the

2nd order PTFT of dimensional sizes $N \times N_1$ with real- and complex-valued sequences. The computation of the PTFT of real-valued sequence for these figures are implemented by jointly using the FSD and radix-8 algorithm. In these two figures, the axis associated with N_1 is indirectly expressed in terms of parameter p_1 . It is clearly shown that the ratios change with the value of p_1 and their averages tend to be about 0.53.

Figure 5.5: The ratios between the numbers of multiplications needed by the fast 2nd order PTFT (with dimensional sizes $N \times N_1$) algorithms for real- and complex-valued inputs.

5.5 Split-radix Algorithms

In this section, we derive split-radix- $m/(ma)$ type algorithms for real-valued input sequence, where m is a positive divisor of the integer a , which is usually denoted as $m|a$. Note that for $m = 1$, split-radix- $1/a$ is equivalent to radix- a algorithm, which has already been analyzed in Section 5.3. Thus, we exclude the case of $m = 1$.

Figure 5.6: The ratios between the numbers of additions needed by the the fast 2nd order PTFT (with dimensional sizes $N \times N_1$) algorithms of real- and complex-valued inputs.

Based on the decomposition process derived in Section 4.4.2, we know that for an $(r+1)$ th order PTFT, the outputs whose indices are divisible by m , for $i = 0, 1, \dots, r$, can be computed by

$$X(mk_0, mk_1, \dots, mk_r) = \sum_{n=0}^{\frac{N}{m}-1} u_m(n, 0, \dots, 0) W_{N/m}^{\theta_r(n, \bar{k})}, \quad (5.5.1)$$

and the other PTFT outputs can be computed according to

$$\begin{aligned} X(mak_0 + l_0, mak_1 + l_1, \dots, mak_r + l_r) \\ = \sum_{n=0}^{\frac{N}{ma}-1} u_{ma}(n, l_0, \dots, l_r) W_N^{\theta_r(n, \bar{l})} W_{N/(ma)}^{\theta_r(n, \bar{k})}, \end{aligned} \quad (5.5.2)$$

where $0 \leq k_i \leq N/(ma) - 1$, $0 \leq l_i \leq ma - 1$, and at least one l_i is not a multiple of m . For real-valued input sequence, applying the properties (5.3.14) to (5.5.2), we have

$$\begin{aligned} X(mak'_0 + l'_0, mak'_1 + l'_1, \dots, mak'_r + l'_r) \\ = X^*(mak_0 + l_0, mak_1 + l_1, \dots, mak_r + l_r). \end{aligned} \quad (5.5.3)$$

i.e., among all the combinations of l_i in (5.5.2), one half of them can be obtained from the other outputs. Therefore, to obtain the outputs of length- N $(r+1)$ th order PTFT of real sequences, we need compute one length- (N/m) $(r+1)$ th order PTFT and $K_{m/(ma)}^{r+1} = a^{r+1}(m^{r+1} - 1)/2$ length- $(N/(ma))$ $(r+1)$ th order PTFTs of complex sequences.

Let us consider an example for $r = 1$ to support the derived split-radix algorithms as follows.

(a). $N = 2^p b$

With $m = 2$ and $a = 2$, a split-radix 2/4 algorithm is obtained. Following (5.5.3), we have $[X_{03}, X_{23}, X_{30}, X_{33}, X_{32}, X_{31}] \implies [X_{01}, X_{21}, X_{10}, X_{11}, X_{12}, X_{13}]$. Thus, the entire computation task includes one length- $(N/2)$ 2nd order PTFT of real-valued sequence and 6 length- $(N/4)$ 2nd order PTFTs of complex-valued sequence.

(b). $N = 4^p b$

With $m = 2$ and $a = 4$, a split-radix 2/8 is obtained. Applying the property in (5.5.3), we need to compute one length- $(N/2)$ 2nd order PTFT of real-valued sequence and 24 length- $(N/8)$ 2nd order PTFTs of complex-valued sequence to obtain all the transform outputs.

In general, it is quite difficult and tedious to derive mathematical expressions for split-radix algorithms with the parameters of m and a . As an example, the details of derivation for split-radix- a/a^2 algorithm are presented here. For an $(r+1)$ th order PTFT of complex-valued input sequence, the number of additions

and multiplications are given by the recursive equations as follows

$$\begin{aligned} A_C(N) &= A_C\left(\frac{N}{a}\right) + a^{r+1}(a^{r+1} - 1)A_C\left(\frac{N}{a^2}\right) + C_{a,add,C}\frac{N}{a^2} \\ &\quad + 2a^{r+1}(a^{r+1} - 1)\left(\frac{N}{a^2} - 1\right) \end{aligned} \quad (5.5.4)$$

$$\begin{aligned} M_C(N) &= M_C\left(\frac{N}{a}\right) + a^{r+1}(a^{r+1} - 1)M_C\left(\frac{N}{a^2}\right) + C_{a,mul,C}\frac{N}{a^2} \\ &\quad + 4a^{r+1}(a^{r+1} - 1)\left(\frac{N}{a^2} - 1\right). \end{aligned} \quad (5.5.5)$$

While for real-valued input sequences, the number of additions and multiplications are expressed as

$$\begin{aligned} A_R(N) &= A_R\left(\frac{N}{a}\right) + K_{a/a^2}^{r+1}A_C\left(\frac{N}{a^2}\right) + C_{a,add,R}\frac{N}{a^2} \\ &\quad + 2K_{a/a^2}^{r+1}\left(\frac{N}{a^2} - 1\right) \end{aligned} \quad (5.5.6)$$

$$\begin{aligned} M_R(N) &= M_R\left(\frac{N}{a}\right) + K_{a/a^2}^{r+1}M_C\left(\frac{N}{a^2}\right) + C_{a,mul,R}\frac{N}{a^2} \\ &\quad + 4K_{a/a^2}^{r+1}\left(\frac{N}{a^2} - 1\right) \end{aligned} \quad (5.5.7)$$

It can be easily seen that the equations for complex-valued and real-valued input sequence can be expressed in the form of

$$f(x) = f\left(\frac{x}{a}\right) + a^{r+1}(a^{r+1} - 1)f\left(\frac{x}{a^2}\right) + v_0x + v_1 \quad (5.5.8)$$

and

$$g(x) = g\left(\frac{x}{a}\right) + K_{a/a^2}^{r+1}f\left(\frac{x}{a^2}\right) + t_0x + t_1, \quad (5.5.9)$$

respectively, where $x = a^p b$, v_0 , v_1 , t_0 and t_1 are constants. After m -th decomposition, (5.5.9) can be written as

$$g(x) = g\left(\frac{x}{a^m}\right) + K_{a/a^2}^{r+1} \sum_{i=2}^{m+1} f\left(\frac{x}{a^i}\right) + t_0x \sum_{i=0}^{m-1} \frac{1}{a^i} + mt_1. \quad (5.5.10)$$

It is obviously that the closed form expression of $g(x)$ requires expression of $f(x)$ which will be analyzed next. The solution to (5.5.8) is in the form

$$f(x) = c_1(a^{r+1})^p + c_2(1 - a^{r+1})^p + c_3a^p b + c_4. \quad (5.5.11)$$

Substituting (5.5.11) back into the recurrence relation, we get:

$$\begin{aligned}
 & c_1(a^{r+1})^p + c_2(1 - a^{r+1})^p + c_3a^pb + c_4 \\
 = & c_1(a^{r+1})^{p-1} + c_2(1 - a^{r+1})^{p-1} + c_3a^{p-1}b + c_4 \\
 & + a^{r+1}(a^{r+1} - 1)[c_1(a^{r+1})^{p-2} + c_2(1 - a^{r+1})^{p-2} + c_3a^{p-2}b + c_4] \\
 & + v_0 a^pb + v_1.
 \end{aligned} \tag{5.5.12}$$

Looking only at the coefficients of a^pb and 1 (constants), respectively,

$$\begin{cases} c_3 = \frac{c_3}{a} + \frac{a^{r+1}(a^{r+1}-1)}{a^2}c_3 + v_0 \\ c_4 = c_4 + a^{r+1}(a^{r+1} - 1)c_4 + v_1, \end{cases} \tag{5.5.13}$$

we obtain the values of c_3 and c_4 as

$$\begin{cases} c_3 = \frac{v_0}{\frac{a-1}{a} - \frac{a^{r+1}(a^{r+1}-1)}{a^2}} \\ c_4 = -\frac{v_1}{a^{r+1}(a^{r+1}-1)} \end{cases}. \tag{5.5.14}$$

To determine the values of c_1 and c_2 , we need two initial values $f(x_0)$ and $f(x_1)$, where $x_0 = a^{p_0}b$, $x_1 = ax_0 = a^{p_1}b$. With the initial values we have

$$\begin{cases} f(x_0) = c_1(a^{r+1})^{p_0} + c_2(1 - a^{r+1})^{p_0} + c_3a^{p_0}b + c_4 \\ f(x_1) = c_1(a^{r+1})^{p_1} + c_2(1 - a^{r+1})^{p_1} + c_3a^{p_1}b + c_4 \end{cases} \tag{5.5.15}$$

The solutions to these equations are

$$\begin{cases} c_1 = \frac{(1-a^{r+1})f(x_0) - f(x_1) - c_3x_0(1-a^{r+1}) + c_4a^{r+1}}{(1-2a^{r+1})(a^{r+1})^{p_0}} \\ c_2 = \frac{a^{r+1}f(x_0) - f(x_1) - c_3a(a^r-1)x_0 - c_4(a^{r+1}-1)}{(1-a^{r+1})^{p_0}(2a^{r+1}-1)} \end{cases}. \tag{5.5.16}$$

Therefore, using two initial values, $f(x_0)$ and $f(x_1)$, the number of operations for complex-valued input sequence can be obtained from (5.5.11) with the coefficients c_1 to c_4 given in (5.5.14) and (5.5.16). Substituting the obtained expression into (5.5.9) and use an initial valued of $g(x_0)$, the number of operations for real-valued input sequence is obtained.

Table 5.5 lists the number of real multiplications and additions that are needed by the split-radix algorithm with $m = 2$ and $a = 2$ and 4. It is seen that the split-radix algorithm is more computational efficient than the radix- a algorithm. Compared to the split-radix algorithms for complex-valued sequence, the proposed algorithms achieve about 50% savings on the computational complexity.

Table 5.5: The number of real multiplications and additions required by the split-radix algorithm

$N =$	Split-radix-2/4		Split-radix-2/8	
	$M_R(N)$	$A_R(N)$	$M_R(N)$	$A_R(N)$
8	4	88	4	88
16	40	276	40	276
32	207	1095	207	1095
64	1038	4394	1087	4471
128	4733	18317	4863	17895
256	20204	73504	20575	74071
512	83707	299203	81663	296775

5.6 Summary

This chapter presents radix- a and split-radix fast algorithms for the computation of the PTFTs of real-valued sequences. Analysis and comparison on the computational complexity are also made in terms of the numbers of real additions and real multiplications. The proposed algorithms provide the flexibility to support various input sequence lengths by setting the parameters of a , b and p . With the use of conjugate symmetric property, the presented fast algorithms, with $a = 3, 4$ and 8, effectively reduce the computational complexity compared to that needed by the reported fast algorithms for complex-valued sequences.

Chapter 6

Pipeline Structure for PTFT Computation

6.1 Introduction

In Chapters 4 and 5, fast algorithms have been reported for the computation of PTFTs of complex-valued and real-valued sequences, respectively. The proposed algorithms can support various input sequence lengths and efficiently compute the PTFTs. However, real-time processing is required in many practical applications and the software implementation may not be fast enough. To support high processing throughput, it is necessary to implement the fast algorithms in hardware that should generally require the chip area and power consumption as small as possible. Therefore, it is critical to select a suitable hardware structure that has the potential for achieving such requirements.

Given a particular computational algorithm and problem size, there are two popular hardware architectures. One is based on pipelined structure and the

other is based on parallel structure. For some applications, high throughput and low latency require the use of parallel architecture at the expense of high cost of hardware and power consumption because many arithmetic functions are implemented in hardware. An alternative is the pipelined structure that allows concurrent operations on a number of hardware elements in an organized manner to support high-speed, real-time applications. These operations can be scheduled in various ways according to the specific application requirements. One type of pipelined structure is particularly useful to deal with sequential input data streams which is the dominant format for communication applications. Furthermore, in the literature, it is often observed that the maximum processing throughput (or capability) that can be offered by the arithmetic units is not fully utilized for applications that have sequential input and output streams. This is because a waiting time is needed for sequentially collecting the data that is simultaneously processed by the parallel arithmetic functions. For example, such phenomena is observed from the earlier reported pipelined FFT processor implementation [105] in which radix-4 butterfly cannot be fully utilized if a sequentially presented data stream is dealt with. To avoid the throughput mismatch, it is necessary to map the algorithm into a hardware structure that contains the arithmetic units whose processing throughput is just enough to cope with the throughput of the input data stream. The purpose of this chapter is to examine some related algorithms and propose new algorithms suited to high-speed pipelined hardware architectures. The rest of this chapter is organized as follows. The algorithms and hardware structures for fast Fourier transform (FFT) of the sequential input data stream are reviewed and discussed in Section 6.2, which serves as a basis for developing the hardware design for the PTFT in the succeeding sections. In addition, a new single-path pipelined

hardware structure for DFT computation based on the radix-2² FFT algorithm is proposed in this section. Section 6.3 presents the description of the single-path pipelined structure based on the radix-2² fast algorithm for the PTFT. The operations of the functional blocks used in the pipelined structure are explained, and the merits of the proposed structure are also discussed. Finally, conclusions are given in Section 6.4.

6.2 Algorithms and Pipeline Structures for FFT

Discrete Fourier transform (DFT) has been an important signal processing function for many applications. A recent application is the orthogonal frequency division multiplexing (OFDM) for digital video broadcasting, digital audio broadcasting and very high-speed digital subscriber line [106, 107].

The DFT of an N -point sequence $x(n)$, $n = 0, 1, \dots, N-1$, is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad (6.2.1)$$

where $k = 0, 1, \dots, N-1$, $W_N = e^{-j2\pi/N}$, $N = 2^r$ and $r > 2$ is an integer. The straightforward DFT calculation requires about N^2 complex multiplications and can be cumbersome for typical values of N . To reduce the computational complexity, many FFT algorithms are reported in the literature [67–73]. This section focuses on the reviews and discussions of hardware-oriented FFT processors.

6.2.1 Algorithms

The FFT algorithms share the same strategy, i.e., decomposing the DFT into two or more DFTs of shorter input sequences recursively. Here, we stick to the principle of decimation in frequency (DIF). Decimation in time (DIT) algorithms can be easily derived by applying the transposition theorem to the DIF type algorithms.

In short, using the radix-2 decomposition, (6.2.1) is decomposed into even and odd frequency components as

$$X(k_1 + 2k_2) = \sum_{n=0}^{N/2-1} \{[x(n) + (-1)^{k_1}x(n + N/2)]W_N^{k_1n}\}W_{N/2}^{k_2n} \quad (6.2.2)$$

where $k_1 = 0, 1$ and $k_2 = 0, 1, \dots, N/2 - 1$. For the radix-4 algorithm, (6.2.1) equates the sum of 4 DFTs of length- $N/4$ sequences, the resulting formula is

$$X(k_1 + 4k_2) = \sum_{n=0}^{N/4-1} \{[x(n) + (-j)^{k_1}x(N/4 + n) + (-1)^{k_1}x(N/2 + n) + (j)^{k_1}x(3N/4 + n)]W_N^{k_1n}\}W_{N/4}^{k_2n} \quad (6.2.3)$$

where $k_1 = 0, 1, 2, 3$ and $k_2 = 0, 1, \dots, N/4 - 1$. A more in-depth description of FFT algorithms can be found in Section 2.6.

The radix-2² fast algorithm [108] considers the first 2 steps of radix-2 decomposition together. The key steps of the radix-2² fast algorithm are reported as follows :

$$\begin{aligned} & X(k_1 + 2k_2 + 4k_3) \quad (6.2.4) \\ &= \sum_{n_3=0}^{N/4-1} \sum_{n_2=0}^1 \sum_{n_1=0}^1 x\left(\frac{N}{2}n_1 + \frac{N}{4}n_2 + n_3\right) W_N^{\left(\frac{N}{2}n_1 + \frac{N}{4}n_2 + n_3\right)(k_1 + 2k_2 + 4k_3)} \\ &= \sum_{n_3=0}^{N/4-1} \sum_{n_2=0}^1 \left\{ B_{N/2}\left(\frac{N}{4}n_2 + n_3, k_1\right) W_N^{\left(\frac{N}{4}n_2 + n_3\right)k_1} \right\} W_N^{\left(\frac{N}{4}n_2 + n_3\right)(2k_2 + 4k_3)} \end{aligned}$$

where k_1 and k_2 are 0 or 1, $k_3 = 0, 1, \dots, N/4 - 1$ and

$$B_{N/2}(\frac{N}{4}n_2 + n_3, k_1) = x(\frac{N}{4}n_2 + n_3) + (-1)^{k_1}x(\frac{N}{4}n_2 + n_3 + \frac{N}{2}) \quad (6.2.5)$$

which is equivalent to a butterfly computation in radix-2 algorithm. In (6.2.4), the twiddle factors inside and outside of the braces can be combined for a further decomposition, which is the key difference from the decomposition process of the radix-2 fast algorithm. It was demonstrated in [108] that the combined twiddle factors can be expressed into

$$W_N^{(\frac{N}{4}n_2+n_3)k_1}W_N^{(\frac{N}{4}n_2+n_3)(2k_2+4k_3)} = (-j)^{n_2(k_1+2k_2)}W_N^{n_3(k_1+2k_2)}W_{N/4}^{n_3k_3}. \quad (6.2.6)$$

Substituting the twiddle factors in (6.2.4) with (6.2.6), (6.2.4) can be written into a set of length- $(N/4)$ DFTs as follows:

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{N/4-1} \left\{ B_{N/4}(n_3, k_1, k_2) W_N^{n_3(k_1+2k_2)} \right\} W_{N/4}^{n_3k_3} \quad (6.2.7)$$

where $B_{N/4}(n_3, k_1, k_2)$ is the butterfly computation at the next stage defined as:

$$B_{N/4}(n_3, k_1, k_2) = B_{N/2}(n_3, k_1) + (-j)^{(k_1+2k_2)}B_{N/2}(n_3 + \frac{N}{4}, k_1). \quad (6.2.8)$$

The entire DFT computation is now decomposed into four length- $(N/4)$ DFTs, defined in (6.2.7), and each of them can be further decomposed in the same way until length-2 or length-4 DFTs are reached.

From the above decomposition procedures, we can see that

- The radix-2 algorithm decomposes the entire DFT into $\log_2 N$ stages, each stage requires the radix-2 butterfly operation, and non-trivial twiddle factors are generally needed at the outputs of every stage (except the last one).

- The radix-4 algorithm decomposes the entire DFT into $\log_4 N$ stages, each stage requires the radix-4 butterfly operation which is fairly complicated, and non-trivial twiddle factors are required at the outputs of every stage (except the last one).
- The radix- 2^2 algorithm also decomposes the entire DFT into $\log_2 N$ stages, each stage requires the radix-2 butterfly operation. However, the non-trivial twiddle factors are necessary only at the outputs of every other stages.

To understand the relationship among the three algorithms, the signal flow graphs (SFG) of the radix-2, radix-4 and radix- 2^2 fast algorithms for a length-16 DFT are shown in Figure 6.1. From the figure, it can be easily observed that for $N = 16$, both the radix-2 algorithm and radix- 2^2 algorithm consist of 4 stages, and each stage requires 16 complex additions/subtractions. Non-trivial multiplications are needed at the outputs of stage 1 and 2 for radix-2 algorithm, but they are only needed at the outputs of stage 2 for radix- 2^2 algorithm. The radix-4 algorithm consists of 2 stages, each needs 32 complex additions/subtractions, and non-trivial multiplications are required at the output of stage 1. From these observations we know that the radix- 2^2 algorithm has the same computational complexity as the radix-4 algorithm, and remains the same radix-2 butterfly structure as the radix-2 algorithm.

6.2.2 Reported Pipelined Structures

Many hardware structures and designs reported in the literature have the potential of dealing with sequential input data streams. In general, in order to provide

Figure 6.1: Signal flow graph of (a) radix-2, (b) radix-4, (c) radix- 2^2 fast algorithm ($N=16$).

the global interconnection, the input stream has to be stored, delayed and distributed timely to the arithmetic units for processing. Thus a pipelined FFT hardware structure consists of five types of primary building blocks: butterfly units, delay units, complex multipliers for the twiddle factors, ROM devices for twiddle factors, and the commutators for the control of data stream movements. The differences between the architectures stem from the structure of the commutators and their associated control signals, and the length of delay units. There are basically three kinds of commutators: multipath delay commutator (MDC), single-path delay feedback (SDF) and single-path delay commutator (SDC).

Radix-2 MDC (R2MDC) and radix-4 MDC (R4MDC) [109] are the most straightforward ways for pipelined implementations of radix-2 and radix-4 FFT algorithms, respectively. The input sequence is broken into 2 or 4 parallel data streams flowing forward. At each stage the MDC distributes the data streams into the data elements with proper delays to enter the butterfly units. The butterfly units and multipliers are in 50 and 25 percent utilization for R2MDC and R4MDC algorithms, respectively. Radix-4 SDF (R4SDF) [110] employs CORDIC iterations and uses the registers more efficiently by storing 3 out of 4 radix-4 butterfly outputs. A single data stream goes through the multiplier at every stage. The utilization of multipliers is increased to 75 percent while the utilization of the butterfly is dropped to 25 percent. With the condition that the processing elements for arithmetic operations are operated at the input data rate with a processing latency of N word cycles, more attention has recently been paid to two single-path pipelined structures reported many years ago. Both structures have the capabilities of naturally interfacing with the sequential input/output stream and performing processing for consecutive DFTs without

requiring any extra cycles for resetting operations. The one with SDF [111] is based on the radix-2 FFT algorithm and the other one with the SDC [112] is based on a modified radix-4 FFT algorithm. Attempts of further improvements on these two types of structures have also reported in the literature. To minimize the power consumption, a novel coefficient ordering is presented in [113] to reduce the switching activities between the successive coefficients fed to the complex multipliers for the structure reported in [112]. A multi-path pipelined structure is reported in [114]¹ to increase the processing throughput.

A radix-2² fast algorithm was reported in [108] to combine the twiddle factors located at two consecutive stages in the radix-2 FFT algorithms into those located at one stage only. Thus the complex multiplier is needed only at the output of every other stage. The number of complex multipliers needed by a pipelined implementation based on the radix-2² algorithms is reduced by about one half, i.e., the same as that needed by the implementation based on the radix-4 algorithm. Because it uses the SDF structure, it also requires $N - 1$ complex data stores.

6.2.3 Proposed Structure

Based on the concepts reported in [112] and the radix-2² fast algorithm, this section reports a new cascaded single-path pipelined structure to reduce the number of complex adders compared to those previously reported (see Table 6.3). The proposed structure requires more data stores than those based on SDF structure, which may be a drawback for implementing DFTs of large block sizes because the data stores used for the shift registers generally need more chip area

¹In table I of reference [114], the comparisons on a few items with the structure reported in [112] were incorrect. Please refer to Table I in [112] for correct information

and power consumption than the RAM. To avoid this drawback, considerations are also given to use RAMs to replace the shift registers. Section 2 briefly

Figure 6.2: Block diagram of the single-path pipelined structure based on the radix-2² algorithm ($N = 16$).

presents the radix-2² fast algorithm and Section 3 gives the general description of the single-path pipelined structure based on the radix-2² fast algorithm. The operations of the functional blocks used in the pipeline are also discussed. The merits of the proposed structure are discussed in Section 4. Finally, conclusions are given in Section 5.

Based on the radix-2² fast algorithm, the length-16 DFT can be computed by a cascaded single-path pipelined hardware structure as shown in Figure 6.2. Each of the four stages contains the functions that are needed by the processing operations for the corresponding stage in Figure 6.1(c). The complex multiplier is used for the non-trivial twiddle factors located at the outputs of stage 2 in Figure 6.1(c). For a length- N DFT, this structure contains $\log_2 N$ cascaded stages. Each stage in Figure 6.2 requires a delay and commutator (DAC) unit for storing input data and timely distributing them in a correct order for processing by the complex adder/subtractor and the complex multiplier. Let us

Table 6.1: Operations of the DAC unit at stage 1 for $N=16$

n	0	...	7	8	...	15
Input	$x(0)$	...	$x(7)$	$x(8)$	...	$x(15)$
Output 1	$\sim$	...	$\sim$	$x(0)$	...	$x(7)$
Output 2	$\sim$	...	$\sim$	$x(8)$	...	$x(15)$
Control	$\sim$	...	$\sim$	0	...	0
n	16	...	23	24	...	31
Input	$\mathbf{x}(0)$	...	$\mathbf{x}(7)$	$\mathbf{x}(8)$	...	$\mathbf{x}(15)$
Output 1	$x(0)$	...	$x(7)$	$\mathbf{x}(0)$	...	$\mathbf{x}(7)$
Output 2	$x(8)$	...	$x(15)$	$\mathbf{x}(8)$	...	$\mathbf{x}(15)$
Control	1	...	1	0	...	0

next consider the functionalities and the implementation of these units used in a stage.

6.2.3.1 Delay and commutator unit

The main function of the DAC unit is to provide correct timing for the input data required by the adder/subtractor for butterfly computations defined in the i th stage of Figure 6.1(c). The DAC unit can be implemented with either shift registers or RAMs to obtain the same function.

The former implementation consists of two shift registers, as shown in Figure 6.3(a), each storing and shifting $N/2^i$ complex data, where $1 \leq i \leq \log_2 N$ is the stage index, and two 2-to-1 complex multiplexers. Let us consider the example for $N = 16$. In the first 8 word cycles, i.e., $n = 0$ to 7, $x(0), x(1), \dots$, and $x(7)$ are sequentially shifted into the first length-8 shift register with $x(0)$ available at its output. At the next word cycle, i.e., $n = 8$, both $x(0)$ and $x(8)$ appear at the two outputs of the DAC unit when the 2-to-1 complex multiplexers are at position 0. Similarly the other seven pairs of $x(n)$ and $x(n+8)$ for n

= 1 to 7 sequentially appear at the outputs of the DAC unit within the next 7 word cycles. Table 6.1 shows the details of the operations in which it is assumed that the first data $x(0)$ arrives at the input of the DAC unit to enter the shift register at the first cycle, i.e., $n = 0$. Before $n = 16$, the entire input sequence is in the two shift registers, and $x(0)$ and $x(8)$ are available at the outputs of the two shift registers. When the 2-to-1 multiplexers are at position 1 in the next 8 word cycles, the same output sequence as that in the previous 8 cycles is obtained at the outputs of the DAC unit. The same operations are repeated for the computation of the next DFT computation. Table 6.1 also shows the 8 outputs (in bold face) of the DAC unit for the next DFT computation.

Figure 6.3: Block diagram of delay and commutator.

When the block size of the DFT is large, i.e., $N > 64$, the use of shift registers to implement the DAC unit may require a large chip area and consume more power since the data is actively shifted all the time. One alternative

implementation is to use RAMs to replace the shift registers, particularly for the front stages of the pipeline, as shown in Figure 6.3 (b). Let us consider stage 1 of the pipelined structure for a length-16 DFT and assume that the RAMs can write and read once in each word cycle. The 16 input data are written into the two RAMs sequentially with the first 8 data in RAM 1 and the next 8 data in RAM 2. From cycle 8, the 2-to-1 multiplexer is at position 0 so that $x(n)$ and $x(n + 8)$, where $n = 0, 1, \dots, 7$, are available at outputs 1 and 2, respectively within word cycles 8 to 15, i.e., for $n = 8$ to 15. In the next 8 cycles, the 2-to-1 multiplexer is at position 1 to obtain the same data sequence as that in the previous 8 cycles. At the same time, the first 8 input data of the next DFT computation are sequentially written into RAM 1 before the data for the current computation are sequentially read out.

Figure 6.4: Block diagram of modified adder/subtractor unit.

The structure of the DAC unit in Figure 6.3 can be used for any stage in the pipeline. For stage i , however, the number of complex data stores in a shift register (or a RAM) should be $N/2^i$, and the control signals for the DAC operations at stage i should be accordingly designed.

Table 6.2: Operations of addition and subtraction at the first stage for $N=16$.

$n =$	8	9	10	11
Add/sub	+	+	+	+
Swap	no	no	no	no
Output	$x(0) + x(8)$	$x(1) + x(9)$	$x(2) + x(10)$	$x(3) + x(11)$
$n =$	12	13	14	15
Add/sub	+	+	+	+
Swap	no	no	no	no
Output	$x(4) + x(12)$	$x(5) + x(13)$	$x(6) + x(14)$	$x(7) + x(15)$
n	16	17	18	19
Add/sub	—	—	—	—
Swap	no	no	no	no
Output	$x(0) - x(8)$	$x(1) - x(9)$	$x(2) - x(10)$	$x(3) - x(11)$
n	20	21	22	23
Add/sub	—	—	—	—
Swap	yes	yes	yes	yes
Output	$-j[x(4) - x(12)]$	$-j[x(5) - x(13)]$	$-j[x(6) - x(14)]$	$-j[x(7) - x(15)]$

6.2.3.2 Butterfly computation

Let us consider the implementation of butterfly computation. Closely observing the operations defined in (6.2.5) and Figure 6.1(c), it is seen that at the first stage of Figure 6.1(c), 8 complex additions are performed sequentially within 8 cycles, and 8 complex subtractions are performed sequentially in the next 8 cycles. Therefore, a controllable adder/subtractor, as shown in Figure 6.4, is sufficient to perform these operations within 16 cycles. Because a trivial twiddle factor, $-j$, at the first and third stages of the DFT computation (see Figure 6.1(c)), a further consideration is to combine the operations needed by the trivial twiddle factor and those for butterfly computation. It is known that the computation $-j[(a + jb) - (c + jd)]$ can be expressed into $(b - d) + j(c - a)$ so that the multiplication by $-j$ can be met by a modified complex subtracter

with swapped inputs and outputs. Figure 6.4 shows the implementation of a complex adder/subtractor for the computation of $[(a + jb) \pm (c + jd)]$ or $-j[(a + jb) - (c + jd)]$. The 2:1 multiplexers are used for swapping the real and imaginary parts of the input/outputs when the trivial twiddle factor is applied. In this figure, the add/subtraction control is specified according to the butterfly operations and a swap control is applied only when the trivial twiddle factor $-j$ is applied. Table 6.2 shows the details and timing information of the butterfly operations and the multiplications with the trivial twiddle factor.

Up to now, the operations for the first stage of Figure 6.1(c) is completed within the 16 cycles. The presented concepts can be easily generalized for the operations of other stages.

6.2.4 Discussions

The proposed single-path pipelined structure is able to continuously process a sequential input stream without requiring any extra cycle for resetting operations between successive DFT blocks, i.e., the proposed structure is able to compute a length- N DFT within exactly N cycles. It can be easily calculated that the structure needs $2(N - 1)$ complex data stores, $\log_2 N$ complex adder/subtractors, and $\log_4 N - 1$ complex multipliers. It also needs $2\log_2 N$ and $\log_2 N$ 2-to-1 complex multiplexers for the DAC unit realized with shift registers. In addition, one more complex 2:1 complex multiplexer is used for any stage that is associated with the trivial twiddle factor.

For the length- N DFT computation, Table 6.3 lists the hardware requirements for the pipelined implementations. Compared with other single-path pipelined structure reported previously, the proposed one uses the same num-

Table 6.3: Comparisons among different pipelined implementations.

Architecture	Multiplier	Adder	Data Stores
R2MDC [109]	$2(\log_4 N - 1)$	$4\log_4 N$	$3N/2 - 2$
R2SDF [111]	$2(\log_4 N - 1)$	$4\log_4 N$	$N - 1$
R4MDC [109]	$3(\log_4 N - 1)$	$8\log_4 N$	$5N/2 - 4$
R4SDF [110]	$\log_4 N - 1$	$8\log_4 N$	$N - 1$
R4SDC [112]	$\log_4 N - 1$	$3\log_4 N$	$2(N - 1)$
R2 ² SDF [108]	$\log_4 N - 1$	$4\log_4 N$	$N - 1$
Proposed	$\log_4 N - 1$	$2\log_4 N$	$2(N - 1)$

ber of complex multipliers as those in [108, 112], and 67% and 50% of the number of complex adder/subtractors used in [112] and in [108], respectively (see Table 6.3).

The proposed structure has about the same processing latency as that in [108], i.e., $(N-1)$ word cycles + $(\log_2 N)T_a + (\log_4 N - 1)T_m$, where T_a and T_m are the latencies for complex adder/subtractors and multipliers, respectively.

6.3 Algorithms and Pipeline Structures for PTFT

6.3.1 Algorithms

Fast algorithms for computing the PTFT, such as radix-2 and radix-4, have been reported in Chapter 4. In this section, we focus on the new hardware-oriented algorithm development for a high speed pipelined implementation of the PTFT. The realization of the design should use minimum multipliers to reduce the hardware complexity. Because the signals in most practical applications follow the model of the second-order PPS, i.e., the linear frequency modulated or chirp signals, we consider the hardware structure for the second-order PTFT, which

is defined by

$$X(k, l) = \sum_{n=0}^{N-1} x(n) W_N^{kn+ln^2} \quad (6.3.1)$$

where $0 \leq k \leq N-1$, $0 \leq l \leq N-1$, $W_N = e^{-j2\pi/N}$, $N = 2^r$ and $r > 2$ is an integer.

By using the same principle of decomposition as the radix-2² FFT algorithm, we consider the first 2 steps of decomposition in radix-2 PTFT algorithm together. Applying a linear index map,

$$\begin{aligned} n &= \left\langle \frac{N}{2}n_1 + \frac{N}{4}n_2 + n_3 \right\rangle_N \\ k &= \left\langle k_1 + 2k_2 + 4k_3 \right\rangle_N \\ l &= \left\langle l_1 + 2l_2 + 4l_3 \right\rangle_N \end{aligned} \quad (6.3.2)$$

where $n_1 = \{0, 1\}$, $n_2 = \{0, 1\}$, $n_3 = 0, 1, \dots, N/4 - 1$, $k_1 = \{0, 1\}$, $k_2 = \{0, 1\}$, $k_3 = 0, 1, \dots, N/4 - 1$, $l_1 = \{0, 1\}$, $l_2 = \{0, 1\}$, $l_3 = 0, 1, \dots, N/4 - 1$. The PTFT takes the form of

$$\begin{aligned} &X(k_1 + 2k_2 + 4k_3, l_1 + 2l_2 + 4l_3) \\ &= \sum_{n_3=0}^{N/4-1} \sum_{n_2=0}^1 \sum_{n_1=0}^1 x\left(\frac{N}{2}n_1 + \frac{N}{4}n_2 + n_3\right) \\ &\quad W_N^{(k_1+2k_2+4k_3)(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3)+(l_1+2l_2+4l_3)(\frac{N}{2}n_1+\frac{N}{4}n_2+n_3)^2} \end{aligned} \quad (6.3.3)$$

Expanding the summation with regard to index n_1 , and using

$$W_N^{(k_1+2k_2+4k_3)(\frac{N}{2}+\frac{N}{4}n_2+n_3)} = (-1)^{k_1} W_N^{(k_1+2k_2+4k_3)(\frac{N}{4}n_2+n_3)} \quad (6.3.4)$$

and

$$W_N^{(l_1+2l_2+4l_3)(\frac{N}{2}+\frac{N}{4}n_2+n_3)^2} = W_N^{(l_1+2l_2+4l_3)(\frac{N}{4}n_2+n_3)^2}, \quad (6.3.5)$$

we have

$$\begin{aligned} & X(k_1 + 2k_2 + 4k_3, l_1 + 2l_2 + 4l_3) \\ &= \sum_{n_3=0}^{N/4-1} \sum_{n_2=0}^1 B_{N/2}\left(\frac{N}{4}n_2 + n_3, k_1\right) W_N^{(k_1+2k_2+4k_3)(\frac{N}{4}n_2+n_3)+(l_1+2l_2+4l_3)(\frac{N}{4}n_2+n_3)^2} \end{aligned} \quad (6.3.6)$$

where

$$B_{N/2}\left(\frac{N}{4}n_2 + n_3, k_1\right) = x\left(\frac{N}{4}n_2 + n_3\right) + (-1)^{k_1}x\left(\frac{N}{2} + \frac{N}{4}n_2 + n_3\right). \quad (6.3.7)$$

Expanding the summation with regard to index n_2 , and using

$$W_N^{(k_1+2k_2+4k_3)n_3+(l_1+2l_2+4l_3)n_3^2} = W_N^{(k_1+2k_2)n_3+(l_1+2l_2)n_3^2} W_{N/4}^{k_3n_3+l_3n_3^2} \quad (6.3.8)$$

$$W_N^{(k_1+2k_2+4k_3)(\frac{N}{4}+n_3)} = (-j)^{k_1+2k_2} W_N^{(k_1+2k_2)n_3} W_{N/4}^{k_3n_3} \quad (6.3.9)$$

and

$$\begin{aligned} & W_N^{(l_1+2l_2+4l_3)(\frac{N}{4}+n_3)^2} e^{-j2\pi\frac{N}{16}(l_1+2l_2+4l_3)} \\ &= e^{-j2\pi\frac{N}{16}l_1} (-1)^{l_1n_3} W_N^{(l_1+2l_2)n_3^2} W_{N/4}^{l_3n_3^2} \\ &= \begin{cases} (-1)^{l_1(1+n_3)} W_N^{(l_1+2l_2)n_3^2} W_{N/4}^{l_3n_3^2} & N = 8 \\ (-1)^{l_1n_3} W_N^{(l_1+2l_2)n_3^2} W_{N/4}^{l_3n_3^2} & N \geq 16 \end{cases}, \end{aligned} \quad (6.3.10)$$

we obtain

$$\begin{aligned} & X(k_1 + 2k_2 + 4k_3, l_1 + 2l_2 + 4l_3) \\ &= \sum_{n_3=0}^{N/4-1} \left[B_{N/4}(n_3, k_1, k_2, l_1) W_N^{(k_1+2k_2)n_3+(l_1+2l_2)n_3^2} \right] W_{N/4}^{k_3n_3+l_3n_3^2} \end{aligned} \quad (6.3.11)$$

where

$$\begin{aligned} & B_{N/4}(n_3, k_1, k_2, l_1) \\ &= \begin{cases} B_{N/2}(n_3, k_1) + (-1)^{l_1(n_3+1)} (-j)^{k_1+2k_2} B_{N/2}\left(\frac{N}{4} + n_3, k_1\right) & N = 8 \\ B_{N/2}(n_3, k_1) + (-1)^{l_1n_3} (-j)^{k_1+2k_2} B_{N/2}\left(\frac{N}{4} + n_3, k_1\right) & N \geq 16 \end{cases}. \end{aligned} \quad (6.3.12)$$

Consider all the combinations of $\{k_1, k_2, l_1, l_2\}$, (6.3.11) indicates that the PTFT of length- N input sequence is decomposed into 16 PTFTs of length $N/4$. The decomposition procedure repeats until PTFTs of length-2 or length-4 remain. As an example, the signal flow graph of the radix- 2^2 fast algorithm for a second order PTFT of length-16 input sequence is shown in Figure 6.5. The first and second stages implement the first two columns of butterflies represented by (6.3.7) and (6.3.12), respectively, wherein only trivial multiplications are required. The non-trivial twiddle factors are needed at the output of stage 2, and then a set of PTFTs of length-4 input sequences are to be computed.

Now, let us analyze the computational complexity of the above derived algorithm. It can be easily seen that each of the first two butterfly structures, represented by (6.3.7) and (6.3.12), respectively, requires N complex additions/subtractions. Full multiplications are required to apply the decomposed twiddle factor $W_N^{(k_1+2k_2)n_3+(l_1+2l_2)n_3^2}$ in (6.3.11). Therefore, the numbers of complex additions and multiplications required by the radix- 2^2 algorithms can be expressed by

$$A(N) = 16A\left(\frac{N}{4}\right) + 2N \quad (6.3.13)$$

$$M(N) = 16M\left(\frac{N}{4}\right) + 15\left(\frac{N}{4} - 1\right). \quad (6.3.14)$$

These recursive equations are the same as those of the radix-4 PTFT algorithm, i.e., the radix- 2^2 algorithm has the same computational complexity as radix-4 algorithm.

Figure 6.5: Signal flow graph of the radix-2² fast algorithm for PTFT of length-16 input sequence.

6.3.2 Pipelined Structure

Based on the radix-2² fast algorithm, the PTFT can be computed by a cascaded pipelined hardware structure. For input sequence of length- N , this structure contains $\log_2 N$ cascaded stages. Each stage contains a delay and commutator (DAC) unit and a controllable adder/subtractor unit. The DAC unit temporarily stores input data and distributes them in a proper order to be processed by the complex adder/subtractor. At the outputs of every other stage (except the last one), another type of delay and commutator unit (DACII) is required to arrange the order of data, and a complex multiplier is used for the non-trivial twiddle factors. Note that the implementation of the adder/subtractor unit of the last stage is different from that of other stage. As an illustration, Figure 6.6 shows the block diagram of the pipelined structure for PTFT of length-16 input sequence. There are four stages, each stage (except the last one) produces one output within each word cycle, and the last stage produces two outputs within one word cycle. It should be stressed that for length- N input sequence, contiguous data blocks can be processed by FFT processor (as discussed in Section 6.2), but for the 2nd order PTFT, a new block of data cannot enter the processor until the former PTFT computation is completed, which is due to the introduction of the DACII unit placed at the output of stage 2. The functionalities and implementations of the building units used in the stages are discussed in details in the following sections.

6.3.2.1 Delay and Commutator Units

There are two different DAC structures used in the implementation. They are used to store and distribute the data required by the controllable adder/subtractor

Figure 6.6: Block diagram of the radix- 2^2 fast algorithm for PTFT of length-16 input sequence.

within each stage and after every other stage, respectively.

The function of the first type of DAC unit is to break the data entering the i th stage into two parallel data streams flowing forward, with $N_i/2$ being the distance of indices between the data elements entering the adder/subtractor, where $N_i = N/2^i$, $1 \leq i \leq \log_2 N$ is the stage index. As discussed in Section 6.2.3.1, this DAC can be implemented with either shift registers or RAMs, and the control signals for the DAC operations at stage i should be designed accordingly. The data required by the implementation of the first butterfly can be arranged using this type of DAC. Let us consider the arrangement of the data required by the second butterfly. The data streams entering the adder/subtractor to implement $B_{N/4}(n_3, k_1, k_2, 0)$ for $n_3 = 0, 1, \dots, N/4 - 1$ are $B_{N/2}(n_3, k_1)$ and $B_{N/2}(n_3 + N/4, k_1)$, and they can be arranged by using the above mentioned DAC unit. Examining the expression (6.3.12), we can find that $B_{N/4}(n_3, k_1, k_2, 1)$ can be obtained from the elements of $B_{N/4}(n_3, k_1, k_2, 0)$ depending on the values of k_1, k_2 and the parity of n_3 . For example, when $N \geq 16$, $B_{N/4}(n_3, 0, 0, 1)$ can be obtained from $B_{N/4}(n_3, 0, 0, 0)$ for even n_3 and

Figure 6.7: Block diagram of the second type of delay and commutator unit (DACII).

Figure 6.8: Block diagram of the index reorder unit.

Figure 6.9: Operations of the second type of delay and commutator unit (DACII).

Figure 6.10: Operations of the index reorder unit after stage 2 for $N = 16$.

$B_{N/4}(n_3, 0, 1, 0)$ for odd n_3 .

This observation leads to the design of the second type of DAC unit (DACII). First, we consider the implementation with shift registers. As shown in Figure 6.7(a), the DACII unit consists of shift registers, an index reorder unit, two 4-to-1 multiplexers and a 1-to-4 demultiplexer. Figure 6.8 depicts the block diagram of the index reorder unit and its operations will be explained in details later. Here we make a few remarks about this unit. It requires $2N_i$ delay elements, has a processing latency of N_i word cycles, and the switches are opened at the same rate as input data rate.

Figure 6.9 shows the operations performed by the DACII unit, where the input sequence and the sequence after passing through the index reorder unit are denoted as $y(n)$ and $y'(n)$, respectively, where $n = 0, 1, \dots, N_{i-2} - 1$. Let us define the following operation:

if $\log_2 N$ is even

$$X = N_i^2, Y = N_i(N_i/4 - 1);$$

else

$$X = 2N_i^2, Y = N_i(N_i/2 - 1). \quad (6.3.15)$$

In the first N_{i-2} word cycles, the control signals are set as $c_0c_1 = 00$, thus the input, $y(0 : N_{i-2} - 1)$, is connected directly to the output of the first 4-to-1 multiplexer. The input is also shifted into the length- X shift register. In the next X word cycles, the control signals are set as $c_0c_1 = 01$, i.e., the output of the first 4-to-1 multiplexer equals the output of the register. From the X -th word cycles, $y(0 : N_{i-2} - 1)$ is shifted out sequentially and appears at the output of the first 4-to-1 multiplexer and the input of the index reorder unit. In the following X word cycles, the control signals are set as $c_0c_1 = 10$. As

mentioned before, the index reorder unit has a latency of N_i word cycles, and the reordered sequence, $y'(n)$, is connected to a length- $(X - N_i)$ shift register. Then from the $2X$ -th word cycle, $y'(0 : N_{i-2} - 1)$ appears as the output of the first 4-to-1 multiplexer. The delayed $y'(n)$ is passing through a length- X shift register too. In the last X word cycles, the control signals are set as $c_0c_1 = 11$. Thus, $y'(0 : N_{i-2} - 1)$ appears at the output of the first 4-to-1 multiplexer from the $3X$ -th word cycle. To analyze the remaining part of DACII unit, we show the operations in the first X word cycles and the corresponding control signals in Figure 6.9. The input sequence is broken into 4 length- N_i sequences separated by a distance of $Y + N_i$. The operations in the following $3X$ word cycles are similar to those in the first X word cycles.

Taking an example for $N = 16$, the DACII is placed at the output of stage 2, where $N_2 = 4$. Since $N_2(N_2/4 - 1) = 0$, the control signals c_4 and c_5 are identical to c_2 and c_3 , respectively. Therefore, the parts from the 1-to-4 demultiplexer to the second 4-to-1 multiplexer just pass the output of the first 4-to-1 multiplexer to the final output. The input of DACII is denoted as $x_2(0 : 15)$ with $x_2(0 : 3) = B_4(0 : 3, 0, 0, 0)$, $x_2(4 : 7) = B_4(0 : 3, 0, 1, 0)$, $x_2(8 : 11) = B_4(0 : 3, 1, 0, 0)$ and $x_2(12 : 15) = B_4(0 : 3, 1, 1, 0)$. Figure 6.10 explains the operations performed by the index reorder unit. The presented concepts can be easily generalized for the operations of other stages.

Denote the output of index reorder unit as $x'_2(n)$, Table 6.4 shows the input and output data sequences of the DACII unit. In the first 16 word cycles, i.e., $n = 12$ to 27, the 4-to-1 multiplier is put at position 0, thus the input, $x_2(0), x_2(1), \dots$ and $x_2(15)$, are sequentially available at the unit output, and the input is also shifted into the length-16 shift register with $x_2(0)$ available as

Table 6.4: Operations of the DACII unit after stage 2 for $N = 16$.

word cycle	12 $\dots$ 27	28 $\dots$ 43	44 $\dots$ 59	60 $\dots$ 75
Input	$x_2(0) \dots x_2(15)$			
control $c_0 c_1$	0 0	0 1	10	1 1
Output	$x_2(0) \dots x_2(15)$	$x_2(0) \dots x_2(15)$	$x'_2(0) \dots x'_2(15)$	$x'_2(0) \dots x'_2(15)$

its output. In the next 16 word cycles, i.e., $n = 28$ to 43, the 4-to-1 multiplier is put at position 1, so $x_2(0), x_2(1), \dots$ and $x_2(15)$ are sequentially shifted out as the unit output, and they are sequentially shifted into the index reorder unit too. Therefor, it is available at the DACII unit output during word cycles $n = 44$ to 59, since the 4-to-1 multiplier is put at position 2. Besides, it is shifted into a length-16 shift register. Thus, in the next 16 word cycles, i.e., $n = 60$ to 75, at that time the 4-to-1 multiplier is put at position 3, $x'_2(0)$ to $x'_2(15)$ appears at the DACII unit output sequentially.

As mentioned in Section 6.2.3.1, when the required memory size is large, the implementation of the DAC unit using shift registers may require a large chip area and consume more power since the data are actively shifted all the time. Besides, the above analysis shows that many shift registers are just used to provide the correct timing. One alternative method is to use RAMs to implement the DAC unit. Figure 6.7(b) shows the corresponding block diagram of the implementation with RAMs. The input signal is written into RAM1 and fed into the index reorder unit simultaneously, and the output of index reorder unit is written into RAM2 immediately. The appropriate portion of the data in the RAMs is read out by passing the corresponding memory address, and the control signal $c_0 c_1$ is designed to obtain the result shown in Figure 6.9. Compared with the implementation using shift registers, using RAMs requires much less memory. Note that the index reorder unit within Figure 6.7(b) can

be eliminated to achieve further memory saving if we realize its function by a memory address controller, but it will increase the complexity.

6.3.2.2 Controllable Adder/Subtractor Units

Let us consider the implementation of butterfly computation. The operations required by the controllable adder/subtractor are listed in Table 6.5. The im-

Table 6.5: Operations required by the controllable adder/subtractor.

... Add/Sub	the last stage	other stage
+	$x_1 + x_2$ and $x_1 - jx_2$	$x_1 + x_2$
-	$x_1 - x_2$ and $x_1 + jx_2$	$x_1 - x_2$ or $-j(x_1 - x_2)$

plementation of the adder/subtractor in other stage except the last one was discussed in Section 6.2.3.2. In the last stage, to implement the required operation within one word cycle, a modified adder/subtractor unit is devised. Figure 6.11 shows its block diagram consisting of two complex adder/subtractors and one NOT gate.

6.3.3 Discussions

For the PTFT of length- N input sequence, the proposed pipelined structure has a latency of $N - 1$ word cycles. When $\log_2 N$ is even, the presented structure is able to compute the PTFT within $N^2/4$ word cycles, and when $\log_2 N$ is odd, the computation can be completed within $N^2/2$ word cycles. It can be easily calculated that the structure needs $\log_2 N + 1$ complex adder/subtractors, and the number of required complex multipliers is $\log_4 N - 1$ when $\log_2 N$ is even and $(\log_2 N - 1)/2$ when $\log_2 N$ is odd. The memory space required by all the

Figure 6.11: Block diagram of the second type of modified adder/subtractor unit (MASII).

DAC units is

$$\sum_{i=1}^{\log_2 N} 2N_i = 2(N-1).$$

It also needs $2 \log_2 N$ and $\log_2 N$ 2-to-1 complex multiplexers for the DAC unit realized with shift registers and RAMs, respectively. For the DACII unit implemented with shift registers, when $\log_2 N$ is even, the memory requirement for all the DACII units is

$$\sum_{t=1}^{\log_4 N-1} \left(\frac{9}{2} N_{2t}^2 - 5 N_{2t} \right) = \frac{9N^2 - 50N + 56}{30},$$

and when $\log_2 N$ is odd, it is

$$\sum_{t=1}^{(\log_2 N-1)/2} (9N_{2t}^2 - 5N_{2t}) = \frac{9N^2 - 25N + 14}{15}.$$

In addition, 2 4-to-1 multiplexers, 1 1-to-4 demultiplexer, 1 1-to-2 multiplexer and 1 1-to-2 demultiplexer are used for each DACII unit. If the DACII unit is

implemented using RAMs, the memory requirement for all the units is

$$\sum_{t=1}^{\log_4 N-1} 10N_{2t} = \frac{10N-40}{3}$$

when $\log_2 N$ is even, and

$$\sum_{t=1}^{(\log_2 N-1)/2} 10N_{2t} = \frac{10N-20}{3}$$

when $\log_2 N$ is odd. Besides, 1 4-to-1 multiplexer, 1 1-to-2 multiplexer and 1 1-to-2 demultiplexer are needed in each DACII unit. Furthermore, the adder/subtractor in each stage (except the last one) requires 4 2-to-1 multiplexers, while in the last stage, it requires 1 NOT gate.

It should be mentioned that the introduction of the DAC and DACII units definitely increases the chip area or hardware requirements, but the delays are important to achieve synchronization between various components for sequential input data streams. When the number of samples is large, it is generally appropriate to implement the units using RAMs to minimize the hardware requirements, but further considerations are needed when a specific application is dealt with. In addition, if PTFT is implemented using the FFT based approach, N FFTs of input $\{x_l(n) = x(n)W_N^{ln^2}, 0 \leq l \leq N-1\}$ are required. Since each FFT requires at least $(N-1)$ data stores (see Table 6.3), the total number of data stores required is $N(N-1)$. Compared with the memory requirement of proposed approach, FFT-based approach requires much more memory space. Thus, the proposed approach is more efficient than the FFT based approach.

6.4 Summary

Based on the radix-2² fast algorithm, this chapter presents pipelined structures for the computation of DFTs and PTFTs that have sequentially presented input and output data streams. The concepts are simple and the structure is regular and easily extendable to any number of stages.

Compared with previously reported structures for the computation of DFTs, the proposed structure reaches the minimum requirement for both the number of complex adders/subtractors and the complex multipliers, which is particularly attractive for applications that directly deal with high speed sequentially presented data streams in real-time.

A hardware-oriented radix-2² algorithm for the computation of second order PTFT is then proposed, the pipelined structure for the implementation is introduced, and the functionalities and the implementation of the building blocks are discussed in details. The analysis of hardware requirement shows that the proposed architecture is simple and can be easily realized by $\log_2 N + 1$ complex adders/subtractors and $\log_2 N/2 - 1$ or $(\log_2 N - 1)/2$ complex multipliers when $\log_2 N$ is even or odd respectively, where N is the length of input sequence.

Chapter 7

Signal Processing for OFDM

7.1 Introduction

In recent years, orthogonal frequency division multiplexing (OFDM) has become popular for digital communication systems. It has been adopted or considered as a standard for digital audio/video broadcasting (DAB/DVB) and wireless local/metropolitan area networks (WLAN/WMAN) such as IEEE 802.11a/g and WiMAX. The principle of OFDM is to divide the data to be transmitted into several bit streams of a much lower bit rate to modulate a number of orthogonal subcarriers. The primary advantages of OFDM include its efficient implementation using FFT techniques and its ability to combat the effects of channel impairments, such as intersymbol interference (ISI) and fading caused by multipath propagation. A more in-depth tutorial of OFDM systems can be found in [107].

One particular problem of OFDM, which is often cited as its major drawback, is that it is orders of magnitude more sensitive to carrier frequency offset

(CFO) than single carrier system [115]. Without proper compensation, the system performance would be severely degraded (the effect of CFO on the system performance will be explained in more details in the next section). The reported CFO estimators can be generally divided into the categories of data-aided methods and blind methods. Data-aided estimators [115–119] rely on the periodical transmission of known data blocks or the embedded pilots in every OFDM symbol. Blind estimators utilize the inherent structure of OFDM signal provided by cyclic prefix (CP) [120] or virtual subcarriers [121] to compensate CFO.

The essential requirements are to obtain low mean square errors (MSEs) of the CFO estimation within a wide range of variations. It has been reported that the use of multiple lags is able to achieve substantial performance improvement [119, 122]. Furthermore, the estimators should be resistant to the multipath effects of the channels. Effective methods were reported in [119] to use one training symbol consisting of multiple small identical training parts to achieve adjustable acquisition range and minimize the multipath effects. Efficient frequency estimation methods were also proposed by using the pulse-pair method at various lags in conjunction with the DFT. Comparisons were also made on performances and the computational complexities required by these methods [122].

This chapter focuses on a data-aided estimation scheme for packet transmission OFDM systems. Inspired by a novel approach using the co-prime lags for high-order ambiguity function (HAF) in [42], a CFO estimator is constructed by using a lag diversity method to solve the frequency ambiguity resolution problem. Compared to other reported estimators in [118, 119], the proposed one achieves wider ranges of improved performance in terms of mean square

errors (MSEs).

The rest of the chapter is organized as follows. The next section discusses the OFDM signal model and explains the effect of CFO on system performance. In Section 7.3, a general form of data-aided CFO estimation method is presented and the corresponding Cramer-Rao lower bound (CRLB) of the estimator is given. The proposed method is described in Section 7.4 and simulation results are shown in Section 7.5. Finally, Section 7.6 concludes the chapter.

7.2 Signal Model

Let us consider a discrete-time baseband OFDM system with N subcarriers, and define $X(k)$ as a symbol of the quadrature amplitude modulation or phase shift keying constellation on the k th subcarrier at the transmitter. The OFDM signal is generated by the inverse discrete Fourier transform of $\{X(k)\}$ and preceded by a CP. The received symbols, after passing through a channel and the CP being removed, are processed by a discrete Fourier transform operation. With the assumption that perfect timing synchronization is achieved, the system performance is only affected by the CFO and the channel. The received signal in the time domain is expressed as

$$r(n) = e^{j2\pi f_{\Delta} n T_s} s(n) + w(n), \quad (7.2.1)$$

where f_{Δ} denotes the CFO, $T_s = T/N$ is the sampling period, T is the OFDM symbol period, $w(n)$ is the complex additive white Gaussian noise (AWGN) with zero mean and variance σ_n^2 , and $s(n)$ is the transmitted signal given by

$$s(n) = \sum_{k=0}^{N-1} H(k) X(k) e^{j2\pi n k / N}, \quad (7.2.2)$$

where $H(k)$ is the channel frequency response associated with the k th subcarrier and the signal-to-noise ratio (SNR) is defined as σ_s^2/σ_n^2 with $\sigma_s^2 = E\{|s(n)|^2\}$. If ν is defined as the CFO normalized to subcarrier spacing $f_{sub} = 1/T$, i.e., $\nu = f_\Delta T$, we have

$$\nu = m + \varepsilon \quad (7.2.3)$$

where $m \in \mathbb{Z}$ is the integer CFO and $\varepsilon \in [-0.5, 0.5)$ denotes the fractional CFO. Now, let us consider their effects on the received signal. The l th subcarrier signal in the frequency domain can be written as [115, 116]

$$\begin{aligned} Y(l) &= \sum_{n=0}^{N-1} r(n) e^{-j2\pi nl/N} \\ &= \frac{\sin(\pi\varepsilon)}{\sin(\frac{\pi\varepsilon}{N})} e^{j\pi\varepsilon(1-1/N)} H(\tilde{l}) X(\tilde{l}) \\ &\quad + \sum_{k=0, k \neq \tilde{l}}^{N-1} H(k) X(k) \frac{\sin(\pi\varepsilon)}{\sin(\frac{\pi(\varepsilon+k-\tilde{l})}{N})} e^{j\pi\varepsilon(1-1/N)} e^{-j\pi(k-\tilde{l})/N} \\ &\quad + \sum_{n=0}^{N-1} w(n) e^{-j2\pi nl/N} \end{aligned} \quad (7.2.4)$$

where $\tilde{l} = \text{mod}(l - m, N)$. The expression of $Y(l)$ consists of three terms: the first term is the desired signal component, the second term denotes the inter-channel interference (ICI) produced by the CFO, and the last term represents the contribution of AWGN. It can be seen that the desired signal component experiences an amplitude reduction and phase shift due to the fractional CFO ε , and if the integer CFO $m \neq 0$, the signal on the l th subcarrier is circularly shifted to the $\tilde{l}$ th subcarrier. Statistical evaluations of SNR degradation due to CFO can be found in [123, 124].

7.3 Data-aided CFO Estimation

For the data-aided CFO estimation methods, the required training information contains at least two identical parts available in the training-symbol-block. Because the CFO produces phase shift between these identical parts, the CFO estimation is considered as a phase estimation. Let τ denotes the delay samples between two identical parts of the training-symbol-block. Based on the observations

$$\begin{aligned} r_n &= e^{j2\pi f_\Delta n T_s} s_n + w_n \\ r_{n+\tau} &= e^{j2\pi f_\Delta (n+\tau) T_s} s_n + w_{n+\tau}, \end{aligned} \quad (7.3.1)$$

a correlation function is defined as

$$P_\tau = \sum_{n=0}^{L-1} r_n^* r_{n+\tau} = e^{j2\pi f_\Delta T_s \tau} \sum_{n=0}^{L-1} |s_n|^2 + \xi_w \quad (7.3.2)$$

where L is the total number of samples used in the summation and ξ_w is

$$\xi_w = \sum_{n=0}^{L-1} [w_n^* s_{n+\tau} e^{j2\pi f_\Delta (n+\tau) T_s} + s_n^* e^{-j2\pi f_\Delta n T_s} w_{n+\tau} + w_n^* w_{n+\tau}]. \quad (7.3.3)$$

In (7.3.2), the phase shift between two identical samples is $\phi_\tau = 2\pi f_\Delta T_s \tau$, and the CFO estimator is formed as

$$\hat{f}_\Delta = \frac{\hat{\phi}_\tau}{2\pi T_s \tau}. \quad (7.3.4)$$

Since $\hat{\phi}_\tau$ can only be unambiguously in the range $[-\pi, \pi)$, the range for the CFO estimate to be accurate is

$$|\hat{f}_\Delta| < \frac{1}{2T_s \tau}, \quad (7.3.5)$$

The performance of CFO estimation given by (7.3.4) was previously studied for frequency estimation in [122] and was called the pulse-pair (PP) method there.

Based on [122, equations(1)-(3)], the estimate $\hat{\nu}$ is asymptotically unbiased, its range and variance are given by

$$|\nu| < \frac{0.5}{\tau/N}, \quad (7.3.6)$$

and

$$\text{var}(\hat{\nu}) = \frac{1}{(2\pi\tau/N)^2 L \text{SNR}} \left(\frac{\min(\tau, L)}{L} + \frac{1}{2 \text{SNR}} \right). \quad (7.3.7)$$

Clearly, both the acquisition range and its variance are enlarged by selecting smaller value of τ , which leads to a tradeoff on the choice of τ . By using the method is [117], the CRLB on the variance of the estimate of normalized CFO can be found to be

$$\text{var}(\hat{\nu}) \geq \frac{1}{(2\pi\tau/N)^2 L \text{SNR}} \left(\frac{\min(\tau, L)}{L} + \frac{1}{2 \text{SNR}} \right). \quad (7.3.8)$$

Follow the derivations in Appendix of [116], it can be easily verified that using the observations (7.3.1) the CFO estimator based on (7.3.4) provides maximum likelihood estimate (MLE) of f_{Δ} . This fact explains why the CRLB is met by the estimator.

In [116], Moose estimated the CFO based on the observation of two consecutive and identical OFDM symbols. The tracking range and variance of this estimator can be obtained from (7.3.6) and (7.3.8) by setting $\tau = L = N$. To enlarge the estimation range, Schmidl and Cox estimator (SCE) and two other ones, known as the pilot assisted estimator (PAE) and the blind estimator (BE), respectively, were proposed [117, 118]. They all employed two OFDM training symbols and N_P pilot symbols that were inserted at known locations in each OFDM symbol. These pilot symbols satisfy the relation

$$c_{n,0}c_{n,1}^* = d_n \quad n \in \mathcal{P} \quad (7.3.9)$$

where $*$ is the complex conjugate operator and $\{d_n, n \in \mathcal{P}\}$ is a pseudo noise sequence known at the receiver and $\mathcal{P}$ is the set of the pilot locations. The first training symbol has two identical parts and serves to estimate the CFO with an ambiguity equal to the subcarrier spacing and the pilot symbols are explored to resolve the ambiguity. Other estimators, such as the estimator proposed in [119], employed only one OFDM training symbol with the assumption that the training symbol consisted of $K = 2Q$ identical parts, where $Q > 1$ is an integer. The estimation performance depends on both parameters K and h , which are related to τ and L via

$$\tau = \frac{hN}{K}, \quad L = N - \tau, \quad h \in [1, K - 1]. \quad (7.3.10)$$

Substituting (7.3.10) into (7.3.6) and (7.3.8), the tracking range and CRLB on the variance of estimate of normalized CFO become $|\hat{\nu}| < K/2h$ and

$$\begin{aligned} \text{var}(\hat{\nu}) \geq & \frac{1}{N(2\pi h/K)^2(1 - h/K) \text{SNR}} \\ & \times \left(\frac{\min(h/K, 1 - h/K)}{1 - h/K} + \frac{1}{2 \text{SNR}} \right). \end{aligned} \quad (7.3.11)$$

It is noted that the above CRLB is different from that in [119] because the term $\left(\frac{\min(h/K, 1 - h/K)}{1 - h/K} + \frac{1}{2 \text{SNR}} \right)$ is absent. It will be shown later that the ignorance of this term results in serious estimation errors.

7.4 CFO Estimation with Lag Diversity

As discussed in the last section, the principle of data-aided CFO estimators is to transform the received signal to obtain a component whose phase is directly related to the CFO. In fact, similar structure also exists in the polynomial-phase signal (PPS) phase coefficient estimation based on the HAF, where the signal

is nonlinearly transformed to a sinusoid at a certain frequency that is directly related to the phase coefficient of the PPS. It is well known that the phase and frequency (normalized to the sampling frequency) can only be unambiguous in the range of $[-\pi, \pi)$ and $[-0.5, 0.5)$, respectively. Thus, the ranges of CFO and PPS phase parameter are limited.

Let us consider an effective estimation method motivated by the lag diversity approach in [42]. Our goal is to enlarge the CFO estimation range without deteriorating the MSE performance. It should be noted that the algorithm reported in [42] uses two co-prime lags. The method presented below removes this restriction. When $f_{\Delta}T_s\tau$ is not within the range of $[-0.5, 0.5)$, as discussed above, the correlation, P_{τ} defined in (7.3.2), rotates a phase larger than π . It means that the CFO is actually estimated from $2\pi f_{\Delta}T_s\tau = 2\pi k + \eta$ for $k \in \mathbb{Z}$ and $\eta \in [-\pi, \pi)$. It is assumed that the estimates are obtained from two lags τ_1 and τ_2 according to the following equations

$$\begin{aligned} 2\pi f_{\Delta}T_s\tau_1 &= 2\pi k_1 + \eta_1 \\ 2\pi f_{\Delta}T_s\tau_2 &= 2\pi k_2 + \eta_2, \end{aligned} \tag{7.4.1}$$

where η_i is the angle of P_{τ_i} for $i = 1$ and 2 . By defining $\gcd(\tau_1, \tau_2)$ as the greatest common divisor of τ_1 and τ_2 , $\tau'_i = \tau_i / \gcd(\tau_1, \tau_2)$, $\eta'_i = \eta_i / (2\pi)$ for $i = 1$ and 2 , and manipulating (7.4.1), we have

$$k_2\tau'_1 - k_1\tau'_2 = \eta'_1\tau'_2 - \eta'_2\tau'_1 \triangleq M, \tag{7.4.2}$$

where M is theoretically an integer. Due to the effect of noise in practice, however, M has to be rounded to its nearest integer value. With the Euclidean

algorithm, it is well known that the solution to (7.4.2) are

$$\begin{aligned} k_1 &= n_1 M + l \tau'_1 \\ k_2 &= n_2 M + l \tau'_2 \end{aligned} \quad (7.4.3)$$

where $l \in \mathbb{Z}$, n_1 and n_2 are two integers satisfying the condition

$$n_2 \tau'_1 - n_1 \tau'_2 = 1. \quad (7.4.4)$$

It is easily seen that k_1 increases by τ'_1 if l increases by one. Thus, there exists a unique k_1 satisfying $-\pi \tau'_1 \leq 2\pi k_1 + \eta_1 < \pi \tau'_1$. Denote this k_1 as k_1^* , the solution of the inequality can be easily found to be

$$k_1^* = n_1 M + \lfloor 0.5 - (n_1 M + \eta'_1) / \tau'_1 \rfloor \tau'_1 \quad (7.4.5)$$

where $\lfloor x \rfloor$ returns the integer smaller than or equal to x . Putting k_1^* into (7.4.1), the CFO estimate is obtained from

$$\hat{f}_\Delta = \frac{k_1^* + \eta'_1}{\tau_1 T_s}. \quad (7.4.6)$$

The tracking range of this estimator is

$$|\hat{f}_\Delta| < \frac{\tau'_1}{2T_s \tau_1} = \frac{1}{2T_s \gcd(\tau_1, \tau_2)} \quad (7.4.7)$$

or

$$|\hat{\nu}| < \frac{N}{2 \gcd(\tau_1, \tau_2)}. \quad (7.4.8)$$

In summary, the steps of the estimation process using lag diversity are:

- (a) Compute η_1 and η_2 from P_{τ_1} and P_{τ_2} , respectively, and obtain the linear Diophantine equation in (7.4.2);

- (b) Find integers n_1 and n_2 such that (7.4.4) is satisfied using the Euclidean algorithm;
- (c) Find k_1^* satisfying $2\pi k_1 + \eta_1 \in [-\pi\tau'_1, \pi\tau'_1)$ according to (7.4.5),
- (d) Obtain the CFO estimate from (7.4.6).

Now, let us consider two special cases as follows.

Case (i): When τ_1 and τ_2 are co-prime, i.e., $\gcd(\tau_1, \tau_2) = 1$, the tracking range of the estimator is

$$|\hat{f}_\Delta| < \frac{1}{2T_s} \text{ or } |\hat{\nu}| < \frac{N}{2} \quad (7.4.9)$$

which is the maximum range of the CFO.

Case (ii): Consider the same training-symbol-block structure as that of [119], i.e., one training symbol with more than two identical parts. When $\tau_1 = h_1 \frac{N}{K}$, and $\tau_2 = h_2 \frac{N}{K}$ and $\gcd(\tau_1, \tau_2) = \gcd(h_1, h_2) \frac{N}{K}$, the tracking range becomes

$$|\hat{f}_\Delta| < \frac{K}{2NT_s \gcd(h_1, h_2)} \text{ or } |\hat{\nu}| < \frac{K}{2 \gcd(h_1, h_2)}. \quad (7.4.10)$$

If h_1 and h_2 are co-prime, the range of $|\hat{\nu}| < K/2$ is reached. While for the multiple- h estimator reported in [119], the estimation range is

$$|\hat{\nu}| < \frac{K}{2 \max(h_1, h_2)}. \quad (7.4.11)$$

Since $\gcd(h_1, h_2) < \max(h_1, h_2)$ for $h_1 \neq h_2$, the estimator based on lag diversity can guarantee wider estimation range.

7.5 Simulations

This section presents several simulation results to validate our analysis and illustrate the improvements achieved by the described method in terms of the

MSE of the normalized CFO estimate, i.e., $\hat{\nu}$. It is assumed that the OFDM system has $N = 64$ subcarriers and a guard interval of $L = 16$ samples. It is operated in either an AWGN or multipath channel environment. The maximum path delay is set to be shorter than the guard interval to eliminate the ISI. It is also assumed that the multipath channel has 16 paths with path delays $\tau_i = i$ samples. The path gains α_i , $0 \leq i \leq 15$, are independent complex Gaussian random variables and the amplitudes are of a Rayleigh distribution with an exponential power delay profile $E\{|\alpha_i|^2\} = \exp(-i/1.5)$. The sum of the power gains $\sum_{i=0}^{15} |\alpha_i|^2$ is normalized to 1. The CFO and the channel path gains are assumed to be constant during the period of the training-symbol-block.

The first simulation is to verify the CRLB derived in Section 7.3. It is assumed that the training-symbol-block has one OFDM symbol consisting of $K = 8$ identical parts with AWGN channels. Figure 7.1 shows the analytic results from (7.3.11) with dashed lines, the results from [119] with solid lines and the simulated ones with circle symbols. It is clearly seen that the simulation results closely agree with their theoretically counterparts given by (7.3.11). In contrast, the curves based on the results in [119] deviate significantly from (7.3.11) except when $h \geq 4$ with high SNRs. It is also observed that the minimum MSEs are obtained around $h = K/3$, and $h = 2K/3$, and the CRLB is symmetric around $h = 4$ when $\text{SNR} > 5$ dB.

The single- h estimator in [119] is simulated to show the ranges in which the lowest MSEs are achieved. With $K = 8$, the system is evaluated with an AWGN channel environment for $\text{SNR} = 5$ and 10 dB, respectively. Figure 7.2 shows the MSE performance of single- h estimator for $h = 1, 3$ and 5. It verifies that the single- h estimator has an estimation range of $\pm K/(2h)$. For $h = 1$, for example,

Figure 7.1: MSE versus h , $\nu = 0.15$ in AWGN channel.

the estimator has the widest estimation range in which the MSE is the largest compared with that for $h = 3$ and 5 .

Figure 7.3 illustrates the performance achieved by the estimator exploring the lag diversity from the same system setup as that used for Figure 7.2. Comparing with Figure 7.2, it is seen that the MSEs for SNR = 5 and 10 dB are reduced, respectively. When h_1 and h_2 are co-prime, the lag diversity method achieves the same tracking range as that shown in Figure 7.2 with $h = 1$. When $h > 1$, however, the tracking range of the single- h estimator is significantly smaller than that achieved with the lag diversity method, even compared with the case that h_1 and h_2 are not co-prime. In general, the estimator based on lag diversity method provides a lower MSE within a wider tracking range compared with that achieved by the single- h estimator.

We also compute the MSE as a function of SNR, for both AWGN and mul-

Figure 7.2: MSE of single- h estimator versus ν in AWGN channel

Figure 7.3: MSE of the proposed estimator versus ν in AWGN channel.

Figure 7.4: $\Pr(\hat{m} \neq m)$ of PAE versus SNR in AWGN and multipath channel.

Figure 7.5: Performance of the proposed estimator, the single- h estimator and the PAE in AWGN and multipath channels, $\nu = 3.25$.

multipath fading channels with $\nu = 3.25$, to compare the performances achieved by the single- h method, the lag-diversity method and the PAE method reported in [118]. For the first two methods, we assume the training-symbol-block has one OFDM symbol with $K = 8$ identical parts. The PAE requires two consecutive OFDM symbols in the training-symbol-block. The first one has two identical parts to estimate the fractional part of the CFO, ε , and the second symbol has 4 pilots satisfying (7.2.3) to decide the integer part of the CFO, which is searched within the range of $(-5, 5)$ in our simulation. To verify the implementation of the PAE method, the failure probabilities of the estimators, $\Pr(\hat{m} \neq m)$, under AWGN and multipath channels, are shown in Figure 7.4. It can be easily seen that the failure probability has an error floor at high SNRs in the multipath channel. Under AWGN channel it has no error floor and has much smaller values compared with the multipath case. These observations coincide with the numerical results shown in [118]. Next, the MSE performance is displayed in Figure 7.5 to show that both the single- h method and the lag diversity method achieve about the same performance for both AWGN and multipath channels. However, the PAE method generally achieves much higher MSEs than the single- h method, especially in multipath fading channels, and the estimation method based on lag diversity achieves the best MSE performance. The feature of the PAE method for estimating integer CFO is to search the maximum of the likelihood function $\Lambda(\tilde{m})$ for different values of $\tilde{m}$ in a predetermined set. The computation of the likelihood function requires two length- N FFTs, then for each value of $\tilde{m}$, circular shift operations and some other complex multiplications and additions are needed by $\Lambda(\tilde{m})$. Thus it has much more complexity than the other two algorithms.

7.6 Summary

This chapter considers the frequency ambiguity problem of CFO estimation in OFDM systems. In addition to providing a more accurate formula for a previously reported single- h estimation method, a lag diversity based method is also introduced to achieve improvements on both estimation accuracy and the range of the desired estimation performance. Comparisons among simulation results from different methods also confirm the advantages of the lag diversity based method.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

The research work carried out in this dissertation is about some aspects of analysis and processing of signals with time-varying frequencies and possible extensions of signal processing methods to communication systems.

The first stage of this study focuses on reviewing and understanding different signal analysis methods. It is shown that the classical time and frequency representations fail to fully characterize the time-varying spectral contents of such signals. This limitation motivates two categories of approaches to study signals with time-varying frequencies. Parametric methods mainly deal with the signals' phase using PPS model and aims at estimating the phase coefficients through ML method based on PTFT or suboptimal methods like HAF, PHAF etc. Non-parametric methods map one-dimensional time-domain signals to two-dimensional or multi-dimensional function to display the temporal localization of the signal's spectral contents. The characteristics of these methods

are discussed and their performances are compared. It has been shown that the choice of the best method for signal analysis depends on the particular problem at hand.

Another goal of the first stage of the study is addressed in Chapter 3 where we have seen that the performance of the standard signal transforms and TFRs deteriorates when impulsive noise is involved. To address this problem, we have derived generalized forms of robust signal transforms and TFRs based on the M -estimation. With the appropriate basis function (for signal transform) or the kernel and exponential functions (for TFR), the corresponding robust signal transform or robust TFR can be easily obtained. Iterative processes are proposed for the implementation of the M -estimation based robust forms. Uniqueness and convergence of the iterative procedure are also analyzed. Then we review the general forms of robust signal transforms and TFRs based on the L -estimation. It is known that clipping is a classical tool to deal with impulsive noise. We also propose a method to automatically determine the level of clipping according to the characteristics of the signal samples. Finally, comparison is made among the above-mentioned methods, and the performance results are presented in terms of visual plots, statistical measures and the required computation time.

The study in the first stage illustrates the statistical advantages of PTFT over suboptimal methods from the aspects of estimates' variance, and output SNRs. Moreover, in some cases we have no prior knowledge about the signal's model or the signal is hard to be described using a particular model. In these cases, the TFR named MLPTFT, which combines PTFT with LPTFT, has been shown to be an appropriate signal analysis tool. For the study of signals

with time-varying frequencies in impulsive signals, an efficient method based on clipping and standard signal transform is proposed. Compared with the M -estimation and L -estimation based methods, it can provide similar performance with greatly reduced computation complexity, especially when the standard transforms can be efficiently computed using fast algorithms. The drawback of PTFT is that it requires a huge amount of computational complexity. A major objective of this dissertation is to address this problem. In Chapter 4, a general class of fast algorithms for the computation of the PTFTs of complex-valued input is presented. The proposed algorithms consist two-step decomposition processes, the first step splits the PTFT into multiple PTFTs of the same dimensional sizes, and then the second step uses the symmetry and periodicity to further decompose the resulting PTFTs of the same dimensional sizes based on a particular radix or a blend of radices. The proposed algorithms provide the flexibility to support arbitrary order of PTFTs and various input sequence lengths $N = a^p \cdot b$ by setting the parameters of a , b and p . Furthermore, they have regular computational structures for an easy implementation. Analysis and comparison on the computational complexity are also reported in terms of the numbers of complex additions and complex multiplications. It is shown that with the use of symmetric property and the presented two-step decomposition processes, the computational complexity of the PTFTs is significantly reduced to support practical applications.

Chapter 5 develops fast algorithms to calculate the PTFT of real-valued sequences. A “conjugate and symmetric” property is discovered to further reduce the required computational complexity. Similar to the fast algorithms for complex-valued sequences, the proposed fast algorithms for real-valued sequences support arbitrary PTFT order, work for various input sequence length

$N = a^p \cdot b$, and have regular computational structures. Note that the proposed fast algorithms also utilize two-step decomposition processes, here the second step decomposes the PTFTs with the same dimensional sizes into PTFTs of both real and complex-valued inputs of smaller lengths. Thus, the whole computation is performed by jointly using fast algorithms for real-valued sequence and those for complex-valued sequences. Analysis of the computational complexity is made in terms of the numbers of real additions and real multiplications, and it is shown that the fast algorithms (except radix-2) for real-valued sequences require about one half of the computational complexity compared to the counterparts of complex-valued sequences.

Chapter 6 is dedicated to the hardware implementation of PTFT. A hardware-oriented radix-2² algorithm for the computation of the second order PTFT is proposed, the pipelined structure for the implementation is introduced, and the functionalities and the implementation of the building blocks are discussed in details.

The last goal of this research work is to apply certain signal processing method to communication systems. In Chapter 7, we consider the ambiguity problem of data-aided CFO estimator in the OFDM system. The principle of data-aided CFO estimation is the same as PPS phase coefficient estimation using HAF. This fact motivates us to modify one method aiming at enlarging HAF estimation range to solve the ambiguity problem of the CFO. The obtained algorithm explores lag diversity and solves linear Diophantine equations via the Euclidean algorithm to recover the correct CFO from ambiguous values. Theoretical analysis and simulations have shown that the proposed method can effectively achieve a wide estimation range with reduced MSEs compared to

other reported algorithms.

8.2 Recommendations for Further Research

The following recommendations are made for possible research directions in the future.

8.2.1 Properties and Practical Implementations of PTFT

More efforts are needed for analyzing the properties of PTFT. For example, the side lobes of PTFT, which are the peaks other than the peaks corresponding to desired signal components, are to be studied further. Since PTFT involves in multi-dimensional representation, the properties of its side lobes are more complicated than those of one-dimensional DFT. All of the fast algorithms proposed in this thesis compute the PTFT exactly, i.e. neglecting floating-point errors, thus the study of floating-point errors accumulated by the PTFT algorithms is also required. Another important issue is to implement the proposed fast PTFT algorithms using hardware architectures other than pipelined structure, because the choice of appropriate hardware structure is depending on the application at hand. The design of such systems will encourage the use of PTFT in real-time applications.

8.2.2 Application of Time-Frequency Analysis for Signals with Time-varying Frequencies

Another valuable research direction is to explore the applications of LPTFT and MLPTFT. Since most real-world signals, including speech, sonar, communication, and biological signals, have time-varying frequencies, and it has been reported that LPTFT and MLPTFT can provide high time-frequency resolution for signals with time-varying frequencies, it is worthy to investigate the use of LPTFT and MLPTFT in diverse application areas. Take the problem of blind source separation (BSS) of signals with time-varying frequencies as an example, which aims at recovering unknown signals from several observed mixtures. A few techniques have been reported in the literature, such as the method in [125] which uses the spatial time-frequency distribution (STFD) to exploit the difference in the TF signatures of the sources to be separated, and the sparse component analysis (SCA) approach in [126] which relies on TFR to change the signal into “sparse” component in TF domains. In these algorithms, one can use any TFR. However, the best results are obtained from high resolution TFRs with reduced or free crossterms, LPTFT and MLPTFT are appropriate choices. Evaluation of the performance of using them and comparison with that of using other TFRs are required to be demonstrated.

8.2.3 Robust Receiver Design in Wireless Communication Systems

The increase in the use of wireless devices has put many such systems operating in vicinity of other wireless devices. This has caused an increase in interfer-

ence from the other users operating in the same band, which makes the noise in presence of interference to deviate from Gaussianity. Thus in practice the Gaussian noise assumption does not hold for some communication systems and scenarios [80, 81]. Robust receiver based on the M -estimation has been reported in [127]. The numerical study in this thesis indicates that robust form based on the M -estimation generally requires a large computational complexity. More efficient methods based on L -estimation and clipping are promising techniques to explore in order to suppress impulsive noise and improve system performance.

Appendix A - Subroutine for 2nd order PTFT based on DIF radix a algorithm

This appendix present Matlab programs, based on the radix- a algorithm, for the computation of the 2nd order PTFT of complex-valued and real-valued sequence of N points. It contains the main functions `cPTFT_2nd_ra(x, a, p, b, Ns)` and `rPTFT_2nd_ra(x, a, p, b, Ns)`, which call function `cPTFT_2nd.Ns(x)` for complex-valued sequence of Ns points and function `rPTFT_2nd.Ns(x)` for real-valued sequence of Ns points. A function `fun_u(x, l0, l1, n)` is also included.

```
function X = cPTFT_2nd_ra(x,a,p,b,Ns);
% This is for 2nd order PTFT of complex sequence based on DIF radix-a algorithm
N = length(x);
if N ~= a^p*b
    error('Error in input parameters!');
end
x = x(:) .'; % make sure x is a row vector
% The length-N PTFTs is divided into length-Ns PTFTs
% This step computes the inputs of the length-Ns PTFTs
D = (N/Ns)^2; d = 1; len = N; aa = a^2;
while d < D
    new_d = d*aa; new_len = len/a;
    new_x = zeros(new_d,new_len);
    n = [0:new_len-1]; w=exp(-j*2*pi/len);
    for i = 0:d-1
        num = 0;
        for l0 = 0:a-1
            for l1 = 0:a-1
                num = num + 1;
```

```

        u = fun_u(x(i+1),:),a,l0,l1,n);
        twiddle = w.^(l0*n + l1*n^2);
        new_x(i*aa+num,:) = u.*twiddle
    end
end
clear x; x = new_x; clear new_x;
d = new_d; len = new_len;
end
% compute the length-Ns PTFTs
X = cell(1,D);
for i = 1:D
    X{i} = cPTFT_2nd_Ns(x(i,:));
end
% obtain the outputs of length-N PTFT from the length-Ns PTFTs
d = D; len = Ns;
while d > 1
    new_d = d/aa; new_len = len*a; new_X = cell(1,new_d);
    k0 = a*[0:len-1]+1;
    for i = 0:new_d-1
        for l0 = 0:a-1
            for l1 = 0:a-1
                new_X{i+1}(k0+l0,k1+l1) = X{1}; X(1) = [];
            end
        end
    end
    clear X; X = new_X; clear new_X;
    d = new_d; len = new_len;
end
X = X{1};

function X = rPTFT_2nd_ra(x,a,p,b,Ns);
% This is for 2nd order PTFT of real sequence based on DIF radix-a algorithm
N = length(x);
if N ~= a^p*b
    error('Error in input parameters!');
end
if any( imag(x)) % x is not purely real
    error('It only supports real-valued input!');
end
x = x(:) .'; % make sure x is a row vector

```

```

aa = a^2; flag = ( mod(a,2) == 0);
if flag % a is even
    a2 = a/2; Ke = (aa/2+1)/(aa-1); a1 = a2-1;
else
    a1 = (a-1)/2;
end
% This step computes the inputs of the length-Ns PTFTs
d = 1; len = N; q = 1;
while len > Ns
    nlen = len/a; q = q*aa;
    if flag
        nd = 1 + Ke*(q-1);
    else
        nd = 1 + (q-1)/2;
    end
    nx = zeros(nd,nlen);
    n = [0:nlen-1]; nn = n.^2;
    w = 2*pi/len; num = 1;
    % (l0,l1)=(0,0)
    nx(num,:) = fun_u(x(1,:),a,0,0,n);
    tmp = zeros(1,nlen);
    for l1 = 1:a1 % l0=0
        num = num + 1; tmp = tmp + nn;
        twi = exp(-j*w*mod(tmp,len));
        nx(num,:) = fun_u(x(1,:),a,0,l1,n) .*twi;
    end
    tmp = zeros(1,nlen);
    for l0 = 1:a1 % l1 = 0
        num = num + 1; tmp = tmp + n;
        twi = exp(-j*w*mod(tmp,len));
        nx(num,:) = fun_u(x(1,:),a,l0,0,n) .*twi;
    end
    if flag
        % (l0,l1)=(a/2,0)
        num = num + 1; w1 = pi/nlen;
        twi = exp(-j*w1*n);
        nx(num,:) = fun_u(x(1,:),a,a2,0,n) .*twi;
        tmp = a2*nn;
        for l0 = 0:a2 % l1=a/2
            num = num + 1;
            twi = exp(-j*w*mod(tmp,len));
            nx(num,:)=fun_u(x(1,:),a,l0,a2,n) .*twi;
        end
    end
end

```

```

        tmp = tmp + n;
    end
end
tmp0 = n;
for l0 = 1:a-1
    for l1 = 1:a1
        num = num + 1; tmp = tmp0 + nn;
        twi = exp(-j*w*mod(tmp,len));
        nx(num,:)=fun_u(x(1,:),a,l0,l1,n).*twi;
    end
    tmp0 = tmp0 + n;
end
for i = 1:d-1
    tmp0 = zeros(1,nlen);
    for l0 = 0:a-1
        tmp = tmp0;
        for l1 = 0:a-1
            num = num + 1;
            twi = exp(-j*w*mod(tmp,len));
            nx(num,:)=fun_u(x(i+1,:),a,l0,l1,n).*twi;
            tmp = tmp + nn;
        end
        tmp0 = tmp0 + n;
    end
end
clear x; x = nx; clear nx; d = nd; len = nlen;
end
% Compute the length-Ns PTFTs
X = cell(1,d);
for i = 1:d
    if any(imag(x(i,:))) % x is not purely real
        X{i} = cPTFT_2nd_Ns(x(i,:));
    else
        X{i} = rPTFT_2nd_Ns(x(i,:));
    end
end
clear x;
% Obtain the outputs of length-N PTFT from the length-Ns PTFTs
while len < N
    q = q/aa;
    if flag
        nd = 1 + Ke*(q-1);

```



```

else
    nd = 1 + (q-1)/2;
end
nlen = len * a; nX = cell(1,nd);
k0 = a*[0:len-1]+1; kk = nlen-k0+2;
nX{1}(k0,k0)=X{1}; X(1) = []; % (l0,l1)=(0,0)
for l1 = 1:a1 % l0=0
    nX{1}(k0,k0+l1) = X{1};
    nX{1}(mod(kk,nlen),kk-l1) = conj(X{1}); X(1)=[];
end
for l0 = 1:a1 % l1=0
    nX{1}(k0+l0,k0) = X{1};
    nX{1}(kk-l0, mod(kk,nlen)) = conj(X{1}); X(1)=[];
end
if flag
    nX{1}(k0+a2,k0)=X{1}; X(1)=[]; % (l0,l1)=(a/2,0)
    nX{1}(k0,k0+a2)=X{1}; X(1)=[]; % (l0,l1)=(0,a/2)
    for l0 = 1:a1 % l1=a/2
        nX{1}(k0+l0,k0+a2)=X{1};
        nX{1}(kk-l0,kk-a2)= conj(X{1}); X(1)=[];
    end
    nX{1}(k0+a2,k0+a2)=X{1}; X(1)=[]; % (l0,l1)=(a/2,a/2)
end
for l0 = 1:a-1
    for l1 = 1:a1
        nX{1}(k0+l0,k0+l1)=X{1};
        nX{1}(kk-l0,kk-l1)=conj(X{1}); X(1)=[];
    end
end
for i = 1:nd-1
    for l0 = 0:a-1
        for l1 = 0:a-1
            nX{i+1}(k0+l0,k0+l1) = X1; X(1)=[];
        end
    end
end
clear X; X = nX; clear nX; len = nlen;
end
X = X{1};

```

```

function X = rPTFT_2nd_Ns(x);
    if any( imag(x))      % x is not purely real
        error( 'In only supports real-valued input');
    end
    x = x(:); % make sure x is a column vector
    N = length(x); X = zeros(N,N);
    n = [0:N-1] .'; nn = n.^2; w = 2*pi/N*mod(nn,N);
    flag = (mod(N,2)==0);
    if flag % N is even
        k1_end = floor(N/4);
    else
        k1_end = (N-1)/2;
    end
    % compute via 1D FFTs
    for k1 = 0:k1_end
        tmp = x.* exp(-j*w*k1); X(:,k1+1) = fft(tmp,N);
    end
    k0 = [0:N-1]; k01 = mod(N-k0,N);
    % when N is even,  $X(N-k_0, N/2-k_1) = \text{conj}(X(k_0+N/2, k_1))$ 
    if flag
        k02 = mod(k0+N/2,N); k1 = [1:k1_end];
        X(k01+1, N/2-k1+1) = conj(X(k02+1, k1+1));
    end
    % Hermitian property:  $X(N-k_0, N-k_1) = \text{conj}(X(k_0, k_1))$ 
    if flag
        k1 = [1:N/2-1];
        X(k02+1, N/2+1) = X(k0+1, 1); %  $X(k_0+N/2, N/2) = X(k_0, 0)$ 
    else
        k1 = [1:k1_end];
    end
    end
    X(k01+1, N-k1+1) = conj(X(k0+1, k1+1));

function X = cPTFT_2nd_Ns(x);
    x = x(:); % make sure x is a column vector
    N = length(x); flag = (mod(N,2)==0);
    X = zeros(N,N); n = [0:N-1] .'; nn = n.^2;
    if flag % N is even
        N2 = N/2; k1_end = N2-1; % range of k1 reduces to one half
    else
        k1_end = N-1;
    end

```

```

end
w = 2*pi/N*mod(nn,N);
for k1 = 0:k1_end
    tmp = x.*exp(-j*w*k1);
    X(:,k1+1) = fft(tmp,N); % compute via 1D FFTs
end
if flag % using symmetric property
    k = [0:k1_end]+1;
    X(k+N2,k+N2) = X(k,k);
    X(k,k+N2) = X(k+N2,k);
end

function u = fun_u(x,a,l0,l1,n);
N = length(x); x = x(:).'; n = n(:).';
P1 = l0; P2 = l1*n; P = P1 + P2 + P2;
phi0 = 2*pi*P/a; ind0 = N/a; phi = phi0;
n1 = n+1; N1 = n1+N;
if rem(a,2), % a odd
    u = x(n1);
    ind1 = n1+ind0; ind2 = N1-ind0;
    for c = 1:(a-1)/2
        s1 = x(ind1)+x(ind2);
        s2 = x(ind1)-x(ind2);
        u = u + s1.*cos(phi)-j*s2.*sin(phi);
        ind1 = ind1 + ind0;
        ind2 = ind2 - ind0;
        phi = phi + phi0;
    end
else % a is even
    a2 = a/2; N2 = N/2; sgn = mod(P1,2) ;
    if sgn
        u = x(n1) - x(n1+N2);
    else
        u = x(n1) + x(n1+N2);
    end
    if mod(a,4)==0
        a4 = a2/2; N4 = N2/2;
        tmp = mod(P,4); tmp = (-j).^tmp;

```

```

    if sgn
        u = u + tmp.*(x(n1+N4)-x(n1+N2+N4));
    else
        u = u + tmp.*(x(n1+N4)+x(n1+N2+N4));
    end
    index_e = a4-1;
else
    index_e = floor(a/4);
end
ind1 = n1+ind0; ind2 = ind1+N2;
ind3 = N1-ind0; ind4 = ind3-N2;
for c = 1:index_e
    if sgn
        s1 = x(ind1) - x(ind2);    s2 = x(ind3)- x(ind4);
    else
        s1 = x(ind1) + x(ind2);    s2 = x(ind3)+ x(ind4);
    end
    u = u + (s1+s2).* cos(phi) - j*(s1-s2) .*sin(phi);
    phi = phi + phi0;
    ind1 = ind1 + ind0; ind2 = ind2 + ind0;
    ind3 = ind3 - ind0; ind4 = ind4 - ind0;
end
end
end

```

Author's Publications

- 1 Yingtuo Ju and Guoan Bi, "Generalized Fast Algorithms for the Polynomial Time-Frequency Transforms," *IEEE Trans. Signal Process.*, vol. 55, no. 10, pp. 4907-4915, 2007.
- 2 Guoan Bi, Yingtuo Ju and Xiumei Li, "Fast Algorithms for Polynomial Time-Frequency Transforms of Real-Valued Sequences," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1905-1915, 2008.
- 3 Guoan Bi and Yingtuo Ju, "Radix-3 Fast Algorithms for Polynomial Time-Frequency Transforms," *Signal Process.*, vol.88, Issue 9, pp. 2316-2322, 2008.
- 4 Yingtuo Ju and Guoan Bi, "CFO Estimation with Lag Diversity Techniques for OFDM Systems", *Revised and Submitted to IEEE Trans. Wireless Commun.*, 2008.
- 5 X.Li, Guoan Bi, Yingtuo Ju and Zhiping Lin, "Improved ISAR imaging based on local polynomial Fourier transform", *Accepted by IEEE Trans. Aerosp. Electron. Syst.*, 2008.
- 6 B. Barkat and Y. Ju, "A Modified Fractional Fourier Series for the Analysis of Finite Chirp Signals & Its Application," *IEEE 7th International Symposium on Signal Processing and its Applications (ISSPA'03)*, vol. 1, pp. 285-288, July, 2003.
- 7 B. Barkat and Y. Ju, "Detection of Known FM Signals in Known Heavy-tailed Noise," *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT'03)*, pp. 130-133, Dec., 2003.
- 8 Y. Ju and B. Barkat, "A New Efficient Chirp Modulation Technique For Multi-user Access Communications Systems," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04)*, vol.4, pp.17-21, May, 2004.

-
- 9 Y. Ju, B. Barkat and S. Attallah, "Analysis of Peak-to-average Power Ratio of a Multicarrier System Based on the Fractional Fourier Transform," *International Conference on Communication Systems (ICCS 2004)*, pp. 165-168, Sept. 2004.

Bibliography

- [1] T. J. Abatzoglou, "Fast maximum likelihood joint estimation of frequency and frequency rate," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-22, no. 6, pp. 708–715, 1986.
- [2] O. Besson, M. Ghogho, and A. Swami, "Parameter estimation for random amplitude chirp signals," *IEEE Trans. Signal Process.*, vol. 47, no. 12, pp. 3208–3219, 1999.
- [3] M. Z. Ikram, K. Abed-Meraim, and Y. Hua, "Fast quadratic phase transform for estimating the parameters of multicomponent chirp signals," *Digital Signal Process.*, vol. 7, no. 2, pp. 127–135, 1997.
- [4] X. Xia, "Discrete chirp-Fourier transform and its applications to chirp rate estimation," *IEEE Trans. Signal Process.*, vol. 48, no. 11, pp. 3122–3133, 2000.
- [5] S. Peleg and B. Porat, "Estimation and classification of polynomial-phase signals," *IEEE Trans. Inf. Theory*, vol. 37, no. 2, pp. 422–430, 1991.
- [6] —, "Linear FM signal parameter estimation from discrete-time observations," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 27, no. 4, pp. 607–616, 1991.
- [7] S. Peleg and B. Friedlander, "The discrete polynomial-phase transform," *IEEE Trans. Signal Process.*, vol. 43, no. 8, pp. 1901–1914, 1995.
- [8] —, "Multicomponent signal analysis using the polynomial-phase transform," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 32, no. 1, pp. 378–387, 1996.
- [9] S. Peleg, B. Porat, and B. Friedlander, "The achievable accuracy in estimating the instantaneous phase and frequency of a constant amplitude signal," *IEEE Trans. Signal Process.*, vol. 41, no. 6, pp. 2216–2224, 1993.

- [10] S. Barbarossa and V. Petrone, "Analysis of polynomial-phase signals by the integrated generalized ambiguity function," *IEEE Trans. Signal Process.*, vol. 45, no. 2, pp. 316–327, 1997.
- [11] S. Barbarossa, A. Scaglione, and G. B. Giannakis, "Product high-order ambiguity function for multicomponent polynomial-phase signal modeling," *IEEE Trans. Signal Process.*, vol. 46, no. 3, pp. 691–708, 1998.
- [12] L. Cohen, *Time-Frequency Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
- [13] B. Boashash, *Advances in Spectrum Analysis and Array Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1991, vol. 1, ch. 9 Time-Frequency Signal Analysis, pp. 418–517.
- [14] V. Katkovnik, "A new form of the Fourier transform for time-varying frequency estimation," *Signal Process.*, vol. 47, no. 2, pp. 187–200, 1995.
- [15] —, "Discrete-time local polynomial approximation of the instantaneous frequency," *IEEE Trans. Signal Process.*, vol. 46, no. 10, pp. 2626–2637, 1998.
- [16] —, "Nonparametric estimation of instantaneous frequency," *IEEE Trans. Inf. Theory*, vol. 43, no. 1, pp. 183–189, 1997.
- [17] J. C. Curlander and R. N. McDonough, *Synthetic Aperture Radar: System & Signal Processing*. New York: Wiley, 1991.
- [18] M. Soumekh, "Reconnaissance with ultra wideband UHF synthetic aperture radar," *IEEE Signal Process. Mag.*, vol. 12, no. 4, pp. 21–40, 1995.
- [19] F. Gini, M. Luise, and R. Reggiannini, "Cramer Rao bounds in the parametric estimation of fading radiotransmission channels," *IEEE Trans. Commun.*, vol. 46, no. 10, pp. 1390–1398, 1998.
- [20] I. Ali, N. Al-Dhahir, and J. E. Hershey, "Doppler characterization for LEO satellites," *IEEE Trans. Commun.*, vol. 46, no. 3, pp. 309–313, 1998.
- [21] "Special Issue on communications in the intelligence transportation system," *IEEE Commun. Mag.*, vol. 34, 1996.
- [22] I. Djurović and S. Stanković, "Estimation of time-varying velocities of moving objects by time-frequency representations," *IEEE Trans. Image Process.*, vol. 12, no. 5, pp. 550–562, 2003.

- [23] S. Stanković, I. Djurović, and R. Herpers, "Velocity and acceleration estimation in video sequences by the local polynomial periodogram," in *International Symposium on Signal Processing and its Applications*, vol. 1, Paris, France, 2003, pp. 145–148.
- [24] B. Boashash, "Estimating and interpreting the instantaneous frequency of a signal-part II: Algorithms," *Proc. IEEE*, vol. 80, pp. 539–569, 1992.
- [25] V. Katkovnik, "Robust M -periodogram," *IEEE Trans. Signal Process.*, vol. 46, no. 11, pp. 3104–3109, 1998.
- [26] I. Djurović and L. Stanković, "Robust Wigner distribution with application to the instantaneous frequency estimation," *IEEE Trans. Signal Process.*, vol. 49, no. 12, pp. 2985–2993, 2001.
- [27] B. Barkat and L. Stanković, "Analysis of polynomial FM signals corrupted by heavy-tailed noise," *Signal Process.*, vol. 84, no. 1, pp. 69–75, 2004.
- [28] I. Djurović, V. Katkovnik, and L. Stanković, "Median filter based realizations of the robust time-frequency distributions," *Signal Process.*, vol. 81, no. 8, pp. 1771–1776, 2001.
- [29] I. Djurović, L. Stanković, and J. Böhme, "Robust L -estimation based forms of signal transforms and time-frequency representations," *IEEE Trans. Signal Process.*, vol. 51, no. 7, pp. 1753–1761, 2003.
- [30] I. Djurović, "Robust adaptive local polynomial Fourier transform," *IEEE Signal Process. Lett.*, vol. 11, no. 2, pp. 201–204, 2004.
- [31] I. Djurović and V. V. Lukin, "Estimation of single-tone signal frequency by using the L-DFT," *Signal Process.*, vol. 87, no. 6, pp. 1537–1544, 2007.
- [32] F. Hlawatsch and G. F. Boudreaux-Bartels, "Linear and quadratic time-frequency signal representations," *IEEE Signal Process. Mag.*, vol. 9, no. 4, pp. 21–67, 1992.
- [33] B. Boashash, "Estimating and interpreting the instantaneous frequency of a signal-part I: Fundamentals," *Proc. IEEE*, vol. 80, pp. 529–538, 1992.
- [34] M. Martone, "A multicarrier system based on the fractional Fourier transform for time-frequency selective channel," *IEEE Trans. Signal Process.*, vol. 49, no. 6, pp. 1011–1020, 2001.

- [35] H. Liu, "Multicode ultra-wideband scheme using chirp waveforms," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 4, pp. 885–891, 2006.
- [36] S. Erküçük, S. Krishnan, and M. Zeytinoglu, "A robust audio watermark representation based on linear chirps," *IEEE Trans. Multimedia*, vol. 8, no. 5, pp. 925–936, 2006.
- [37] A. Ramalingam and S. Krishnan, "Robust image watermarking using a chirp detection-based technique," *IEE Proc. Vis. Image Signal Process.*, vol. 152, no. 6, pp. 771–778, 2005.
- [38] S. Peleg and B. Porat, "The Cramer-Rao lower bound for signals with constant amplitudes and polynomial phase," *IEEE Trans. Signal Process.*, vol. 39, no. 3, pp. 749–752, 1991.
- [39] Y. Wei, "New algorithms for the analysis of signals with time-varying instantaneous frequency," Ph.D. dissertation, Electrical & Electronic Engineering, Nanyang Technological University, 2005.
- [40] A. Scaglione and S. Barbarossa, "Statistical analysis of the product high-order ambiguity function," *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 343–356, 1999.
- [41] M. Z. Ikram, K. Abed-Meraim, and Y. Hua, "Estimating the parameters of chirp signals: an iterative approach," *IEEE Trans. Signal Process.*, vol. 46, no. 12, pp. 3436–3441, 1998.
- [42] G. T. Zhou and Y. Wang, "Exploring lag diversity in the high-order ambiguity function for polynomial phase signals," *IEEE Signal Process. Lett.*, vol. 4, no. 8, pp. 240–242, 1997.
- [43] M. Z. Ikram and G. T. Zhou, "Estimation of multicomponent polynomial phase signals of mixed orders," *Signal Process.*, vol. 81, no. 11, pp. 2293–2308, 2001.
- [44] S. Barbarossa, "Analysis of multicomponent LFM signals by a combined Wigner-Hough transform," *IEEE Trans. Signal Process.*, vol. 43, no. 6, pp. 1511–1515, 1995.
- [45] S. Kay, "Statistically/computationally efficient frequency estimation," in *IEEE Int. Conf. Acoust., Speech, and Signal Process.*, New York, 1988, pp. 2292–2295.

- [46] B. Boashash and B. Ristic, "Polynomial time-frequency distributions and time-varying higher-order spectra: Application to the analysis of multi-component FM signal and to the treatment of multiplicative noise," *Signal Process.*, vol. 67, no. 1, pp. 539–569, 1998.
- [47] B. Barkat and B. Boashash, "Instantaneous frequency estimation of polynomial FM signals using the peak to the PWVD: Statistical performance in the presence of additive Gaussian noise," *IEEE Trans. Signal Process.*, vol. 47, no. 9, pp. 2480–2490, 1999.
- [48] B. Barkat, "Instantaneous frequency estimation of nonlinear frequency-modulated signals in the presence of multiplicative and additive noise," *IEEE Trans. Signal Process.*, vol. 49, no. 10, pp. 2214–2222, 2001.
- [49] V. C. Chen and S. Qian, "Joint time-frequency transform for radar range-Doppler imaging," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 34, no. 2, pp. 486–499, 1998.
- [50] X. Ouyang and M. G. Amin, "Short-time Fourier transform receiver for nonstationary interference excision in direct sequence spread spectrum communications," *IEEE Trans. Signal Process.*, vol. 49, no. 4, pp. 851–863, 2001.
- [51] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE Trans. Speech and Audio Process.*, vol. 13, no. 5, pp. 1035–1047, 2005.
- [52] S. Chikkerur, A. N. Cartwright, and V. Govindaraju, "Fingerprint enhancement using STFT analysis," *Pattern Recognition*, vol. 40, no. 1, pp. 198–211, 2007.
- [53] P. Rao and F. Taylor, "Estimation of IF using the discrete Wigner-Ville distribution," *Electr. Lett.*, vol. 26, no. 4, pp. 246–248, 1990.
- [54] H. Choi and W. Williams, "Improved time-frequency representation of multicomponent signals using exponential kernels," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 6, p. 862871, 1989.
- [55] L. E. Atlas, Y. Zhao, and R. J. Marks, "The use of cone shape kernels for generalized time-frequency representations of nonstationary signals," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 7, pp. 1084–1091, 1990.

- [56] B. Barkat and B. Boashash, "A high-resolution quadratic time-frequency distribution for multicomponent signal analysis," *IEEE Trans. Signal Process.*, vol. 49, no. 10, pp. 2232–2239, 2001.
- [57] L. Stanković, "A method for time-frequency signal analysis," *IEEE Trans. Signal Process.*, vol. 42, no. 1, pp. 225–229, 1994.
- [58] S. Stanković and L. Stanković, "An architecture for the realization of a system for time-frequency signal analysis," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 44, no. 7, pp. 600–604, 1997.
- [59] B. Boashash and P. O'shea, "Polynomial Wigner-Ville distributions and their relationship to time-varying higher order spectra," *IEEE Trans. Signal Process.*, vol. 42, no. 1, pp. 216–220, 1994.
- [60] B. Barkat and B. Boashash, "Design of higher order polynomial Wigner-Ville distributions," *IEEE Trans. Signal Process.*, vol. 47, no. 9, pp. 2608–2611, 1999.
- [61] B. Boashash and V. Sucic, "Resolution measure criteria for the objective assessment of the performance of quadratic time-frequency distributions," *IEEE Trans. Signal Process.*, vol. 51, no. 5, pp. 1253–1263, 2003.
- [62] V. Katkovnik, "New method for varying adaptive bandwidth selection," *IEEE Trans. Signal Process.*, vol. 47, no. 9, pp. 2567–2571, 1999.
- [63] I. Djurović, T. Thayaparan, and L. Stanković, "Adaptive local polynomial Fourier transform in ISAR," *Journal of Applied Signal Process.*, vol. 2006, pp. 1–15, 2006.
- [64] F. Zhang, Y. Chen, and G. Bi, "Adaptive harmonic fractional Fourier transform," *IEEE Signal Process. Lett.*, vol. 6, no. 11, pp. 281–283, 1999.
- [65] L. Stanković, "A measure of some time-frequency distributions concentration," *Signal Process.*, vol. 3, no. 621–631, p. 2001, 81.
- [66] Y. Wei and G. Bi, "Efficient analysis of time-varying multi-component signals with modified LPTFT," *Journal of Applied Signal Process.*, vol. 2005, no. 8, pp. 1261–1268, 2005.
- [67] P. Duhamel, "Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 2, pp. 285–295, 1986.

- [68] H. Sorensen, M. T. Heideman, and C. S. Burrus, "On computing the split-radix FFT," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 1, pp. 152–156, 1986.
- [69] Y. Suzuki, T. Sone, and K. Kido, "A new FFT algorithm of radix 3, 6, and 12," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 2, pp. 380–383, 1986.
- [70] M. Vetterli and P. Duhamel, "Split-radix algorithms for length p^m DFT's," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 1, pp. 57–64, 1989.
- [71] C. M. Rader, "Discrete Fourier transforms when the number of data samples is prime," *Proc. IEEE*, vol. 56, no. 6, p. 1107, 1968.
- [72] D. P. Kolba and T. W. Parks, "A prime factor FFT algorithm using high-speed convolution," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 4, pp. 281–294, 1977.
- [73] P. Duhamel and M. Vetterli, "Fast Fourier transforms: a tutorial review and a state of the art," *Signal Process.*, vol. 19, no. 4, pp. 259–299, 1990.
- [74] D. B. Harris, J. H. McClellan, D. S. K. Chan, and H. W. Schuessler, "Vector radix fast Fourier transform," in *IEEE Int. Conf. Acoust., Speech, and Signal Process.*, vol. 2, 1977, pp. 548–551.
- [75] S. C. Chan and K. L. Ho, "Split vector-radix fast Fourier transform," *IEEE Trans. Signal Process.*, vol. 40, no. 8, pp. 2029–2039, 1992.
- [76] B. Friedlander, "Parametric signal analysis using the polynomial phase transform," in *Proceedings of the IEEE signal processing workshops on high order statistics*, South Lake Tahoe, CA, 1993, pp. 151–159.
- [77] S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1998.
- [78] —, *Fundamentals of Statistical Signal Processing: Detection Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1998.
- [79] I. Pitas and A. N. Venetsanopoulos, "Order statistics in digital image processing," *Proc. IEEE*, vol. 80, no. 12, pp. 1893–1921, 1992.
- [80] K. M. Blackard and C. W. Bostian, "Measurements and models of radio frequency impulsive noise for indoor wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 11, no. 9, pp. 991–1001, 1993.

- [81] J. L. Fernández and J. Salter, "Modelling impulsive interference in DVB-T: Statistical analysis, test waveforms and receiver performance, BBC R&D white paper [online], available : <http://www.bbc.co.uk/rd/pubs/whp/whp080.html>."
- [82] P. Huber, *Robust Statistics*. John Wiley & Sons Ltd, 1981.
- [83] D. Middleton, "Statistical-physical models of electromagnetic interference," *IEEE Trans. Electromagn. Compat.*, vol. EMC-19, no. 8, pp. 106–127, 1977.
- [84] C. L. Nikias and M. Shao, *Signal Processing with Alpha-Stable Distributions and Applications*. New York: John Wiley & Sons Ltd, 1995.
- [85] S. M. Zabin and H. V. Poor, "Efficient estimation of the class A parameters via the EM algorithm," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 60–72, 1991.
- [86] E. F. Fama and R. Roll, "Parameter estimates for symmetric stable distributions," *J. Amer. Sci. Assoc.*, vol. 66, no. 334, pp. 331–338, 1971.
- [87] E. E. Kuruoglu, C. Molina, and W. J. Fitzgerald, "Approximation of α -stable probability densities using finite mixtures of Gaussians," in *European Signal Process. Conf.*, Rhodes, Greece, 1998, pp. 989–992.
- [88] J. Friedmann, H. Messer, and J. Cardoso, "Robust parameter estimation of a deterministic signal in impulsive noise," *IEEE Trans. Signal Process.*, vol. 48, no. 4, pp. 935–942, 2000.
- [89] Z. Guo, L. G. Durand, and H. C. Lee, "Comparison of time-frequency distribution techniques for analysis of simulated Doppler ultrasound signals of the femoral artery," *IEEE Trans. Biomed. Eng.*, vol. 41, no. 4, pp. 332–342, 1994.
- [90] Y. Hu, K. D. K. Luk, W. W. Lu, A. Holmes, and J. C. Y. Leong, "Comparison of time-frequency distribution techniques for analysis of spinal somatosensory evoked potential," *Medical and Biological Engineering and Computing*, vol. 39, no. 3, pp. 375–380, 2001.
- [91] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proc. IEEE*, vol. 78, no. 4, pp. 678–689, 1990.

- [92] I. Pitas and P. Tsakalides, "Multivariate ordering in color image filtering," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 3, pp. 247–259, 1991.
- [93] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press, 2001.
- [94] I. Djurović, "Estimation of the sinusoidal signal frequency based on the marginal median DFT," *IEEE Trans. Signal Process.*, vol. 55, no. 5, pp. 2043–2051, 2007.
- [95] S. Barbarossa, "Detection and imaging of moving objects with synthetic aperture radar-part 1: optimal detection and parameter estimation theory," *Radar and Signal Process., IEE Proceed. F*, vol. 139, no. 1, pp. 79–88, 1992.
- [96] M. Z. Ikram, A. Belouchrani, D. Gesbert, and K. Abed-Meraim, "Parametric estimation and suppression of non-stationary interference in DS-spread spectrum communications," in *The 32nd Asilomar Conf. on Signals, Systems and Comput.*, 1998, pp. 1401–1405.
- [97] P. O'shea, "Fast parameter estimation algorithms for linear FM signals of frequency and frequency rate," in *IEEE Int. Conf. Acoust., Speech, and Signal Process.*, vol. 4, 1994, pp. 17–20.
- [98] G. Bi and Y. Chen, "Fast DFT algorithms for length $N = q \times 2^m$," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 45, no. 6, pp. 685–690, 1998.
- [99] G. Bi and Y. Wei, "Fast computation for third-order polynomial time frequency transforms," *Electr. Lett.*, vol. 40, no. 5, pp. 349–351, 2004.
- [100] Y. Wei and G. Bi, "Fast algorithm for polynomial time frequency transform," *Signal Process.*, vol. 87, no. 5, pp. 789–798, 2007.
- [101] G. Bi and Y. Wei, "Split-radix algorithms for arbitrary order of polynomial time frequency transforms," *IEEE Trans. Signal Process.*, vol. 55, no. 1, pp. 134–141, 2007.
- [102] G. Bi, Y. Wei, G. Li, and C. Wang, "Radix-2 DIF fast algorithms for polynomial time-frequency transforms," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 42, no. 4, pp. 1540–1546, 2006.

- [103] H. Murakami, "Real-valued decimation-in-time and decimation-in-frequency algorithms," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 41, no. 12, pp. 808–816, 1994.
- [104] G. Bi, "New split-radix algorithm for the discrete Hartley transform," *IEEE Trans. Signal Process.*, vol. 45, no. 2, pp. 297–302, 1997.
- [105] E. E. Swartzlander, W. K. W. Young, and S. J. Joseph, "A radix-4 delay commutator for fast Fourier transform processor implementation," *IEEE J. Solid-State Circuits*, vol. 19, no. 5, pp. 702–709, 1984.
- [106] R. V. Nee and R. Prasad, *OFDM wireless multimedia communications*. Boston: Artech House, 2000.
- [107] J. A. C. Bingham, "Multicarrier modulation for data transmission: An idea whose time has come," *IEEE Commun. Mag.*, vol. 28, no. 5, pp. 5–14, 1990.
- [108] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in *URSI International Symposium on Signals, Systems, and Electronics*, 1998, pp. 257–262.
- [109] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Prentice-Hall, 1975.
- [110] A. M. Despain, "Fourier transform computers using CORDIC iteration," *IEEE Trans. Comput.*, vol. 10, no. 993–1001, p. 1974, C-23.
- [111] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementation," *IEEE Trans. Comput.*, vol. C-33, no. 5, pp. 414–426, 1984.
- [112] G. Bi and E. V. Jones, "A pipelined FFT processor for word sequential data," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 12, pp. 1982–1985, 1989.
- [113] M. Hasan, T. Arslan, and J. Thompson, "Novel coefficient ordering based low power pipelined radix-4 FFT processor for wireless LAN applications," *IEEE Trans. Consum. Electron.*, vol. 49, no. 1, pp. 128–134, 2003.
- [114] Y. Chang and K. K. Parhi, "An efficient pipelined FFT architecture," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 50, no. 6, pp. 322–325, 2003.

- [115] T. Pollet, M. V. Bladel, and M. Moeneclaey, "BER sensitivity of OFDM systems to carrier frequency offset and weiner phase noise," *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 191–193, 1995.
- [116] P. H. Moose, "A technique for orthogonal frequency division multiplexing frequency offset correction," *IEEE Trans. Commun.*, vol. 42, no. 10, pp. 2908–2914, 1994.
- [117] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Trans. Commun.*, vol. 45, no. 12, pp. 1613–1621, 1997.
- [118] M. Morelli, A. N. D'Andrea, and U. Mengali, "Frequency ambiguity resolution in OFDM systems," *IEEE Commun. Lett.*, vol. 4, no. 4, pp. 134–136, 2000.
- [119] Z. Zhang, W. Jiang, H. Zhou, Y. Liu, and J. Gao, "High accuracy frequency offset correction with adjustable acquisition range in OFDM systems," *IEEE Trans. Wireless Commun.*, vol. 4, no. 1, pp. 228–237, 2005.
- [120] J. van de Beek, M. Sandell, and P. O. Borjesson, "ML estimation of time and frequency offset in OFDM systems," *IEEE Trans. Signal Process.*, vol. 45, no. 7, pp. 1800–1805, 1997.
- [121] H. Liu and U. Tureli, "A high efficiency carrier estimator for OFDM communications," *IEEE Commun. Lett.*, vol. 2, no. 4, pp. 104–106, 1998.
- [122] S. S. Abeysekera, "Efficient frequency estimation using the pulse-pair method at various lags," *IEEE Trans. Commun.*, vol. 54, no. 9, pp. 1542–1546, 2006.
- [123] K. Sathananthan and C. Tellambura, "Probability of error calculation of OFDM systems with frequency offset," *IEEE Trans. Commun.*, vol. 49, no. 11, pp. 1884–1888, 2001.
- [124] W. Hwang, H. Kang, and K. Kim, "Approximation of SNR degradation due to carrier frequency offset for OFDM in shadowed multipath channels," *IEEE Commun. Lett.*, vol. 7, no. 12, pp. 581–583, 2003.
- [125] A. Belouchrani and M. G. Amin, "Blind source separation based on time-frequency signal representations," *IEEE Trans. Signal Process.*, vol. 46, no. 11, pp. 2888–2897, 1998.

- [126] O. Fourt and M. Benidir, "Separation of a polynomial phase signals mixture using sparsity," in *The 32nd Annual Conf. of the IEEE Industrial Electronics Society*, Paris, France, 2006, pp. 3531–3536.
- [127] X. Wang and H. Poor, "Robust multiuser detection in non-Gaussian channels," *IEEE Trans. Signal Process.*, vol. 47, no. 2, pp. 289–305, 1999.