Employing

Data Fusion & Diversity

in the

Applications

of

Adaptive Signal Processing

Thiernithi V.

under supervision of
Prof. Danilo P. Mandic

A Thesis submitted in fulfilment of requirements for the degree of
Doctor of Philosophy (*Ph.D.*) in Electrical & Electronic Engineering
at Imperial College London

Department of Electrical & Electronic Engineering
Imperial College London
March 12, 2020

Declaration of Originality

I hereby certify to the best of my knowledge that the copy of my thesis, of which the contents have been seen by my supervisor before presentation and submitted for assessment on the programme of study leading to the degree of Ph.D. in Electrical & Electronic Engineering, entirely embodies the intellectual essence of my own course of study and research, and has not been taken from the work of others, apart from such work which either is a reprint of the material published elsewhere with permission to redistribute from the license owners, or has been cited and acknowledged within the text of my work.

Copyright Declaration

Abstract

The paradigm of adaptive signal processing is a simple yet powerful method for the class of system identification problems. The classical approaches consider standard one-dimensional signals whereby the model can be formulated by flat-view matrix/vector framework. Nevertheless, the rapidly increasing availability of large-scale multisensor/-multinode measurement technology has render no longer sufficient the traditional way of representing the data. To this end, the author, who from this point onward shall be referred to as 'we', 'us', and 'our' to signify the author myself and other supporting contributors i.e. my supervisor, my colleagues and other overseas academics specializing in the specific pieces of research endeavor throughout this thesis, has applied the adaptive filtering framework to problems that employ the techniques of data diversity and fusion which includes quaternions, tensors and graphs. At the first glance, all these structures share one common important feature: invertible isomorphism. In other words, they are algebraically one-to-one related in real vector space. Furthermore, it is our continual course of research that affords a segue of all these three data types. Firstly, we proposed novel quaternion-valued adaptive algorithms named the n-moment widely linear quaternion least mean squares (WL-QLMS) and c-moment WL-LMS. Both are as fast as the recursive-least-squares method but more numerically robust thanks to the lack of matrix inversion. Secondly, the adaptive filtering method is applied to a more complex task: the online tensor dictionary learning named online multilinear dictionary learning (OMDL). The OMDL is partly inspired by the derivation of the c-moment WL-LMS due to its parsimonious formulae. In addition, the sequential higher-order compressed sensing (HO-CS) is also developed to couple with the OMDL to maximally utilize the learned dictionary for the best possible compression. Lastly, we consider graph random processes which actually are multivariate random processes with spatiotemporal (or vertex-time) relationship. Sim-

ilar to tensor dictionary, one of the main challenges in graph signal processing is sparsity constraint in the graph topology, a challenging issue for online methods. We introduced a novel splitting gradient projection into this adaptive graph filtering to successfully achieve sparse topology. Extensive experiments were conducted to support the analysis of all the algorithms proposed in this thesis, as well as pointing out potentials, limitations and as-yet-unaddressed issues in these research endeavor.

Acknowledgment

I would like to thank my supervisor, Prof. Danilo Mandic, who has been the main advisor of all matters regarding the completion of this thesis.
I also would like to thank my colleagues who have contributed additional advice and suggestion that form part of the thesis.
Thirdly, I would like to thank Imperial College who has sponsored me financially to the end of my PhD course.
Lastly, I would like to thank my beloved family and significant other who have been mental supporters to the end of and long after the completion of this thesis.

Contents

List of Figures

List of Tables

Abbreviations

DSP: Digital Signal Processing

LMS: Least Mean Squares

MSE: Mean Square Error

QLMS: Quaternion Least Mean Squares

QRLS: Quaternion Recursive Least Squares

QWL: Quaternion Widely Linear

RLS: Recursive Least Squares

WL-QLMS: Widely Linear Quaternion Least Mean Squares

WL-QRLS: Widely Linear Quaternion Recursive Least Squares

WL-QRTLS: Widely Linear Quaternion Recursive Total Least Squares

TKD: Tucker Decomposition

CPD: Canonical Polyadic Decomposition

HOS: Higher-Order Statistics

OMDL: Online Multilinear Dictionary Learning

CS: Compressed Sensing

HO-CS: Higher-Order Compressed Sensing

DL: Dictionary Learning

ODL: Online Dictionary Learning

ETF: Equiangular Tight-Frame

TMOD: Tensorial Method of Optimal Directions

NRMSE: Normalized Root Mean Square Error

ARE: Average Representation Error

HSI: Hyperspectral Imaging

GSO: Graph Shift Operator

ARMA: Autoregressive Moving Average

Chapter 1

Introduction

DIGITAL signal processing began its root in electrical engineering but recently has intertwined more with the field of applied mathematics, especially machine learning and artificial intelligence. All these fields share one attribute: the use of linear algebra as an analytical building block which discretizes all the analysis and synthesis of problem solving. Almost half a century the digitization has simplified the analogue systems from natural processes and opened up more creative mathematical manoeuvre for better computing technology. The field has branched into many areas of research spanned across many applications. This thesis extends our previous MSc dissertation [4], which concentrated on an unconventional data structure called *quaternions*, by considering the utilization of data diversity and data fusion and their related adaptive signal processing usage.

1.1 Brief on Adaptive Signal Processing

As suggested in the title, the main topics of the thesis are adaptive signal processing and its applications on different different data structures. Its necessary basics is explained in details in this section. Everything would revolve around the concept of stochastic gradient descent which requires knowledge of linear algebra, linear estimation, optimisation techniques and probability. For complete treatment of classical adaptive filtering, please refer to [5].

The stochastic gradient descent is literally the gradient descent method applied to a stochastic optimization problem i.e. optimization problem where the data is generated by some underlying probability distribution. The ultimate philosophy of such tools is to implicitly capture necessary underlying statistics of the stochastic processes via the stochastic gradient without explicitly calculating the autocorrelation and cross-correlation) [6]. Moreover, the flexibility in the construction of the gradient descent means the properties of memoryless or with memory can be adjusted to suit specific applications dealing with stationary/non-stationary processes [7]. This learning technique is now popular across different fields with different names (e.g. online, real-time, etc), but ultimately boils down to the creation of a system that can adapt itself to the environment at hands.

In this section, we review two of the most well known stochastic gradient algorithms: the Least Mean Squares (LMS) and the Recursive Least Squares (RLS).

1.1.1 Least Mean Squares Algorithm

The LMS algorithm minimizes the *instantaneous* complex-valued quadratic cost function $\mathcal{J}_t$ associated with epoch t

$$\mathcal{J}_t = e_t e_t^* \tag{1.1a}$$

$$e_t = y_t - \mathbf{w}_{t-1}^H \mathbf{x}_t \tag{1.1b}$$

where $e_t \in \mathbb{C}$ is the instantaneous *a priori* error of the target signal $y_t \in \mathbb{C}$, adaptive weight (coefficient) vector $\mathbf{w}_t \in \mathbb{C}^p$, and tap input data $\mathbf{x}_t = [x_t \; x_{t-1} \; x_{t-2} \; \cdots \; x_{n-p+1})]^T \in \mathbb{C}^p$, for a filter of lag order p. The subsequent update of the coefficient vector is then expressed as

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mu e_t \mathbf{x}_t^* \tag{1.2}$$

where scalar μ is the step size. The value of $\mathbf{w}_0$ is defaulted to zero.

Due to its simplicity and mathematical elegance [5], the LMS algorithm has always been the mainstay of the adaptive filter related research topics. It can be shown even further that these elegant formulae eqs. (1.1) and (1.2) also hold up in a more general non-commutative construct like quaternion LMS (see [8]).

1.1.2 Recursive Least Squares Algorithm

The original derivation of the RLS algorithm relies on the least squares problem, which could be interpreted as an approximation to stochastic quadratic cost function *with memory* and given by [5]

$$\mathcal{J}_t = \sum_{\tau=1}^{t} \lambda^{t-\tau} e_\tau e_\tau^* \tag{1.3}$$

where $\lambda \in (0, 1]$ is the forgetting factor, giving the algorithm *memory*, and the output error e_t is defined as in eq. (1.1b). Observe that the cost function of RLS takes *more than one* values of error into account, with full memory ($\lambda = 1$), fading memory ($0 < \lambda < 1$), or *no* memory ($\lambda = 0$), which becomes an LMS algorithm! The complete recursion of the RLS algorithm is as follows [5]

$$\mathbf{k}_t = \frac{\mathbf{P}_{t-1}\mathbf{x}_t}{\lambda + \mathbf{x}_t^H \mathbf{P}_{t-1}\mathbf{x}_t} \tag{1.4a}$$

$$\mathbf{P}_t = \frac{1}{\lambda}[\mathbf{P}_{t-1} - \mathbf{k}_t \mathbf{x}_t^H \mathbf{P}_{t-1}] \tag{1.4b}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} + e_t \mathbf{k}_t^* \tag{1.4c}$$

where $\mathbf{P}_t \in \mathbb{C}^{p\times p}$ and $\mathbf{k}_t \in \mathbb{C}^p$ are referred to respectively as the *inverse correlation matrix* and the *Kalman gain vector*, such that $\mathbf{x}_t = [x_t \; x_{t-1} \; x_{t-2} \; \cdots \; x_{n-p+1})]^T$ for a filter of lag order p.

The RLS algorithm is clearly more complicated to implement than the LMS algorithm but achieves faster convergence and attains lowest steady state error in the case of stationary processes [5]. However, the RLS is more likely to be unstable numerically than the LMS algorithm [9] and hence its steady-state performance could be worse than that of the LMS due to its sensitivity to input disturbance [10]. Similarly, eqs. (1.3) and (1.4c) are still valid in non-commutative ring of quaternions via some manipulation of eqs. (1.4a) and (1.4b) (see [11]).

1.2 Thesis Overview

1.2.1 Motivations

The adaptive filters has been intensively studies with many production-ready relevant algorithms, still there are rooms for exploring and improving upon. When classical methods are augmented into higher dimension to cope with more complex data, the overall analysis needs more comprehensive re-work but generally stays unchanged [12]. This has initially meant to tackle multidimensional data until its usage can transcend into different data sources fused into the same model as long as the underlying correlation exists [13]. The way this thesis put an emphasis on data diversity and fusion also stemmed from this respect, since the unconventional data types studied in the thesis, namely quaternions, tensors and graphs, are also *multivariate* in their nature. In our previous work [4], the novel recursive algorithm based on quaternion-valued total least squares was proposed. In this continuation, the data types mentioned above were studied and employed to underpin the new algorithms grounded on the construction of adaptive filtering techniques motivating this entire thesis to transpire which we hope that would add noteworthy contribution to the research community, despite the scale of improvement.

1.2.2 Objectives

The objectives of this thesis are as follows:

- to provide the working knowledge of quaternions. tensors and graphs in relation to signal processing;

- to develop new recursive algorithms for quaternions, tensors and graphs for the application requiring real-time estimation/prediction;

- to give empirical comparison of the proposed algorithms with relevant benchmarks to testify its practical usefulness.

1.2.3 Outline

The thesis is divided into eight chapters and organized as follows. In chapter 2, relevant mathematical principles are briefly reviewed. The WL-QLMS [14] and WL-QRLS [11] algorithms are then reviewed and compared in chapter 3. Chapter 4 begins with a proof for the existence of low-rank matrix approximation in the quaternion domain, which is afterwards used to obtain the solutions of QTLS. The results from chapter 4 are then used to construct quaternion recursive total least squares (QRTLS) algorithms in chapter 5, where two methods are proposed: the RSVDQ-based method and the QRQI-based method. In chapter 6, simulations for the derived algorithm are conducted to compare performance, under input perturbations, with the pre-existing algorithms. Chapter 7 provides conclusions, followed by possible future directions.

Chapter 2

Quaternions

STANDARD complex-valued mathematical models arose in the light of a signal described not only by its *magnitude*, but also its *phase* [15]. they also find thier use in *data fusion technique* where different types of data fused together into the model and somehow the estimation result improves due to their underlying correlation [13]. One of the most straightforward approach to data fusion is a multivariate model in which the target variables are in the form of vectors, which casts the model into a matrix factorization problem [16]. Further, the model becomes *tensor* factorization problem when the desired output is in the form of matrices, or even tensors themselves. All these will be explored from Chapter 4 onwards where tensors and graphs are prime topics. Another way is to extend the model in its algebraic field i.e. from complex to hyper-complex numbers (*or quaternions*) which will be introduced rigorously in this chapter. Concepts relating to quaternions such as higher-order statistics, widely linear models, quaternion random processes and etc, are also discussed.

2.1 Basics of Quaternions

Quaternions was first conceived by Sir W.R. Hamilton in 1843 and applied to three-dimensional classical mechanics [17]. They became really popular again when their advantage was found in computer graphics replacing the method of Euler angles, which creates

the Gimbal lock problem [18]. From the perspective of abstract algebra, The division algebra renders 3D/4D models succinct, elegant, and less unwieldy than matrix factorization models [18, 19] and found applications in many research fields such as color image processing [20, 21], robotics [22, 23], Kalman filtering [24–26], information theory [27–29], pattern recognition [30], machine learning [31,32], wireless communication [33,34], sensing technology [35,36], and renewable energy [37]. Next, we dive in the concept of quaternions from the very beginning.

2.1.1 Basic Properties

A quaternion can be thought of as a generlization of complex number with one real part (denoted by the subscript a) and three imaginary parts (denoted by the subscript b, c and d). A quaternion $q \in \mathbb{H}$ can be expressed mathematically as

$$q = q_a + q_b\imath + q_c\jmath + q_d\kappa$$

where q_a, q_b, q_c, $q_d \in \mathbb{R}$ and $\imath$, $\jmath$, κ are *imaginary units* such that the they follow *sort of* rotating properties e.g.

$$\imath\jmath = \kappa \quad \jmath\kappa = \imath \quad \kappa\imath = \jmath$$

$$\imath^2 = \jmath^2 = \kappa^2 = \imath\jmath\kappa = -1$$

However, this imaginary rotation renders quaternion multiplication quaternions non-commutative ($\imath\jmath = \kappa \neq \jmath\imath = -\kappa$). The norm $\|q\|$ of a quaternion q is defined as

$$\|q\| = \sqrt{qq^*} = \sqrt{{q_a}^2 + {q_b}^2 + {q_c}^2 + {q_d}^2}$$

The quaternion q is called a *unit* quaternion if $\|q\| = 1$ and $q^* := q_a - q_b\imath - q_c\jmath - q_d\kappa$ is the conjugate of q. With this, the inverse of the quaternion q is expressed as

$$q^{-1} \triangleq \frac{q^*}{\|q\|^2}.$$

After all, among all basic properties presented so far, nothing is more germane to our

algorithm development than the *self-inverse mapping* property, also known as *involutions*, a very important properties unique to quaternions [38], given by

$$q^{\imath} = -\imath q \imath = q_a + q_b \imath - q_c \jmath - q_d \kappa$$

$$q^{\jmath} = -\jmath q \jmath = q_a - q_b \imath + q_c \jmath - q_d \kappa$$

$$q^{\kappa} = -\kappa q \kappa = q_a - q_b \imath - q_c \jmath + q_d \kappa$$

Involutions are called *self-inverse mappings* because the inverse of its inverse results in the original quaternion i.e. $(q^{\imath})^{\imath} = q$. The involutions are also distributive, $(q_1 q_2)^{\imath} = q_1{}^{\imath} q_2{}^{\imath}$. By looking closely, it can be seen as a *more complete* version of quaternionic conjugates. In a complex number, the components can be expressed by a linear combination of the complex number and its conjugate. Similarly in a quaternion, the components can be expressed by a linear combination of the quaternion and its involutions, that is

$$q_a = \frac{1}{4}[q + q^{\imath} + q^{\jmath} + q^{\kappa}]$$

$$q_b = \frac{1}{4}[q + q^{\imath} - q^{\jmath} - q^{\kappa}]$$

$$q_c = \frac{1}{4}[q - q^{\imath} + q^{\jmath} - q^{\kappa}]$$

$$q_d = \frac{1}{4}[q - q^{\imath} - q^{\jmath} + q^{\kappa}]$$

These relations are important in making an estimation/prediction model in hypercomplex domains. It was already known that in a complex-valued regression technique, both the complex numbers and their conjugates have to be taken into account in order to capture all *up-to-second-order* statistics of the data for the learning algorithm to be as rigorous as possible [39]. In quaternions, the quaternion and all its involutions are required to accomplish such rigor as well [40], and the model is called widely linear model.

2.1.2 Hilbert Spaces over Quaternions

The concept of Hilbert Spaces does not only form the basis of the linear algebra of a quaternion, but also its calculus, to be called $\mathbb{HR}-$calculus from this point onwards. Due to it part of the division ring, the space defined over quaternions will be either *left* or *right*, simply meaning that the multiplicative operators will be applied on the *left* or *right* side of the expression, respectively. This is a non-trivial result as a great deal of research effort has been put into realizing these simplistic yet elegant conclusion. An all-inclusive summary of this quaternionic analysis can be found in [41, 42], which will be the main source from which this section draws.

Since the division ring (the algebraic structure where quaternions live) is not a field [43], how multiplication is applied to the *left* or *right* of a quaternion equation matters. In our case, deriving an algorithm over a wrong quaternion space would result in quaternionic equation [44], which creates *nested algorithm* (algorithm-in-algorithm) problem. From our own practical perspective, the *right* quaternion vector space, consequently producing *right* quaternion constant rule [42], would ease all our quaternion expressions involving quaternionic derivatives. This is actually intuitively implied by the fact that the *right* vector space defined over division ring is associated with column space [45], a concept already familiar and common in standard complex-valued equivalents.

The main takeaway point from this section is the fact that all quaternion-related algorithms in this thesis will be based on the *right* constant rule [42] so that analytical closed-form formulae are achieved for the algorithm recursion. For all detailed discussions on these basics, please see [4].

2.2 $\mathbb{HR}$ Calculus

As in the name of a stochastic gradient algorithm, it is necessary to obtain a derivative of a quaternion function. The main obstacle, however, is our quaternion function of interest (the MSE objective) is real-valued and is not analytic in $\mathbb{H}$ [46]. In complex analysis, there exists $\mathbb{CR}$ calculus invented specifically to calculate the *pseudo*-derivative of real-

valued complex function due to a sheer number of this type of expression in real-world problems [47]. The $\mathbb{HR}$ calculus was also devised in the same spirit and this section provides necessary concepts used in our algorithms. For a full coverage of the material, please be referred to [42, 48].

2.2.1 Analytic Function in $\mathbb{H}$

Like multiplication, the *true* derivatives of quaternion-valued function have both *left* and *right* forms, respectively given by [49, 50]

$$\lim_{h\to 0}[(f(q+h)-f(q))h^{-1}]$$
$$\lim_{h\to 0}[h^{-1}(f(q+h)-f(q))]$$

where $q, h \in \mathbb{H}$ and $f : \mathbb{H} \to \mathbb{H}$. It is necessary that $f(q)$ must be in the form $\omega q + \lambda$ for the left derivative to exist where $\omega, \lambda \in \mathbb{H}$ and in the form $q\omega + \lambda$ for the right one [50]. Similarly to the complex analysis, the quaternion derivatives are only defined for analytic functions and it was shown that both forms of $f(q)$ are analytic by definition [46]. However, most real-world optimization problems require the objective functions to be real-valued. A particular example, including ours,is the mean square function given by

$$\mathcal{J}(q) = \|f(q)\|^2$$

and it was shown to be not analytic because $\|f(q)\|^2 = f(q)f(q)^*$ and the conjugate of $f(q)$, $f(q)^*$, breaks the analyticity [46]. In order to make sense of the *derivative* of the quaternion function of real value, A new calculus framework was proposed, known as $\mathbb{HR}$ calculus [42, 48].

2.2.2 Isomorphism between $\mathbb{H}$ and $\mathbb{R}^4$

Driven by the desire to make sense of the *derivative* of the real-valued complex function, the $\mathbb{CR}$ calculus was invented through the observation of isomorphism between the fields $\mathbb{C}$ and $\mathbb{R}^2$ [51], resulting in a framework which expands traditional complex analysis

(standard Cauchy-Riemann differentials) to include the differentials of real-valued complex functions [47]. The invention of $\mathbb{HR}$ calculus is motivated by exactly the same reason, the desire to take derivatives of the real-valued function of quaternion variables by establishing the isomorphic relationship (or duality) between the differentials of quaternion-valued functions in $\mathbb{H}$ and those of corresponding *quadrivariate* real-valued functions in $\mathbb{R}^4$. With this, we arrive at a version of hypercomplex calculus which allows taking a derivative of a function of quaternion variables, regardless of its analyticity.

2.2.3 Differentials with Respect to q and q^*

Let q and q^* respectively be an arbitrary quaternion and its corresponding conjugate. The key novelty of $\mathbb{HR}$ calculus (as well as $\mathbb{CR}$ calculus) is the definitions are provided for both the derivatives w.r.t. q and q^*. Unlike the $\mathbb{CR}$ calculus, in the $\mathbb{H}$ field, there are *left* and *right* derivatives of q and q^*.

Definition 1. *([48]) If $f : \mathbb{H} \to \mathbb{H}$ is real-differentiable [50], then the left $\mathbb{HR}$ derivatives of the function f with respect to q and q^* are defined as*

$$\begin{aligned} \frac{\partial_l f}{\partial q} &\triangleq \frac{1}{4}\left(\frac{\partial f}{\partial q_a} - \frac{\partial f}{\partial q_b}\imath - \frac{\partial f}{\partial q_c}\jmath - \frac{\partial f}{\partial q_d}\kappa\right), \\ \frac{\partial_l f}{\partial q^*} &\triangleq \frac{1}{4}\left(\frac{\partial f}{\partial q_a} + \frac{\partial f}{\partial q_b}\imath + \frac{\partial f}{\partial q_c}\jmath + \frac{\partial f}{\partial q_d}\kappa\right), \end{aligned} \tag{2.1}$$

and the right $\mathbb{HR}$ derivatives of the function f with respect to q and q^* are defined as

$$\begin{aligned} \frac{\partial_r f}{\partial q} &\triangleq \frac{1}{4}\left(\frac{\partial f}{\partial q_a} - \imath\frac{\partial f}{\partial q_b} - \jmath\frac{\partial f}{\partial q_c} - \kappa\frac{\partial f}{\partial q_d}\right), \\ \frac{\partial_r f}{\partial q^*} &\triangleq \frac{1}{4}\left(\frac{\partial f}{\partial q_a} + \imath\frac{\partial f}{\partial q_b} + \jmath\frac{\partial f}{\partial q_c} + \kappa\frac{\partial f}{\partial q_d}\right), \end{aligned} \tag{2.2}$$

where $q = q_a + q_b\imath + q_c\jmath + q_d\kappa$ and q_a, q_b, q_c, $q_d \in \mathbb{R}$.

These relationships are not trivial and importantly driven by the involution properties. For a complete treatment of how involution could further generalize the $\mathbb{HR}$ derivatives, please be referred to [42], but eqs. (2.1) and (2.2) suffice for our proposed algorithms.

There is two important simplifications in case of real-valued quaternion function i.e. $f : \mathbb{H} \to \mathbb{R}$. If this is the case, observe that all the sub-

derivatives $\partial f/\partial q_a, \partial f/\partial q_b, \partial f/\partial q_v, \partial f/\partial q_d$ will be real-valued and consequently, eq. (2.1) and eq. (2.2) are identical, meaning the left and right derivatives are equal if f is real. The other simplification is an extension of Brandwood's results from the $\mathbb{CR}$ calculus which states that, for the real-valued function of complex variables, the conjugate derivative yields the maximal change in the optimization space [47]. This also turns out to hold up in $\mathbb{H}$ as well [48]. Ultimately, the $\mathbb{HR}$ derivative for which can be reduced to

$$\frac{\partial f}{\partial q^*} \triangleq \frac{1}{4}\left(\frac{\partial f}{\partial q_a} + \frac{\partial f}{\partial q_b}\imath + \frac{\partial f}{\partial q_c}\jmath + \frac{\partial f}{\partial q_d}\kappa\right). \tag{2.3}$$

Note that the formula resembles the *left* derivative. This is just for ease of expression and does not give priority or connection to *left* derivative and without losing generality, all expressions from this point onwards are assumed with left $\mathbb{HR}$ derivatives.

2.2.4 The Novel Product and Chain Rules

One immediate problem faced by using the $\mathbb{HR}$ calculus is the traditional product rule is no longer valid. With an effort to overcome this, a more generalized product rule was derived [42]; if the functions $f, g : \mathbb{H} \to \mathbb{H}$ are real-differentiable, then so too is their product, that is,

$$\frac{\partial(fg)}{\partial q^*} = f\frac{\partial g}{\partial q^*} + \frac{\partial f}{\partial q^{g*}}g, \tag{2.4}$$

where $q^{g*} = gq^*g^{-1}$ is a quaternion rotation (the right derivative will have a flip of g or f but it is out of focus here). If either f or g is $\mathbb{H} \to \mathbb{R}$, then eq. (2.4) reduces to traditional product rule, which is

$$\frac{\partial(fg)}{\partial q^*} = f\frac{\partial g}{\partial q^*} + \frac{\partial f}{\partial q^*}g. \tag{2.5}$$

So basically, the novel product rule is a generalization of the traditional one. Likewise, the $\mathbb{HR}$ calculus also needs a new, more inclusive chain rule to handle more cases it covers. If $g : \mathcal{S} \to \mathbb{H}$ and $f : \mathcal{T} \to \mathbb{H}$ are real-differentiable at the respective interior points, $q \in \mathcal{S} \subseteq \mathbb{H}$ and $g(q) \in \mathcal{T} \subseteq \mathbb{H}$, then the derivative of the composite function $f(g(q))$ is

given by [42]

$$\begin{aligned}\frac{\partial f(g)}{\partial q^*} &= \frac{\partial f}{\partial g}\frac{\partial g}{\partial q^*} + \frac{\partial f}{\partial g^\imath}\frac{\partial g^\imath}{\partial q^*} + \frac{\partial f}{\partial g^\jmath}\frac{\partial g^\jmath}{\partial q^*} + \frac{\partial f}{\partial g^\kappa}\frac{\partial g^\kappa}{\partial q^*} \\ &= \frac{\partial f}{\partial g^*}\frac{\partial g^*}{\partial q^*} + \frac{\partial f}{\partial g^{\imath *}}\frac{\partial g^{\imath *}}{\partial q^*} + \frac{\partial f}{\partial g^{\jmath *}}\frac{\partial g^{\jmath *}}{\partial q^*} + \frac{\partial f}{\partial g^{\kappa *}}\frac{\partial g^{\kappa *}}{\partial q^*}\end{aligned} \tag{2.6}$$

For the problems we are working in this thesis, the objective is of real value i.e. $f : \mathcal{T} \to \mathbb{R}$ where this chain rule is still valid.

2.3 Stochastic Processes of Quaternion-valued Signals

Now, the focus is once again shifted onto the quaternionic random variables. For a complex random vector, both a correlation matrix and a pseudo-correlation matrix are necessary to fully capture matrix second-order statistics which arises out of a quadratic function of complex random variables. This notion of pseudo-correlation or pseudo-covariance led to the idea of widely linear modelling in order to deal with improper processes, that is, complex-valued random processes whose real and imaginary parts independently follow different probability density functions (PDFs) [39]. In the case of quaternion random variables, the classical correlation matrix is not enough to generally explain each real-valued component [52]

In this thesis, the conditions of $\mathbb{Q}$-properness will be based on [53] for an arbitrary axis and angle of rotation φ, $q \mathrel{\mathring{=}} e^{v\varphi}q \quad \forall\varphi$ for any pure unit quaternion v (the real part equals zero) and $\mathring{=}$ denotes equality in terms of PDF. This section provides important summary of the second-order statistics of quaternion random variables as well as the conditions for a complete description of the second-order statistics of general $\mathbb{Q}$-improper signals. For a full coverage of the topic, please refer to [40].

2.3.1 Augmented Second-order Quaternion Statistics

Similar to the case of complex-valued random processes, in order to exploit complete second-order information, it is necessary to take into account the pseudo-correlation matrices. The standard correlation matrix $\mathbf{R}_{\mathbf{xx}}$ of a quaternion random vector $\mathbf{x} \in \mathbb{H}^n$ is

given by

$$\mathbf{R}_{\mathbf{xx}} = E\{\mathbf{x}\mathbf{x}^H\} \tag{2.7}$$

and its elemental structure is detailed in Table 2.1. Note that the real and imaginary parts of $\mathbf{R}_{\mathbf{xx}}$ are linear combinations of the autocorrelation and cross-correlation matrices of its real-valued component vectors $\mathbf{x}_a$, $\mathbf{x}_b$, $\mathbf{x}_c$ and $\mathbf{x}_d \in \mathbb{R}^n$. From Table 2.1, the cross-correlation matrices have the symmetry property that $\mathbf{R}_{ab}^T = \mathbf{R}_{ba}$. To summarize, $\Re\{\mathbf{R}_{\mathbf{xx}}\}$ is symmetric, whereas $\Im\{\mathbf{R}_{\mathbf{xx}}\}$ is skew-symmetric, indicating that $\mathbf{R}_{\mathbf{xx}}$ is Hermitian [4].

Since the second-order information of the quaternion random vector $\mathbf{x}$ cannot be characterised completely by the standard correlation matrix $\mathbf{R}_{\mathbf{xx}}$ alone [52], it is necessary to define something similar to the pseudo-correlation matrices in a complex domain. However, unlike complex numbers, the quaternions have involution instead of conjugate of the complex. Therefore, the *complementary* correlation matrices [40] were introduced based on the involutions: $\imath$-correlation $\mathbf{R}_{\imath\mathbf{x}}$, $\jmath$-correlation $\mathbf{R}_{\jmath\mathbf{x}}$, and κ-correlation $\mathbf{R}_{\kappa\mathbf{x}}$

$$\mathbf{R}_{\imath\mathbf{x}} = E\{\mathbf{x}^\imath\mathbf{x}^H\} \tag{2.8a}$$

$$\mathbf{R}_{\jmath\mathbf{x}} = E\{\mathbf{x}^\jmath\mathbf{x}^H\} \tag{2.8b}$$

$$\mathbf{R}_{\kappa\mathbf{x}} = E\{\mathbf{x}^\kappa\mathbf{x}^H\} \tag{2.8c}$$

Also, the structure of these complementary correlation matrices are given in table 2.1.

Observe that all components of $\mathbf{R}_{\imath\mathbf{x}}$ are symmetric, except for the i-component $\Im_\imath\{\mathbf{R}_{\imath\mathbf{x}}\}$ which is skew-symmetric, resulting in the i-Hermitian property. This is also similar for $\mathbf{R}_{\jmath\mathbf{x}}$ and $\mathbf{R}_{\kappa\mathbf{x}}$ in tht they are respectively $\jmath$-Hermitian and κ-Hermitian, such that

$$\mathbf{R}_{\imath\mathbf{x}} = \mathbf{R}_{\imath\mathbf{x}}^{\imath H} \tag{2.9a}$$

$$\mathbf{R}_{\jmath\mathbf{x}} = \mathbf{R}_{\jmath\mathbf{x}}^{\jmath H} \tag{2.9b}$$

$$\mathbf{R}_{\kappa\mathbf{x}} = \mathbf{R}_{\kappa\mathbf{x}}^{\kappa H} \tag{2.9c}$$

	$\mathbf{R}_{\mathbf{xx}}$	$\mathbf{R}_{\imath\mathbf{x}}$	$\mathbf{R}_{\jmath\mathbf{x}}$	$\mathbf{R}_{\kappa\mathbf{x}}$
$\Re\{\cdot\}$	$\mathbf{R}_a + \mathbf{R}_b + \mathbf{R}_c + \mathbf{R}_d$	$\mathbf{R}_a + \mathbf{R}_b - \mathbf{R}_c - \mathbf{R}_d$	$\mathbf{R}_a - \mathbf{R}_b + \mathbf{R}_c - \mathbf{R}_d$	$\mathbf{R}_a - \mathbf{R}_b - \mathbf{R}_c + \mathbf{R}_d$
$\Im_\imath\{\cdot\}$	$\mathbf{R}_{ba} - \mathbf{R}_{ab} + \mathbf{R}_{dc} - \mathbf{R}_{cd}$	$\mathbf{R}_{ba} - \mathbf{R}_{ab} - \mathbf{R}_{dc} + \mathbf{R}_{cd}$	$-\mathbf{R}_{ba} - \mathbf{R}_{ab} - \mathbf{R}_{dc} - \mathbf{R}_{cd}$	$-\mathbf{R}_{ba} - \mathbf{R}_{ab} + \mathbf{R}_{dc} + \mathbf{R}_{cd}$
$\Im_\jmath\{\cdot\}$	$\mathbf{R}_{ca} - \mathbf{R}_{ac} + \mathbf{R}_{bd} - \mathbf{R}_{db}$	$-\mathbf{R}_{ca} - \mathbf{R}_{ac} + \mathbf{R}_{bd} + \mathbf{R}_{db}$	$\mathbf{R}_{ca} - \mathbf{R}_{ac} - \mathbf{R}_{bd} + \mathbf{R}_{db}$	$-\mathbf{R}_{ca} - \mathbf{R}_{ac} - \mathbf{R}_{bd} - \mathbf{R}_{db}$
$\Im_\kappa\{\cdot\}$	$\mathbf{R}_{da} - \mathbf{R}_{ad} + \mathbf{R}_{cb} - \mathbf{R}_{bc}$	$-\mathbf{R}_{da} - \mathbf{R}_{ad} - \mathbf{R}_{cb} - \mathbf{R}_{bc}$	$-\mathbf{R}_{da} - \mathbf{R}_{ad} + \mathbf{R}_{cb} + \mathbf{R}_{bc}$	$\mathbf{R}_{da} - \mathbf{R}_{ad} - \mathbf{R}_{cb} + \mathbf{R}_{bc}$

Table 2.1: Structures of quaternion-valued correlation matrices in terms of real-valued quadrivariate counterparts.

Now, we arrive at one of the most important formulae in the quaternion-related algorithms: the augmented correlation matrix, which is actually a correlation matrix of an augmented quaternion random vector $\mathbf{q} := [\mathbf{x}^T \ \mathbf{x}^{\imath T} \ \mathbf{x}^{\jmath T} \ \mathbf{x}^{\kappa T}]^T$, denoted by

$$\mathbf{R}_{\mathbf{qq}} \triangleq E\{\mathbf{q}\mathbf{q}^H\} = \begin{bmatrix} \mathbf{R}_{\mathbf{xx}} & \mathbf{R}^{\imath}_{\imath\mathbf{x}} & \mathbf{R}^{\jmath}_{\jmath\mathbf{x}} & \mathbf{R}^{\kappa}_{\kappa\mathbf{x}} \\ \mathbf{R}_{\imath\mathbf{x}} & \mathbf{R}^{\imath}_{\mathbf{xx}} & \mathbf{R}^{\jmath}_{\kappa\mathbf{x}} & \mathbf{R}^{\kappa}_{\jmath\mathbf{x}} \\ \mathbf{R}_{\jmath\mathbf{x}} & \mathbf{R}^{\imath}_{\kappa\mathbf{x}} & \mathbf{R}^{\jmath}_{\mathbf{xx}} & \mathbf{R}^{\kappa}_{\imath\mathbf{x}} \\ \mathbf{R}_{\kappa\mathbf{x}} & \mathbf{R}^{\imath}_{\jmath\mathbf{x}} & \mathbf{R}^{\jmath}_{\imath\mathbf{x}} & \mathbf{R}^{\kappa}_{\mathbf{xx}} \end{bmatrix}. \tag{2.10}$$

As seen from eq. (2.10), the augmented correlation matrix contains the correlation and all complementary matrices altogether. This implies that it should capture all the second-order statistics of the quaternion variable x. To verify this, we resort to the same idea employed when $\mathbb{HR}$ derivatives were defined; there must be an isomorphism with the corresponding real-valued quadrivariate correlation matrix $\mathbf{R}^r$, defined as

$$\mathbf{R}^r = E\{\mathbf{q}^r\mathbf{q}^{rT}\} = \begin{bmatrix} \mathbf{R}_a & \mathbf{R}_{ab} & \mathbf{R}_{ac} & \mathbf{R}_{ad} \\ \mathbf{R}_{ba} & \mathbf{R}_b & \mathbf{R}_{bc} & \mathbf{R}_{bd} \\ \mathbf{R}_{ca} & \mathbf{R}_{cb} & \mathbf{R}_c & \mathbf{R}_{cd} \\ \mathbf{R}_{da} & \mathbf{R}_{db} & \mathbf{R}_{dc} & \mathbf{R}_d \end{bmatrix}. \tag{2.11}$$

By the concept of linear algebra, this isomorphism between matrices is simply an invertible

matrix factorization which was shown to be [40]

$$\mathbf{R}^r = \frac{1}{16}\mathbf{A}^H\mathbf{R}_{\mathbf{qq}}\mathbf{A} \tag{2.12a}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \imath\mathbf{I} & \jmath\mathbf{I} & \kappa\mathbf{I} \\ \mathbf{I} & \imath\mathbf{I} & -\jmath\mathbf{I} & -\kappa\mathbf{I} \\ \mathbf{I} & -\imath\mathbf{I} & \jmath\mathbf{I} & -\kappa\mathbf{I} \\ \mathbf{I} & -\imath\mathbf{I} & -\jmath\mathbf{I} & \kappa\mathbf{I} \end{bmatrix}, \tag{2.12b}$$

where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix. The augmented second-order statistics introduced can consequently support general second-order modelling of quaternion random processes[1].

2.3.2 Second-order Circularity in $\mathbb{H}$: $\mathbb{Q}$-properness

The second-order circularity (properness) in the complex domain refer to the vanishing pseudocorrelation [54]. Based on [53], a quaternion-valued second-order circular ($\mathbb{Q}$-proper) variable should have equal powers for each component, such that each component is pairwise uncorrelated. All these conditions can be wrapped up into 4 properties below,

$$\begin{array}{lll} \text{P1:} & E\{x_\delta^2\} = \sigma^2, & \forall\delta = a,b,c,d \\ \text{P2:} & E\{x_\delta x_\varepsilon\} = 0, & \forall\delta,\varepsilon = a,b,c,d \text{ and } \delta \neq \varepsilon \\ \text{P3:} & E\{xx\} = -2E\{x_\delta^2\} = -2\sigma^2, & \forall\delta = a,b,c,d \\ \text{P4:} & E\{xx^*\} = 4E\{x_\delta^2\} = 4\sigma^2, & \forall\delta = a,b,c,d \end{array}$$

The first property, P1, states that all real-valued component signals of a quaternion random variable have equal variance (power). The property P2 implies that the cross-component signals of x are uncorrelated. Property P3 indicates that the pseudo-correlation matrix of a $\mathbb{Q}$-proper variable does not vanish (in stark contrast to the complex case [54]). The last property states that the total power of a quaternion random variable is the sum of the power of each component. Note that properties P1 and P2 yield P3 and P4.

From the properties above together with the relations between real-valued components of a quaternion and those of involution counterparts in the previous, a $\mathbb{Q}$-proper

[1]Note that $\mathbf{A}^{-1} = \frac{1}{4}\mathbf{A}^H$.

quaternion random vector $\mathbf{x} \in \mathbb{H}^n$ is not correlated with its involutions $\mathbf{x}^\imath$, $\mathbf{x}^\jmath$, $\mathbf{x}^\kappa$, that is,

$$\mathbf{R}_{\imath\mathbf{x}} = \mathbf{0} \qquad \mathbf{R}_{\jmath\mathbf{x}} = \mathbf{0} \qquad \mathbf{R}_{\kappa\mathbf{x}} = \mathbf{0} \tag{2.13}$$

Thus such a vector has vanishing complementary correlation matrices (specified in table 2.1). Therefore, for a $\mathbb{Q}$-proper quaternion random vector, it follows that the standard correlation matrix eq. (2.5) is real-valued, diagonal, and positive-semidefinite, while the complementary correlation matrices in eq. (2.6) are zero matrices [40]. This makes the augmented correlation matrix $\mathbf{R}_{\mathbf{qq}}$ to have much simpler form, which is identity, as

$$\mathbf{R}_{\mathbf{qq}} = \begin{bmatrix} \mathbf{R}_{\mathbf{xx}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^\imath_{\mathbf{xx}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}^\jmath_{\mathbf{xx}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}^\kappa_{\mathbf{xx}} \end{bmatrix} = 4\sigma^2\mathbf{I}, \tag{2.14}$$

where σ is the power of each component of the quaternion. For more detail about the $\mathbb{Q}$-properness of a quaternion random variable, please see [40].

2.3.3 Quaternion Widely Linear Model

If the augmentation is the basic building block of our quaternion-related algorithms, then the quaternion widely linear (QWL) model is the prime machinery utilizing the building block to create our finalized algorithms. The widely linear model is a form of linear regression which takes into account each real-valued component of the complex-/quaternion-valued random vectors. To illustrate how the QWL model can capture the augmented second-order information established so far, consider the mean square error (MSE) estimate y of the desired signal d via the observed inputs $\mathbf{x}$. In probability, the estimate y that minimizes the MSE error is the conditional expectation [55]

$$y = E\{d|\mathbf{x}\}.$$

Note that the noise is absorbed into the desired signal d for ease of presentation, and since the noise is assumed i.i.d. Gaussian, its expectation w.r.t. the observed inputs, $\mathbf{x}$, is zero. As the main focus is on a *linear* model, only the *linear* MSE estimator was employed in this thesis, and is given by

$$y = \mathbf{w}^T \mathbf{x} \tag{2.15}$$

where here $\mathbf{w} \in \mathbb{H}^n$ is a vector of filter coefficients and $\mathbf{x} \in \mathbb{H}^n$ is an input vector. The estimator in eq. (2.15) is only valid for a $\mathbb{Q}$-proper variable [40], and needs to be expanded to cover every component of a quaternion variable for a complete second-order statistics explained in the previous sections. In terms of probability, the standard estimator $y = E\{d|\mathbf{x}\}$ shall span the condition into

$$y_\delta = E\{d_\delta|\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d\}, \qquad \delta \in \{a, b, c, d\}$$

Hence, the *widely* MSE estimator of a quaternion random vector becomes

$$\begin{aligned} y =& E\{d_a|\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d\} + E\{d_b|\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d\}\imath \\ &+ E\{d_c|\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d\}\jmath + E\{d_d|\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d\}\kappa, \end{aligned} \tag{2.16}$$

With the isomorphism between real-valued components of a quaternion and their quaternion-valued involution counterparts, the real components of quaternions in eq. (2.16) can be replaced by the quaternions themselves and their involutions. The *widely* MSE estimator can have an alternative form as

$$\begin{aligned} y =& E\{d|\mathbf{x}, \mathbf{x}^\imath, \mathbf{x}^\jmath, \mathbf{x}^\kappa\} + E\{d^\imath|\mathbf{x}, \mathbf{x}^\imath, \mathbf{x}^\jmath, \mathbf{x}^\kappa\}\imath \\ &+ E\{d^\jmath|\mathbf{x}, \mathbf{x}^\imath, \mathbf{x}^\jmath, \mathbf{x}^\kappa\}\jmath + E\{d^\kappa|\mathbf{x}, \mathbf{x}^\imath, \mathbf{x}^\jmath, \mathbf{x}^\kappa\}\kappa, \end{aligned} \tag{2.17}$$

which can be linearized into the corresponding QWL model as follows [40]

$$\begin{aligned} y &= \mathbf{u}^T\mathbf{x} + \mathbf{v}^T\mathbf{x}^\imath + \mathbf{g}^T\mathbf{x}^\jmath + \mathbf{h}^T\mathbf{x}^\kappa \\ &= \mathbf{w}^T\mathbf{q} \end{aligned} \tag{2.18}$$

where $\mathbf{w} = [\mathbf{u}^T\ \mathbf{v}^T\ \mathbf{g}^T\ \mathbf{h}^T]^T$ and $\mathbf{q} = [\mathbf{x}^T\ \mathbf{x}^{\imath T}\ \mathbf{x}^{\jmath T}\ \mathbf{x}^{\kappa T}]^T$. The Wiener solution, $\mathbf{w}_{op}$, which minimizes the MSE, $E\{|d-y|^2\}$, based on the QWL model in eq. (2.16) is given by [11,40]

$$\mathbf{w}_{op}^* = \mathbf{R}_{\mathbf{qq}}^{-1}\mathbf{r}_{\mathbf{q}d}, \tag{2.19}$$

where $\mathbf{R}_{\mathbf{qq}} = E\{\mathbf{q}\mathbf{q}^H\}$, $r_{\mathbf{q}d} = E\{\mathbf{q}d^*\}$ and $(\cdot)^*$ is the (quaternion) conjugate operator. An interesting point is that eq. (2.19) is also valid for the real and complex domains, but the augmented correlation matrix $\mathbf{R}_{\mathbf{qq}}$ in eq. (2.10) will vary with respect to the specific domains. For $\mathbb{Q}$-proper signals, the augmented correlation matrix becomes identity eq. (2.14), simplifying eq. (2.19)

$$\mathbf{w}_{op}^* = \frac{1}{4\sigma^2}\mathbf{r}_{\mathbf{q}d}. \tag{2.20}$$

The QWL Wiener solution in eq. (2.19) is particularly important in deriving the WL-QRLS algorithm [8, 11]. Nevertheless, the method has been shown multiple times to be numerically unstable [1,2,8,11]. Fortunately, with the advent of the $\mathbb{HR}$ calculus, there is a leeway to acquire as powerful algorithms without resorting to eq. (2.19), our quaternion adaptive filters proposed in this thesis.

2.4 Summary

In this chapter, we have outlined all the relevant basics of a quaternion and its analysis relating to adaptive filtering like $\mathbb{HR}$ calculus and QWL models. These tools will allow us to make sense of the proposed quaternion adaptive filters to be expounded in chapter 3.

Chapter 3

Quaternion-Valued Adaptive Filters Based on Extrapolated Gradient Methods

THE topic of quaternion-valued adaptive signal processing first took shape when the quaternion LMS (QLMS) adaptive filters were first introduced in [56] for moving average (MA) and autoregressive (AR) types of linear models. The ARMA model followed

quickly [57]. These simplest forms of quaternion-valued adaptive filters were already shown to exceed both bivariate complex LMS and quadrivariate LMS for multichannel data processing in terms of performance. This is because the quaternion algebra naturally accounts for the coupling between the signal components, resulting in a superior data fusion [58]. These papers neither took into account the widely linear modelling, nor properly captured second-order statistics until it was formally introduced [14]. Despite many potential research topics available in quaternions like quaternionic frequency domain [59], quaternion wavelet [60], etc, our effort is to build on the exiting knowledge of the quaternion-valued adaptive signal processing in pursuit of better and more versatile methods.

3.1 Accelerated Gradient Methods

The QLMS algorithms employ the instantaneous error to calculate the stochastic gradient. While this makes the QLMS convergent robustly and versatile in tracking a variety of signals whether it is stationary or not, its limitations comes in when the signals become complicated like non-linear signals [39], and its rather slow convergence [14]. On the other hand, we have QRLS-type algorithms inspired from the real-valued RLS methods. This algorithm, if able to, can cope with a little more complicated signals than the QLMS with much faster convergence. However, their numerical robustness is a real concern where they have tendency to diverge more frequent than the LMS [5], and the situation was shown to go worse in complex RLS [39] and further downhill in QRLS [1, 2]. These observation has been empirically, but rigorous investigation is still lacking. In this thesis, however, we took this notion as an underlying caveat and proceeded to avoid matrix inversion at all cost as it is largely believed to be the source of numerical instability [5]. In this section, we extend the methods of heavy-ball gradient descents [61] to the quaternion domains by offering two *accelerated gradient descent* schemes: the one inspired by Nesterov's optimal method [62] and by Chebyshev's iterative method [63], called n-moment and c-moment algorithms, respectively. These are key to the derivation of the proposed algorithms.

3.2 Quaternion-Valued Adaptive Filters with Extrapolated Gradient

Now, the main algorithms are proposed in 2 alternative schemes: n-moment and c-moment. First, the algorithm is derived for the exact optimal gradient with some adjustments to suit stochastic cost function instead of classical deterministic function, leading to the n-moment algorithm. Then, the simplified scheme, the n-moment algorithm, which interestingly resembles conjugate gradient descent is rendered. The results in this chapter has been published in IEEE proceedings and the interested readers should refer to [1, 2]

3.2.1 Accelerated Gradient Descents - Generic Statement

In real domains, it was proved that when the objective function to be minimized is a convex quadratic function, Nesterov's accelerated gradient simplifies to the conjugate gradient descent [61], an optimal form of Chebyshev's iterative method. This is however proved on the basis of deterministic function, but our problem is a stochastic one. Generally, most optimization techniques would perform not so differently from the deterministic cases *in case of white Gaussian error* [5, 55] where our problem lies. However, the Nesterov's algorithm is probably among the exception as its convergence guarantee has only been verified for deterministic cost function [62]. Also, the analysis to connect these two algorithms in quaternion and stochastic setting has proven challenging for us so far. Nevertheless, we managed to provide *sketch* proof of how the n-moment algorithm converges locally and how it relates to the c-moment algorithm, which will be provided later in this chapter. Moreover, extensive empirical simulations seem to validate our hypothesis as both of our proposed methods perform on par with each other in almost all experiment settings.

Before embarking on the main content, we shall begin with the notion of quaternion gradient. Consider a function $f(\mathbf{q}) : \mathbb{H}^{M\times 1} \to \mathbb{H}$, where $\mathbf{q} = (q_1, q_2, ..., q_M)^T \in \mathbb{H}^M$. Then, the quaternion gradient and conjugate gradient are respectively given by [42]

$$\nabla_{\mathbf{q}} f \triangleq \frac{\partial f}{\partial \mathbf{q}} = \left(\frac{\partial f}{\partial q_1}, ..., \frac{\partial f}{\partial q_M} \right)^T \in \mathbb{H}^M,$$
$$\nabla_{\mathbf{q}^*} f \triangleq \frac{\partial f}{\partial \mathbf{q}^*} = \left(\frac{\partial f}{\partial q_1^*}, ..., \frac{\partial f}{\partial q_M^*} \right)^T \in \mathbb{H}^M.$$

As clearly explained in the previous chapter, the conjugate derivative will yield the steepest descent direction of the function f. Moreover, since our cost function of interest is of real value, the concern for left or right derivative also vanishes. Now, let's state the generic problem. Consider the signals $y_n, x_n \in \mathbb{H}$, $n = 1, ..., N$ as the output and input signals, respectively. Hence, the widely linear estimator of y_n, $\hat{y}_n$, can be expressed as [1, 2, 55]

$$\hat{y}_n = \langle \hat{\mathbf{w}}, \mathbf{q}_n \rangle \triangleq \hat{\mathbf{w}}^H \mathbf{q}_n \tag{3.1}$$

where $\hat{\mathbf{w}}$ is an estimate of the optimal solution $\mathbf{w} \in \mathbb{H}^{4M}$, and $\mathbf{q}_n \in \mathbb{H}^{4M}$ is the augmented input signal defined as

$$\mathbf{q}_n = [\mathbf{x}_n^T,\ \mathbf{x}_n^{\imath T},\ \mathbf{x}_n^{\jmath T},\ \mathbf{x}_n^{\kappa T}]^T \tag{3.2}$$

and $\mathbf{x}_n = [x_n, x_{n-1}, ..., x_{n-M+1}]^T$ is an input vector of filter order M. In case of a strictly linear model, $\mathbf{q}_n := \mathbf{x}_n$ (i.e. data is circular [40] and all complementary parts vanish). By the probability theory, this linear estimator minimizes the mean square error given by

$$\mathcal{J}_n(\hat{\mathbf{w}}) = E\{\|e_n(\hat{\mathbf{w}})\|_2^2\} \tag{3.3a}$$

$$e_n(\hat{\mathbf{w}}) \triangleq y_n - \hat{y}_n = y_n - \hat{\mathbf{w}}^H \mathbf{q}_n. \tag{3.3b}$$

In the adaptive filtering framework, a stochastic optimization is performed on the MSE eq. (3.3) as the cost function. The utilization of accelerated gradient methods would result in the generic heavy-ball recursive update below [64]

$$\mathbf{w}_n = \mathbf{w}_{n-1} - \left(\nabla_{\hat{\mathbf{w}}^*} \mathcal{J}_n(\hat{\mathbf{w}}) |_{\hat{\mathbf{w}}=\mathbf{v}} \right) \alpha_n + (\mathbf{w}_{n-1} - \mathbf{w}_{n-2}) \beta_n \tag{3.4}$$

where $\mathbf{w}_n$ is the value of $\hat{\mathbf{w}}$ at the epoch n, $\mathbf{v}$ is a point on $\mathbb{H}^M$ space which lies within

the closed line segment $\mathcal{V}$, i.e. $\mathbf{v} \in \mathcal{V}$, parameterized as

$$\mathcal{V} = \{(\nu + 1)\mathbf{w}_{n-1} + \nu(\mathbf{w}_{n-1} - \mathbf{w}_{n-2})\beta_n | \nu \in [0, 1] \subset \mathbb{R}\},$$

and α_n, β_n are quaternion scalars. Here in eq. (3.4), the second term of the RHS is the gradient trem while the rightmost term is the momentum (extrapolation) term. α_n is called a stepsize and β_n is an extrapolation ratio. As considering the quaternion domains, we maintain the *right* constant rule. This will be obvious throughout this chapter how the quaternionic Sylvester's equation is avoided so that we could obtain the closed-form recursive formulae as a result.

Now, eq. (3.4) can be re-formulated into 2 distinct notions in terms of how the extrapolation takes place, at the point of descent, or at the gradient descent. The former hence has the rightmost term of eq. (3.4) collapses into $\mathbf{w}_{n-1}$, to yield

$$\mathbf{v}_n = \mathbf{w}_{n-1} + (\mathbf{w}_{n-1} - \mathbf{w}_{n-2})\beta_n, \tag{3.5a}$$

$$\mathbf{w}_n = \mathbf{v}_n - \left(\nabla_{\hat{\mathbf{w}}^*}\mathcal{J}_n(\hat{\mathbf{w}})|_{\hat{\mathbf{w}}=\mathbf{v}_n}\right)\alpha_n. \tag{3.5b}$$

The expression in eq. (3.5) is the main recursion of the proposed n-moment gradient method. On the other hand, the c-moment gradient method has the extrapolation implemented at the gradient term of eq. (3.4) by collapsing the rightmost term into

$$\mathbf{d}_n\alpha_n \triangleq \mathbf{w}_n - \mathbf{w}_{n-1}, \tag{3.6}$$

and we thus arrive at the following expression

$$\mathbf{d}_n = -\nabla_{\hat{\mathbf{w}}^*}\mathcal{J}_n(\hat{\mathbf{w}})|_{\hat{\mathbf{w}}=\mathbf{w}_{n-1}} + \mathbf{d}_{n-1}\beta_n, \tag{3.7a}$$

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{d}_n\alpha_n. \tag{3.7b}$$

where $\mathbf{d}_n$ is the extrapolated descent direction and β_n absorbs extra constants α_n, α_{n-1} for lean formulae. Note that eq. (3.5) differs from eq. (3.7) not only in terms of formulae, but also in the argument of the gradient $\nabla_{\hat{\mathbf{w}}^*}\mathcal{J}_n(\hat{\mathbf{w}})$ where the n-moment method uses

as $\hat{\mathbf{w}}$ eq. (3.5a), while the c-moment one uses $\mathbf{w}_{n-1}$. This distinction actually transpires from how the underlying ODE of eq. (3.4) is discretized [63].

As stated above, the right constant rule is maintained for the scalars $\alpha_n, \beta_n \in \mathbb{H}$ to avoid quaternion Sylvester's equation which could destroy the analytical form our our recursion and create nested algorithm problem [44]. In the next section, we will focus on the main research results of our quaternion-valued adaptive signal processing.

3.2.2 The Conjugate Gradient of WL-QLMS Algorithm with Memory

We named our proposed methods as QLMS due to the cost function to be minimized is off eq. (3.3), which is by definition a least mean squares [5]. However, not only instantaneous input is used to calculate the gradient in the n-moment algorithm; instead, the RLS-like cost function is utilized as an approximate to the cost in eq. (3.3a), that is

$$\mathcal{J}_n(\hat{\mathbf{w}}) \approx \frac{\Phi_n(\hat{\mathbf{w}})}{\sum\limits_{k=1}^{n} \lambda^{n-k}} = \frac{\sum\limits_{k=1}^{n} \lambda^{n-k} \|e_k(\hat{\mathbf{w}})\|^2}{\sum\limits_{k=1}^{n} \lambda^{n-k}} \tag{3.8}$$

where $\lambda \in (0, 1)$ is a real-valued forgetting factor used to suppress the effect of early data which may no longer contribute to the data of the current epoch. Here, if $\lambda = 0$, then the cost function reduces to a standard instantaneous error, the normal QLMS. Observe eq. (3.8) that its denominator is constant w.r.t. the gradient operand $\hat{\mathbf{w}}$, and thus could be left out in our derivation as it will be absorbed into any constants left in our final recursion. Consequently, $\Phi_n(\hat{\mathbf{w}})$ will be used as our cost function in our quaternion research. Now, we can rewrite $\Phi_n(\hat{\mathbf{w}})$ as

$$\Phi_n(\hat{\mathbf{w}}) = \hat{\mathbf{w}}^H \mathbf{R}_n \hat{\mathbf{w}} - 2\Re\{\hat{\mathbf{w}}^H \mathbf{r}_n\} + \sum_{k=1}^{n} \lambda^{n-k} |y_k|^2 \tag{3.9}$$

where $\Re\{\cdot\}$ is an operator selecting only real part, and

$$\mathbf{R}_n = \sum_{k=1}^{n} \lambda^{n-k} \mathbf{q}_k \mathbf{q}_k^H = \lambda \mathbf{R}_{n-1} + \mathbf{q}_n \mathbf{q}_n^H, \tag{3.10}$$

$$\mathbf{r}_n = \sum_{k=1}^{n} \lambda^{n-k} \mathbf{q}_k y_k^* = \lambda \mathbf{r}_{n-1} + \mathbf{q}_n y_n^*. \tag{3.11}$$

We then arrive at one of our main results of this thesis, the $\mathbb{HR}$ gradient of the WL-QLMS algorithms with extrapolation

Theorem 1. *Let $\hat{\mathbf{w}} \in \mathbb{H}^M$ be a quaternion vector and $\Phi_n(\hat{\mathbf{w}}) : \mathbb{H}^M \to \mathbb{R}$, then, the gradient of $\Phi_n(\hat{\mathbf{w}})$, denoted by $\mathbf{g}_n(\hat{\mathbf{w}}) \in \mathbb{H}^M$, can be recursively given by*

$$\mathbf{g}_n(\hat{\mathbf{w}}) \triangleq 2\nabla_{\hat{\mathbf{w}}^*}\Phi_n(\hat{\mathbf{w}}) = \sum_{k=1}^{n} \lambda^{n-k}\mathbf{q}_k e_k^*(\hat{\mathbf{w}}) = \lambda\mathbf{g}_{n-1}(\hat{\mathbf{w}}) + \mathbf{q}_n e_n^*(\hat{\mathbf{w}}) \tag{3.12}$$

Proof. By the above expression of quaternion gradient, we have

$$\nabla_{\hat{\mathbf{w}}^*}\Phi_n(\hat{\mathbf{w}}) = \frac{\partial\Phi_n(\hat{\mathbf{w}})}{\partial\mathbf{q}^*}$$

and by the novel product rule given by eq. (2.4), we have

$$\frac{\partial\Phi_n(\hat{\mathbf{w}})}{\partial\mathbf{q}^*} = \sum_{k=1}^{n} \lambda^{n-k} e_k^*(\hat{\mathbf{w}})\frac{\partial e_k(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}^*} + \frac{\partial e_k^*(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}^{e_k*}} e_k(\hat{\mathbf{w}})$$

Now, with the derivative rule of $\mathbb{HR}$ calculus [42] and eq. (3.3b), the above derivative terms are calculated as

$$e_k^*(\hat{\mathbf{w}})\frac{\partial e_k(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}^*} = e_k^*(\hat{\mathbf{w}})\frac{\partial(y_k - \hat{\mathbf{w}}^H\mathbf{q}_k)}{\partial\hat{\mathbf{w}}^*} = -e_k^*(\hat{\mathbf{w}})\frac{\partial\hat{\mathbf{w}}^H\mathbf{q}_k}{\partial\hat{\mathbf{w}}^*} = -e_k^*(\hat{\mathbf{w}})\Re(\mathbf{q}_k)$$

and

$$\frac{\partial e_k^*(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}^{e_k*}} e_k(\hat{\mathbf{w}}) = \frac{\partial(y_k^* - \mathbf{q}_k^H\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}^{e_k*}} e_k(\hat{\mathbf{w}}) = -\frac{\partial\mathbf{q}_k^H\hat{\mathbf{w}}}{\partial\hat{\mathbf{w}}^{e_k*}} e_k(\hat{\mathbf{w}}) = \frac{1}{2}\mathbf{q}_k^* e_k^*(\hat{\mathbf{w}})$$

Finally, we arrive at

$$\begin{aligned}
\frac{\partial\Phi_n(\hat{\mathbf{w}})}{\partial\mathbf{q}^*} &= \sum_{k=1}^{n} \lambda^{n-k}\frac{1}{2}\mathbf{q}_k^* e_k^*(\hat{\mathbf{w}}) - e_k^*(\hat{\mathbf{w}})\Re(\mathbf{q}_k) \\
&= \sum_{k=1}^{n} \lambda^{n-k}\left(\frac{1}{2}\mathbf{q}_k^* - \Re(\mathbf{q}_k)\right) e_k^*(\hat{\mathbf{w}}) \\
&= \frac{1}{2}\sum_{k=1}^{n} \lambda^{n-k}\mathbf{q}_k e_k^*(\hat{\mathbf{w}}) \\
&= \frac{1}{2}\left(\sum_{k=1}^{n-1} \lambda^{n-k}\mathbf{q}_k e_k^*(\hat{\mathbf{w}}) + \mathbf{q}_n e_n^*(\hat{\mathbf{w}})\right) \\
&= \frac{1}{2}\left(\lambda\mathbf{g}_{n-1}(\hat{\mathbf{w}}) + \mathbf{q}_n e_n^*(\hat{\mathbf{w}})\right)
\end{aligned}$$

which concludes our result. □

3.2.3 The n-Moment WL-QLMS Algorithm

We will now derive the first proposed method which is inspired by the classical Nesterov's optimal gradient descent. In classical optimization, the optimal method utilizes the notions of *strong convexity* and *estimate sequences*, given respectively by

Definition 2. *A continuously differentiable function $f(\mathbf{w}) : \mathbb{R}^M \to \mathbb{R}$ is strongly convex on $\mathbb{R}^M$ if there exists a constant $\sigma > 0$ such that for any $\mathbf{w}, \mathbf{v} \in \mathbb{R}^M$, we have*

$$f(\mathbf{v}) \geq f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), \mathbf{v} - \mathbf{w} \rangle + \frac{\sigma}{2} \|\mathbf{v} - \mathbf{w}\|^2$$

Definition 3. [62] *For any $\mathbf{w} \in \mathbb{R}^M$ and all $n \geq 0$, a pair of estimate sequences of $f(\mathbf{w}) : \mathbb{R}^M \to \mathbb{R}$, $\{\phi_n(\mathbf{w}) : \mathbb{R}^M \to \mathbb{R}\}_{n=0}^{\infty}$ and $\{\eta_n \in (0,1)\}_{n=0}^{\infty}$, satisfies, if $\eta_n \to 0$,*

$$\phi_n(\mathbf{w}) \leq (1 - \eta_n) f(\mathbf{w}) + \eta_n \phi_0(\mathbf{w})$$

Due to all these real-valued sequences despite quaternion-valued input vectors, this definition can be used to derive the quaternion version of the traditional Nesterov optimal method almost trivially the same way as in the original paper [62], except for the right constant rule which is strictly enforced throughput the derivation. However, in our problem, the function to be minimized, $\Phi(\mathbf{w})$, is stochastic. The analysis to date has been challenging and found limited success, not only by us but also across topic enthusiasts. From the perspective of estimate sequences, β_n in eq. (3.4) would form a part if an iterative equation involving η_n, which is real-valued. Therefore, β_n by definition 3 will be of real value too. This enables Nesterov's formulae to be straightforwardly applied to our n-moment algorithm, yielding

$$\beta_n = \frac{\eta_{n-1}(1 - \eta_{n-1})}{(\eta_{n-1})^2 + \eta_n} \tag{3.13}$$

with $\eta_n \in (0,1)$, such that

$$(\eta_n)^2 = (1-\eta_n)(\eta_{n-1})^2 + \left(\frac{\sigma_n}{L_n}\right)\eta_n \tag{3.14}$$

where

$$\sigma_n = \|\lambda_{\min}(\mathbf{R}_n)\|_2 \tag{3.15}$$

and

$$L_n = \|\lambda_{\max}(\mathbf{R}_n)\|_2. \tag{3.16}$$

The $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ respectively represent the minimum and maximum eigenvalues of $\mathbf{R}_n$. Together with the recursions eqs. (3.5), (3.10) and (3.11) and our $\mathbb{HR}$ gradient result in eq. (3.12), the update equation for n-moment algorithm can be expressed by

$$\mathbf{v}_n = \mathbf{w}_{n-1} + (\mathbf{w}_{n-1} - \mathbf{w}_{n-2})\beta_n, \tag{3.17a}$$

$$\check{\mathbf{g}}_n \triangleq \mathbf{R}_n\mathbf{v}_n - \mathbf{r}_n, \tag{3.17b}$$

$$\mathbf{w}_n = \mathbf{v}_n - \check{\mathbf{g}}_n\alpha_n. \tag{3.17c}$$

These equations comprise the n-moment WL-QLMS algorithm. Its complete algorithm is summarized in the following section.

3.2.4 The Sketched Analysis of Convergence of the n-moment method

The convergence analysis of the n-moment algorithm would be self-proven in its derivation (similar to the original analysis [62]) if the objective $\Phi_n(\hat{\mathbf{w}})$ is deterministic, that is, $\mathbf{R}_n$ and $\mathbf{r}_n$ are constants. Our true struggle during studying this topic is to extend the analysis to a stochastic case where the sense of optimality is also compromised. The essence of convergence analysis is to ensure that the objective function decreases with time in a statistical sense. While certainly $\Phi_n(\mathbf{w}_n) < \Phi_n(\mathbf{v}_n)$ due to exact local gradient descent, it is not guaranteed that $\Phi_n(\mathbf{w}_n) < \Phi_n(\mathbf{w}_{n-1})$, an important condition to prove the convergence of the algorithm. While the analysis of the deterministic case is self-validated via the way the algorithm is derived from estimate sequences, the case of stochastic cost

function is still limited. To this end, we developed a heuristic method to ensure that $\Phi_n(\mathbf{w}_n) < \Phi_n(\mathbf{w}_{n-1})$ by resetting β_n whenever this condition is violated. to put it into algorithmic practice, we utilized 2 checking conditions: gradient check [65]:

$$\langle \mathbf{w}_n - \mathbf{v}_n, \mathbf{w}_n - \mathbf{w}_{n-1} \rangle > 0, \tag{3.18}$$

and contraction mapping [66]:

$$\|\mathbf{w}_n - \mathbf{v}_n\|_2 < \epsilon_0 \|\mathbf{w}_{n-1} - \mathbf{v}_{n-1}\|_2. \tag{3.19}$$

The first condition eq. (3.18) is necessary to accommodate our convergence proof later on in this chapter. It basically mandates that the effective gradient of the recursion, $\mathbf{w}_n - \mathbf{w}_{n-1}$, is a descent direction i.e. positive inner product with $\check{\mathbf{g}}_n$. Nevertheless, this poses a challenging interpretation of what is a *positive quaternion*. In complex cases, it is suggested that the positive real part of a complex product would yield the same desired outcome [65]. As there has been no similar recommendation made for quaternions, it is, at the first glace, better to play safe and resort to eq. (3.19) to guarantee the convergence of n-moment algorithm. Moreover, we also observed that the convergence rate of the algorithm was very inconsistent across trials and samples. It is obvious that as the objective is no longer deterministic, β_n is also no longer critically optimal; instead, it can cause underdamped convergence trajectory by overestimating the extrapolation step [62], making the algorithm performs inefficiently. To prevent this, we need to ensure that β_n lies within the range that makes $\Phi_n(\mathbf{w}_{n-1}) < \Phi_n(\mathbf{w}_n)$ by significant *order of growth*. In the original Nesterov's method, second-order growth condition,

$$(\epsilon_0/2)\|\mathbf{w}_n - \mathbf{w}_{n-1}\|^2 < \Phi_n(\mathbf{w}_{n-1}) - \Phi_n(\mathbf{w}_n), \tag{3.20}$$

is naturally employed as a result of $\Phi_n(\hat{\mathbf{w}})$ being assumed as strongly convex function definition 2. With $\Phi_n(\hat{\mathbf{w}})$ given in eq. (3.9), we can manipulate eqs. (3.19) and (3.20) to yield the following theorem [1]

Theorem 2. *For the stepsize $\alpha_n < 1/L_n$, the conditions in eqs.* (3.19) *and* (3.20) *are*

satisfied if

$$\beta_n < \sqrt{\frac{1-\alpha_n\epsilon_0}{1-\alpha_n\sigma_n}}\frac{\|\mathbf{w}_n-\mathbf{w}_{n-1}\|_2}{\|\mathbf{w}_{n-1}-\mathbf{w}_{n-2}\|_2} \tag{3.21}$$

The expression in theorem 2 is elegant because not only does it succinctly combine two convergence criteria (eqs. (3.19) and (3.20)) in one compact formula, but we also arrive at the algorithm guaranteed to converge at the rate of second order. Nevertheless, to this point, it may beg a question that while these additional expressions may not worth the convergence speed gained from them and whether we could devise an algorithm comparable in terms of performance but less algorithmic steps involved. Consequently, we proposed another method, the c-moment WL-QLMS algorithm, which has even more succinct formulae while empirically achieving comparable performance.

3.2.5 The c-Moment WL-QLMS Algorithm

The "c" connotes the Chebyshev's iterative method which underpins eq. (3.7), the main equations for the c-moment algorithm. This algorithm has more succinct formulae and more computational friendly than the n-moment one. This results from the use of dual recursion where all parameters will be calculated from their past values. This is not the case in the n-moment algorithm because the explicit extrapolation precludes the iterative computation of the gradient $\check{\mathbf{g}}_n$ in eq. (3.17b) and therefore requires full calculation in every iteration. From this perspective, a huge computational reduction in c-moment algorithm can be seen and it starts by finding the optimal value of the stepsize α_n via the following equation:

$$\frac{\partial\Phi_n(\mathbf{w}_n)}{\partial\alpha_n} = 0 \tag{3.22}$$

and substituting eq. (3.7b) into eq. (3.22), we have

$$\alpha_n = -\frac{\langle\mathbf{d}_n, \mathbf{g}_n(\mathbf{w}_{n-1})\rangle}{\langle\mathbf{d}_n, \mathbf{h}_{n|n}\rangle} \tag{3.23}$$

where

$$\mathbf{h}_{n|n} \triangleq \mathbf{R}_n\mathbf{d}_n. \tag{3.24}$$

Observe that $\langle \mathbf{d}_n, \mathbf{h}_{n|n} \rangle$ is real-valued and thus no left or right constant rule applicable. Now, to reuse past calculations, we need to give distinction between *a priori* and *a posteriori* gradients, $\mathbf{g}_{n|n-1}$ and $\mathbf{g}_{n|n}$ respectively, as

$$\mathbf{g}_{n|n-1} \triangleq \mathbf{g}_n(\mathbf{w}_{n-1}), \tag{3.25}$$

and

$$\mathbf{g}_{n|n} \triangleq \mathbf{g}_n(\mathbf{w}_n). \tag{3.26}$$

Then, with eqs. (3.7b) and (3.12), we obtain first pair of dual recursion, the *gradient recursion*, given by

$$\mathbf{g}_{n|n-1} = \lambda \mathbf{g}_{n-1|n-1} - \mathbf{q}_n e_n^*, \tag{3.27}$$

$$\mathbf{g}_{n|n} = \mathbf{g}_{n|n-1} + \mathbf{h}_{n|n} \alpha_n. \tag{3.28}$$

It is obvious that by shifting the extrapolation into the gradient rather than the argument variable, we gain seamless expression which eases the computation greatly. this would conclude our algorithm unless the extrapolated descent direction $\mathbf{d}_n$ is used. We see from the previous section that although we gain better convergence, the additional computational complexity outweighs its main utility due to many extra steps to ensure its robustness. In this algorithm, we would like to give the structure of $\mathbf{d}_n$ in eq. (3.7a) from the outset. To this end, we introduced a novel property to force the behavior of $\mathbf{d}_n$ and it is called *Markov conjugacy* [2], that is

Definition 4. *A set of descent directions* $\{\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_n\}$ *is Markov conjugate if, at any iteration i,*

$$\mathbf{d}_k^H \mathbf{R}_k \mathbf{d}_{k-1} = 0 \text{ for } k = 2, 3, ..., n. \tag{3.29}$$

Note that this definition resembles the conjugate gradient condition on a traditional optimization framework. Now, we pre-multiply eq. (3.7a) with $(\mathbf{d}_{n-1})^H \mathbf{R}_n$ and substitute eq. (3.29) into eq. (3.7a) to give the following formulae,

$$\beta_n = \frac{\langle \mathbf{d}_{n-1}, \mathbf{v}_n \rangle}{\langle \mathbf{d}_{n-1}, \mathbf{h}_{n|n-1} \rangle} \tag{3.30}$$

where

$$\mathbf{h}_{n|n-1} = \mathbf{R}_n \mathbf{d}_{n-1} \tag{3.31}$$

$$\mathbf{v}_n \triangleq \mathbf{R}_n \mathbf{g}_{n|n-1}. \tag{3.32}$$

and after some manipulation, we obtain the second dual recursion, the *gain vector recursion*

$$\mathbf{h}_{n|n-1} = \lambda \mathbf{h}_{n-1|n-1} + \mathbf{q}_n(\mathbf{q}_n^H \mathbf{d}_{n-1}) \tag{3.33}$$

$$\mathbf{h}_{n|n} = -\mathbf{v}_n + \mathbf{h}_{n|n-1}\beta_n. \tag{3.34}$$

It is clear that the computational bottleneck only lies at eq. (3.32), accounting for around $\mathcal{O}(M^2)$, while in n-moment algorithm, the cost to find the minimum and maximum eigenvalues would be of order $\mathcal{O}(M^3)$. This roughly illustrate the reduction of computational complexity of the c-moment algorithm over the n-moment one. Regarding convergence, it is actually straightforward that, with eqs. (3.23) and (3.30), the c-moment algorithm also satisfies the second-order growth condition eq. (3.20), and we decided to forgo detailed proof for the sake of time limits at the time of writing this thesis. We summerized both algorithms below:

3.3 Numerical Experiments

To this point, the readers would have noticed that the proofs of convergence of our algorithms are sketched proofs, meaning it provides solid, rough still, mathematical lines of reasoning to show that it would be highly likely to converge. Candidly, the exact analysis has been a difficult challenge to us until now. We add up to this lacking by performing exhaustive numerical experiments to testify the validity of our proposed methods.

For rigor, the experiments were conducted via a widely linear quaternion moving average model of order 3 (WLQMA(3)), where the data x_n was drawn from 1200 samples of the same distribution $\mathcal{N}(0,1)$ for each component of x_n, with a moderate SNR of 20dB. The coefficients were drawn from a mixture of distribution $w_m \sim \mathcal{U}(-1,-0.45)+\mathcal{U}(0.45,1)$ and zeroed out if below 0.25. Then, all 1200 samples of x_n we fed into the WLQMA(3) to produce 1200 outputs, y_n. Note that the first 200 samples of y_n were left out so that

Algorithm 1: the n-Moment WL-QLMS Algorithm

Input : x_n, y_n, M, λ and δ
Output: $\mathbf{w}$

Initialize $\mathbf{w}_0 = \mathbf{p}_0 = \mathbf{0}$ and $\mathbf{R}_0 = \underline{\mathbf{0}}$;
$n = 0$;
do
  $n = n + 1$;
  Update $\mathbf{q}_n$ according to (3.2);
  $\hat{y}_n - (\mathbf{w}_{n-1})^H \mathbf{q}_n$;
  $e_n = y_n - \hat{y}_n$;
  $\mathbf{R}_n = \lambda \mathbf{R}_{n-1} + \mathbf{q}_n \mathbf{q}_n^H$;
  $\mathbf{r}_n = \lambda \mathbf{r}_{n-1} + \mathbf{q}_n y_n^*$;
  $\sigma_n = \lambda_{\min}(\mathbf{R}_n)$ (smallest eigenvalue);
  $L_n = \lambda_{\max}(\mathbf{R}_n)$ (largest eigenvalue);
  $\alpha_n = \frac{1}{L_n}$;
  Find $\mu_n \in (0, 1)$ such that $(\mu_n)^2 = (1 - \mu_n)(\mu_{n-1})^2 + \left(\frac{\sigma_n}{L_n}\right) \mu_n$;
  $\beta_n = \frac{\mu_{n-1}(1-\mu_{n-1})}{(\mu_{n-1})^2 + \mu_n}$ ($= 0$ if $(\mu_{n-1})^2 + \mu_n = 0$);
  $\mathbf{v}_n = \mathbf{w}_{n-1} + \beta_n(\mathbf{w}_{n-1} - \mathbf{w}_{n-2})$;
  $\mathbf{w}_n = \mathbf{v}_n - \frac{1}{L_n}(\mathbf{R}_n \mathbf{v}_n - \mathbf{r}_n)$;
  $\epsilon_0 = \frac{\sigma_n}{10}$;
  if $\beta_n \geq \sqrt{\frac{1-\alpha_n \epsilon_0}{1-\alpha_n \sigma_n}} \frac{\|\mathbf{w}_n - \mathbf{w}_{n-1}\|_2}{\|\mathbf{w}_{n-1} - \mathbf{w}_{n-2}\|_2}$ **then**
    $\mu_n = 1$;
    $\mathbf{w}_n = \mathbf{w}_{n-1} - \frac{1}{L_n}(\mathbf{R}_n \mathbf{w}_{n-1} - \mathbf{r}_n)$;
  end
while $\|e_n\| > \delta$ or $n \leq N$;
$\mathbf{w}^{op} = \mathbf{w}_n$.

Algorithm 2: The c-Moment WL-QLMS Algorithm

Input : x_n, y_n, M, λ, ϵ, δ and *update scheme*
Output: $\hat{y}_n$ and $\mathbf{w}^{op}$

Initialize $\mathbf{w}_0 = \mathbf{d}_0 = \mathbf{g}_{0|0} = \mathbf{h}_{0|0} = \mathbf{0}$ and $\mathbf{R}_0 = \underline{\mathbf{0}}$;
$n = 0$;
do
  $n = n + 1$;
  Update $\mathbf{q}_n$ according to (3.2);
  $\hat{y}_n - (\mathbf{w}_{n-1})^H \mathbf{q}_n$;
  $e_n = y_n - \hat{y}_n$;
  $\mathbf{R}_n = \lambda \mathbf{R}_{n-1} + \mathbf{q}_n \mathbf{q}_n^H$;
  $\mathbf{g}_{n|n-1} = \lambda \mathbf{g}_{n-1|n-1} - \mathbf{q}_n e_n^*$;
  $\mathbf{v}_n = \mathbf{R}_n \mathbf{g}_{n|n-1}$;
  if *Markov conjugate scheme* **then**
    $\mathbf{h}_{n|n-1} = \lambda \mathbf{h}_{n-1|n-1} + \mathbf{q}_n (\mathbf{q}_n^H \mathbf{d}_{n-1})$;
    $\beta_n = \frac{\langle \mathbf{d}_{n-1}, \mathbf{v}_n \rangle}{\langle \mathbf{d}_{n-1}, \mathbf{h}_{n|n-1} \rangle} \quad (\beta_1 = 0)$;
    $\mathbf{d}_n = -\mathbf{g}_{n|n-1} + \mathbf{d}_{n-1} \beta_n$;
    $\mathbf{h}_{n|n} = -\mathbf{v}_n + \mathbf{h}_{n|n-1} \beta_n$;
    $\alpha_n = -\frac{\langle \mathbf{d}_n, \mathbf{g}_{n|n-1} \rangle}{\langle \mathbf{d}_n, \mathbf{h}_{n|n} \rangle}$;
  else *steepest descent scheme*
    $\mathbf{d}_n = -\mathbf{g}_{n|n-1}$;
    $\alpha_n = \frac{1}{\pi_n + \epsilon_n}$ where $\pi_n = \frac{\langle \mathbf{g}_{n|n-1}, \mathbf{v}_n \rangle}{\|\mathbf{g}_{n|n-1}\|^2}$;
  end
  $\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{d}_n \alpha_n$;
  $\mathbf{g}_{n|n} = \mathbf{g}_{n|n-1} + \mathbf{h}_{n|n} \alpha_n$;
while $\|e_n\| > \delta$ or $n \leq N$;
$\mathbf{w}^{op} = \mathbf{w}_n$.

the rest 1000 were in steady state and hence used in our experiments, every of which was conducted over 200 independent trails to assure the consistency of the empirical outcome. Both proposed algorithms were performed against other more basic algorithms of the same class - the original WL-QLMS [14] and WL-RLS [11]. For a fair comparison, the stepsizes α_n for the original WL-QLMS was calculated the same way as our c-moment algorithm in eq. (3.22) (this would ultimately render a normalized version of WL-QLMS). The metric used to benchmark the algorithms was the normalized misalignment, ζ_n, with $\mathbf{w}^{op}$ the true value of $\mathbf{w}$, defined as

$$\zeta_n \triangleq \frac{\|\mathbf{w}_n - \mathbf{w}^{op}\|^2}{\|\mathbf{w}^{op}\|^2}. \tag{3.35}$$

Remark 1. *All of the experiments and figures in this section have been published in [1, 2]. The original .fig files of some images were lost, so the legend of some images shows the alternative names of the proposed algorithms in this chapter.*

In the first experiment, we compared our proposed algorithms with the WL-QRLS routine because all of them minimize the same cost function eq. (3.9). Two variants of λ, 0.95 and 0.91 were considered. As seen in Fig. 3.1, both proposed momentum-based algorithms, n- and c-moment, converged as fast and achieved steady-state misalignment as low as the WL-QRLS. When $\lambda = 0.95$, the WL-QRLS started off the convergence rate faster but then was caught up by the momentum algorithms and eventually reached almost the same steady-state misalignment. For $\lambda = 0.91$, while the WL-QRLS now clearly converges slightly faster than the remaining two, it somehow abruptly diverges away at the epoch ~ 360 whereas the momentum routines remained robust. This is due to the absence of matrix inversion in our algorithms and hence no concern for non-invertible $\mathbf{R}_n$ which could potentially arise. The other lesser-known cause of this divergence is the fact that λ will be a denominator in a WL-QRLS algorithm and therefore could lead to exponentially growing sequence. In comparing between the 2 proposed routines, it can be seen that the c-moment WL-QLMS actually performed better numerically and computationally than the n-moment one. Consequently, for the sake of space saving, c-moment algorithm will be mostly considered from this point onwards.

For the next simulation, the effect of λ was illustrated in Fig. 3.2 where the standard

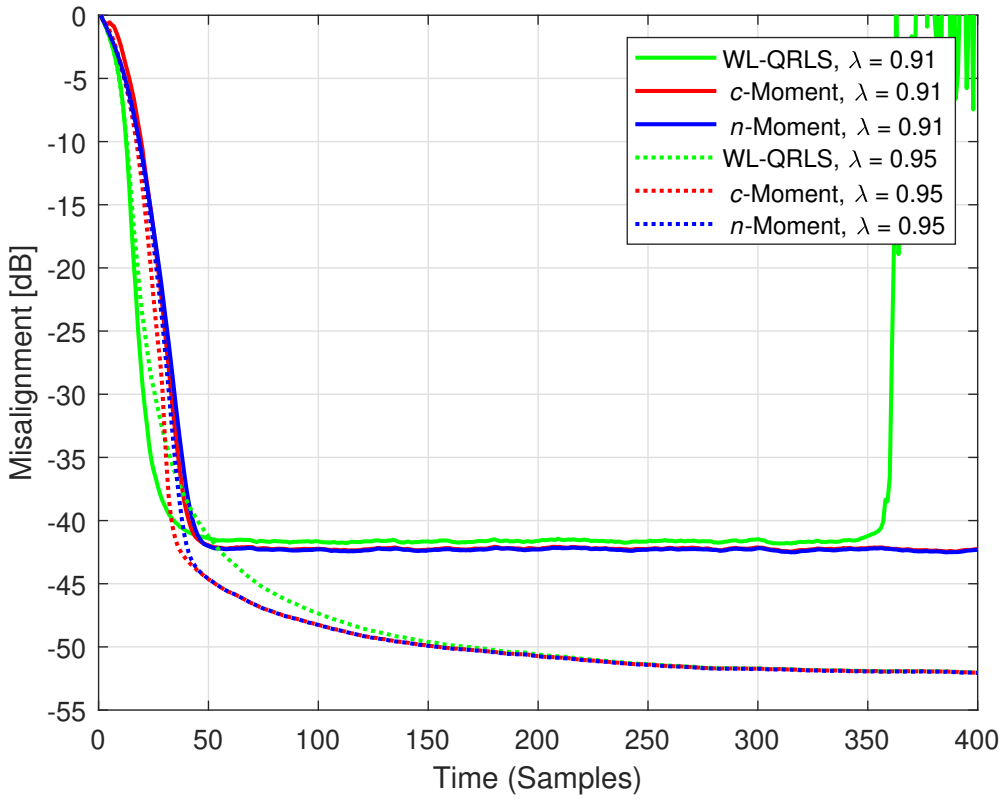


Figure 3.1: [1] The normalized misalignment of WL-QRLS (green), c-moment WL-QLMS (red) and n-moment WL-QLMS (blue) algorithms for the values of λ 0.91 (line) and 0.95 (dash), averaged over 200 independent trials, when employed for the identification of a WLQMA(3) process, at an SNR of 20dB.

WL-QLMS was put as a benchmark. In this experiment, the value of β_n was set to zero to gauge how c-moment WL-QLMS routine would perform at its worst. When the forgetting factor is close to unity, the convergence rate decreases close to the standard WL-QLMS but the steady-state misalignment is lower. When the value of λ decreases further from unity, the algorithms converge faster but misalignment increasingly approaches that of the standard WL-QLMS. Between the WL-QRLS and the c-moment WL-QLMS, the c-moment routine clearly achieves lower misalignment, regardless of the value of λ; however, when λ is low, its convergence get caught up by that of the WL-QRLS by a slight margin. Now with the same setting from fig. 3.2, we looked into how the presence of β_n affects the c-moment WL-QLMS algorithm as a whole in the third experiment. Now via fig. 3.3, the momentum from β_n pushed the convergence to be even better, but a quite slight increase

in misalignment was also observed.

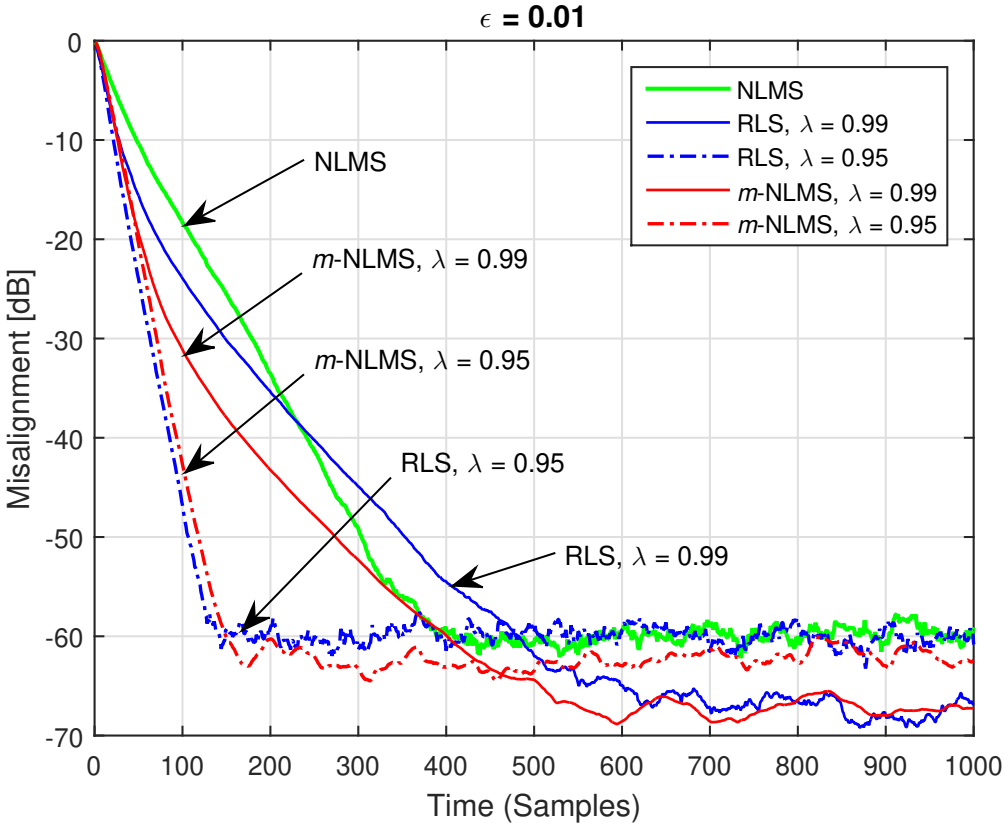


Figure 3.2: [2] The normalized misalignment of standard WL-QLMS (NLMS with $\epsilon = 0.01$), WL-QRLS (RLS), and c-moment WL-QLMS with $\beta_n = 0$ (m-NMLS) algorithms for the values of λ 0.99 (line) and 0.95 (dash), averaged over 200 independent trials, when employed for the identification of a WLQMA(3) process, at an SNR of 20dB.

The forth experiment exhibited the versatility of our proposed methods by combining variable stepsize techniques into the routine. figs. 3.4 and 3.5 show the results of the forth experiment where three fixed-stepsize routines (WL-QRLS and c-moment WL-QLMS) were benchmarked against three variable-stepsize ones (normalized WL-QLMS and c-moment WL-QLMS with GSER [67] and GNGD [68] schemes). By inspection of c-moment WL-QLMS at $\lambda = 0.99$ and 0.95 as well as comparing the results with the second experiment in fig. 3.2, it is clear that with the right choice of variable step size scheme, both fast convergence and low misalignment could be attained simultaneously.

For the fifth experiment, we wanted to see how our proposed methods fare against non-stationary signals where the coefficients could suddenly change at random. Such a candi-

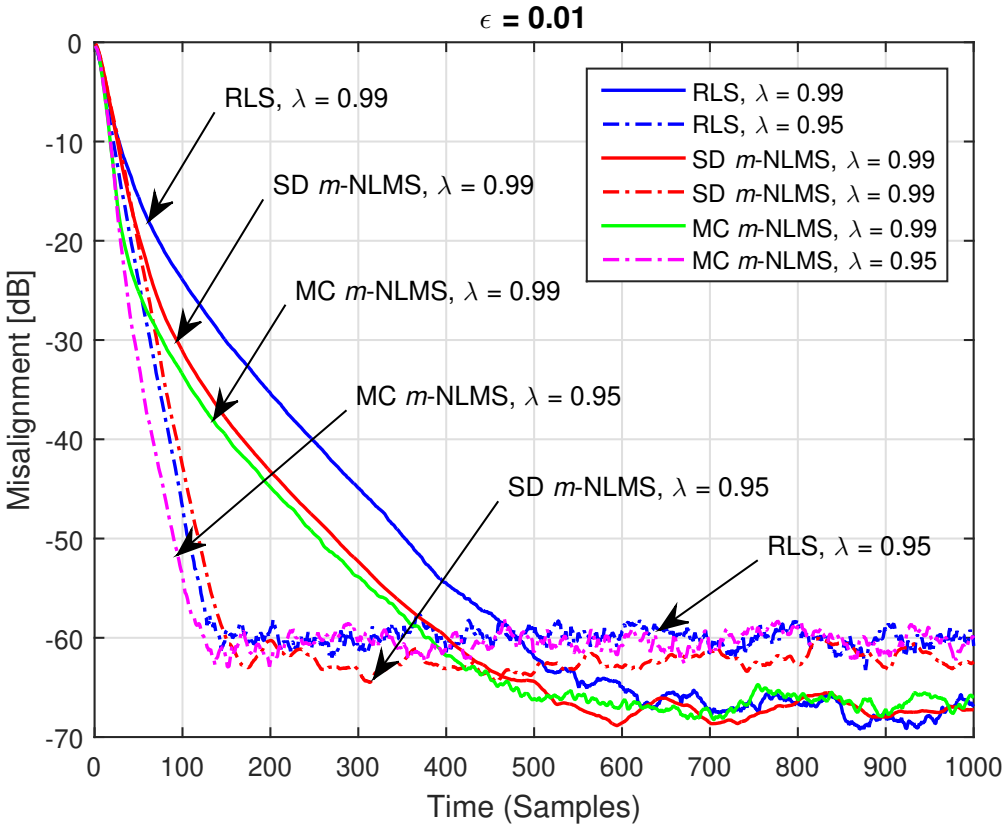


Figure 3.3: [2] The normalized misalignment of WL-QRLS (RLS), c-moment WL-QLMS with $\beta_n = 0$ (SD m-NLMS), and normal c-moment WL-QLMS (MC m-NLMS) algorithms for the values of λ 0.99 (line) and 0.95 (dash), averaged over 200 independent trials, when employed for the identification of a WLQMA(3) process, at an SNR of 20dB.

date used in our simulation is Saito's chaotic signals which will be modelled via a widely linear quaternion autoregresive models of order 3 and 6 - WLQAR(3) and WLQAR(6), both of which were set to make 3-step prediction. This experiment were visualized in figs. 3.6 to 3.8 and used NMSE as a measure of performance. It is obvious right away that the WL-QRLS failed to re-adjust itself after the weight coefficients altered fig. 3.6. This is again due to its inherent numerical stability from matrix inversion. While not visualized here, the stadard WL-QLMS were too slow to even converge to steady state before the change in the coefficients. Our proposed methods surely reigned supreme in this non-stationary case. Nevertheless, the momentum methods struggled to identify the system during the transient state fig. 3.7 as the nature of this spike pattern is not Gaussian [55] as assumed in our model. After all, the ability of our methods to re-adjust in a steady-state condition

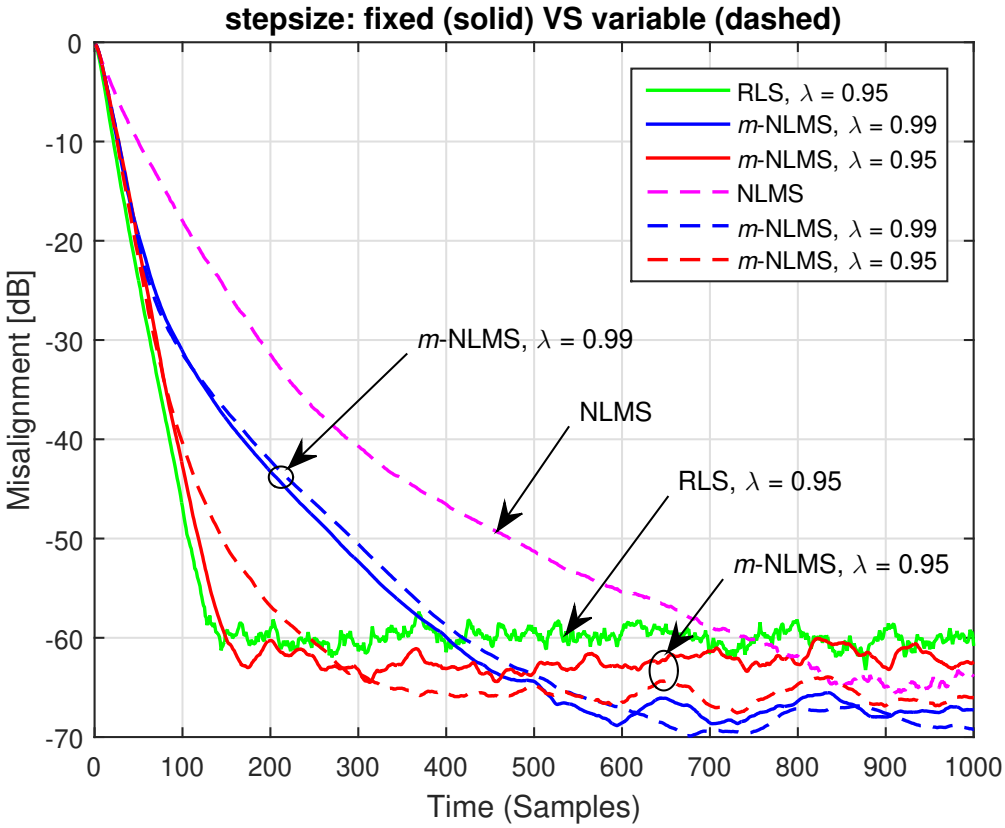


Figure 3.4: [2]The normalized misalignment of WL-QRLS (RLS), standard WL-QLMS (NLMS), and c-moment WL-QLMS (m-NMLS) algorithms for the fixed stepsize (line) and GSER stepsize (dash) schemes, averaged over 200 independent trials, when employed for the identification of a WLQMA(3) process, at an SNR of 20dB.

was validated. The original Saito's signal and its predictions were in fig. 3.8.

3.4 Summary

This chapter presents our first work out of three: the quaternion-valued adaptive signal processing based on extrapolated gradient methods. Firstly, the theory of optimal gradient descent was re-visited and revised for the case of quaternions. The fortunate aspect of our first topic is the fact that the function to be minimized is a real-valued function of quaternion-valued arguments and hence the analysis would be theoretically identical to that of the real-valued arguments, were the function to be deterministic. By assuming that the second-order statistics of the input data varies slowly (which is actually likely to be true

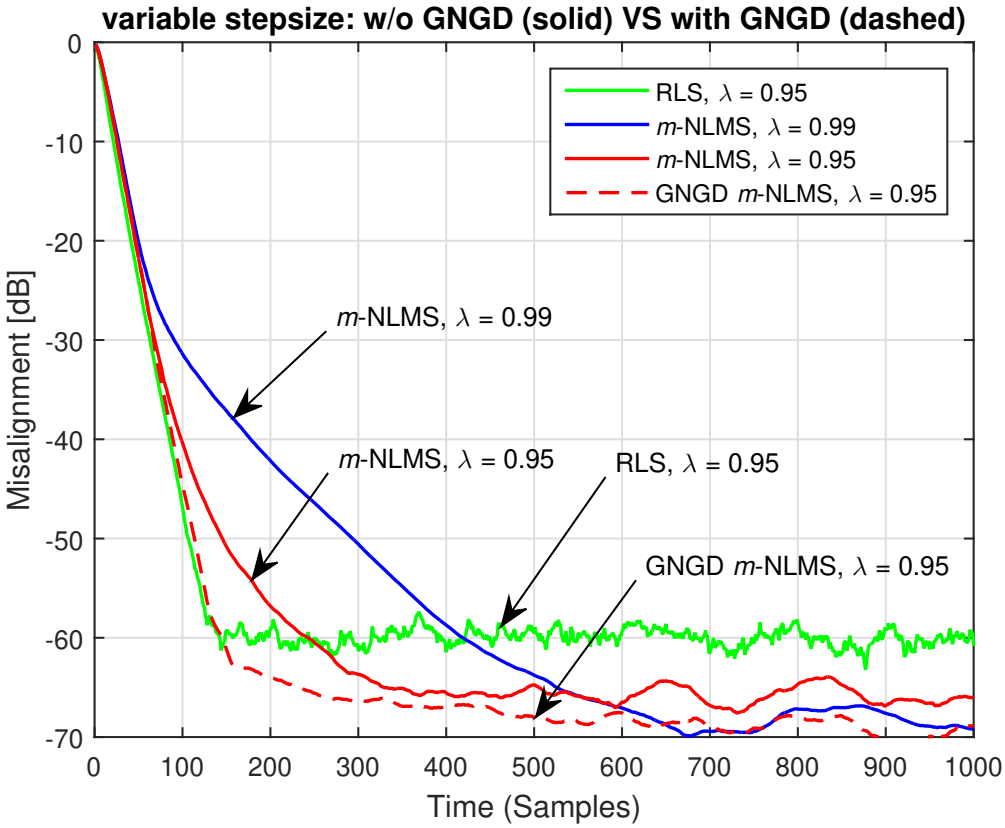


Figure 3.5: [2] The normalized misalignment of WL-QRLS (RLS), c-moment WL-QLMS (m-NMLS) algorithms both fixed (line) stepsize and GNGD stepsize (dash) schemes, averaged over 200 independent trials, when employed for the identification of a WLQMA(3) process, at an SNR of 20dB.

according to eqs. (3.10) and (3.11)), the local descent of the gradient was validated through numerical simulations to be sufficient for our methods to be robust. Two approaches were introduced: the n-moment WL-QLMS and the c-moment algorithms. These two methods are different in the way their underlying heay-ball update equation is expressed; the former follows Nesterov's optimal method while the latter follows Chebyshev's iterative method. The n-moment WL-QLMS almost take the same formulae as the classical Nesterov's except for the additional step of gradient conditioning to ensure both local descent direction and second-order reduction of cost function. On the other hand, the c-moment WL-QLMS can be interpreted as a simplified n-moment routine with less computation but almost identical numerical performance. While its outward expression resembles conjugate gradient algorithm, it only captures the immediate momentum of the most recent past

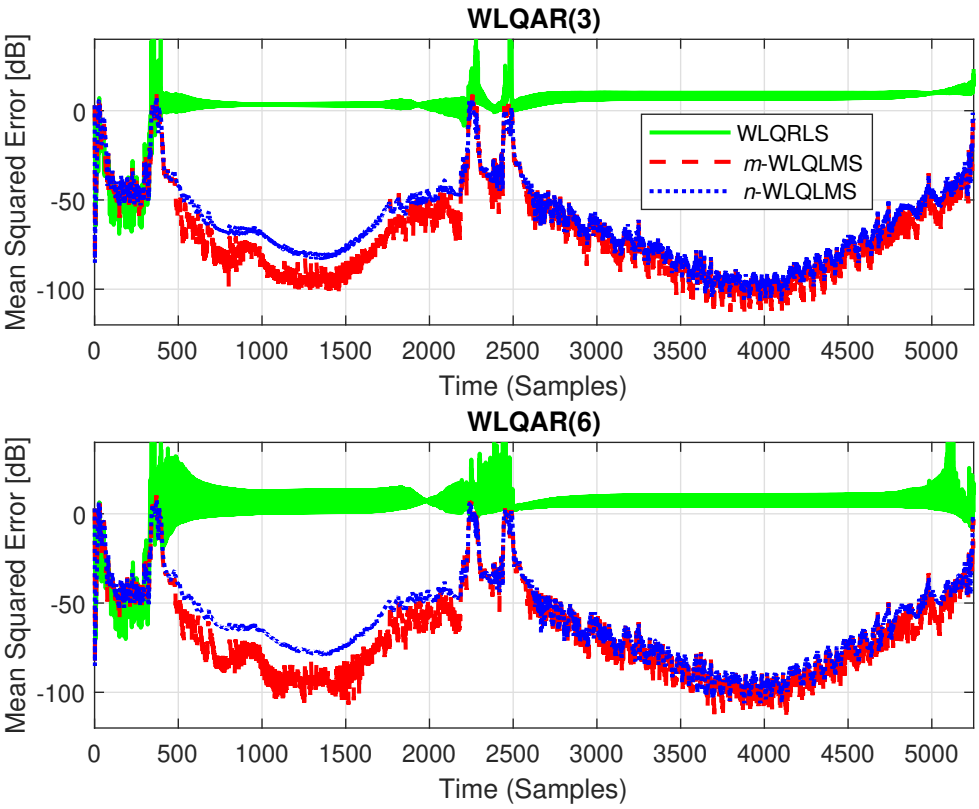


Figure 3.6: [1] The normalized MSE of WL-QRLS, n-moment WL-QLMS (n-WLQLMS) and c-moment WL-QLMS (m-WLQLMS) algorithms with $\lambda = 0.95$, when employed for the identification of Saito's circuit through 3-step predictive WLQAR(3) and WLQAR(6) models.

descent direction while the traditional conjugate gradient would capture the momentum of all past directions, but that is due to the deterministic nature of the problem. The empirical experiments confirm our analysis and potential of the proposed quaternion-valued adaptive filters with the performance on par with the fast-converging WL-QRLS but numerically robust like the slow-converging standard WL-QLMS. Finally, we consider that the idea of momentum-based approaches derived so far is too beneficial to be a one-trick enabling concept used only in this work. Our second work about dictionary learning in tensors see this concept as the main scheme for updating the dictionary, which will be presented in the next two chapters.

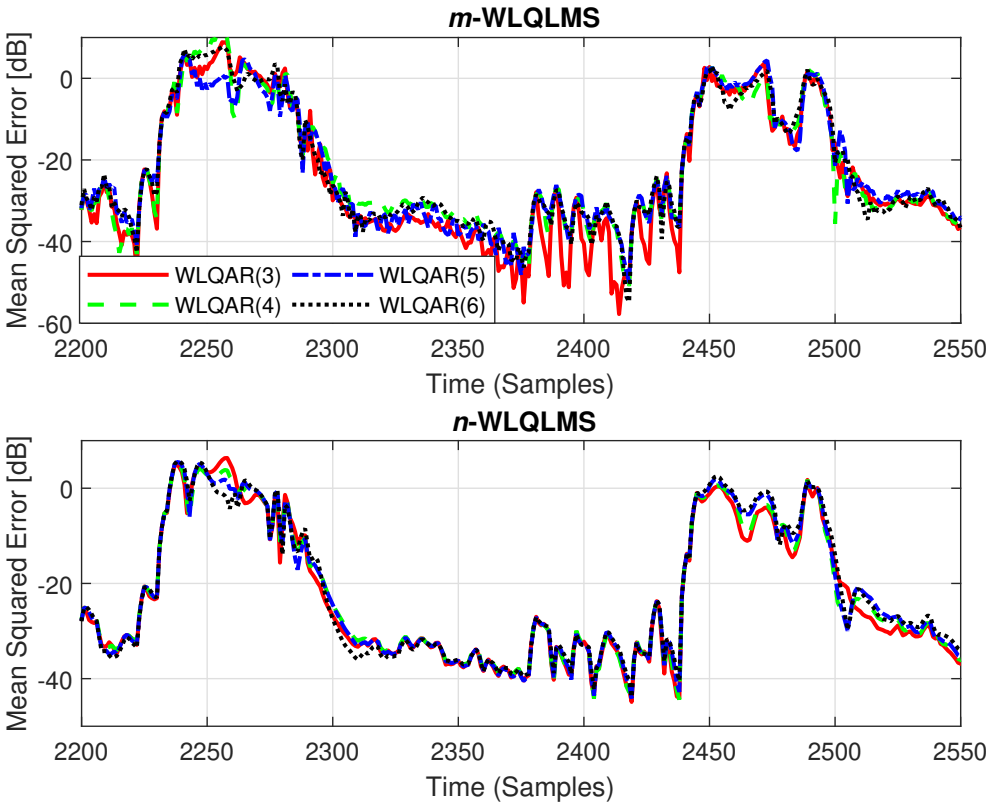


Figure 3.7: [1] The transient normalized MSE of n-moment WL-QLMS (n-WLQLMS) and c-moment WL-QLMS (m-WLQLMS) algorithms with $\lambda = 0.95$, when employed for the identification of Saito's circuit through WLQAR models of orders from 3 to 6.

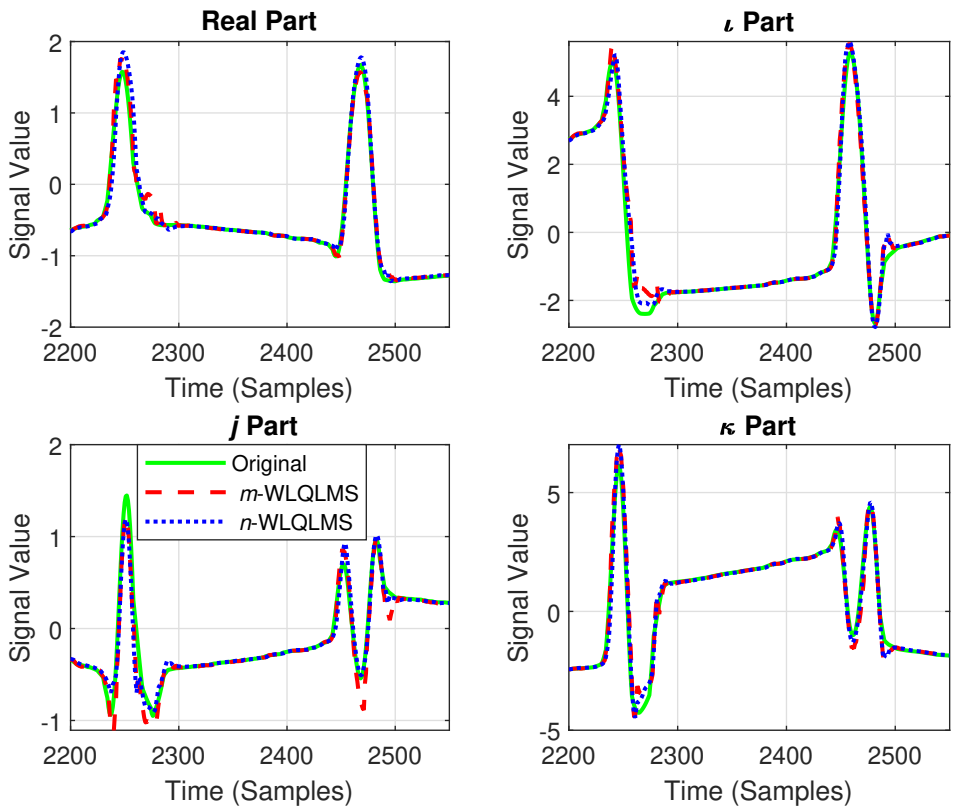


Figure 3.8: [1] The values of each component of Saito's signal presented in the original (green) and its estimates through WLQAR(3) in c-moment WL-QLMS (red dash) and n-moment WL-QLMS (blue dot) algorithms with $\lambda = 0.95$.

Chapter 4

Tensor Decompositions for Signal Processing

TENSORS are the second unconventional-structure data we study in this thesis. As a matter of fact, all three types of data - quaternions, tensors and graphs - naturally occur in the setting of ever growing multisensor/sultinode data acquisition. In fact, all these three data types have share isomorphism between each other. To put simplest, a quaternion matrix ($\mathbb{H}^{M\times N}$) is isomorphic with a real-valued tensor ($\mathbb{R}^{M\times N\times 4}$). It is under this umbrella of data fusion research we have been working on that gives meaning to the smooth transition from quaternions to tensors. Unlike matrices, it has been shown through various research article that data analysis techniques for tensor decomposition are more flexible, the decomposed latent components are usually more insightful in terms of hidden features, and algebraic properties are usually guaranteed under more natural and milder conditions. By these interesting attributes of tensor analysis, we are inspired to extend to tensors the data fusion technique of adaptive filtering. In this chapter, the necessary concepts of multilinear algebra are explained and discussed in order to form the basic for our second research work: the multilinear online dictionary learning. This chapter will provide explanation of important tensor decomposition techniques relevant to our tensor dictionary learning algorithm to be introduced in the next chapter.

4.1 Tensor Basics: History, Motivation and Preliminaries

The material in this chapter can be found with much more additional details that are not relevant for our work in [69–71].

4.1.1 Brief History of Tensors as Signal Processing Tools

Probably, the most widely known use of a tensor is the Einstein notation in his general theory of relativity. As a multiway data analysis, it can be traced back to the studies of homogeneous polynomials in the 19th century. The very first tensor representation as a *multidimensional array* (or *multiway*) was the introduction of Tucker decomposition (TKD) for analyzing psychometrics [72]. Shortly, the canonical polyadic decomposition (CPD) was independently re-invented as canonical decomposition (CANDECOMP) [73] and parallel factor model (PARAFAC) [74]. After many adoption in diverse fields of data analysis [75], it entered the field of signal processing as a potential tool to analyze higher-order statistics (HOS), which is higher-order tensors. Many novel applications regarding HOS have been innovated afterwards [76–78].

4.1.2 Why tensors?

Before diving into the material review, we need to highlight the benefits of the tensors as a data structure. The first obvious benefit of a tensor is that it is a straightforward representation of multidimensional data. This tensor representation can be unfolded to a matrix form via its equivalent Kronecker structure [69] which unnecessarily and geometrically increases the size of data. Therefore, direct analysis on a tensor itself would save any redundant computation. For example, the correlation matrix from unfolding a correlation tensor will be dependent on only the main correlation sub-matrices along the diagonal while the remaining off-diagonals will be the product of their respective diagonal ones [71]. All in all, analyzing tensor data directly helps reduce the redundancy arising from tensors unfolding to matrices - the blessing of dimensionality. The second benefit is from the perspective of data fusion technique where more and more data diversity (channel

$\times$ time $\times$ frequency $\times$ trial $\times$ subject $\times$ etc) can be aggregated and hence joint analyzed for more profound insights to better understand the data at hand.

Tensor representation is not without its limits though. While much existing work considers *tensorizing* data by concatenating matrix and vector data, we quite disagree with this practice for three reasons. Firstly, as mentioned above, the natural way to unfold a tensor into a matrix is its Kronecker structure equivalent whereby some properties in auto-/cross-correlation are enforced. *Blind transform* of matrices into a tensor is therefore dangerous because it may create false artifacts into the original matrices that may not have a Kronecker structure. Secondly, thanks to the advance in computing technology, the maneuvering of large-scale data has no longer been hindered by the sophistication of a mathematical algorithm. While general tensor decomposition algorithms could achieve much more massive dimensionality reduction than any matrix/vector counterparts, it may be no longer necessary in today's hardware scope. Last but not least, although the analysis of already decomposed tensors is doubtless computational-friendly, the decomposition algorithms themselves are not, of which many can exceed its target tensor in terms of computational dimension such as *exact* CPD, which is actually NP-complete [79]. Many active research in the field endeavors to propose novel algorithms which can achieve both computational affordability and the dimensionality reduction as close as the theoretical bound simultaneously. The important consideration before employing tensor decomposition is therefore whether it is still worth it when the cost of decomposition weighs in.

Nevertheless, the research field regarding tensor decomposition has been very active and new approaches have been introduced continually. We may eventually arrive at effective methods whereby the computational complexity is out of concern. Only the Kronecker structure inherited in a tensor product is the remaining issue.

4.1.3 Notation and Preliminaries

A tensor can be deemed a multi-way array whereby the term *ways* or *modes* is the order of the tensor. A real-valued tensor of order N is symbolized by a boldface calligraphic uppercase letter as $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ with its scalar entries by italic lowercase letters as

$a_{i_1 i_2 \ldots i_N}$. As a side note, a matrix, denoted by a boldface uppercase letter, $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$, can be considered a 2nd-order tensor or 2-way array.

A subarray of a tensor is a subset of the tensor where part of the indices is fixed. For example, a vector is a subarray of a matrix. For tensors, their matrix (2-way) subarrays are termed *slices* by fixing all but any two indices. If we now fix all but one index, the subarrays will be called *fibers*. The *Frobenius norm* of a tensor $\boldsymbol{\mathcal{A}}$ is the square root of the sum of the squares of all its entries, that is,

$$\|\boldsymbol{\mathcal{A}}\|_F \triangleq \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} a_{i_1 i_2 \ldots i_N}^2}.$$

Observe that this is similar to Frobenius norm of a matrix. The *Frobenius inner product* of two equal-sized tensors $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}$ is the sum of the products of their elements, that is,

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle_F \triangleq \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} a_{i_1 i_2 \ldots i_N} b_{i_1 i_2 \ldots i_N},$$

and it is obvious that $\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}} \rangle_F = \|\boldsymbol{\mathcal{A}}\|_F^2$. Moreover, $\boldsymbol{\mathcal{A}}$ can be called a *rank-one* tensor if it can be expressed as the outer product of *factor vectors*, $\boldsymbol{\psi}_n \in \mathbb{R}^{I_n}$ that is,

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\psi}_1 \circ \boldsymbol{\psi}_2 \circ \cdots \circ \boldsymbol{\psi}_N,$$

where the operator $\circ$ represents outer product, signifying that each element of the tensor is the product of the corresponding elements of the respective factor vectors, i.e.

$$a_{i_1 i_2 \ldots i_N} = \psi_{i_1} \psi_{i_2} \cdots \psi_{i_N} \quad \text{for all} \quad 1 \leq i_n \leq I_n.$$

$\boldsymbol{\mathcal{A}}$ is called a *diagonal tensor* if its elements $a_{i_1 i_2 \ldots i_N} \neq 0$ only if $i_1 = i_2 = \cdots = i_N$. So far, we have defined many special tensors and two tensor products. Now, we will review three more tensor products crucial in deriving our tensor dictionary learning algorithm. Given a matrix $\boldsymbol{\Psi}_n \in \mathbb{R}^{J_n \times I_n}$, a *mode-n product* between $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\Psi}_n$ yields another tensor

$\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times J_n \cdots \times I_N}$ given by

$$(\boldsymbol{\mathcal{C}})_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = (\boldsymbol{\mathcal{A}} \times_n \boldsymbol{\Psi}_n)_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} \triangleq \sum_{i_n=1}^{I_n} a_{i_1 i_2 \ldots i_N} \psi_{j_n i_n},$$

where, put simply, each mode-n fiber of $\boldsymbol{\mathcal{A}}$ is multiplied by the matrix $\boldsymbol{\Psi}_n$. This mode-n product is the most basic building block of most tensor decomposition techniques. Now, given $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{J_1 \times J_2 \cdots \times J_M}$ and $I_n = J_m = K$, we can define a *mode-(n, m) contracted product* or *contraction* between $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{D}}$ which yields an $(N + M - 2)^{th}$-order tensor

$$(\boldsymbol{\mathcal{A}} \times_m^n \boldsymbol{\mathcal{D}})_{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N j_1 \ldots j_{m-1} j_{m+1} \ldots j_M} \triangleq \sum_{k=1}^{K} a_{i_1 \ldots i_{n-1} k i_{n+1} \ldots i_N} d_{j_1 \ldots j_{m-1} k j_{m+1} \ldots j_M}.$$

The definition above displays contraction in a single common mode, but in reality, tensors can be contracted in several modes simultaneously. In fact, the essence of our proposed online tensor dictionary learning routine relies on the mode-wise operation where two tensors $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{D}}$, now $I_k = J_k$ for $k \neq n$, is *contracted* into a matrix $\mathbf{X} \in \mathbb{R}^{I_n \times J_n}$ via the mode-'all-but-n' contraction, of which the element $x_{i_n j_n}$ is given by

$$x_{i_n j_n} = (\boldsymbol{\mathcal{A}} \times_{/n}^{/n} \boldsymbol{\mathcal{D}})_{i_n j_n} \triangleq \sum_{i_1=1}^{I_1} \cdots \sum_{i_{n-1}=1}^{I_{n-1}} \sum_{i_{n+1}=1}^{I_{n+1}} \cdots \sum_{i_N=1}^{I_N} a_{i_1 i_2 \ldots i_N} d_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N}.$$

Note that the *all-mode* contraction is the same as the Frobenius inner product defined above. This contraction of tensors into a matrix forms part of the main strategies in our proposed tensor dictionary algorithm for data fusion and dimensionality reduction. The last product, which is not a tensor product, we would like to cover is the Kronecker product, the matrix product that underpins tensor operation. Given matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$, the Kronecker product between $\mathbf{A}$ and $\mathbf{B}$, denoted by $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{IK \times JL}$,

given by

$$\mathbf{A} \otimes \mathbf{B} \triangleq \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \vdots & a_{IJ}\mathbf{B} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_1 \otimes \mathbf{b}_2 & \mathbf{a}_1 \otimes \mathbf{b}_3 & \cdots & \mathbf{a}_J \otimes \mathbf{b}_{L-1} & \mathbf{a}_J \otimes \mathbf{b}_L \end{bmatrix}.$$

While the above expression appears large, it is very redundant in its formation and thus possesses correlation properties explained earlier. with this view, tensor representation is a more natural and more compact way to store those data with inherent Kronecker structure. For more detailed reading, please be referred to [69–71].

4.2 Major Decomposition Techniques

All the preliminaries so far have been built up for the most important concept of the tensor section of this thesis: the tensor decomposition. Here, we give a summary of three decomposition methods relevant to our tensor dictionary learning: the CPD, the TKD, and the higher-order compressed sensing.

4.2.1 Canonical Polyadic Decomposition

Although the CPD is not directly related to our tensor algorithm, It provides relational basics with respect to the other decomposition. The term *polyadic decomposition* means an Nth-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{J_1 \times J_2 \cdots \times J_N}$ can be represented as a linear combination of rank-1 tensors, i.e.

$$\boldsymbol{\mathcal{X}} = \sum_{\ell=1}^{L} \lambda_\ell \left(\boldsymbol{\psi}_{1\ell} \circ \boldsymbol{\psi}_{2\ell} \circ \cdots \circ \boldsymbol{\psi}_{N\ell} \right).$$

Alternatively, with the notion of mode-n product, $\boldsymbol{\mathcal{X}}$ can be expressed as a *full multilinear product*, that is

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{D}} \times_1 \boldsymbol{\Psi}_1 \times_2 \boldsymbol{\Psi}_2 \cdots \times_N \boldsymbol{\Psi}_N$$

where $\boldsymbol{\mathcal{D}}$ is a diagonal tensor with $d_{\ell\ell\cdots\ell} = \lambda_\ell$ and $\boldsymbol{\Psi}_n = [\boldsymbol{\psi}_{n1} \quad \boldsymbol{\psi}_{n2} \cdots \boldsymbol{\psi}_{nL}] \in \mathbb{R}^{J_n \times L}$. This polyadic decomposition becomes *canonical* when the value L is smallest while the equlity still holds *exactly*; this smallest possible L is termed as the CPD rank which appears to be similar to the SVD rank of a matrix in the way the CPD operates like the SVD. This is however out of the thesis' scope and the really important point from this introduction of the CPD is the expression of a full multilinear product as a compact representation of tensor decomposition.

4.2.2 Tucker Decomposition

The TKD can be regarded as a more general form of the full multilinear product expression of the tensor $\boldsymbol{\mathcal{X}}$ with a core tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{L_1 \times L_2 \cdots \times L_N}$ and factor matrices $\boldsymbol{\Psi}_n = [\boldsymbol{\psi}_{n1} \quad \boldsymbol{\psi}_{n2} \cdots \boldsymbol{\psi}_{nL_n}] \in \mathbb{R}^{J_n \times L_n}$, given by

$$\boldsymbol{\mathcal{X}} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} s_{r_1 r_2 \ldots r_N} \left(\boldsymbol{\psi}_{1r_1} \circ \boldsymbol{\psi}_{2r_2} \circ \cdots \circ \boldsymbol{\psi}_{Nr_N} \right),$$

or equivalently

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{S}} \times_1 \boldsymbol{\Psi}_1 \times_2 \boldsymbol{\Psi}_2 \cdots \times_N \boldsymbol{\Psi}_N. \tag{4.1}$$

Moreover, it can be *unfolded* into a matrix/vector form via the Kronecker product, that is

$$\operatorname{vec}(\boldsymbol{\mathcal{X}}) = [\boldsymbol{\Psi}_N \otimes \boldsymbol{\Psi}_{N-1} \otimes \cdots \otimes \boldsymbol{\Psi}_1] \operatorname{vec}(\boldsymbol{\mathcal{S}}).$$

There are many ways to condition the TKD. The classical one is the orthogonality. A much more recent one, for example, is the minimal size of the core tensor $\boldsymbol{\mathcal{S}}$, where L_n would then represent the mode-n rank of the tensor $\boldsymbol{\mathcal{X}}$ (it is actually the rank of the mode-n fiber of the tensor $\boldsymbol{\mathcal{X}}$), and the N-tuple $(L_1, L_2, \ldots, L_N)$ is as a result defined as the *multilinear rank*, or the *higher-order SVD* rank of the tensor $\boldsymbol{\mathcal{X}}$.

It is obvious by these examples that the concept of a *tensor rank* is more sophisticated than a matrix as it can be, but not limited to, the one number which represents the whole tensor overall, or it can be a collection of numbers which represents their corresponding subarrays. Whether to use which is a subject of specific applications [70, 71].

4.2.3 Higher-Order Compressed Sensing

The TKD is the main decomposition which underpins our online tensor dictionary learning model. Nevertheless, the constraint for our TKD model is neither orthogonaliy nor multilinear rank. The higher-order compressed sensing (HO-CS) model is a class of TKD where the core tensor $\boldsymbol{\mathcal{S}}$ is *overcomplete* and *sparse*. Overcompleteness dictates that $J_n < L_n$, $\forall n$, and sparsity means that $\boldsymbol{\mathcal{S}}$ is barely filled with most entries equal 0. The dimensionlity reduction in the HO-CS model thus happens in the reverse fashion with the other decomposition techniques in that the result of the decomposition, $\boldsymbol{\mathcal{S}}$, is much larger in dimensionality than the original tensor $\boldsymbol{\mathcal{X}}$ and it is the sparse elements of the core $\boldsymbol{\mathcal{S}}$ that could be projected to another compressed tensor, $\boldsymbol{\mathcal{Y}}$, smaller than $\boldsymbol{\mathcal{X}}$. Given the similar expression as in eq. (4.1), the HO-CS problem extends the traditional CS task [80] into a form of the multilinear product, given by

$$\begin{aligned} &\min_{\boldsymbol{\mathcal{S}}} \ \|\boldsymbol{\mathcal{S}}\|_0 \quad \text{s.t.} \\ &\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{S}} \times_1 \boldsymbol{\Theta}_1 \times_2 \boldsymbol{\Theta}_2 \cdots \times_N \boldsymbol{\Theta}_N, \end{aligned} \tag{4.2}$$

where $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$, with $I_n < J_n$, $\forall n$, is measurement tensor and $\boldsymbol{\Theta}_N \triangleq \boldsymbol{\Phi}_N \boldsymbol{\Psi}_N \in \mathbb{R}^{I_n \times L_n}$, $\forall n$ is defined as a mode-n *sensing* matrix. Note that the matrix $\boldsymbol{\Psi}_n$ is teh HO-CS setting is now called a mode-n *sparsifying dictionary*. With eq. (4.1), $\boldsymbol{\mathcal{Y}}$ can be expressed in terms of $\boldsymbol{\mathcal{X}}$ as

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_1 \boldsymbol{\Phi}_1 \times_2 \boldsymbol{\Phi}_2 \cdots \times_N \boldsymbol{\Phi}_N, \tag{4.3}$$

where $\boldsymbol{\Phi}_n \in \mathbb{R}^{I_n \times J_n}$ is called a mode-n projection matrix with $I_n \leq J_n$, $\forall n$. From eqs. (4.1) to (4.3), dimensionality reduction is finally achieved with $\boldsymbol{\mathcal{Y}}$ which can recover $\boldsymbol{\mathcal{X}}$ by any generic classes of pursuit algorithms [81]. It is noteworthy that the problem in (4.2) is equivalent to the conventional CS problem with a Kronecker structure [82]:

$$\text{vec}\,(\boldsymbol{\mathcal{Y}}) = \boldsymbol{\Theta}\,\text{vec}\,(\boldsymbol{\mathcal{S}}) \triangleq (\boldsymbol{\Theta}_N \otimes \boldsymbol{\Theta}_{N-1} \otimes \cdots \otimes \ \boldsymbol{\Theta}_1)\,\text{vec}\,(\boldsymbol{\mathcal{S}}) \tag{4.4}$$

where it is clear that, if we define $\boldsymbol{\Phi} \triangleq \boldsymbol{\Phi}_N \otimes \boldsymbol{\Phi}_{N-1} \otimes \cdots \otimes \ \boldsymbol{\Phi}_1$ and $\boldsymbol{\Psi} \triangleq \boldsymbol{\Psi}_N \otimes \boldsymbol{\Psi}_{N-1} \otimes \cdots \otimes \ \boldsymbol{\Psi}_1$ and use the mixed-product property of the Kronecker product, we have

$$\boldsymbol{\Theta} = \boldsymbol{\Phi}\boldsymbol{\Psi}. \tag{4.5}$$

In our tensor dictionary learning algorithm, which we name the *online multilinear dictionary learning* (OMDL) algorithm, both the mode-n dictionaries, $\mathbf{\Psi}_n$, and the mode-n projection matrices, $\mathbf{\Phi}_n$, are learned in a sequential manner with the former is regarded as the *online* dictionary update and the latter as the sequential HO-CS step, as it fits in our whole algorithm.

4.3 Summary

This chapter provided necessary and sufficient knowledge of tensor decomposition techniques important to the to-be-proposed OMDL algorithm, especially the TKD and the HO-CS. The notation and basic preliminaries are first given. Then building on these, many types of tensor structures are provided. Then, three major decomposition methods are reviewed. With the CPD and the TKD, the concept of tensor ranks is shown to be distinct from those of matrices as tensors have different uniqueness condition and algebraic properties which ultimately leading to a tensor rank much smaller than its unfolded matrix form [83]. In our OMDL routine, the main task is cast into the TKD expression eq. (4.1) and its sub-task, the sequential HO-CS, will take the formulae in eqs. (4.2) and (4.3). With the analytic tools summarized in this chapter, we could proceed to our second proposed algorithm of this thesis, the OMDL algorithm which is formulated and derived in the next chapter.

Chapter 5

Online Multilinear Dictionary Learning

DICTIONARY learning (DL) is a class of representation learning viewed from the perspective of a matrix factorization problem, and there exists a body of research [84–86], most of which are batch method and not suitable for streaming data or the data too massive to analyze all at once. To address this, an *online* dictionary learning (ODL) has been created via the mechanism of the LMS adaptive algorithm with rank-1 stochastic gradient [87]. This was followed by the block co-ordinate descent (BCD) ODL method [88] which is basically a variation of momentum-based adaptive algorithms discussed and proposed in the previous chapters. As a result of incorporating past information into the learning gradient, the performance is improved. Other alternative methods of ODL include, for example, recursive-least-square (RLS)-DL [89], discriminative learning [90, 91], kernel dictionary [92] and ODL with pruning [93].

Many times, the compressed sensing problem is also considered a part of the DL

due to the fact that both leverage the sparsity prior [80] as the main constraint. In the CS task, if the data is sparse or can be sparisified in some arbitrary transformed bases, it can be compressed to a smaller set of samples than those dictated by the Shannon-Nyquist criterion [94] and , still, can be exactly reconstructed back to its original information. The reason why CS and DL tasks usually come together is that the CS task assumes that the data to be compressed is sparse, but unfortunately most natural data is not; that is where the DL application comes in. Early CS effort started with random measurements [80] and later evolved into the problem of finding the optimal sensing matrices. To date, one of the most robust approaches mostly agreed in the literature is the closest tight-frame Gram matrix [95–97], a very flexible scheme which can be combined with additional constraints like robustness to measurement error [98] and joint optimization of projection matrix and dictionary [99, 100].

So far the DL problems in the above mentioned work consider traditional flat-view matrix/vector data. To cope with higher dimensional information like tensors as discussed in the previous chapter, many efforts have been made in this direction. Beginning with the concept of higher-order compressed sensing (HO-CS) [82]. Many tensor-based dictionary learning, or multilinear dictionary learning (MDL), methods have been introduced including the Kronecker OMP [101], K-CPD [102], K-HOSVD [103], T-MOD [104] and the joint optimization between MDL and HO-CS [105]. At the time of researching this, we observed that no online implementation of the MDL had been proposed yet, and hence comes the online multilinear dictionary learning (OMDL) algorithm, the second class of the main algorithms in this thesis which has been published in [3].

In the following sections, we quickly go though the traditional DL and CS problems, then go straight to their multilinear equivalents. Then, we derive the algorithm for the mode-wise dictionary update recursion as well as showcasing the empirical evidence of convergence. We afterwards proceed to render the HO-CS task sequential to complete the whole algorithm. Finally, the convergence analysis is provided before extensive numerical experiments are illustrated to affirm the whole analysis of the algorithm before the chapter is concluded.

5.1 Briefs on the Classical DL/CS Problem

5.1.1 Linear DL

In the classical matrix/vector setting, the goal of DL is to factorize a signal of interest, $\mathbf{x} \in \mathbb{R}^J$, into its sparse representation, $\mathbf{s} \in \mathbb{R}^L$, and a (sparsifying) dictionary, $\mathbf{\Psi} \in \mathbb{R}^{J \times L}$, which is, as a result, overcomplete ($J < L$). This factorization is expressed in the form of linear equation as $\mathbf{x} \cong \mathbf{\Psi}\mathbf{s}$. The sparse vector, $\mathbf{s}$, which only has S non-zero elements where $\|\mathbf{s}\|_0 = S \ll L$.

A classical DL problem considers a finite set of t *unlabeled* training signals, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t] \in \mathbb{R}^{J \times t}$, with their corresponding sparse representations, $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_t] \in \mathbb{R}^{L \times t}$ with respect to the representation dictionary $\mathbf{\Psi}$, and can be cast into

$$\begin{aligned} &\min_{\mathbf{\Psi}, \mathbf{S}} \sum_{\tau=1}^{t} w_\tau \ell_u(\mathbf{x}_\tau, \mathbf{\Psi}, \mathbf{s}_\tau) \\ &\text{s.t. } \mathbf{\Psi} \in \boldsymbol{C} \subset \mathbb{R}^{J \times L} \text{ and } \|\mathbf{s}_\tau\|_0 \leq S, \forall \tau \in \boldsymbol{t} \end{aligned} \tag{5.1}$$

where $w_\tau \geq 0$ is a weighting parameter similar to the way a forgetting factor in the previous chapter works, $\boldsymbol{t} \triangleq \{1, 2, \ldots, t\}$, $\boldsymbol{C}$ is a constraint space of $\mathbf{\Psi}$, and $\ell_u(\cdot)$ is an objective function where the index u emphasizes the *unsupervised* characteristics of the DL problem. Also similar to the quaternion-valued adaptive algorithms, the linear least squares can be employed as a cost function here, yielding

$$\ell_u(\mathbf{x}_\tau, \mathbf{\Psi}, \mathbf{s}_\tau) = \|\mathbf{x}_\tau - \mathbf{\Psi}\mathbf{s}_\tau\|_2^2. \tag{5.2}$$

Unlike the quaternion-valued adaptive algorithms, however, the problem in eq. (5.1) is non-convex because there are two distinct unknown variables, $\mathbf{\Psi}$ and $\mathbf{S}$. The most common method to deal with this situation is a strategy called alternating minimization where one variable is optimized while the other is fixed and vice versa in an alternate fashion. In this particular case, the optimization of $\mathbf{\Psi}$ with fixed $\mathbf{S}$ is called the *dictionary update* task, while finding $\mathbf{S}$ with fixed $\mathbf{\Psi}$ is known as the *sparse coding* problem.

In the dictionary update step, let $\ell_u(\mathbf{x}_\tau, \mathbf{\Psi}) = \ell_u(\mathbf{x}_\tau, \mathbf{\Psi}, \hat{\mathbf{s}}_\tau)$ where $\hat{\mathbf{s}}_\tau, \forall \tau \in \boldsymbol{t}$ is

calculated from the sparse coding task in the preceding alternate step. The problem in eq. (5.1) is then re-formulated to

$$\min_{\mathbf{\Psi}} \sum_{\tau=1}^{t} w_\tau \ell_u(\mathbf{x}_\tau, \mathbf{\Psi}) \;\; \text{s.t. } \mathbf{\Psi} \in \boldsymbol{C} \subset \mathbb{R}^{J \times L}. \tag{5.3}$$

Usually, the dictionary $\mathbf{\Psi}$ shall conform to some condition, especially norm condition to prevent $\mathbf{\Psi}$ from becoming arbitrarily large, and thus constraint space $\boldsymbol{C}$ is required. The dictionary update problem in eq. (5.3) can be solved in either a *batch* ([84–86]) or an *online* ([87–89]) manner.

In the sparse coding step, now let $\ell_u(\mathbf{x}_\tau, \mathbf{s}_\tau) = \ell_u(\mathbf{x}_\tau, \hat{\mathbf{\Psi}}, \mathbf{s}_\tau)$ where $\hat{\mathbf{\Psi}}$ is calculated from the dictionary update step in the preceding alternate step.. Likewise, eq. (5.1) becomes

$$\min_{\mathbf{S}} \sum_{\tau=1}^{t} w_\tau \ell_u(\mathbf{x}_\tau, \mathbf{s}_\tau) \;\; \text{s.t. } \|\mathbf{s}_\tau\|_0 \leq S, \, \forall \tau \in \boldsymbol{t}.$$

Notice that in the sparse coding setting, the constraint depends on each single sparse vector $\mathbf{s}_\tau$; therefore, the problem above can be independently solved for the most optimal $\mathbf{s}_\tau$ for every τ, i.e.

$$\min_{\mathbf{s}_\tau} \ell_u(\mathbf{x}_\tau, \mathbf{s}_\tau) \;\; \text{s.t. } \|\mathbf{s}_\tau\|_0 \leq S, \, \forall \tau \in \boldsymbol{t}. \tag{5.4}$$

Additionally, it should be noted that sparse coding is an older problem than the *learning* of dictionaries, having existed since the dictionaries were still *pre-determined* (overcomplete Fourier and wavelets). More importantly, it is the intertwining part of the DL problems which connect to the compressed sensing paradigm.

5.1.2 Compressed Sensing

The compressed sensing (CS) [80,81,94] can be thought of as a class of sampling strategies which achieves better compression than the Shannon-Nyquist policy to obtain a measurement signal, $\mathbf{y} \in \mathbb{R}^I$, but still able to recover without loss the sparse signal, $\mathbf{s} \in \mathbb{R}^L$ where $I < L$, by solving [94]

$$\min_{\mathbf{s}} \; \|\mathbf{s}\|_0 \;\; \text{s.t. } \mathbf{y} = \mathbf{\Theta}\mathbf{s}.$$

where $\mathbf{\Theta} \in \mathbb{R}^{I \times L}$ is called a *sensing matrix*. As mentioned before, natural signals are rarely sparse outright, and that is how a DL task practically comes as a factorization technique to obtain the sparse representation of the signals. With the DL framework, assuming that the signal of interest, $\mathbf{x}_\tau$, is in the form $\mathbf{x}_\tau = \mathbf{\Psi}\mathbf{s}_\tau$, $\forall \tau \in \boldsymbol{t}$, as described above, the reconstruction problem [81] is recast into

$$\min_{\mathbf{s}_\tau} \|\mathbf{s}_\tau\|_0 \quad \text{s.t. } \mathbf{y}_\tau = \mathbf{\Theta}\mathbf{s}_\tau \triangleq \mathbf{\Phi}\mathbf{\Psi}\mathbf{s}_\tau, \forall \tau \in \boldsymbol{t} \tag{5.5}$$

where $\mathbf{\Phi} \in \mathbb{R}^{I \times J}$ is termed a *projection matrix* [80]. The combinatorial nature of ℓ_0-norm, $\|\cdot\|_0$ makes eq. (5.5) an NP-hard problem. A common way to cope with this includes greedy algorithms, ℓ_1 norm as an alternative to ℓ_0-norm, or Bregman iteration [81]. These approximate approaches can attain arbitrarily small error if the sensing matrix $\mathbf{\Theta}$ obeys an *orthonomality-like* conditions such as mutual coherence [94] or restricted isometry property (RIP) [106][1].

The current direction in CS problem, apart from sparse coding, to design the optimal projection matrix $\mathbf{\Phi}$ with respect to the learned dictionary $\mathbf{\Psi}$ so that $\mathbf{\Theta}$ satisfies those conditions. Of these two, the mutual coherence of $\mathbf{\Theta}$, denoted by $\mu(\mathbf{\Theta})$, is analytically simpler and leads to the spring of many designs of projection matrix $\mathbf{\Phi}$ based on the optimization of the Gram matrix of $\mathbf{\Theta}$, defined as $\mathbf{\Theta}^T\mathbf{\Theta}$, in order to be as close as possible to a target equiangular tight-frame (ETF) Gram matrix $\mathbf{\Gamma} \in \boldsymbol{G}_{\underline{\mu}}$, i.e. [97]

$$\min_{\mathbf{\Theta}} \|\mathbf{\Gamma} - \mathbf{\Theta}^T\mathbf{\Theta}\|_F^2 \triangleq \min_{\mathbf{\Phi}} \|\mathbf{\Gamma} - \mathbf{\Psi}^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{\Psi}\|_F^2 \tag{5.6}$$

where $\boldsymbol{G}_{\underline{\mu}}$ is a set of relaxed ETF Gram matrices defined as

$$\begin{aligned} \boldsymbol{G}_{\underline{\mu}} \triangleq \{\mathbf{\Gamma} \in \mathbb{R}^{L \times L} : \mathbf{\Gamma} = \mathbf{\Gamma}^T, \operatorname{diag}(\mathbf{\Gamma}) = 1, \\ \max_{i \neq j} |\mathbf{\Gamma}(i,j)| \leq \underline{\mu}\}. \end{aligned} \tag{5.7}$$

[1] These conditions are crucial part of the CS problem as it makes possible the full recovery of the signals of interest, $\mathbf{x}_\tau$, from the *undersampled* measurement $\mathbf{y}_\tau$ via $\mathbf{\Phi}$ with $I < J$ [106].

The parameter $\underline{\mu}$ is the lower bound of $\mu(\boldsymbol{\Theta})$ given by [107]

$$\underline{\mu} = \sqrt{\frac{L-I}{I(L-1)}} \leq \mu(\boldsymbol{\Theta}) \leq 1. \tag{5.8}$$

Observe that eq. (5.6) is highly non-convex. Therefore, its solution is likely to be sub-optimal. Most existing algorithms do not tackle this directly but attempt to reduce the chance of setting stuck in local minimum by rendering the update equation as gradual as possible [95–98]. It is also surprisingly interesting that the formulation of eq. (5.6) looks similar to that of the PCA, even though there is nothing to do with it.

5.2 Online Dictionary Learning for Tensors

In our proposed OMDL routine, we employ the accelerated gradient method used in our already introduced momentum-based quaternion-valued adaptive algorithms [1,2] and extend it to the setting of online tensor dictionary. Thanks to the isomorphism discussed in the previous chapter, the analysis of algorithm can be used almost interchangeably, as will be manifested here as well. Since the sparse coding is out of scope in our research effort, the Kronecker othogonal matching pursuit [101] and multipath mathching pursuit [108], of which both guarantees the local minimum of the solution, are employed for the sparse coding.

5.2.1 Preliminaries

Let $\boldsymbol{\mathcal{X}}^{(\tau)} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}, \forall \tau \in \boldsymbol{t}$ be an observed sequence of t N^{th}-order tensors. It multilinear dictionary factorization can be expressed in the TKD form as [82]

$$\boldsymbol{\mathcal{X}}^{(\tau)} = \boldsymbol{\mathcal{S}}^{(\tau)} \times_1 \boldsymbol{\Psi}_1 \times_2 \boldsymbol{\Psi}_2 \cdots \times_N \boldsymbol{\Psi}_N + \boldsymbol{\mathcal{E}}^{(\tau)}, \forall \tau \in \boldsymbol{t}, \tag{5.9}$$

where $\boldsymbol{\Psi}_n \in \mathbb{R}^{J_n \times L_n}$ is a mode-n overcomplete dictionary (i.e. $J_n < L_n$), $\forall n \in \boldsymbol{N}$, $\boldsymbol{\mathcal{S}}^{(\tau)} \in \mathbb{R}^{L_1 \times L_2 \times \cdots \times L_N}$ are the S-sparse core tensors associated with $\boldsymbol{\mathcal{X}}^{(\tau)}$, and $\boldsymbol{\mathcal{E}}^{(\tau)}$ are the representation error. For the DL problem, it usually is desirable that the number

of non-zero-elements of $\boldsymbol{\mathcal{S}}^{(\tau)}$ is much fewer than the total dimension of the observation i.e. $S \ll \prod_{n=1}^{N} J_n$. Almost analogous to its classical equivalent eq. (5.3), the multilinear dictionary learning task is given by

$$\min_{\{\boldsymbol{\Psi}\}} \sum_{\tau=1}^{t} w^{(\tau)} \ell_u(\boldsymbol{\mathcal{X}}^{(\tau)}, \{\boldsymbol{\Psi}\}) \;\; \text{s.t. } \boldsymbol{\Psi}_n \in \boldsymbol{C}_n, \forall n \in \boldsymbol{N}. \tag{5.10}$$

where $\{\boldsymbol{\Psi}\} = \{\boldsymbol{\Psi}_n, \;\; \forall n \in \boldsymbol{N}\}$ is a set of all mode-wise dictionaries, $\boldsymbol{C}_n \subset \mathbb{R}^{J_n \times L_n}$ is a mode-n constraint space controlling the size of $\boldsymbol{\Psi}_n$. Now, the cost function $\ell_u(\cdot)$ in eq. (5.2) takes a *multilinear* least-square form [103, 104]:

$$\ell_u(\boldsymbol{\mathcal{X}}^{(\tau)}, \{\boldsymbol{\Psi}\}) = \|\boldsymbol{\mathcal{X}}^{(\tau)} - \boldsymbol{\mathcal{S}}^{(\tau)} \times_1 \boldsymbol{\Psi}_1 \times_2 \boldsymbol{\Psi}_2 \cdots \times_N \boldsymbol{\Psi}_N\|_F^2. \tag{5.11}$$

Unlike the traditional matrix/vector counterpart, fixing just $\boldsymbol{\mathcal{S}}^{(\tau)}$ is insufficient to make eqs. (5.10) and (5.11) convex; instead, it is multi-convex due to its multilinear structure. This can be resolved by resorting to the alternating linear scheme [109], whereby one mode-n dictionary is optimized at a time while the rest mode-n dictionaries are fixed and so on[2]. Hence, let $\mathcal{J}_n^{(t)}(\{\boldsymbol{\Psi}\})$, or $\mathcal{J}_n^{(t)}$ in short, be an effective cost function of (5.10) and (5.11) overall, *contracted* into a mode-n expression as

$$\mathcal{J}_n^{(t)} \triangleq \frac{1}{2} \sum_{\tau=1}^{t} w^{(\tau)} \left\| \boldsymbol{\mathcal{X}}^{(\tau)} - \tilde{\boldsymbol{\mathcal{S}}}_n^{(\tau)} \times_n \boldsymbol{\Psi}_n \right\|_F^2, \tag{5.12}$$

where $\tilde{\boldsymbol{\mathcal{S}}}_n^{(\tau)} = \boldsymbol{\mathcal{S}}^{(\tau)} \times_1 \boldsymbol{\Psi}_1 \times_2 \boldsymbol{\Psi}_2 \cdots \times_{n-1} \boldsymbol{\Psi}_{n-1} \times_{n+1} \boldsymbol{\Psi}_{n+1} \cdots \times_N \boldsymbol{\Psi}_N$. With the knowledge of tensor mode-all-but-n contraction detailed in the previous chapter trace-norm relation of a matrix [16], the right-hand side of eq. (5.12) can be manipulated into a quadratic

[2] the alternating linear scheme can be applied without the loss of generality on condition that all mode-n dictionaries are separable (i.e. each multilinear atom is only in the form of a rank-1 tensor) [109].

equation of a matrix form as

$$\begin{aligned}\mathcal{J}_n^{(t)} =& \mathrm{Tr}\left(\frac{1}{2}\boldsymbol{\Psi}_n\left(\sum_{\tau=1}^{t} w^{(\tau)}\left[\tilde{\boldsymbol{\mathcal{S}}}_n^{(\tau)} \times_{/n}^{/n} \tilde{\boldsymbol{\mathcal{S}}}_n^{(\tau)}\right]\right)\boldsymbol{\Psi}_n^T\right.\\ &-\left(\sum_{\tau=1}^{t} w^{(\tau)}\left[\boldsymbol{\mathcal{X}}^{(\tau)} \times_{/n}^{/n} \tilde{\boldsymbol{\mathcal{S}}}_n^{(\tau)}\right]\right)\boldsymbol{\Psi}_n^T\\ &\left.+\frac{1}{2}\sum_{\tau=1}^{t} w^{(\tau)}\left[\boldsymbol{\mathcal{X}}^{(\tau)} \times_{/n}^{/n} \boldsymbol{\mathcal{X}}^{(\tau)}\right]\right)\end{aligned} \tag{5.13}$$

where the symbol $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix. With eq. (5.13), the all-mode multilinear problem in eq. (5.10) turns into n mode-wise *linear* sub-problems, that is

$$\min_{\boldsymbol{\Psi}_n} \mathcal{J}_n^{(t)} \quad \text{s.t. } \boldsymbol{\Psi}_n \in \boldsymbol{C}_n. \tag{5.14}$$

Most existing work in the MDL problem utilizes eq. (5.14) for an *offline* alternating-least-squares approach whereby all training pairs $(\boldsymbol{\mathcal{X}}^{(\tau)}, \tilde{\boldsymbol{\mathcal{S}}}_n^{(\tau)})$, $\forall \tau \in \boldsymbol{t}$ are considered at once in a single batch ([103–105]).

5.2.2 Alternating Linear Scheme for Online MDL

To realize the OMDL algorithm based on eqs. (5.13) and (5.14), we follow the same derivation of the c-moment WL-QLMS [2] over the n-moment one [1] due to its parsimonious formulae without loss in performance as illustrated in the Chapter 3. For each mode-n cost function $\mathcal{J}_n^{(t)}$, let $w^{(\tau)} = \lambda^{t-\tau}$ and $\tilde{\mathbf{S}}_n^{(\tau)}$ and $\tilde{\mathbf{Q}}_n^{(\tau)}$ be

$$\mathbf{S}_n^{(t)} \triangleq \tilde{\boldsymbol{\mathcal{S}}}_n^{(t)} \times_{/n}^{/n} \tilde{\boldsymbol{\mathcal{S}}}_n^{(t)}, \tag{5.15}$$

$$\mathbf{Q}_n^{(t)} \triangleq \boldsymbol{\mathcal{X}}^{(t)} \times_{/n}^{/n} \tilde{\boldsymbol{\mathcal{S}}}_n^{(t)}. \tag{5.16}$$

Since the rightmost term of eq. (5.13) does not depend on $\boldsymbol{\Psi}_n$, it is left out *for the purpose of clean display only.* (5.14) is therefore equivalent *analytically* to

$$\min_{\boldsymbol{\Psi}_n} \hat{\mathcal{J}}_n^{(t)} \quad \text{s.t. } \boldsymbol{\Psi}_n \in \boldsymbol{C}_n \tag{5.17}$$

where

$$\hat{\mathcal{J}}_n^{(t)} = \mathrm{Tr}\left(\frac{1}{2}\mathbf{\Psi}_n\mathbf{R}_n^{(t)}\mathbf{\Psi}_n^T - \mathbf{P}_n^{(t)}\mathbf{\Psi}_n^T\right) \tag{5.18}$$

with the recursion very similar to eqs. (3.10) and (3.11) of the quaternion-valued adaptive filtering algorithms,

$$\mathbf{R}_n^{(t)} \triangleq \sum_{\tau=1}^{t}\lambda^{t-\tau}\left[\tilde{\boldsymbol{\mathcal{S}}}_n^{(\tau)} \times_{/n}^{/n} \tilde{\boldsymbol{\mathcal{S}}}_n^{(\tau)}\right] = \lambda\mathbf{R}_n^{(t-1)} + \mathbf{S}_n^{(t)}, \tag{5.19}$$

$$\mathbf{P}_n^{(t)} \triangleq \sum_{\tau=1}^{t}\lambda^{t-\tau}\left[\boldsymbol{\mathcal{X}}^{(\tau)} \times_{/n}^{/n} \tilde{\boldsymbol{\mathcal{S}}}_n^{(\tau)}\right] = \lambda\mathbf{P}_n^{(t-1)} + \mathbf{Q}_n^{(t)}, \tag{5.20}$$

and $\lambda \in (0, 1]$ is a forgetting parameter similar to that of the proposed momentum WL-QLMS algorithms. To solve for the descent direction, we take the stochastic conjugate form as in eq. (3.7), yielding

$$\mathbf{D}_n^{(t)} = -\mathbf{G}_n^{(t)} + \beta_n^{(t)}\mathbf{D}_n^{(t-1)} \tag{5.21}$$

and the mode-n gradient of $\mathcal{J}_n^{(t)}$ is given by

$$\mathbf{G}_n^{(t)} \triangleq \left.\frac{\partial\mathcal{J}_n^{(t)}}{\partial\mathbf{\Psi}_n}\right|_{\mathbf{\Psi}_n=\mathbf{\Psi}_n^{(t-1)}} = \mathbf{\Psi}_n^{(t-1)}\mathbf{R}_n^{(t)} - \mathbf{P}_n^{(t)}. \tag{5.22}$$

Analogous to eq. (3.29), we arrive at the following theorem [2](without proof as it is similar to that of the Chapter 3),

Theorem 3. *A set of matrices* $\{\mathbf{D}_n^{(1)}, \mathbf{D}_n^{(2)}, ..., \mathbf{D}_n^{(t)}\}$ *of the form* (5.21) *and satisfying*

$$\mathrm{Tr}\left(\mathbf{D}_n^{(t-1)}\mathbf{R}_n^{(t)}\mathbf{D}_n^{(t)^T}\right) = 0, \ \forall t, \tag{5.23}$$

is a descent direction of the objective function (5.18)*,*

With eqs. (5.21) and (5.23), we obtain the value of a extrapolation parameter $\beta_n^{(t)}$ as

$$\beta_n^{(t)} = \frac{\left\langle \mathbf{H}_n^{(t)}, \mathbf{G}_n^{(t)}\right\rangle_F}{\left\langle \mathbf{H}_n^{(t)}, \mathbf{D}_n^{(t-1)}\right\rangle_F} \tag{5.24}$$

where $\langle \cdot, \cdot \rangle_F$ is a Frobenius inner product and

$$\mathbf{H}_n^{(t)} = \mathbf{D}_n^{(t-1)} \mathbf{R}_n^{(t)}. \tag{5.25}$$

Finally, now not exactly similar to the original c-moment WL-QLMS, the mode-n dictionary $\mathbf{\Psi}_n^{(t)}$ is iteratively calculated as

$$\mathbf{\Psi}_n^{(t)} = \Pi_{\boldsymbol{C}_n} \left[\mathbf{\Upsilon}_n^{(t)} \right] = \Pi_{\boldsymbol{C}_n} \left[\mathbf{\Psi}_n^{(t-1)} + \mathbf{D}_n^{(t)} \mathbf{A}_n^{(t)} \right] \tag{5.26}$$

where $\mathbf{A}_n^{(t)}$ is a diagonal matrix, of which the diagonals are $\alpha_n^{(t)}(l) = \mathbf{R}_n^{(t)}[l, l]$, and $\Pi_{\boldsymbol{C}_n}[\cdot]$ is a projection operator onto the constraint space $\boldsymbol{C}_n$. Since the solution to $\mathbf{\Psi}_n^{(t)}$ is subject to affine transformation due to the multi-convexity of the problem, it is desirable that the overall size of $\mathbf{\Psi}_n^{(t)}$ is confined by its atoms. In other words, the convex set $\boldsymbol{C}_n$ is a linear map which conserves a space spanned by the dictionary atoms (the column space), which turns (5.26) into

$$\mathbf{\Psi}_n^{(t)} = \mathbf{\Upsilon}_n^{(t)} \mathbf{\Pi}_n^{(t)} \tag{5.27}$$

and $\mathbf{\Pi}_n^{(t)}$ is a diagonal matrix the diagonals of which, $\pi_n^{(t)}(l)$, are given by

$$\pi_n^{(t)}(l) = \frac{1}{\max\left(\|\mathbf{u}_n^{(t)}(l)\|_2, 1 \right)}, \forall l = 1, 2, \ldots, L_n \tag{5.28}$$

where $\mathbf{u}_n^{(t)}(l)$ is the lth column vector of $\mathbf{\Upsilon}_n^{(t)}$. At last, The proposed OMDL algorithm is summarized in Algorithm 1 below, where $\delta > 0$ is used as a stopping criterion.

5.2.3 Insights into Convergence

If $\boldsymbol{\mathcal{S}}^{(t)}$ is known, then the OMDL problem reduces to a class of adaptive filtering algorithm where the convergence behavior is analytically interchangeable across the class [110]. As a matter of fact, it is the sparse coding stage that dictates the convergence characteristics [111, 112]. There exist the case where then the DL algorithm attains global optimal solution when sparse coding can give one too [113]. To further verify this claim, an extra experiment is provided on the synthetic scenario where the sparse core is assumed known.

Algorithm 3: The OMDL Algorithm

Input : $\boldsymbol{\mathcal{X}}^{(t)} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ (inputs), T (number of inputs), $\boldsymbol{\Psi}_n^{(0)} \in \mathbb{R}^{J_n \times L_n}$ (initial dictionaries), N (number of modes), λ (forgetting factor)

Output: $\boldsymbol{\Psi}_n^{(t)}$ (modewise dictionaries)

Initialize $\mathbf{R}_n^{(0)} = \underline{\mathbf{0}}$, $\mathbf{P}_n^{(0)} = \underline{\mathbf{0}}$ and $\mathbf{D}_n^{(0)} = \underline{\mathbf{0}}$ $\forall n$;

for $t = 1$ to T **do**

  Obtain sparse core tensor $\boldsymbol{\mathcal{S}}^{(t)}$ via appropriate sparse coding scheme e.g. [101];

  for $n = 1$ to N **do**

    Update $\mathbf{S}_n^{(t)}$ and $\mathbf{Q}_n^{(t)}$ by eqs. (5.15) and (5.16);

    $\mathbf{R}_n^{(t)} = \lambda \mathbf{R}_n^{(t-1)} + \mathbf{S}_n^{(t)}$;

    $\mathbf{P}_n^{(t)} = \lambda \mathbf{P}_n^{(t-1)} + \mathbf{Q}_n^{(t)}$;

    $\mathbf{G}_n^{(t)} = \boldsymbol{\Psi}_n^{(t-1)} \mathbf{R}_n^{(t)} - \mathbf{P}_n^{(t)}$;

    $\mathbf{H}_n^{(t)} = \mathbf{D}_n^{(t-1)} \mathbf{R}_n^{(t)}$;

    $\beta_n^{(t)} = \frac{\left\langle \mathbf{H}_n^{(t)}, \mathbf{G}_n^{(t)} \right\rangle_F}{\left\langle \mathbf{H}_n^{(t)}, \mathbf{D}_n^{(t-1)} \right\rangle_F}$, $(\beta_n^{(1)} = 0)$;

    $\mathbf{D}_n^{(t)} = -\mathbf{G}_n^{(t)} + \beta_n^{(t)} \mathbf{D}_n^{(t-1)}$;

    Update $\mathbf{A}_n^{(t)}$ where its diagonals $\alpha_n^{(t)}(l) = \mathbf{R}_n^{(t)}[l, l]$;

    $\boldsymbol{\Upsilon}_n^{(t)} = \boldsymbol{\Psi}_n^{(t-1)} + \mathbf{D}_n^{(t)} \mathbf{A}_n^{(t)}$;

    Update $\boldsymbol{\Pi}_n^{(t)}$ by eq. (5.28);

    $\boldsymbol{\Psi}_n^{(t)} = \boldsymbol{\Upsilon}_n^{(t)} \boldsymbol{\Pi}_n^{(t)}$;

  end

end

The core tensor $\boldsymbol{\mathcal{S}}^{(t)} \in \mathbb{R}^{20\times20\times20}$ with equal mode-wise sparsity, $S_n = 8(n = 1, 2, 3)$, has non-zero elements randomly selected from a Gaussian distribution and each mode-n dictionary, $\boldsymbol{\Psi}_n \in \mathbb{R}^{10\times20}(n = 1, 2, 3)$, was generated by Gaussian random variable with mean and variance equal 0 and 1 respectively. With SNR set to 0 dB, the 3rd-order tensor data, $\boldsymbol{\mathcal{X}}^{(t)} \in \mathbb{R}^{10\times10\times10}$, was generated via eq. (5.9). The metric for successful recovery of the the mode-wise dictionary $\boldsymbol{\Psi}_n \in \mathbb{R}^{10\times20}(n = 1, 2, 3)$ is given by θ which can be thought of as an *angular distance* between the real dictionary atoms, $\boldsymbol{\psi}_{real}$, and the recovered ones, $\boldsymbol{\psi}_{learned}$. If this distance is below some predefined *threshold*, then that particular atom is considered *successfully recovered*, that is,

$$\frac{\boldsymbol{\psi}_{real} \cdot \boldsymbol{\psi}_{learned}}{|\boldsymbol{\psi}_{real}||\boldsymbol{\psi}_{learned}|} > cos(\theta).$$

The result was averaged over 100 independent trails for comparing three tensor dictionary learning routines: the proposed MODL, TMOD [104] and the TKSVD [105] as shown in Fig. 5.1. Three lines of the same color represent 3 mode-wise dictionaries.

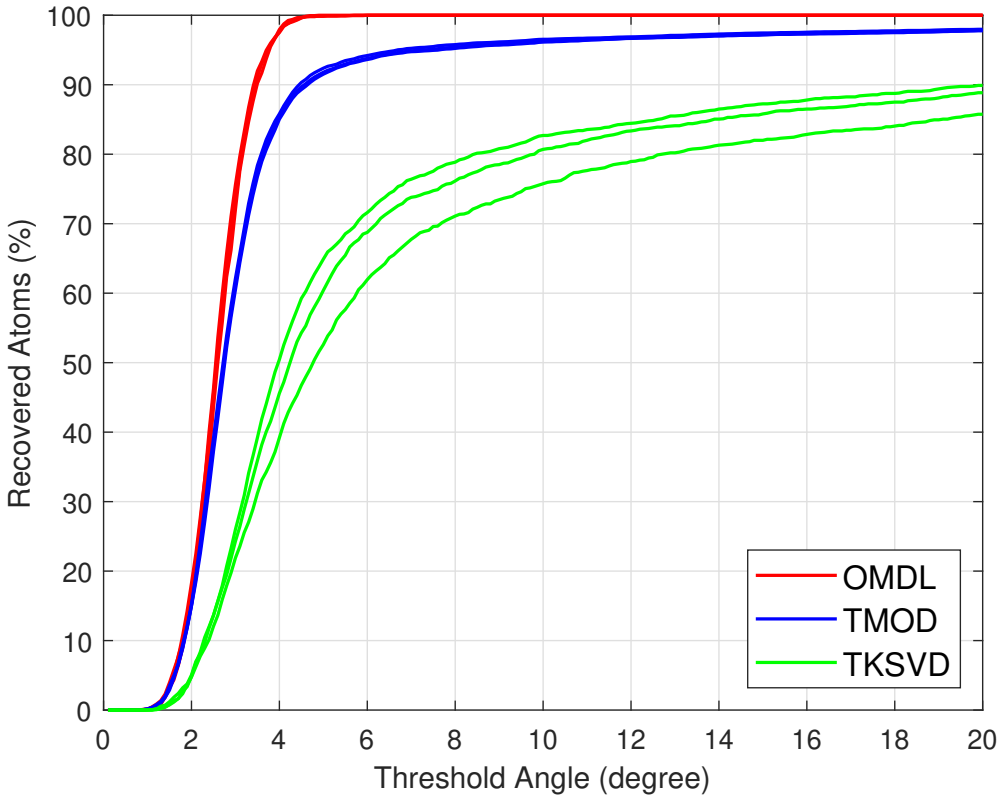


Figure 5.1: [3] Successful recovery of atoms with respect to different 'threshold' angle for all 3 modes grouped in the same color, with red (MODL), blue (TMOD) and green (TKSVD) ($L_n = 20$, $J_n = 10$, $S_n = 8$, $n = 1, 2, 3$)

From Fig. 5.1, only the MODL routine, if the metric threshold is above 5, could achieve 100 percent recovery rate while the other two were unable to do so regardless of how large the threshold is. To treat the test fairly, the poor performance of the TKSVD is likely due to the fact that the TKSVD method keeps changing the sparse core tensors iteratively with the dictionaries and thus loses the advantage gained from the experiment setting of known core tensors. Overall, this little simulation illustrates that, when the sparse coding can effectively give optimal sparse core tensors, not only could the OMDL algorithm lead to global optimality, but the alternating optimization is not shown to impede its global convergence *per se* as typically feared. For a little more rigorous proof of convergence, we can cast the problem in eqs. (5.10) and (5.11) into a flat matrix model via the Kronecker structure in eq. (4.4), and as a result follow the same mechanism in [88]

as a special case. Note that the proof of convergence in [88] only guarantees convergence to a stationary point, not necessarily global.

5.3 Joint Design for Sequential HO-CS

Many HO-CS methods have been invented mostly for an offline setting. They either implement a basic identity matrix as a target Gram matrix [105], or include the error of projection as a criterion for optimality [98, 100]. In this thesis, we follow the simplified robust design with the use of relaxed ETF scheme [99]. By simplification, we leave out the effect of projected error by realizing its upper-bound, which is the *scaled* norm of the projection matrix instead. This notion can be straightforwardly applied in the case of MODL paradigm.

5.3.1 Preliminaries

Building on the base of HO-CS problem in eqs. (4.2) and (4.3) together with the coupled dictionary representation eq. (5.9), the sequential HO-CS task can be initially written as

$$\begin{aligned} &\min_{\boldsymbol{\mathcal{S}}^{(\tau)}} \left\|\boldsymbol{\mathcal{S}}^{(\tau)}\right\|_0 \quad \text{s.t.} \\ &\boldsymbol{\mathcal{Y}}^{(\tau)} = \boldsymbol{\mathcal{S}}^{(\tau)} \times_1 \boldsymbol{\Theta}_1 \times_2 \boldsymbol{\Theta}_2 \cdots \times_N \boldsymbol{\Theta}_N, \forall \tau \in \boldsymbol{t}, \end{aligned} \tag{5.29}$$

where $\boldsymbol{\mathcal{Y}}^{(\tau)} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ is measurement tensor signal and $\boldsymbol{\Theta}_N \triangleq \boldsymbol{\Phi}_N \boldsymbol{\Psi}_N \in \mathbb{R}^{I_n \times L_n}$, $\forall n \in \boldsymbol{N}$ is a mode-n sensing matrix. With (5.9), $\boldsymbol{\mathcal{Y}}^{(\tau)}$ can be expressed in terms of $\boldsymbol{\mathcal{X}}^{(\tau)}$ as

$$\boldsymbol{\mathcal{Y}}^{(\tau)} = \boldsymbol{\mathcal{X}}^{(\tau)} \times_1 \boldsymbol{\Phi}_1 \times_2 \boldsymbol{\Phi}_2 \cdots \times_N \boldsymbol{\Phi}_N, \forall \tau \in \boldsymbol{t} \tag{5.30}$$

where $\boldsymbol{\Phi}_n \in \mathbb{R}^{I_n \times J_n}$ is called a mode-n projection matrix with $I_n \leq J_n$, $\forall n \in \boldsymbol{N}$. With the Kronecker representation of (5.29) is eqs. (4.4) and (4.5) in the previous chapter, extended from the conventional Gram matrix problem in (5.6)-(5.8), the Kronecker-structure equivalent is

$$\min_{\boldsymbol{\Theta}} \|\boldsymbol{\Gamma} - \boldsymbol{\Theta}^T \boldsymbol{\Theta}\|_F^2 = \min_{\boldsymbol{\Phi}} \|\boldsymbol{\Gamma} - \boldsymbol{\Psi}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{\Psi}\|_F^2 \tag{5.31}$$

with $\mathbf{\Gamma} \in \boldsymbol{G}_{\underline{\mu}}$ given by

$$\begin{aligned}\boldsymbol{G}_{\underline{\mu}} \triangleq \{\mathbf{\Gamma} \in \mathbb{R}^{L_1 L_2 \ldots L_N \times L_1 L_2 \ldots L_N} : \mathbf{\Gamma} = \mathbf{\Gamma}^T, \\ \operatorname{diag}(\mathbf{\Gamma}) = 1, \max_{i \neq j} |\mathbf{\Gamma}(i,j)| \leq \underline{\mu}\}.\end{aligned} \tag{5.32}$$

and

$$\underline{\mu} = \sqrt{\left(\prod_{n=1}^{N} L_n - \prod_{n=1}^{N} I_n\right) \Big/ \left(\prod_{n=1}^{N} I_n (\prod_{n=1}^{N} L_n - 1)\right)}. \tag{5.33}$$

Eq. (5.31) can be solved via traditional CS paradigm; however, the massive dimensionality of $\mathbf{\Theta}$ due to the Kronecker structure [82] is the prime bottleneck of direct manipulation. Moreover, there is no explicit constraint to enforce the Kronecker structure of the resulting $\mathbf{\Theta}$. Also by the separable structure of (5.29), we will solve for each mode-n projection matrix in an alternating fashion [101, 114] like the OMDL above, as long as each mode-n projection matrix conforms to standard RIP or mutual coherence conditions (more rigorous theories can be found in [82]). All these efforts utilized an identity matrix as a target Gram matrix because it is the simplest matrix with Kronecker structure; however, we believe that is quite too simple and more sophisticated Gram structure could lead to better performance. Hence, the relaxed ETF scheme is employed for $\mathbf{\Gamma}$ in our thesis.

In order to implement an alternating and online minimization of (5.31), we instead solve an alternative problem

$$\min_{\mathbf{\Theta}} \|\mathbf{\Gamma} - \mathbf{\Theta}^T \mathbf{\Theta}\|_F^2 = \min_{\mathbf{\Phi}} \|\mathbf{\Gamma} - \mathbf{\Psi}^T \mathbf{\Phi}^T \mathbf{\Phi} \mathbf{\Psi}\|_F^2 \tag{5.34}$$

where $\mathbf{\Gamma} \triangleq \mathbf{\Gamma}_N \otimes \mathbf{\Gamma}_{N-1} \otimes \cdots \otimes \mathbf{\Gamma}_1$ with $\mathbf{\Gamma}_n \in \boldsymbol{G}_{\underline{\mu}_n}$ given by

$$\begin{aligned}\boldsymbol{G}_{\underline{\mu}_n} \triangleq \{\mathbf{\Gamma}_n \in \mathbb{R}^{L_n \times L_n} : \mathbf{\Gamma}_n = \mathbf{\Gamma}_n^T, \\ \operatorname{diag}(\mathbf{\Gamma}_n) = 1, \max_{i \neq j} |\mathbf{\Gamma}_n(i,j)| \leq \underline{\mu}_n\}.\end{aligned} \tag{5.35}$$

and

$$\underline{\mu}_n = \min\left(\sqrt{\frac{L_n - I_n}{I_n(L_n - 1)}}, \underline{\mu}\right). \tag{5.36}$$

Here through (5.35) and (5.36), $\mathbf{\Gamma}$ is guaranteed to satisfy (5.32), i.e. $\mathbf{\Gamma} \in \boldsymbol{G}_{\underline{\mu}}$. As a result,

the Kronecker Gram matrices satisfy the global constraint in eq. (5.32) while some room is left for a tighter solution for each mode-wise Gram matrix via eq. (5.35).

5.3.2 Alternating Scheme for Mode-Wise Projection Matrix Design

In order to design robust projection matrices, the projected error should be as small as possible for the corresponding CS system to perform well in practice [96, 98]. This is equal to adding the projected error as a regularizer into (5.34), thus yielding the following optimization problem:

$$\min_{\mathbf{\Phi},\mathbf{\Gamma}} \|\mathbf{\Gamma} - \mathbf{\Psi}^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{\Psi}\|_F^2 + \sigma\|\mathbf{\Phi}\mathbf{e}\|_F^2 \tag{5.37}$$

where $\mathbf{e}$ is the vectorized SRE, $\text{vec}(\boldsymbol{\mathcal{E}})$, defined in (5.9) and σ is a weighting parameter. Without any assumptions, it is obvious that

$$\|\mathbf{\Phi}\mathbf{e}\|_F \leq \|\mathbf{\Phi}\|_F\|\mathbf{e}\|_2 = \left(\prod_{n=1}^{N}\|\mathbf{\Phi}_n\|_F\right)\|\boldsymbol{\mathcal{E}}\|_F. \tag{5.38}$$

In other words, the size of the projected error is bounded above by the sizes of the SRE and the projection matrices in all modes. Since $\|\boldsymbol{\mathcal{E}}\|_F$ is minimized at the dictionary learning stage, then $\prod_{n=1}^{N}\|\mathbf{\Phi}_n\|_F$ can be considered a surrogate of $\|\mathbf{\Phi}\mathbf{e}\|_F$ and minimized instead. Furthermore, if $\boldsymbol{\mathcal{E}}$ can be modelled as Gaussian noise and the number of training data, t, is large enough, then the equality holds in (5.37) [115]. These assumptions therefore simplify (5.37) to

$$\min_{\mathbf{\Phi},\mathbf{\Gamma}} \mathcal{V}^{(t)}(\mathbf{\Phi},\mathbf{\Gamma})$$

with

$$\mathcal{V}^{(t)}(\mathbf{\Phi},\mathbf{\Gamma}) = \|\mathbf{\Gamma} - \mathbf{\Psi}^{(t)^T}\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{\Psi}^{(t)}\|_F^2 + \sigma\prod_{n=1}^{N}\|\mathbf{\Phi}_n\|_F^2 \tag{5.39}$$

where the optimal $\mathbf{\Phi}$ is independent of $\boldsymbol{\mathcal{E}}$. This significantly facilitates online computation.

To address this non-convex problem, an alternating minimization algorithm is used. It is worth noting that this expression is the same as eq. (23) in [105] where alternating gradient descent is used for the non-separable approach employed in our paper as it was shown to outperform and more computationally efficient than the separable one. Firstly, a

shrinking operation is applied to (5.39) to obtain $\mathbf{\Gamma}_n$ mode by mode [96,116]. By defining $\mathbf{\Theta}_n^{(t)} \triangleq \mathbf{\Phi}_n^{(t)}\mathbf{\Psi}_n^{(t)}$ and $\mathbf{\Theta}_n^{(t^*)} \triangleq \mathbf{\Phi}_n^{(t-1)}\mathbf{\Psi}_n^{(t)}$, we now obtain

$$\mathbf{\Gamma}_n^{(t)}[i,j] = \begin{cases} \gamma_n(i,j), & |\gamma_n(i,j)| \leq \underline{\mu}_n \\ \text{sgn}[\gamma_n(i,j)]\underline{\mu}_n, & |\gamma_n(i,j)| > \underline{\mu}_n \\ 1, & i = j \end{cases} \tag{5.40}$$

$$\mathbf{\Gamma}_n^{(t^*)}[i,j] = \begin{cases} \gamma_n^*(i,j), & |\gamma_n^*(i,j)| \leq \underline{\mu}_n \\ \text{sgn}[\gamma_n^*(i,j)]\underline{\mu}_n, & |\gamma_n^*(i,j)| > \underline{\mu}_n \\ 1, & i = j \end{cases} \tag{5.41}$$

where γ_n and γ_n^* are the (i,j)-elements of the corresponding normalized Gram matrices $\mathbf{\Theta}_n^{(t)^T}\mathbf{\Theta}_n^{(t)}$ and $\mathbf{\Theta}_n^{(t^*)^T}\mathbf{\Theta}_n^{(t^*)}$ respectively. Then, $\bar{\mathbf{\Phi}}$ is iteratively calculated per mode. By defining the three following parameters:

$$\rho_n^{(t)} = \prod_{k=1}^{n-1} \|\mathbf{\Theta}_k^{(t)^T}\mathbf{\Theta}_k^{(t)}\|_F^2 \prod_{k=n+1}^{N} \|\mathbf{\Theta}_k^{(t^*)^T}\mathbf{\Theta}_k^{(t^*)}\|_F^2 \tag{5.42}$$

$$\begin{aligned} \omega_n^{(t)} = &\prod_{k=1}^{n-1} \text{Tr}\left(\mathbf{\Theta}_k^{(t)}\mathbf{\Gamma}_k^{(t)}\mathbf{\Theta}_k^{(t)^T}\right) \times \\ &\prod_{k=n+1}^{N} \text{Tr}\left(\mathbf{\Theta}_k^{(t^*)}\mathbf{\Gamma}_k^{(t^*)}\mathbf{\Theta}_k^{(t^*)^T}\right) \end{aligned} \tag{5.43}$$

$$\zeta_n^{(t)} = \prod_{k=1}^{n-1} \|\mathbf{\Phi}_k^{(t)}\|_F^2 \prod_{k=n+1}^{N} \|\mathbf{\Phi}_k^{(t-1)}\|_F^2 \tag{5.44}$$

the update equation for projection matrix becomes

$$\mathbf{\Phi}_n^{(t)} = \mathbf{\Phi}_n^{(t-1)} - \eta_n \mathbf{V}_n^{(t)} \tag{5.45}$$

where η_n is a stepsize parameter and the mode-wise gradient $\mathbf{V}_n^{(t)}$ is given by

$$\begin{aligned} \mathbf{V}_n^{(t)} \triangleq \left.\frac{\partial \mathcal{V}(\bar{\mathbf{\Phi}}, \bar{\mathbf{\Gamma}})}{\partial \mathbf{\Phi}_n}\right|_{\mathbf{\Phi}_n = \mathbf{\Phi}_n^{(t-1)}} &= \rho_n^{(t)}\left[\mathbf{\Theta}_n^{(t^*)}\mathbf{\Theta}_n^{(t^*)^T}\mathbf{\Theta}_n^{(t^*)}\mathbf{\Psi}_n^{(t)^T}\right] \\ &- \omega_n^{(t)}\left[\mathbf{\Theta}_n^{(t^*)}\mathbf{\Gamma}_n^{(t^*)}\mathbf{\Psi}_n^{(t)^T}\right] \\ &+ \sigma\zeta_n^{(t)}\left[\mathbf{\Phi}_n^{(t-1)}\right]. \end{aligned} \tag{5.46}$$

Note that all constants are absorbed into the regularizer σ. The joint optimization algorithm is summarized in Algorithm 2.

Algorithm 4: Joint Optimization Algorithm

Input : $\boldsymbol{\mathcal{X}}^{(t)} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ (inputs), T (number of inputs), $\boldsymbol{\Psi}_n^{(0)} \in \mathbb{R}^{J_n \times L_n}$ (initial dictionaries), $\boldsymbol{\Phi}_n^{(0)} \in \mathbb{R}^{I_n \times J_n}$ (initial sensing matrices), N (number of modes), λ (forgetting factor)

Output: $\boldsymbol{\Phi}_n^{(t)}$ (modewise sensing matrices), $\boldsymbol{\Psi}_n^{(t)}$ (modewise dictionaries)

```
Initialize R_n^(0) = 0, P_n^(0) = 0 and D_n^(0) = 0  ∀n;
for t = 1 to T do
    Obtain Ψ_n^(t) ∀n via the OMDL in Algorithm 1;
    for n = 1 to N do
        for k = 1 to n-1 do
            Θ_k^(t) = Φ_k^(t) Ψ_k^(t);
        end
        for k = n to N do
            Θ_k^(t*) = Φ_k^(t-1) Ψ_k^(t);
        end
        γ_n(i,j) = Θ_n^(t)^T Θ_n^(t)[i,j];
        γ*_n(i,j) = Θ_n^(t*)^T Θ_n^(t*)[i,j];
        Update Γ_n^(t) and Γ_n^(t*) via eqs. (5.40) and (5.41);
        Calculate ρ_n^(t), ω_n^(t), ζ_n^(t) via eqs. (5.42) to (5.44);
        Obtain V_n^(t) via eq. (5.46);
        Φ_n^(t) = Φ_n^(t-1) − η_n V_n^(t);
    end
end
```

5.4 Experimental Validation

A series of experiments were conducted to explore the performance of the proposed algorithms. The performance was evaluated against two criteria, the Normalized Root Mean Squared Error (NRMSE) and the Average Representation Error (ARE) [99], respectively given by

$$\sigma_{nrmse} = \frac{\|\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{S}} \times_1 \boldsymbol{\Psi}_1 \times_2 \boldsymbol{\Psi}_2 \cdots \times_N \boldsymbol{\Psi}_N\|_F}{\|\boldsymbol{\mathcal{X}}\|_F} \tag{5.47}$$

$$\sigma_{are} = \frac{\|(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{E}}) - \boldsymbol{\mathcal{S}} \times_1 \boldsymbol{\Psi}_1 \times_2 \boldsymbol{\Psi}_2 \cdots \times_N \boldsymbol{\Psi}_N\|_F}{\text{lens}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{E}})} \tag{5.48}$$

where lens$(\cdot)$ denotes the total number of elements of the operand.

A. Online Multilinear Dictionary Learning

In the first experiment, we compared the dictionary learning stage of the OMDL with those of other similar algorithms, namely TMOD and TKSVD. A set of observed data contained 1000 3-mode tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{8\times8\times8}$ generated by a full multilinear product between 1000 sparse core tensors $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{16\times16\times16}$ and mode-n dictionary $\boldsymbol{\Psi}_n \in \mathbb{R}^{8\times16}, n = 1, 2, 3$ with an average SNR of 20 dB and a forgetting factor $\lambda = 0.95$. For the TMOD and TKSVD, we adjusted the hyperparameters according to [104, 105] so that they yielded the best respective performance for fair comparison; all simulations were averaged over 100 realizations.

In Fig. 5.2, the two measures (NRMSE and ARE) were computed against different levels of sparsity of the core tensors for the three considered algorithms. For simplicity, the sparsity levels were equal mode-wise sparsity i.e. block sparsity [101]. As defined, the proposed online algorithm consistently outperformed the batch methods. Fig. 5.3 compares the measure difference between the OMDL and TMOD with respect to different block sparsity and SNR. The surface has values below 0 dB, thus verifying the OMDL consistently yielded better performance than the TMOD.

B. Online Joint Learning of Multilinear Dictionary-Projection Matrices

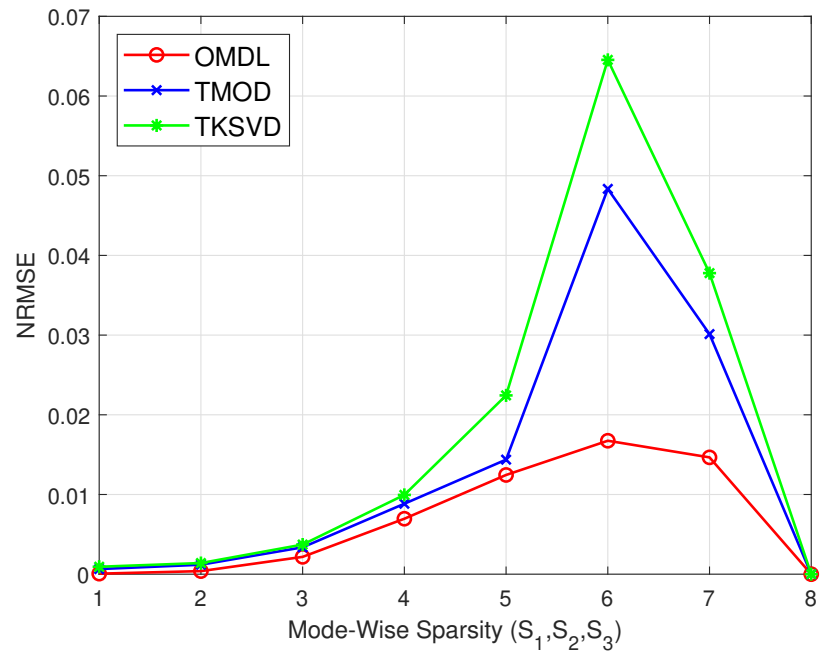
In the second experiment, we compared the performance of the whole joint optimization of the online method was assessed with different rules for $\boldsymbol{\Gamma}$ for the proposed OMDL used. We generated 1000 3-mode tensor data $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{16\times16\times16}$ by a full multilinear product between 1000 sparse core tensors $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{40\times40\times40}$, and mode-$n$ dictionary, $\boldsymbol{\Psi}_n \in \mathbb{R}^{16\times40}, n = 1, 2, 3$. With an average SNR of 20 dB and a forgetting factor $\lambda = 0.95$ for all simulations, we tested for different values of block sparsity (S_n) and projection size (I_n), strictly $S_n \leq I_n$. The measure used in this experiment was the NRMSE of the reconstructed tensor data $\boldsymbol{\mathcal{X}}$. We employed the multipath matching pursuit [108] to recover the sparse core tensor $\boldsymbol{\mathcal{S}}$.

With $\mathbf{\Gamma}$ set as the proposed ETF scheme, identity and $\mathbf{\Psi}^T\mathbf{\Psi}$, respectively, Fig. 5.4a shows that when the data is extremely sparse (S_n is small), there is no much difference in the way how $\mathbf{\Gamma}$ is set. However, as the sparsity level increases, $\mathbf{\Gamma}$ begins to affect the performance of the algorithm, with the ETF scheme starting to beat the others. In Fig. 5.4b where sparsity level was fixed ($S_n = 6, n = 1, 2, 3$), the ETF scheme clearly yielded better results. Observe that as the projection size grows, the more accurate the reconstruction becomes.

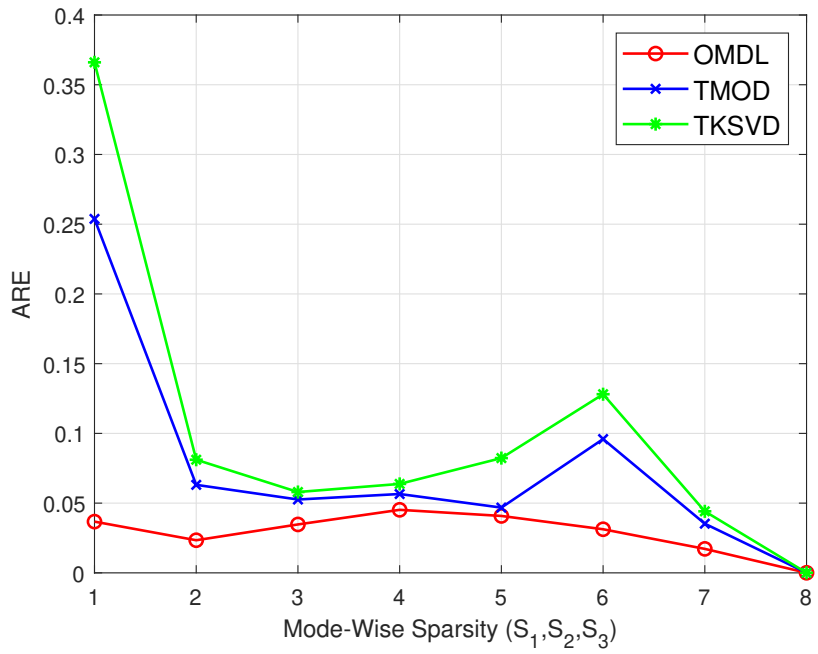
C. Learning Performance on Hyperspectral Images

Finally, we verify the performance of the proposed OMDL algorithm against real-world hyperspectral images sourced from the Indian Pins data of 145×145 pixels and 224 spectral bands, i.e. a tensor of size $145 \times 145 \times 224$, collected by 1992 AVIRIS sensor. We randomly selected 600 patches each of size $24 \times 24 \times 24$ from the Indian Pines data for learning. First, the dictionary learning step was tested alone with the overcomplete sparse core tensor set at $40 \times 40 \times 40$ (overcompleteness level per mode ≈ 1.6). The NRMSE measure was used to compare the performance of MODL vs. TMOD algorithms with varying levels of mode-wise sparsity, ranging from 1 to 20. The second experiment tested joint optimization with MODL fixed as dictionary learning, but comparing the ETF and Identity scheme for the compressed sensing step. Here, the sparse core tensor is the same as previously and the mode-wise sparsity was set at 15. The NRMSE was measured from the reconstructed results from different mode-wise projection size. The depth-first variant of multipath matching pursuit [108] was applied on the vectorized data to recover the original hyperspectral images.

It can be seen from Fig. 5.5a that the MODL performed on par with its offline version. With an addition of peripheral variables like forgetting factor, stepsize etc, we could fine-tune the OMDL to yield marginally better results, but at no significant additional cost in terms of implementation. Fig. 5.5b confirms that the ETF scheme could, albeit slightly, reduce the reconstruction error compared to other compressed sensing strategies.



(a)

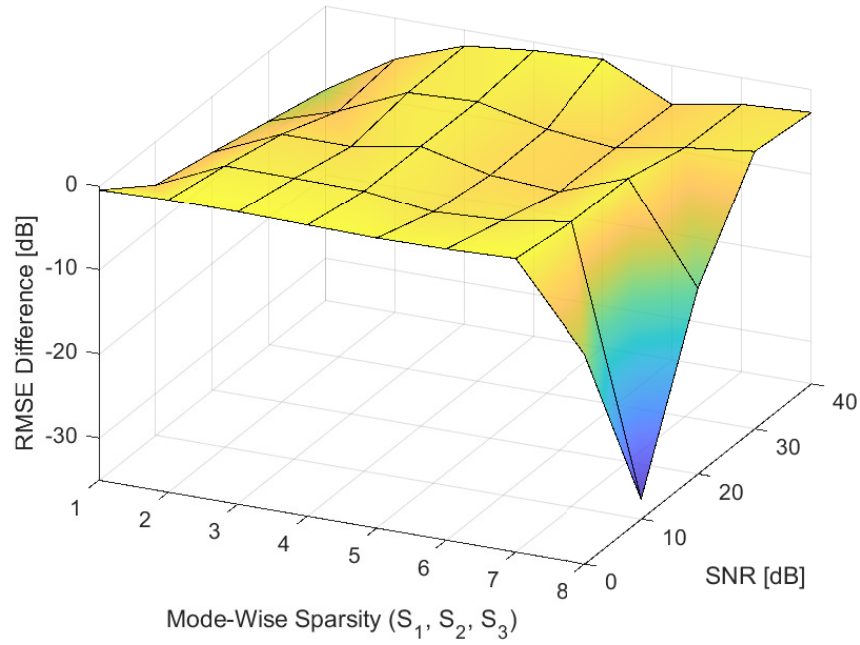


(b)

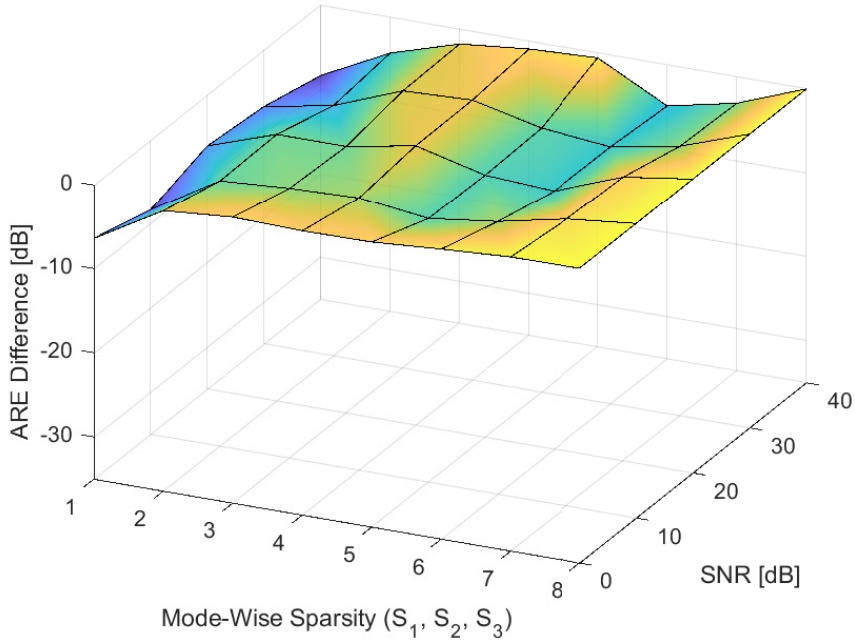
Figure 5.2: [3] Performance measures, (a) NRMSE and (b) ARE, of three considered algorithms with respect to different sparsity levels, for 1000 3-mode tensor data $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{8\times8\times8}$ generated by full multilinear product between 1000 sparse core tensors $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{16\times16\times16}$ and mode-n dictionary $\boldsymbol{\Psi}_n \in \mathbb{R}^{8\times16}, n = 1, 2, 3$, with an average SNR of 20 dB and a forgetting factor $\lambda = 0.95$, averaged over 100 realizations

5.5 Summary

This chapter introduces the extension of the classical dictionary learning methods combined with compressed sensing paradigm to propose the jointly optimized dictionary-projection matrix learning for tensor data to the online learning paradigm. In this way, the batch TMOD method has been modified to operate in a sequential fashion, to obtain the online multilinear dictionary learning (OMDL) algorithm. In addition, we have also proposed modified compressed sensing for a Tucker model (HO-CS), where the target Gram matrix is relaxed from identity to equiangular tight frame (ETF). Although the overall performance improvement is not massive, it has been shown to enable a more computational-friendly method in cases where all the data may not be available altogether or computing all data at once is prohibitive. The advantages have been demonstrated by experiments on both synthetic and real-world data.

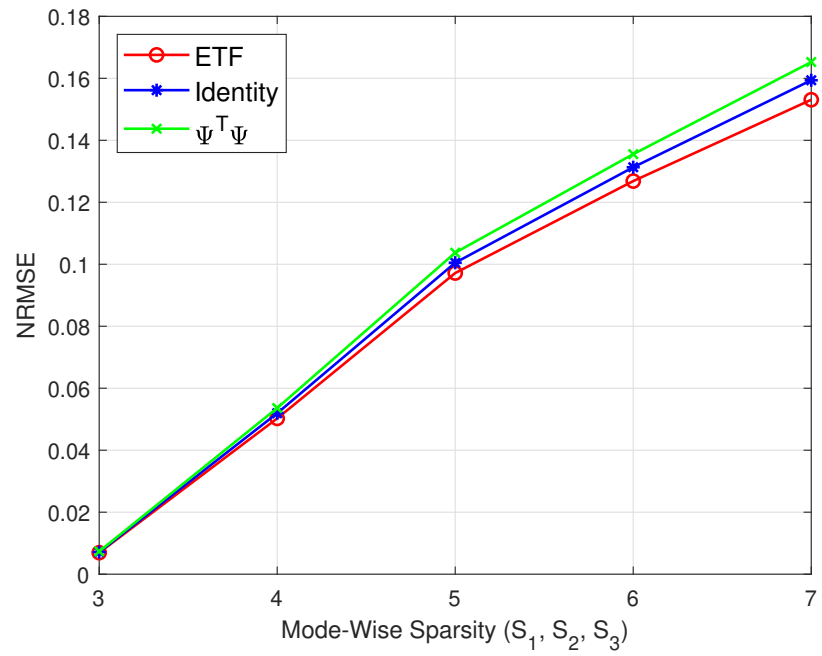


(a)

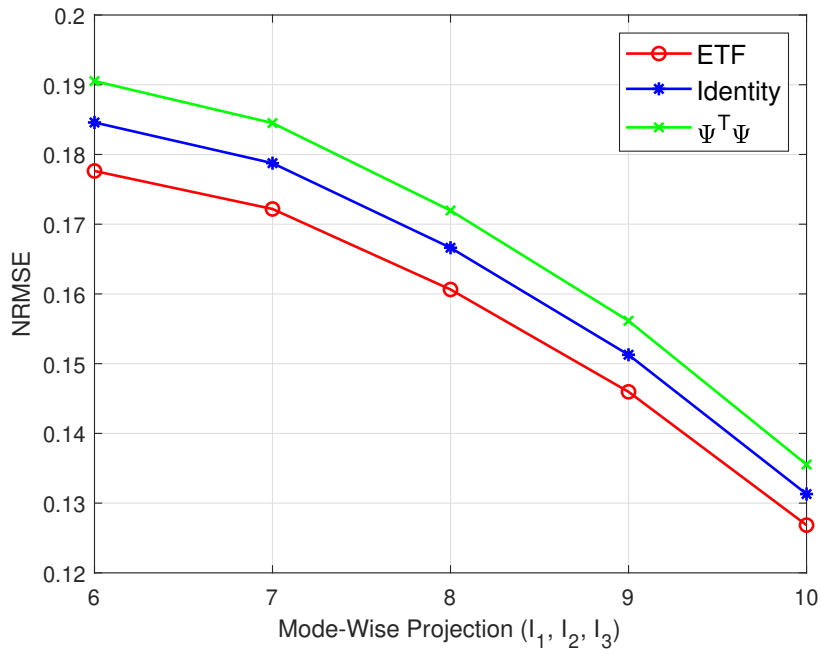


(b)

Figure 5.3: [3] Performance difference, (a) NRMSE and (b) ARE, between OMDL and TMOD with respect to different sparsity and SNR, for 1000 3-mode tensor data $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{8\times8\times8}$ generated by a full multilinear product between 1000 sparse core tensors $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{16\times16\times16}$ and mode-n dictionary $\boldsymbol{\Psi}_n \in \mathbb{R}^{8\times16}, n = 1, 2, 3$, with the forgetting factor $\lambda = 0.95$, averaged over 100 realizations

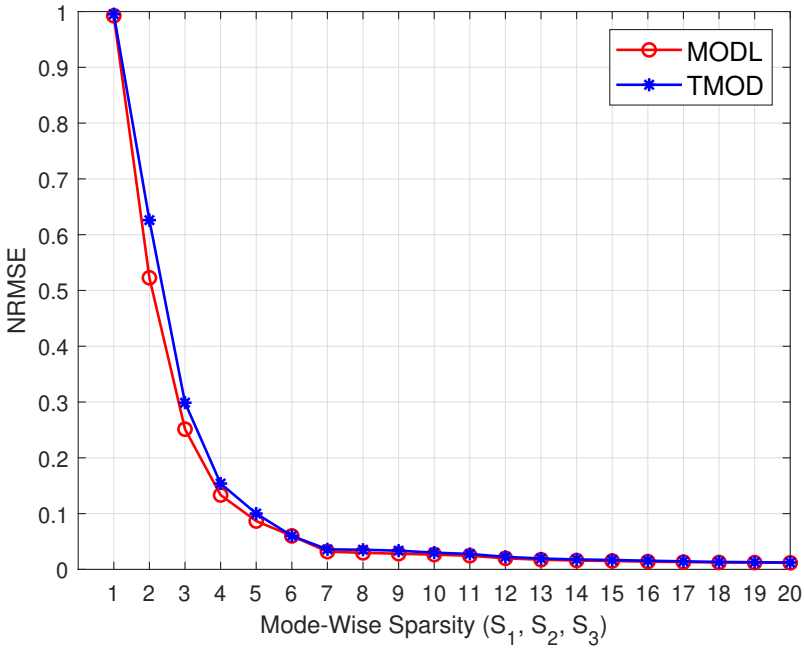


(a)

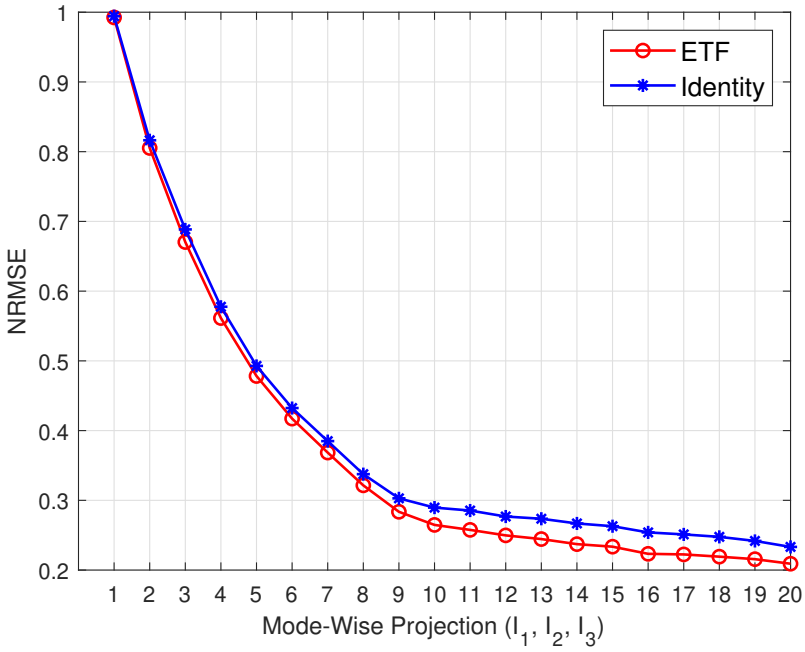


(b)

Figure 5.4: NRMSE Performance measures of different $\bar{\mathbf{\Gamma}}$ with respect to (a) different sparsity for $I_n = 10, \forall n$, and (b) different projection size for $S_n = 6, \forall n$, for 1000 3-mode tensor data $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{16\times16\times16}$ generated by a full multilinear product between 1000 sparse core tensors $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{40\times40\times40}$ and mode-n dictionary $\mathbf{\Psi}_n \in \mathbb{R}^{16\times40}, n = 1, 2, 3$, with an average SNR of 20 dB and a forgetting factor $\lambda = 0.95$, averaged over 100 realizations



(a)



(b)

Figure 5.5: [3] NRMSE Performance for (a) different dictionary learning schemes with varying sparsity and fixed $L_n = 40, \forall n$, and (b) different compressed sensing schemes with varying projection size but fixed $J_n = 15$ and $S_n = 40, \forall n$, for 600 3-mode HSI patches of size $24 \times 24 \times 24$ extracted from the Indian Pines dataset of size $145 \times 145 \times 224$

Chapter 6

Graphs and Signal Processing

GRAPHS are the last unconventional-structure data considered in this thesis. Originally established outside signal processing paradigm, a graph has recently gained in rapid popularity among signal processing community as the same result as quaternions and tensors - the fast growing availability of multisensor and multinode data acquisition devices. Typically, the measurement is obtained from a large-scale sensor array, with possibly sparse and arbitrarily distributed sensors which provide streaming data. These challenges require us to move further the traditional methods to introduce more domain-specific solutions, including graphs. As all three data types are isomorphic as discussed earlier, each type is still unique on its own in its domain-specific representation: 4D data for quaternions, multiway data for tensors, and network data for graphs. This network data is by nature massive, sparse, distributed and ad hoc, and graphs as a data structure directly represent this perfectly. It would be very beneficial to analyze these graph signals and take advantage from the space-time structure of graphs to give physical meaning to sensing locations, sensor importance, and local/global sensor association. One may beg a question of how graphs are relevant to the other two structures apart from isomorphism.

In fact, it is the data sparsity that steers our interest from tensor dictionary from the previous chapter to another problem where, like tensors, data is multidimensional and likely to be sparse [117]; graphs fit these characteristics exactly. Also, we are working on adaptive signal processing. Hence, this smooth combination of graph and signal processing is a budding fruit of our continuing research endeavor.

This chapter aims to provide basics of graphs in relation to the adaptive graph signal processing to be introduced in the next chapter which include graph signals, graph random processes and graph stationarity.

6.1 Graph Signals

Consider a weighted random graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where the vertex set $\mathcal{V} = \{v_1, v_2, ..., v_N\}$, $\mathcal{E}$ is the edge set, and the matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ is the associated shift operator whose entries $w_{ij} \neq 0$ only if $(i, j) \in \mathcal{E}$. The matrix $\mathbf{W}$ captures the local, usually sparse, patterns of $\mathcal{G}$, the examples of which include (weighted) adjacency matrix, graph Laplacian, and their respective generalized forms [117, 118].

A graph signal is then a function which maps the vertex set, $\mathcal{V}$, onto the set of real or complex numbers, e.g. $f : \mathcal{V} \to \mathbb{R}$, and is conveniently represented by a vector $\mathbf{x} = [x_1, ..., x_N]^T \in \mathbb{R}^N$ where x_n denotes the signal value at vertex n. At a particular time instance, the interaction of all elements of a graph signal are modelled according to the graph shift operator (GSO), $\mathbf{W}$, which represents a linear transformation which describes how the graph signals localize across the network.

Similar to the standard shift in time, we can introduce a graph filter which shifts in vertices, $H_L : \mathbb{R}^N \to \mathbb{R}^N$, defined as a polynomial of graph shift operators in the form

$$H_L(\mathbf{W}, \mathbf{h}_k) \triangleq \sum_{l=0}^{L} h_{kl} \mathbf{W}^l \tag{6.1}$$

where $\mathbf{h}_k = [h_{k0}, h_{k1}, ..., h_{kL}]^T$ is a vector of coefficients; the definition of $\mathbf{h}_k$ is given in this way to ease the problem formulation later in the paper. It is noteworthy that the

filter $H_L(\mathbf{W}, \mathbf{h}_k)$ is commutative with respect to the shift operator $\mathbf{W}$, that is

$$H_K(\mathbf{W}, \mathbf{a})H_L(\mathbf{W}, \mathbf{b}) = H_L(\mathbf{W}, \mathbf{b})H_K(\mathbf{W}, \mathbf{a}). \tag{6.2}$$

This property, called the *shift invariance* [117], will be necessary in the estimation of $\mathbf{W}$ as it implies that the structures of graph processes are not entirely arbitrary.

If $\mathbf{W}$ is an adjacency matrix, then its entries $w_{ij} \geq 0$ for $i \neq j$ and $w_{ij} = 0$ for $i = j$. When used as a GSO, a graph Laplacian $\mathbf{L}$ (of $\mathbf{W}$) will have a zero row sum with entries $l_{ij} = -w_{ij}$ for $i \neq j$ and $l_{ij} = \sum_i w_{ij}$ for $i = j$. Other alternative GSOs have also been recently proposed [119], the most suitable choice of which depends on the application at hand. For example, electric circuits are mainly modelled using adjacency matrices while diffusion-on-graph problems naturally employ the Laplacian. Here, to maintain the generality of this study, the only two assumptions made on $\mathbf{W}$ are the shift invariance and sparsity, common features shared by most GSOs in practice [117, 118].

6.2 Vertex-Time ARMA Processes

It is important to note that the shift across vertices does not account for the shift in time which reflects the dynamics of real-world signals. Consider a general time-varying N-dimensional signal, $\mathbf{x}_t$, generated from another time-varying N-dimensional signal, $\mathbf{v}_t$, through a multivariate autoregressive moving average (ARMA) graph process, to give

$$\mathbf{x}_t = \sum_{p=1}^{P} \mathbf{\Psi}_p \mathbf{x}_{t-p} + \sum_{q=0}^{Q} \mathbf{\Phi}_q \mathbf{v}_{t-q} \tag{6.3}$$

where $\mathbf{\Psi}_p$ and $\mathbf{\Phi}_q$ are coefficient matrices of $\mathbf{x}_t$, so that these matrices are not fixed; *please note that the symbols* $\mathbf{\Psi}$ *and* $\mathbf{\Phi}$ *here have nothing to do with those in the preceding OMDL chapters.* For a graph signal, the coefficients (matrix elements) explain how each dimension interacts with all others, and will naturally assume a form of graph shift operators. An intuitive approach would be for the coefficients to assume a form of a graph filter, although there are other interesting basis functions to consider as an alternative, e.g. radial basis

functions. Much existing literature [120–125] employs polynomial graph filter, also adopted here, whereby $\mathbf{\Psi}_p$ and $\mathbf{\Phi}_q$ in (6.3) can be expressed as

$$\mathbf{\Psi}_p \triangleq H_{L_p}(\mathbf{W}, \mathbf{h}_p) \tag{6.4}$$

$$\mathbf{\Phi}_q \triangleq H_{K_q}(\mathbf{W}, \mathbf{h}_q) \tag{6.5}$$

where L_p and K_q are positive integers denoting the maximal shifts of the specific graph filters. With eqs. (6.4) and (6.5), $\mathbf{x}_t$ and $\mathbf{v}_t$ become graph signals, of which the elements relate to their respective vertex. The values of L_p and K_q are arbitrary and have to be determined for every problem at hand [124]. In this work, we narrow down the scope of the problem by restricting the random graph process to be purely autoregressive and causal [125], thus reducing (6.3) to

$$\mathbf{x}_t = \sum_{p=1}^{P} \mathbf{\Psi}_p \mathbf{x}_{t-p} + \mathbf{v}_t \tag{6.6}$$

where $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and

$$\mathbf{\Psi}_p \triangleq H_p(\mathbf{W}, \mathbf{h}_p). \tag{6.7}$$

Remark 2. The causality assumption in (6.6) implies that $L_p = p$ and this interpretation is interesting in that the the maximum vertex shifts at a particular time lag cannot exceed the time lag itself. This signifies that *a shift in vertices occurs in tandem with a shift in time i.e. no more than one shift operator is allowed per time instance.* This assumption is rather reasonable as we are analyzing discrete-time models where the sampling policy can be adjusted accordingly.

In practice, there may exist a more complicated system where shifts in vertices happen asynchronously with time shift, however small the sampling rate; this is beyond the scope of our work as we believe this scenario is rare. We therefore focus on the vertex-time AR model given in (6.6) and (6.7).

6.3 Weak Stationarity

We shall now briefly explain how the shift invariance of the graph filter can be interpreted as a form of 'stationarity'. Analogous to the autocorrelation in time series, the autocorre-

lation of a graph signal should depend on the 'distance' of vertex shifts regardless of the position in vertex domain where the signal initially resides, i.e. however many times the signal has been shifted. Therefore, as long as the total number of shifts is the same, then so should be the correlation. This property is called 'weak stationarity' as in Defnition 1 in [118]. While the concept of stationarity in graphs is still an open research topic, we employ the notion in [118] as it suits our problem setting since the AR model in (6.6) is inherently weakly stationary, as it is made up of a sum of shift-invariant graph filters.

6.4 Summary

This chapter provided necessary basic concepts of signal processing on graphs. Firstly, the notion of random graph signals based on polynomial graph filters is given. Then, the vertex-time (space-time) autoregressive processes are defined with the assumption of causality and polynomial graph filtering. Lastly, graph sttionarity is briefly mentioned to provide relationship between graph shift invariance and weak stationarity of graphs. In the next chapter, the last piece of our work will be derived and properly proposed - the first *true* adaptive graph signal processing.

Chapter 7

Adaptive Graph Signal Processing

THE proposed adaptive graph signal processing considers the phenomena which can be described by multivariate statistical models parameterized by the irregular-structure data represented as a graph, whereby the underlying statistical model follows graph-topological structure. The existing work in signal processing on graphs includes the tracking the time-varying graph signals [120, 121] and the use of space transforms and dimensionality reduction to reduce the problem complexity [122, 123]. These results assume an autoregressive model for graph data, and have recently been generalized to autoregressive moving average models [124].

It is important to note that the common assumption made in much of the existing work is that the graph topology is known *a priori*. However, in many network-related problems, like social media, financial assets, or neuron connectivity, the topology (relationship between nodes, assets or neurons) needs to be learned, not to mention that the topology is also often time-varying. To discover the topology of the GSO which generated the observed graph data, the work in [125] assumes a vertex-time autoregressive casual process; however, like all above mentioned articles, this offers a batch method where all

the data are considered at once. To the best of our knowledge, a *truly* adaptive approach to this problem is still lacking.

To this end, we propose an *online* adaptive filtering algorithm for streaming graph data. Similarly to [125], we design the algorithm in a system identification setting, whereby the task boils down to recovering the structure of the underlying GSO. The proposed approach employs modified stochastic gradient descent methods [1,2], in addition to graph-specific structures such as sparsity or commutativity, which are enforced naturally by graph topology. As the existing work has already shown the potentials and limits of the graph topology identification problem, this chapter aims to further explore the possibility of applying the techniques of adaptive signal processing to random graph processes.

7.1 Regularized Least Squares Estimation

The problem in eqs. (6.6) and (6.7) pertains to the class of multivariate linear regression problems, for which the optimal linear estimator is the MSE estimator [126]. Here, we adopt the least squares method - a deterministic counterpart of the MSE estimator [1,2,5]. The least squares problem of eqs. (6.6) and (6.7) is then given by

$$\min_{\mathbf{W},\mathbf{h}} \frac{1}{2}\sum_{\tau=1}^{t}\lambda^{t-\tau}\left\|\mathbf{x}_\tau - \sum_{p=1}^{P} H_p(\mathbf{W},\mathbf{h}_p)\mathbf{x}_{\tau-p}\right\|_2^2 \tag{7.1}$$

where $\mathbf{h} = [\mathbf{h}_1^T, ..., \mathbf{h}_P^T]^T \in \mathbb{C}^M$ with $M = P(P+3)/2$ (See eq. (7.7)) and P is the order of this AR random graph process, $\mathbf{x}_i = \mathbf{0}$ for $i \leq 0$ and $\lambda \in (0, 1]$. Observe that (7.1) represents a non-convex polynomial problem with many minima, for which many solutions have been proposed [117,120–123], none of which guarantees a global optimum, even under some quite restrictive assumptions [125]. A more plausible metric would be therefore to identify whether an edge between any pair of vertices exist with the least chance of misses and false alarms [122], and the order P should be as small as possible. The above setting with term $H_p(\mathbf{W}, \mathbf{h}_p)$, $\forall p$ defined in (6.1) implies that $\mathbf{W}$ and $\mathbf{h}$ should be sparse, so that rather than solving the polynomial problem, we can cast (7.1) into alternating steps of regularized least squares sub-problems, outlined in the following sections.

7.1.1 Solving for $\mathbf{\Psi}_p = H_p(\mathbf{W}, \mathbf{h}_p)$

The minimization problem in eq. (7.1) is now solved with respect to $\mathbf{\Psi}_p = H_p(\mathbf{W}, \mathbf{h}_p)$, instead of $\mathbf{W}$ and $\mathbf{h}$; this makes the problem quadratic in $\mathbf{\Psi}_p$ and hence standard stochastic gradient descent is applicable. Denote by $\hat{\mathbf{\Psi}}_p$ an estimate of $\mathbf{\Psi}_p$. With the assumption of sparsity, we now arrive at the optimization problem,

$$\min_{\mathbf{\Psi}} \ \frac{1}{2}\sum_{\tau=1}^{t} \lambda^{t-\tau} \left\| \mathbf{x}_\tau - \sum_{p=1}^{P} \mathbf{\Psi}_p \mathbf{x}_{\tau-p} \right\|_2^2 + \sum_{p=1}^{P} \mu_p \|\mathrm{vec}(\mathbf{\Psi}_p)\|_1 \tag{7.2}$$

where $\mathbf{\Psi} = [\mathbf{\Psi}_1, ..., \mathbf{\Psi}_P] \in \mathbb{R}^{N \times NP}$, $\mathrm{vec}(\cdot)$ is a vectorization operator and $\|\cdot\|_1$ is an ℓ_1 norm, while μ_p is a constant which adjusts the degree of sparsity of the corresponding $\mathbf{\Psi}_p$. From (6.7), it is obvious that $\mathbf{\Psi}_p$ grows less sparse with an increase in p, and thus μ_p should be set in a decreasing fashion.

The above equation does not account for the shift invariance property of $\mathbf{\Psi}_p$. However, from (6.2), we can add another regularizing term to enforce this constraint, using the following commutator [125]

$$[\mathbf{\Psi}_i, \mathbf{\Psi}_j] \triangleq \mathbf{\Psi}_i \mathbf{\Psi}_j - \mathbf{\Psi}_j, \mathbf{\Psi}_i. \tag{7.3}$$

Inserting (7.3) into (7.2) yields

$$\min_{\mathbf{\Psi}} \ \frac{1}{2}\sum_{\tau=1}^{t} \lambda^{t-\tau} \left\| \mathbf{x}_\tau - \sum_{p=1}^{P} \mathbf{\Psi}_p \mathbf{x}_{\tau-p} \right\|_2^2 + \sum_{p=1}^{P} \mu_p \|\mathrm{vec}(\mathbf{\Psi}_p)\|_1 + \gamma \sum_{i \neq j} \| [\mathbf{\Psi}_i, \mathbf{\Psi}_j] \|_F^2. \tag{7.4}$$

The final term makes the above problem of a quartic programming type, rendering the convergence analysis more difficult. This becomes evident in the simulations where the addition of this regularizer did not improve the algorithm performance significantly, insteadeven slightly deteriorate it in some cases.

7.1.2 Estimating W from $\hat{\mathbf{\Psi}}_1$

From (6.1) and (6.7), observe that $\mathbf{\Psi}_1$ is a linear function of $\mathbf{W}$ and thus its estimate, $\hat{\mathbf{\Psi}}_1$, could represent a good estimate of $\mathbf{W}$, that is, $\hat{\mathbf{W}}$. To find a true $\mathbf{W}$ after obtaining $\hat{\mathbf{\Psi}}$

from (7.2), another regularized least squares sub-problem is needed, and given by

$$\min_{\mathbf{W}} \ \frac{1}{2}\|\mathbf{\Psi}_1 - \mathbf{W}\|_2^2 + \mu_1\|\text{vec}(\mathbf{W})\|_1 + \gamma \sum_{p=2}^{P} \| \left[\mathbf{W}, \mathbf{\Psi}_p\right] \|_F^2. \tag{7.5}$$

The rightmost term ensures that $\hat{\mathbf{W}}$ is as commutative as possible with all $\hat{\mathbf{\Psi}}_p$ to ensure the shift invariance property. Note that when (7.4) is employed to calculate $\hat{\mathbf{\Psi}}_1$, the shift invariance property has already been enforced so that this optimization sub-problem might be bypassed by setting $\hat{\mathbf{W}} = \hat{\mathbf{\Psi}}_1$. The implementation strategy is further elucidated in the simulation section.

7.1.3 Estimating h

After obtaining $\hat{\mathbf{W}}$, the original problem (7.1) turns into a quadratic programming one with respect to $\mathbf{h}$. Also, by assuming that $\mathbf{h}$ is sparse, we can rearrange (6.7) and (7.1) into

$$\min_{\mathbf{h}} \ \frac{1}{2}\sum_{\tau=1}^{t} \lambda^{t-\tau}\|\mathbf{x}_\tau - \mathbf{Y}_\tau \mathbf{h}\|_2^2 + \zeta\|\mathbf{h}\|_1 \tag{7.6}$$

with

$$\mathbf{Y}_t = \left[\mathbf{x}_{t-1}, \hat{\mathbf{W}}\mathbf{x}_{t-1}, ..., \mathbf{x}_{t-P}, ..., \hat{\mathbf{W}}^P\mathbf{x}_{t-P}\right] \tag{7.7}$$

Note that $\mathbf{Y}_t \in \mathbb{R}^{N\times M}$ contains all possible combination of past P vertex-time instances of the graph signal, $\mathbf{x}_t$. While $M = P(P+3)/2$ appears rather large, in practice, the actual order is quite low, with even $M < N$ a likely case. In addition, this step is optional as our main goal is to recover $\mathbf{W}$.

7.2 Adaptive Graph Signal Processing

We now proceed to build upon the alternating optimization problem detailed in the previous section, to introduce a class of adaptive algorithms based on the optimization criterion in (7.1), with a sparse solution, called *sparsity-aware adaptive algorithm* [127]. Although many methods, such as ℓ_1-regularized least mean square [128] or oracle algorithm [127], lead to convergence with competitively small MSE, these solutions are rarely sparse, if not

at all [129], as they do not explicitly zero out the elements of the GSO matrix like their offline counterparts such as basis pursuit. On the other hand, methods like ADMM or ALM have proved valuable in solving offline ℓ_1-regularized problems, and in our endeavor, as the topic is still in its infancy, we adopt the standard stochastic gradient descent but rewrite our main cost function to naturally enforce the solution to 'project' on acceptable values [130] (the prospect of online ADMM and ALM is promising if the underlying algorithm - as in this thesis - is designed to work well). To this end, we re-formulate (7.2) and (7.4) to (7.6) by splitting the desired variables ($\hat{\mathbf{\Psi}}$, $\hat{\mathbf{W}}$ and $\hat{\mathbf{h}}$) into their positive and negative parts, that is

$$\hat{\mathbf{\Psi}} \triangleq \hat{\mathbf{\Psi}}_+ - \hat{\mathbf{\Psi}}_-, \;\; \hat{\mathbf{W}} \triangleq \hat{\mathbf{W}}_+ - \hat{\mathbf{W}}_-, \;\; \hat{\mathbf{h}} \triangleq \hat{\mathbf{h}}_+ - \hat{\mathbf{h}}_- \tag{7.8}$$

where $(\cdot)_+ \geq \mathbf{0}$ and $(\cdot)_- \geq \mathbf{0}$ contain respectively only the positive and negative parts of $(\cdot)$. Note that if $\mathbf{W}$ is an adjacency matrix, then $\hat{\mathbf{\Psi}}_- = \hat{\mathbf{W}}_- = \mathbf{0}$ which makes the problem easier. For the Laplacian, this is much more difficult because while clearly it can be split into the positive on- and negative off-diagonals, the real bottleneck is the zero row sum, an equality constraint which is awkward to solve iteratively as it could involve Lagrangian methods. Since the stucture of GSO vary with applications, we here study the general unconstrained $\mathbf{W}$ for generality and analytic insights.

7.2.1 Form of the Algorithm

Based on (7.8), the $\|\cdot\|_1$ operator can be expressed through a product-weighted sum i.e.

$$\begin{aligned}
\hat{\mathbf{\Psi}} &= \mathrm{Tr}\left(\mathbf{1}_{N\times N}\hat{\mathbf{\Psi}}_+\right) + \mathrm{Tr}\left(\mathbf{1}_{N\times N}\hat{\mathbf{\Psi}}_-\right), \\
\hat{\mathbf{W}} &= \mathrm{Tr}\left(\mathbf{1}_{N\times N}\hat{\mathbf{W}}_+\right) + \mathrm{Tr}\left(\mathbf{1}_{N\times N}\hat{\mathbf{W}}_-\right), \\
\hat{\mathbf{h}} &= \mathbf{1}_N^T\hat{\mathbf{h}}_+ - \mathbf{1}_N^T\hat{\mathbf{h}}_-
\end{aligned} \tag{7.9}$$

where $\mathrm{Tr}(\cdot)$ is a trace operator, and $\mathbf{1}_{N\times N} \in \mathbb{R}^{N\times N}$ and $\mathbf{1}_N \in \mathbb{R}^N$ both have all their elements equal to 1. These formulae enable us to separate the derivatives with respect to the positive and negative parts, where gradient projection can be used to force invalid

values to zero, leading to sparsity as a by-product. We now proceed to minimize (6.4) by calculating the gradient with respect to $(\hat{\mathbf{\Psi}}_p)_+$, denoted by $\nabla^{(t)}_{(\hat{\mathbf{\Psi}}_p)_+}$, and given by

$$\nabla^{(t)}_{(\hat{\mathbf{\Psi}}_p)_+} = \sum_{\tau=1}^{t} \lambda^{t-\tau} \left(\sum_{k=1}^{P} \hat{\mathbf{\Psi}}_{k,t-1} \mathbf{x}_{\tau-k} \mathbf{x}_{\tau-p}^T - \mathbf{x}_\tau \mathbf{x}_{\tau-p}^T \right) + \mu_{p,t} \mathbf{1}_{N\times N} + \gamma \mathbf{Q}_{p,t} \tag{7.10}$$

where

$$\mathbf{Q}_{p,t+1} = \sum_{k=2}^{P} \left(\left[\hat{\mathbf{\Psi}}_{p,t}, \hat{\mathbf{\Psi}}_{k,t} \right] \hat{\mathbf{\Psi}}_{k,t}^T - \hat{\mathbf{\Psi}}_{k,t}^T \left[\hat{\mathbf{\Psi}}_{p,t}, \hat{\mathbf{\Psi}}_{k,t} \right] \right). \tag{7.11}$$

Note that $\hat{\mathbf{\Psi}}_{p,t}$ denotes $\hat{\mathbf{\Psi}}_p$ at the time instant t in the algorithm. Now, let $\hat{\mathbf{\Psi}}_t$, $\mathbf{M}_t$, $\mathbf{Q}_t \in \mathbb{R}^{N\times NP}$ be respectively defined as

$$\hat{\mathbf{\Psi}}_t \triangleq \left[\hat{\mathbf{\Psi}}_{1,t}, \hat{\mathbf{\Psi}}_{2,t}, ..., \hat{\mathbf{\Psi}}_{P,t} \right] := \hat{\mathbf{\Psi}}_{+_t} - \hat{\mathbf{\Psi}}_{-_t}, \tag{7.12}$$

$$\mathbf{M}_t \triangleq \left[\mu_{1,t} \mathbf{1}_{N\times N}, \mu_{2,t} \mathbf{1}_{N\times N}, ..., \mu_{P,t} \mathbf{1}_{N\times N} \right], \tag{7.13}$$

$$\mathbf{Q}_t \triangleq \left[\mathbf{Q}_{1,t}, \mathbf{Q}_{2,t}, ..., \mathbf{Q}_{P,t} \right], \tag{7.14}$$

and with the following variables,

$$\mathbf{G}_t \triangleq \hat{\mathbf{\Psi}}_{t-1} \mathbf{R}_t - (\mathbf{P}_t - \gamma \mathbf{Q}_t), \tag{7.15}$$

$$\mathbf{R}_t \triangleq \sum_{\tau=1}^{t} \lambda^{t-\tau} \mathbf{x}_{P,\tau} \mathbf{x}_{P,\tau}^T = \lambda \mathbf{R}_{t-1} + \mathbf{x}_{P,t} \mathbf{x}_{P,t}^T, \tag{7.16}$$

$$\mathbf{P}_t \triangleq \sum_{\tau=1}^{t} \lambda^{t-\tau} \mathbf{x}_\tau \mathbf{x}_{P,\tau}^T = \lambda \mathbf{P}_{t-1} + \mathbf{x}_t \mathbf{x}_{P,t}^T, \tag{7.17}$$

where $\mathbf{x}_{P,t} \in \mathbb{R}^{NP}$ is given by

$$\mathbf{x}_{P,t} \triangleq \left[\mathbf{x}_{t-1}^T, \mathbf{x}_{t-2}^T, ..., \mathbf{x}_{t-P}^T \right]^T. \tag{7.18}$$

We can now express the update for $\hat{\mathbf{\Psi}}_{+_t}$ as a gradient projection, that is

$$\hat{\mathbf{\Psi}}_{+_t} = \left(\hat{\mathbf{\Psi}}_{+_{t-1}} - (\mathbf{M}_t + \mathbf{G}_t)(\mathbf{A}_t \otimes \mathbf{I}_{N\times N}) \right)_+ \tag{7.19}$$

where $\mathbf{A}_t = \text{diag}(\alpha_1, \alpha_2, ..., \alpha_P) \in \mathbb{R}^{P\times P}$ is a diagonal matrix of stepsizes. Similarly, we can obtain the update equation for $\hat{\mathbf{\Psi}}_{-_t}$ as

$$\hat{\mathbf{\Psi}}_{-_t} = \left(\hat{\mathbf{\Psi}}_{-_{t-1}} - (\mathbf{M}_t - \mathbf{G}_t)(\mathbf{A}_t \otimes \mathbf{I}_{N\times N})\right)_+. \tag{7.20}$$

The next step involves finding the shift operator $\hat{\mathbf{W}}$. For example, we can easily let $\hat{\mathbf{W}}_t = \hat{\mathbf{\Psi}}_{1,t}$, with the derivation so far based on (7.4) where the commutative property of $\hat{\mathbf{\Psi}}$ is already taken into account. Another approach may employ a simplified version of (7.2) with $\mathbf{Q}_t = \mathbf{0}$ for all t. Since the commutativity is not enforced in eq. (7.2), but is needed when estimating $\hat{\mathbf{W}}_t$, we repeat the same procedure as in (7.5), resulting in the following,

$$\hat{\mathbf{W}}_t = \hat{\mathbf{W}}_{+_t} - \hat{\mathbf{W}}_{-_t}, \tag{7.21}$$

$$\hat{\mathbf{W}}_{+_t} = \left(\hat{\mathbf{W}}_{+_{t-1}} - \beta_t(\mu_{1,t}\mathbf{1}_{N\times N} + \mathbf{V}_t)\right)_+ \tag{7.22}$$

$$\hat{\mathbf{W}}_{-_t} = \left(\hat{\mathbf{W}}_{-_{t-1}} - \beta_t(\mu_{1,t}\mathbf{1}_{N\times N} - \mathbf{V}_t)\right)_+ \tag{7.23}$$

with

$$\mathbf{V}_t = \hat{\mathbf{W}}_{t-1} - (\hat{\mathbf{\Psi}}_{1,t} - \gamma\mathbf{S}_t) \tag{7.24}$$

and

$$\mathbf{S}_t = \sum_{k=2}^{P} \left(\left[\hat{\mathbf{W}}_{t-1}, \hat{\mathbf{\Psi}}_{k,t}\right] \hat{\mathbf{\Psi}}_{k,t}^T - \hat{\mathbf{\Psi}}_{k,t}^T \left[\hat{\mathbf{W}}_{t-1}, \hat{\mathbf{\Psi}}_{k,t}\right]\right). \tag{7.25}$$

Where $\hat{\mathbf{W}}$ is an adjacency matrix, $\hat{\mathbf{\Psi}}_{-_t}$ and $\hat{\mathbf{W}}_{-_t}$ are both set to zero for all t.

Since the objective functions in (7.2) and (7.4) are not pure MSE with regularizing terms, this makes them multi-convex and the solution will thus be biased [125] and not optimal in terms of MSE. The whole procedure to this point has been to identify the *causative* elements of an GSO without necessarily their correct values. To this end, we employ an approach known as *debiasing*, where in order to eliminate the regularization biases, we fix the zero entries of the obtained GSO $\hat{\mathbf{W}}_t$, and only optimize the non-zero entries via a least squares cost. It comes with a caveat that, by reducing data sample size, the noise could be distorted from normality, thus affecting the minimal MSE criterion from the outset [131] if the original data is rather noisy, or of insufficiently large size.

The final step, the calculation of $\hat{\mathbf{h}}$, is discretionary as our prime purpose is to identify $\hat{\mathbf{W}}$ and as stated above, the components of $\hat{\mathbf{W}}$ may not be accurately computed due to the biased objectives, leading to even erroneous $\hat{\mathbf{h}}$. On the other hand, if desiring to fully identify the temporal structure of the vertex-time AR process, we can solve for $\hat{\mathbf{h}}$ via (7.6) and (7.7), but $\hat{\mathbf{W}}$ needs to be *debiased* in order for $\hat{\mathbf{h}}$ to be mathematically meaningful. Unlike the two earlier optimization sub-problems, $\hat{\mathbf{h}}$ is not strictly conditioned, and its sparsity constraint aims mainly to render the model succinct. Hence, GAR-LMS [129] is employed to arrive at the update equation of $\hat{\mathbf{h}}$, given by

$$\hat{\mathbf{h}}_t = \hat{\mathbf{h}}_{t-1} + \rho_t \left(\mathbf{C}_t \hat{\mathbf{h}}_{t-1} - \mathbf{u}_t + \eta_t \mathbf{b}_t \right) \tag{7.26}$$

where

$$\mathbf{C}_t = \lambda \mathbf{C}_{t-1} + \mathbf{Y}_t^T \mathbf{Y}_t \tag{7.27}$$

$$\mathbf{u}_t = \lambda \mathbf{u}_{t-1} + \mathbf{Y}_t^T \mathbf{x}_t \tag{7.28}$$

$$\mathbf{b}_t : \; b_{i,t} = \frac{\text{sign}(\hat{h}_{i,t-1})}{\epsilon + \hat{h}_{i,t-1}} \tag{7.29}$$

and

$$\mathbf{Y}_t = \left[\mathbf{x}_{t-1}, \hat{\mathbf{W}}_t \mathbf{x}_{t-1}, ..., \mathbf{x}_{t-P}, ..., \hat{\mathbf{W}}_t^P \mathbf{x}_{t-P} \right], \tag{7.30}$$

with ϵ a small positive number. This step could be further simplified by only taking the instantaneous samples into (7.26), that is, $\lambda = 0$, to yield

$$\hat{\mathbf{h}}_t = \hat{\mathbf{h}}_{t-1} + \rho_t \left(\mathbf{Y}_t^T \mathbf{e}_t + \eta_t \mathbf{b}_t \right) \tag{7.31}$$

where

$$\mathbf{e}_t = \mathbf{x}_t - \mathbf{Y}_t \hat{\mathbf{h}}_{t-1} \tag{7.32}$$

The so derived algorithms are summarized in Algorithms 1 & 2.

As mentioned earlier, two paths are possible for Algorithm 1; either to ignore Step 9, i.e. $\mathbf{Q}_t = \mathbf{0}$, and consider only Step 18 (we will call this **Path 1**), or vice versa - to consider Step 9 and ignore Step 18 by letting $\hat{\mathbf{W}}_t = \hat{\mathbf{\Psi}}_{1,t}$ (**Path 2**).

Algorithm 5: Identifying the topology of $\hat{\mathbf{W}}$ ($\hat{\mathbf{W}}^*$)

Input : $\mathbf{x}$, P
Output: $\hat{\boldsymbol{\Psi}}$, $\hat{\mathbf{W}}^*$

Initialize $\hat{\boldsymbol{\Psi}}_0 = \hat{\boldsymbol{\Psi}}_{+_0} = \hat{\boldsymbol{\Psi}}_{-_0} = \mathbf{P}_0 = \mathbf{Q}_1 = \mathbf{0}$, $\hat{\mathbf{W}}_0 = \hat{\mathbf{W}}_{+_0} = \hat{\mathbf{W}}_{-_0} = \mathbf{S}_1 = \mathbf{0}$ and $\mathbf{R}_0 = \mathbf{0}$;
$t = 0$;
do
  $t = t + 1$;
  Solving for $\hat{\boldsymbol{\Psi}}_t$;
  $\mathbf{x}_{P,t} = \left[\mathbf{x}_{t-1}^T, \mathbf{x}_{t-2}^T, ..., \mathbf{x}_{t-P}^T\right]^T$;
  $\mathbf{R}_t = \lambda \mathbf{R}_{t-1} + \mathbf{x}_{P,t}\mathbf{x}_{P,t}^T$;
  $\mathbf{P}_t = \lambda \mathbf{P}_{t-1} + \mathbf{x}_t\mathbf{x}_{P,t}^T$;
  $\mathbf{Q}_t = [\mathbf{Q}_{1,t}, \mathbf{Q}_{2,t}, ..., \mathbf{Q}_{P,t}]$ with $\mathbf{Q}_{p,t}$ according to eq. (7.11);
  calculate $\boldsymbol{\mu}_t$;
  $\mathbf{M}_t = [\mu_{1,t}\mathbf{1}_{N\times N}, \mu_{2,t}\mathbf{1}_{N\times N}, ..., \mu_{P,t}\mathbf{1}_{N\times N}]$;
  $\mathbf{G}_t = \hat{\boldsymbol{\Psi}}_{t-1}\mathbf{R}_t - (\mathbf{P}_t - \gamma\mathbf{Q}_t)$;
  calculate $\mathbf{A}_t$;
  $\hat{\boldsymbol{\Psi}}_{+_t} = \left(\hat{\boldsymbol{\Psi}}_{+_{t-1}} - (\mathbf{M}_t + \mathbf{G}_t)(\mathbf{A}_t \otimes \mathbf{I}_{N\times N})\right)_+$;
  $\hat{\boldsymbol{\Psi}}_{-_t} = \left(\hat{\boldsymbol{\Psi}}_{-_{t-1}} - (\mathbf{M}_t - \mathbf{G}_t)(\mathbf{A}_t \otimes \mathbf{I}_{N\times N})\right)_+$;
  $\hat{\boldsymbol{\Psi}}_t = \hat{\boldsymbol{\Psi}}_{+_t} - \hat{\boldsymbol{\Psi}}_{-_t}$;
  Estimating $\hat{\mathbf{W}}_t$;
  $\mathbf{S}_t$ according to eq. (7.25);
  $\mathbf{V}_t = \hat{\mathbf{W}}_{t-1} - (\hat{\boldsymbol{\Psi}}_{1,t} - \gamma\mathbf{S}_t)$;
  $\hat{\mathbf{W}}_{+_t} = \left(\hat{\mathbf{W}}_{+_{t-1}} - \beta_t(\mu_{1,t}\mathbf{1}_{N\times N} + \mathbf{V}_t)\right)_+$;
  $\hat{\mathbf{W}}_{-_t} = \left(\hat{\mathbf{W}}_{-_{t-1}} - \beta_t(\mu_{1,t}\mathbf{1}_{N\times N} - \mathbf{V}_t)\right)_+$;
  $\hat{\mathbf{W}}_t = \hat{\mathbf{W}}_{+_t} - \hat{\mathbf{W}}_{-_t}$;
while $t < T^*$ (an epoch with steady state reached);
$\hat{\mathbf{W}}^* = \hat{\mathbf{W}}_{T^*}$.

7.2.2 Fine-tuning Peripheral Parameters

For desirable accuracy and *fidelity* of the outcome, there are still some minor parameters which need to be fine tuned. These include the regularization constants $\boldsymbol{\mu}_t := [\mu_{1,t}, \mu_{2,t}, ..., \mu_{P,t}]^T$, η_t, γ and ϵ; stepsizes $\mathbf{A}_t$, β_t and ρ_t; and the forgetting factor λ.

For the ℓ_1-norm related constants, these can be initially expressed via [132]

$$\mu_{p,t} = \mu_p \|\mathbf{P}_{p,t} - \gamma \mathbf{Q}_{p,t}\|_\infty \tag{7.33}$$

$$\eta_t = \eta \left\|\mathbf{Y}_t^T \mathbf{x}_t\right\|_\infty \tag{7.34}$$

where $\boldsymbol{\mu} := [\mu_1, ..., \mu_P]^T$ is a constant vector with entry values decreasing with p, and $\mathbf{P}_{p,t} \in \mathbb{R}^{N \times N}$ is a pth block of $\mathbf{P}_t := [\mathbf{P}_{1,t}, \mathbf{P}_{2,t}, ..., \mathbf{P}_{P,t}]$. For the stepsizes, Armijo backtracking is employed to yield suitable values of $\mathbf{A}_t$, β_t and ρ_t, while the parameters $\boldsymbol{\mu}$, η, γ and λ have to be determined manually. Notice that while some prior knowledge is available for $\boldsymbol{\mu}$ (decreasing-valued entries) and λ (closed to unity), η and γ are rather unconstrained.

7.2.3 Discussion on Convergence

A rigorous convergence analysis of the graph random processes can be found in [125]. However, the assumptions for successful convergence are quite restrictive because the graph signal has not only to obey specific sparsity structure (Assumption **A5** in [125]), but also to exhibit a very strong stability condition (Assumptions **A4** and **A6** in [125]), to which only a few classes of topologies conform, like K-regular graphs. While the proof in [125] is without doubt rigorous, it is largely theoretical and limited to real-world cases. Attempts to relax the assumptions underpinning the proof have had limited success; this is partly due to that fact that the base problem (7.2) is inherently biased; for example, the ℓ_1-norm regularizing terms usually exhibit a side effect of underestimating the non-zero elements [132], not to mention a more complex commutator term. Therefore, it may be more favorable to take a *different* convergence measure. In other words, rather than the mean squared error, we could use the percentage of correctly recovered elements of $\mathbf{W}$, regardless of their correct values, a topic of future work.

Algorithm 6: Determining the *unbiased* $\hat{\mathbf{W}}$ and $\hat{\mathbf{h}}$

Input : $\mathbf{x}$, P, δ
Output: $\hat{\mathbf{W}}$, $\hat{\mathbf{h}}$

All recursive variables resume from Algorithm 1;
$t = T^*$;
do
  $t = t + 1$;
  Recovering $\hat{\mathbf{W}}_t$;
  $\mathbf{R}_t = \lambda \mathbf{R}_{t-1} + \mathbf{x}_{P,t}\mathbf{x}_{P,t}^T$;
  $\mathbf{P}_t = \lambda \mathbf{P}_{t-1} + \mathbf{x}_t\mathbf{x}_{P,t}^T$;
  $\mathbf{G}_t = \left(\hat{\mathbf{\Psi}}_{t-1}\mathbf{R}_t - \mathbf{P}_t\right)_{\hat{\mathbf{W}}}$ where $(\cdot)_{\hat{\mathbf{W}}}$ is the projection to non-zero elements of $\hat{\mathbf{\Psi}}$ considering $\hat{\mathbf{W}}$;
  calculate $\mathbf{A}_t$;
  $\hat{\mathbf{\Psi}}_t = \hat{\mathbf{\Psi}}_{t-1} - \mathbf{G}_t(\mathbf{A}_t \otimes \mathbf{I}_{N\times N})$;
  Setting $\hat{\mathbf{W}}_t = \hat{\mathbf{\Psi}}_{1,t}$;
  Estimating $\hat{\mathbf{h}}$;
  $\mathbf{Y}_t = \left[\mathbf{x}_{t-1}, \hat{\mathbf{W}}_t\mathbf{x}_{t-1}, ..., \mathbf{x}_{t-P}, ..., \hat{\mathbf{W}}_t^P\mathbf{x}_{t-P}\right]$;
  $\mathbf{e}_t = \mathbf{x}_t - \mathbf{Y}_t\hat{\mathbf{h}}_{t-1}$;
  $\mathbf{b}_t : \ b_{i,t} = \frac{\text{sign}(\hat{h}_{i,t})}{\sigma + \hat{h}_{i,t}}$;
  $\hat{\mathbf{h}}_t = \hat{\mathbf{h}}_{t-1} + \rho_t\left(\mathbf{Y}_t^T\mathbf{e}_t + \eta_t\mathbf{b}_t\right)$;
while $t < T$ (an epoch with $\|\mathbf{e}_T\| < \delta$);
$\hat{\mathbf{W}} = \hat{\mathbf{W}}_T$, $\hat{\mathbf{h}} = \hat{\mathbf{h}}_T$.

7.3 Simulation Results

Since the suitable choice of the GSOs varies with applications, in our simulation, we tested our algorithm with synthetic graph processes which are consistent with the underlying assumptions of sparsity and shift invariance. Three different topologies of graphs were considered: arbitrarily random graph (R), random graph with power-law degree distribution (PL) [133], and Stochastic Block-Model (SBM) [134]. For each topology, the synthetic graph signal $\mathbf{x}_t \in \mathbb{R}^{12\times 12}$ was generated by feeding an i.i.d. input signal $\mathbf{w}_t \in \mathbb{R}^{12\times 12}$ into the stochastic processes in (6.6) and (6.7), with the number of vertices $N = 12$ and time lag order $P = 3$ throughout all simulations.

7.3.1 Convergence Performance against NMSE

In the first experiment, we examined how the overall algorithm performs in terms of the *normalized mean squared error* (NMSE) of $\mathbf{x}$ and $\mathbf{W}$, respectively denoted by

$$\sigma_t \triangleq \frac{\|\mathbf{e}_t\|_2^2}{\|\mathbf{x}_t\|_2^2}, \tag{7.35}$$

$$\zeta_t \triangleq \frac{\|\mathbf{W} - \hat{\mathbf{W}}_t\|_F^2}{\|\mathbf{W}\|_F^2} \tag{7.36}$$

where $\|\cdot\|_F$ indicates the Frobenius norm. The GSO, $\mathbf{W}$, was generated following the R/PL/SBM topologies chosen at random with 20 realizations in total. For an arbitrarily random topology, the weighted edges were drawn from $\mathcal{N}(0,1)$ and then thresholded to between 0.3 and 0.7 times the maximum absolute value of the components. Finally, the GSO matrix was normalized by 1.5 times its largest eigenvalue (to ensure stable processes).

The PL topology started from three random initial nodes connecting one another with probability 0.8; then new nodes were connected with the probability following the preferential attachment process [133] which is proportional to the total weight of the existing nodes. If connected, the weighted edges were drawn from $\mathcal{N}(0,1)$ and thresholded to between 0.05 and 0.95 times the maximum absolute value of the components, together with normalizing the GSO matrix by 1.5 times its largest eigenvalue. In the SBM case, the

network was clustered into 3 groups of $3, 4$ and 5 vertices each. The inter-/intra-cluster probability of connection was allocated by 3×3 matrix in the form of $0.25\mathbf{I} + \mathcal{U}(0.05, 0.2)$. Then, all the assigned edges were weighted by an exponential distribution with the rate $\lambda = 2$ and the matrix was finally normalized by 1.5 times its largest eigenvalue. All these specifications yielded the sparsity in $\mathbf{W}$ of approximately 0.2.

After $\mathbf{W}$ was created, $\mathbf{x}_t$ was obtained with the coefficients h_{ij} for $1 \leq i \leq P$ and $0 \leq j \leq i$, generated sparsely from a mixture of distribution $h_{ij} \sim \frac{1}{2^{i+j}}(\mathcal{U}(-1, -0.45) + \mathcal{U}(0.45, 1))$. The data was created for over 1100 samples, with first 500 samples left out due to their transient behavior, and the latter 600 samples kept for the simulation. We employed Algorithm 1 (**Path 1**) for the first 400 samples and Algorithm 2 for the remaining 200 samples, to recover $\mathbf{x}$ and $\mathbf{W}$, with the hyper-parameters μ_1, μ_2, μ_3, η chosen from the interval $(0, 5]$ with the step 0.1, γ from $(0, 2]$ with the step 0.1 and λ from $(0.8, 0.99]$ with the step 0.01. For this specific experiment, the selected hyperparameters would minimize the steady-state σ_t i.e. the averaged σ_t for t such that σ_t is in steady state. This step was repeated 20 times to obtain 20 realizations which were then averaged to display the outcome.

In terms of NMSE, the regularized algorithm (Algorithm 1) failed to minimize the 'normed' error of both $\mathbf{x}$ and $\mathbf{W}$, diverging away and levelling at a certain level; a jittery pattern was observed in the NMSE of $\mathbf{W}$. Afterwards, the debiasing process (Algorithm 2) managed to significantly reduce the error to a very low level, as expected from a generic adaptive algorithm. At the first glance, one may question the utility of the first step (Algorithm 1) as the standard stochastic algorithm (Algorithm 2) can accomplish the whole task. The answer is that Algorithm 2 (debiasing) only manipulates the explanatory part of the $\mathbf{W}$, determined by Algorithm 1. Therefore, it would be a disadvantage to neglect the *capability of identifying the correct topology of* $\mathbf{W}$, which is considered in the next experiment.

7.3.2 Identifiability of the GSO Topologies

The same data formulation as in the previous section was used Here, we focus on the likelihood that Algorithm 1 would successfully identify the right topology i.e. the non-zero elements of $\mathbf{W}$, regardless of their correct values. To this end, we compared the rate of *false alarm* (taking zero element as non-zero) and *miss* (failing to identify non-zero element) for every specific topology. Each case was calculated based on the average of 10 repeated random trials.

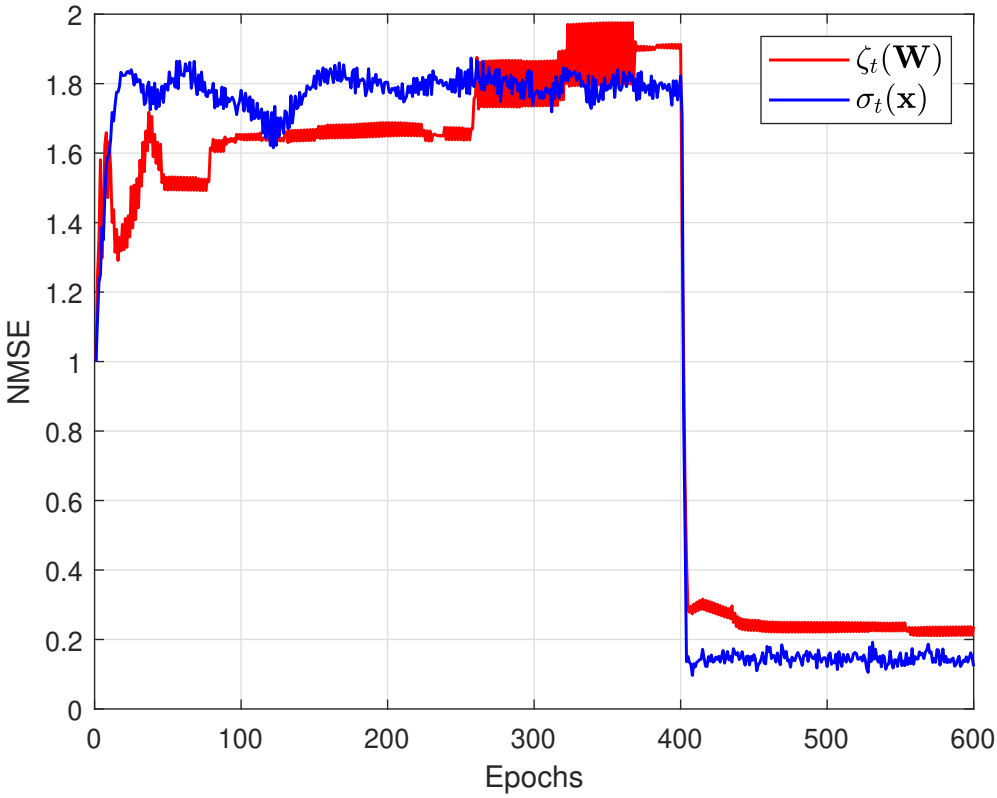


Figure 7.1: NMSE performance of the proposed algorithm with measures ζ_t (red) and σ_t (blue) which represent respectively the NMSE of $\mathbf{W}$ and $\mathbf{x}$. Algorithm 1 was implemented for the first 400 epochs of data samples and Algorithm 2 for the last 200 epochs.

As a consistent benchmark of the outcome, we tuned the hyperparameters such that, in each simulation, the sum of total false alarms (P_{FA}) and total misses (P_M) was minimal with respect to the hyperparameter grids described in the previous experiment. We tested our data twice with the **Path 1** and the **Path 2** of Algorithm 1, to establish

if this shifting order of the commutator term affects the learning performance. Table I shows the probabilities of false alarm (P_{FA}) and miss (P_M) via Path 1 of Algorithm 1 while Table II shows those of Path 2. Observe that Path 1 (using commutator term when estimating $\hat{\mathbf{W}}_t$ rather than at solving for $\hat{\mathbf{\Psi}}_t$) yielded more accurate recovery than

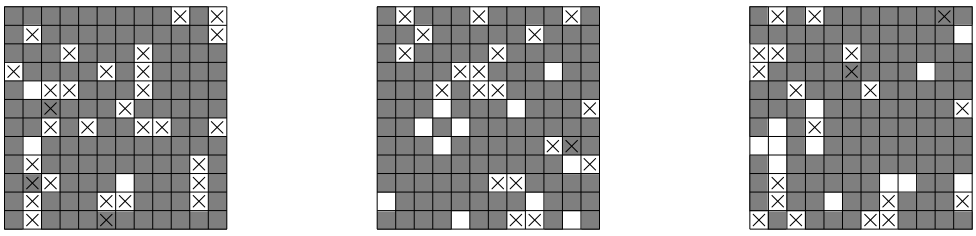
$\mathbf{W}$	P_{FA}	P_M
random	0.1033	0.1967
SBM	0.02	0.353
PL	0.067	0.42

Table 7.1: Results for Path 1

$\mathbf{W}$	P_{FA}	P_M
random	0.22	0.2167
SBM	0.0633	0.4567
PL	0.15	0.6

Table 7.2: Results for Path 2

the Path 2 (using the commutator term together with solving for $\hat{\mathbf{\Psi}}_t$ and letting $\hat{\mathbf{W}}_t = \hat{\mathbf{\Psi}}_{1,t}$). Regarding topology-wise comparison, the results expectedly show that specific topologies affect algorithm performance. When testing the arbitrarily random topology, we noticed the resulting recovery was *not* consistent, as indicated by the sum of P_{FA} and P_M varying considerably from experiment to experiment. Fig 2 (a) shows one of random-topology trials which are close to the average. When considering $\mathbf{W}$ with a clearly specified topology (SBM and PL), the recovery rate was more consistent with most SBM and PL trials, giving the recovery outcome close to the average. Fig 2 (b) and (c) visualize trial cases for both topologies. It should be noted that the recovery results of SBM and PL



(a) (b) (c)

Figure 7.2: Examples of visual representation of (a) arbitrarily random, (b) SBM and (c) power-law topologies of the graph shift operator, $\mathbf{W}$, from one simulation trial where blank spaces designate the non-zero entries, grey spaces the zero entries, and crosses the recovered entries.

topologies were not outstanding as some trails of the arbitrarily random topology gave

more precise identification; an example is shown in Fig. 2. While the structure of SBM and PL graphs ensured that the algorithm was less susceptible to identifying wrong edges, they disproportionately lacked in the ability to detect all the right ones (their P_M's were quite high compared to the *random* benchmark). When attempting to reduce high P_M's, their P_{FA} outgrew the intended reduction; in other words, after reaching some *point*, the algorithm began to wrongly identify edges at a rapid rate. Visually, we still could not distinguish what characteristics of $\mathbf{W}$ would render the algorithm more effective.

Finally, we would like to mention that these simulations were run on a small-scale problem due to computational limits, and hence the 12×12 size of $\mathbf{W}$ and 10 trails per topology could give biased and more varying results compared with the experiments involving thousands of nodes and hundreds of trials [125]. Nevertheless, the findings in this work indicate that there is much more room to discover in this research topic, since topology constraints play a crucial role in selecting appropriate optimization techniques to devise learning algorithms. This already suggests that other topological statistics like graph diameter, node degrees and many others could help with the design of the optimal algorithms.

7.4 Summary

This chapter presents a first design of adaptive graph signal processing implemented jointly by formulating a problem and devising a novel online algorithm accordingly. The model is based on vector autoregression (VAR) where the coefficient matrices are constrained by graph topology via a graph shift operator (GSO). The vertex-time relationship has been explained through a graph filter (vertex part) embedded into a VAR time series (time part), where causality has been imposed on the model to determine lag characteristics of the vertex-time models. To alleviate the non-convex nature arising from the polynomial structure of the graph filters, the problem has been divided into three sub-problems which themselves are re-expressed as convex problems. The algorithm has then been derived based on the split gradient projection method [130] which groups the first two sub-problems into Algorithm 1 and the last into Algorithm 2. The reason is that the

method is expected to produce biased results due to heavy-handed regularization of the problem. Therefore, after Algorithm 1, only the non-zero entries of the resulting GSO are computed in Algorithm 2 to *debias* the solution. Finally, the experiments have been carried out to illustrate the potentials and limits of the proposed method. The fine-tuning of hyperparameters poses another challenge as the empirical results from the algorithm is shown to be highly susceptible to how these hyperparameters are collectively set.

Chapter 8

Conclusions and Future Work

IN this final chapter we provide the recapitulation on what this thesis has accomplished and contributed, together with the indication of what could be further enhanced. The chapter first concludes the main discoveries and improvements of the thesis. Afterwards, it suggests interesting, yet unused, findings as well as pointing out some important assumptions and hypotheses assumed in the procedure of proof and derivation.

8.1 Conclusions

The thesis primarily cope with the extension of adaptive signal processing paradigm to more unconventional data types, particularly those arising out of the increasing availability of multisensor/multinode measurements, popularly known as Big Data; the main theme of this thesis is to employ Big Data techniques onto regular-size data to enable the impact of data diversity and fusion on normal data. The first data type considered is a quaternion, which is a continuing interest from the author's Master's research. The adaptive algorithms based on accelerated gradient have been proposed for a quaternion random processes - the widely linear quaternion least mean squares (WL-QLMS). Two approaches are offered, the n-moment WL-QLMS inspired by exact application of Nesterov's optimal algorithm and the c-moment WL-QLMS taken from Chebyshev's iterative method as an approximation of the former. The exact derivation poses a challenge as the stochastic nature increases the

chance of being underdamped (chance of diverging away), so the algorithmic step to reset the momentum is required. Empirically, the approximate c-moment shows no inferior performance to the more computationally complex n-moment, and thus the c-moment WL-QLMS comes out as the better all round. overall, these two proposed algorithms achieve fast convergence as the widely linear quaternion recursive least squares (WL-QRLS) but with much more numerical robustness (less likely to diverge away) and with quite similar computational complexity.

As we have been working on data fusion topic, it is a natural sague from quaternions to tensors thanks to their invertible isomorphism. Tensors address the direct shortfall of quaternions - 4-way analysis at maximum. As tensors can scale to as many dimensionality as one may like, this is actually a blessing to capture the data with such massive diversity, and then break down that data into a series of factors with massively less size. In our second effort is dictionary learning of tensor data via Tucker decomposition because dictionary learning is actually adaptive filtering algorithms with sparse coding scheme. Without hesitation, the mechanism obtained from the c-moment WL-QLMS is re-applied to derive the proposed online multilinear dictionary learning (OMDL) routine. Given the sparse coding is perfect (its solution is always optimal), it is shown via simulation that the OMDL could recover all the dictionaries perfectly despite its alternating minimization scheme. Furthermore, the joint higher-order compressed sensing (HO-CS) is also proposed to complement the OMDL in order to achieve best dimensionality reduction as in traditional CS paradigm. The relaxed ETF scheme is serialized to obtain the sensing matrix sequentially along with the sparsifying dictionaries from OMDL. The experiments compared the novel OMDL against similar tensor routines and clearly demonstrate better performance, sometimes slightly and sometimes by a large margin.

The sparsity is a typical constraint in today's modelling of large-scale data because in natural phenomena, information comes as ad hoc, random, and sparse in many real applications. This has made the transition from tensors to graphs very smooth for us as the signals on defined on both graphs and tensors usually share sparsity as a common feature. In our final research effort, adaptive filtering algorithm is extended to graph

signals in which we to the best of knowledge believe this is the first adaptive graph signal processing routine. Basically, graph signals are under the field of multivariate signal processing whereby the vector time series follows graph topology, mathematically represented by graph shift operator (GSO). Unlike traditional multivariate signal processing, the sparsity as a result of topology is challenging to be enforced in a real-time setting. To this end, we proposed a novel algorithm based on split gradient projection method to update positive and negative parts of the model parameters separately. This enables the adaptive algorithm to achieve truly sparse model for graph random processes. The experiments demonstrate the potentials and limits of the algorithm, the way to fix bias as a result of regularization parameters, the effect of fine-tuning minor parameters and varying performance against different GSO topologies.

In the final paragraph of this section, we would like to address some shortfalls of all the work in this thesis. In the quaternion part, we obtain the criterion to guarantee local convergence. We have found very challenging a proof of global convergence of a quaternion-valued function as many mathematical analysis utilized in classical setting is no longer applicable in quaternions such as the concept of strong convexity. We complement this shortfall by supplying extensive numerical simulation to show the proven robustness of the proposed algorithm. For the tensor dictionary, since our OMDL routine can be recast into a traditional DL problem with Kronecker structure, there exists a rigorous proof that the traditional DL algorithm converge to stationary point [88]. So far, the global convergence analysis is still an open topic, even with the traditional problem. Lastly for the graphs, owing to its heavy-handed regularization in the model, the algorithm is shown empirically to always diverge unless a *debiasing* algorithm is used to post-process the result. There also exists a proof which, however, relies on very restrictive assumptions to which very few natural signals would conform. Consequently, we decided to forgo the convergence analysis of the adaptive graph filtering algorithm and instead demonstrates empirical insights as we believe it would be more useful at the embryonic stage of the topic.

Bibliography

[1] T. Variddhisaï, M. Xiang, S. C. Douglas, and D. P. Mandic, "Quaternion-valued adaptive filtering via nesterov's extrapolation," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 4868–4872.

[2] T. Variddhisaï and D. P. Mandic, "On an rls-like lms adaptive filter," in *2017 22nd International Conference on Digital Signal Processing (DSP)*, Aug 2017, pp. 1–5.

[3] T. Variddhisaï and D. Mandic, "Online multilinear dictionary learning," 2017.

[4] T. Variddhisai, "An approach to hypercomplex adaptive filters and their real-world applications," M.Sc. thesis, Imperial College London, London, UK, Sep 2014.

[5] A. Sayed, *Adaptive Filters.* Wiley-IEEE Press, 2008.

[6] E. Chong and S. Zak, *An Introduction to Optimization.* John Wiley and Sons, 2013.

[7] B. Widrow, J. McCool, M. Larimore, and C. Johnson, "Stationary and nonstationary learning characteristics of the lms adaptive filter," *Proceedings of the IEEE*, vol. 64, no. 8, pp. 1151–1162, Aug 1976.

[8] C. Jahanchahi, C. Took, and D. Mandic, "On gradient calculation in quaternion adaptive filtering," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, March 2012, pp. 3773–3776.

[9] A. Liavas and P. Regalia, "Numerical stability issues of the conventional recursive least squares algorithm," in *Acoustics, Speech and Signal Processing, 1998. Proceed-*

ings of the 1998 IEEE International Conference on, vol. 3, May 1998, pp. 1409–1412 vol.3.

[10] C. Davila, "Recursive total least squares algorithms for adaptive filtering," in *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, Apr 1991, pp. 1853–1856 vol.3.

[11] C. Jahanchahi, C. Took, and D. Mandic, "The widely linear quaternion recursive least squares filter," in *Cognitive Information Processing (CIP), 2010 2nd International Workshop on*, Jun 2010, pp. 87–92.

[12] C. Mertler, *Advanced and Multivariate Statistical Methods: Practical Application and Interpretation.* Glendale, CA: Pyrczak Publishing, 2013.

[13] C. Took and D. Mandic, "Fusion of heterogeneous data sources: A quaternionic approach," in *Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on*, Oct 2008, pp. 456–461.

[14] ——, "A quaternion widely linear adaptive filter," *Signal Processing, IEEE Transactions on*, vol. 58, no. 8, pp. 4427–4431, Aug 2010.

[15] A. Oppenheim and J. Lim, "The importance of phase in signals," *Proceedings of the IEEE*, vol. 69, no. 5, pp. 529–541, May 1981.

[16] G. Golub and C. Van Loan, *Matrix Computations.* Baltiore, MD-US: Johns Hopkins University Press, 2013.

[17] W. Hamilton, *Elements of Quaternions*, 2nd ed. London, UK: Longmans, Green & Company, 1899.

[18] J. Gallier, *Geometric Methods and Applications For Computer Science and Engineering.* Springer, 2011.

[19] S. Altmann, *Rotations, Quaternions and Double Groups.* Oxford, UK: Dover Publications, 1986.

[20] Z. Shao, J. Wu, J. Coatrieux, G. Coatrieux, and H. Shu, "Quaternion gyrator transform and its application to color image encryption," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*, Sept 2013, pp. 4579–4582.

[21] C. Perez, C. Navarro, D. Schulz, J. Saravia, and C. Aravena, "Pattern detection using a new haralick quaternion color extraction model and support vector machine classifier," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, Oct 2013, pp. 3300–3304.

[22] D. Biamino, G. Cannata, M. Maggiali, and A. Piazza, "Mac-eye: a tendon driven fully embedded robot eye," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, Dec 2005, pp. 62–67.

[23] X. Yun and E. Bachmann, "Design, implementation, and experimental results of a quaternion-based kalman filter for human body motion tracking," *Robotics, IEEE Transactions on*, vol. 22, no. 6, pp. 1216–1227, Dec 2006.

[24] J. Marins, X. Yun, E. Bachmann, R. Mcghee, and M. Zyda, "An extended kalman filter for quaternion-based orientation estimation using marg sensors," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 4, 2001, pp. 2003–2011.

[25] D. Choukroun, I. Y. Bar-Itzhack, and Y. Oshman, "Novel quaternion kalman filter," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 42, no. 1, pp. 174–190, Jan 2006.

[26] C. Jahanchahi and D. Mandic, "A class of quaternion kalman filters," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 3, pp. 533–544, Mar 2014.

[27] J. C. Belfiore and G. Rekaya, "Quaternionic lattices for space-time coding," in *Information Theory Workshop, 2003. Proceedings. 2003 IEEE*, March 2003, pp. 267–270.

[28] T. Unger and N. Markin, "Quadratic forms and space-time block codes from generalized quaternion and biquaternion algebras," *Information Theory, IEEE Transactions on*, vol. 57, no. 9, pp. 6148–6156, Sept 2011.

[29] R. Vehkalahti, C. Hollanti, and F. Oggier, "Fast-decodable asymmetric space-time codes from division algebras," *Information Theory, IEEE Transactions on*, vol. 58, no. 4, pp. 2362–2385, April 2012.

[30] J. Cifuentes, M. T. Pham, R. Moreau, F. Prieto, and P. Boulanger, "An arc-length warping algorithm for gesture recognition using quaternion representation," in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, July 2013, pp. 6248–6251.

[31] in *Neural Networks in Multidimensional Domains*, ser. Lecture Notes in Control and Information Sciences, P. Arena, L. Fortuna, G. Muscato, and M. Xibilia, Eds. Springer, 1998, vol. 234.

[32] V. Risojevic and Z. Babic, "Unsupervised learning of quaternion features for image classification," in *Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), 2013 11th International Conference on*, vol. 01, Oct 2013, pp. 345–348.

[33] B. Wysocki, T. Wysocki, and J. Seberry, "Modeling dual polarization wireless fading channels using quaternions," in *Mobile Future, 2006 and the Symposium on Trends in Communications. SympoTIC '06. Joint IST Workshop on*, June 2006, pp. 68–71.

[34] L. G. P. Meloni, "Hypercomplex OFDM schemes for cross-polarized antennas," in *Communications and Information Technologies (ISCIT), 2012 International Symposium on*, Oct 2012, pp. 502–507.

[35] S. Miron, N. Le Bihan, and J. Mars, "Quaternion-MUSIC for vector-sensor array processing," *Signal Processing, IEEE Transactions on*, vol. 54, no. 4, pp. 1218–1229, April 2006.

[36] N. ur Rehman and D. Mandic, "Empirical mode decomposition for trivariate signals," *Signal Processing, IEEE Transactions on*, vol. 58, no. 3, pp. 1059–1068, Mar 2010.

[37] C. Took, D. Mandic, and K. Aihara, "Quaternion-valued short term forecasting of wind profile," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, Jul 2010, pp. 1–6.

[38] T. Ell and S. Sangwine, "Quaternion involutions and anti-involutions," *Computers & Mathematics with Applications*, vol. 53, no. 1, pp. 137–143, 2007.

[39] D. Mandic and V. Goh, *Complex Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models.* West Sussex, UK: John Wiley and Sons, 2009.

[40] C. Took and D. Mandic, "Augmented second-order statistics of quaternion random signals," *Signal Processing*, vol. 91, no. 2, pp. 214–224, 2011.

[41] K. Viswanath, "Normal operators on quaternionic hilbert spaces," *Transactions of the American Mathematical Society*, vol. 162, pp. 337–350, Dec 1971.

[42] D. Xu, C. Jahanchahi, C. Took, and D. Mandic, "Enabling quaternion derivatives: the generalized hr calculus," *Royal Society*, Aug 2015.

[43] P. A. Grillet, *Abstract Algebra.* Springer, 2007.

[44] V. Shpakivskyi, "Linear quaternionic equations and their systems," *Advances in Applied Clifford Algebras*, vol. 21, no. 3, pp. 637–645, 2011.

[45] G. Ehrlich, *Fundamental Concepts of Abstract Algebra.* Dover Publications, Feb 2012.

[46] S. D. Leo and P. Rotelli, "Quaternionic analyticity," *Applied Mathematics Letters*, vol. 16, no. 7, pp. 1077 – 1081, 2003.

[47] K. Kreutz-Delgado, "The complex gradient operator and the cr-calculus," 2009.

[48] D. P. Mandic, C. Jahanchahi, and C. C. Took, "A quaternion gradient operator and its applications," *IEEE Signal Processing Letters*, vol. 18, no. 1, pp. 47–50, Jan 2011.

[49] C. A. Deavours, "The quaternion calculus," *The American Mathematical Monthly*, vol. 80, no. 9, pp. 995–1008, 1973.

[50] A. Sudbery, "Quaternionic analysis," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 85, no. 2, p. 199–225, 1979.

[51] W. Wirtinger, "Zur formalen theorie der funktionen von mehr komplexen veränderlichen," *Mathematische Annalen*, vol. 97, pp. 357–376, 1927.

[52] J. Navarro-Moreno, "Arma prediction of widely linear systems by using the innovations algorithm," *Signal Processing, IEEE Transactions on*, vol. 56, no. 7, pp. 3061–3068, July 2008.

[53] N. Le Bihan, "On properness of quaternion-valued random variables," *Proc. Int. Conf. Mathematics (IMA) Signal Processing*, pp. 23–26, Dec 2004.

[54] B. Picinbono, "On circularity," *Signal Processing, IEEE Transactions on*, vol. 42, no. 12, pp. 3473–3482, Dec 1994.

[55] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 4th ed. McGraw Hill, 2002.

[56] C. Took and D. Mandic, "The quaternion lms algorithm for adaptive filtering of hypercomplex processes," vol. 57, no. 4, pp. 1316–1327, Apr 2009.

[57] ——, "Quaternion-valued stochastic gradient-based adaptive IIR filtering," *Signal Processing, IEEE Transactions on*, vol. 58, no. 7, pp. 3895–3901, Jul 2010.

[58] C. Took, D. Mandic, and J. Benesty, "Study of the quaternion lms and four-channel lms algorithms," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, April 2009, pp. 3109–3112.

[59] S. Sangwine, "Fourier transforms of colour images using quaternion or hypercomplex, numbers," *Electronics Letters*, vol. 32, no. 21, pp. 1979–1980, Oct 1996.

[60] Y. Liu, J. Jin, Q. Wang, and S. Yi, "Ultrasound extended-field-of-view imaging based on motion estimation using quaternion wavelet," in *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, May 2012, pp. –.

[61] D. Bertsekas, *Nonlinear Programming.* Athena Scientific, 1999.

[62] Y. Nesterov, "A method for solving the convex programming problem with convergence rate $o(1/k^2)$," 1983.

[63] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[64] D. Bertsekas and M. I. of Technology, *Convex Optimization Algorithms.* Athena Scientific, 2015.

[65] B. O'donoghue and E. Candès, "Adaptive restart for accelerated gradient schemes," *Found. Comput. Math.*, vol. 15, no. 3, p. 715–732, Jun. 2015.

[66] D. P. Mandic and J. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms,Architectures and Stability.* USA: John Wiley & Sons, Inc., 2001.

[67] J. Lee, J.-W. Chen, and H.-C. Huang, "Performance comparison of variable step-size nlms algorithms," 2009.

[68] D. P. Mandic, "A generalized normalized gradient descent algorithm," *IEEE Signal Processing Letters*, vol. 11, no. 2, pp. 115–118, Feb 2004.

[69] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, p. 455–500, Aug. 2009.

[70] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. PHAN, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, March 2015.

[71] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, July 2017.

[72] L. R. Tucker, "The extension of factor analysis to three-dimensional matrices," in *Contributions to mathematical psychology.*, H. Gulliksen and N. Frederiksen, Eds. New York: Holt, Rinehart and Winston, 1964, pp. 110–127.

[73] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, pp. 283–319, 1970.

[74] R. Harshman, "Foundations of the parafac procedure: models and conditions for an 'exploratory' multimodal factor analysis," in *UCLA Working Papers in Phonetics*, 1970, pp. 1–84.

[75] P. Kroonenberg, *Applied Multiway Data Analysis*, 01 2008, vol. 702.

[76] N. D. Sidiropoulos, R. Bro, and G. B. Giannakis, "Parallel factor analysis in sensor array processing," *IEEE Transactions on Signal Processing*, vol. 48, no. 8, pp. 2377–2388, Aug 2000.

[77] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[78] P. Comon and C. Jutten, Eds., *Contributors.* Oxford: Academic Press, 2010.

[79] I. Domanov and L. De Lathauwer, "On the uniqueness of the canonical polyadic decomposition of third-order tensors—part ii: Uniqueness of the overall decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 3, pp. 876–903, 2013.

[80] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb 2006.

[81] S. Qaisar, R. M. Bilal, W. Iqbal, M. Naureen, and S. Lee, "Compressive sensing: From theory to applications, a survey," *Journal of Communications and Networks*, vol. 15, no. 5, pp. 443–456, Oct 2013.

[82] M. F. Duarte and R. G. Baraniuk, "Kronecker product matrices for compressive sensing," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2010, pp. 3650–3653.

[83] I. Domanov and L. De Lathauwer, "On the uniqueness of the canonical polyadic decomposition of third-order tensors—part i: Basic results and uniqueness of one factor matrix," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 3, pp. 855–875, 2013.

[84] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, Nov 2006.

[85] K. Engan, K. Skretting, and J. H. Husøy, "Family of iterative ls-based dictionary learning algorithms, ils-dla, for sparse signal representation," *Digital Signal Processing*, vol. 17, no. 1, pp. 32 – 49, 2007.

[86] M. Yaghoobi, T. Blumensath, and M. E. Davies, "Dictionary learning for sparse approximations with the majorization method," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2178–2191, June 2009.

[87] M. Aharon and M. Elad, "Sparse and redundant modeling of image content using an image-signature-dictionary," *SIAM Journal on Imaging Sciences*, vol. 1, no. 3, pp. 228–247, 2008.

[88] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, p. 19–60, Mar. 2010.

[89] K. Skretting and K. Engan, "Recursive least squares dictionary learning algorithm," *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2121–2130, April 2010.

[90] G. Zhang, Z. Jiang, and L. S. Davis, "Online semi-supervised discriminative dictionary learning for sparse representation," in *Computer Vision – ACCV 2012*, K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 259–273.

[91] F. Yang, Z. Jiang, and L. S. Davis, "Online discriminative dictionary learning for visual tracking," in *IEEE Winter Conference on Applications of Computer Vision*, March 2014, pp. 854–861.

[92] S. Kim, "Online kernel dictionary learning," in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec 2015, pp. 103–107.

[93] Y. Naderahmadian, S. Beheshti, and M. A. Tinati, "Correlation based online dictionary learning algorithm," *IEEE Transactions on Signal Processing*, vol. 64, no. 3, pp. 592–602, Feb 2016.

[94] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, March 2008.

[95] L. Zelnik-Manor, K. Rosenblum, and Y. C. Eldar, "Sensing matrix optimization for block-sparse decoding," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4300–4312, Sep. 2011.

[96] V. Abolghasemi, S. Ferdowsi, and S. Sanei, "A gradient-based alternating minimization approach for optimization of the measurement matrix in compressive sensing," *Signal Processing*, vol. 92, no. 4, pp. 999 – 1009, 2012.

[97] W. Chen, M. R. D. Rodrigues, and I. J. Wassell, "Projection design for statistical compressive sensing: A tight frame based approach," *IEEE Transactions on Signal Processing*, vol. 61, no. 8, pp. 2016–2029, April 2013.

[98] G. Li, X. Li, S. Li, H. Bai, Q. Jiang, and X. He, "Designing robust sensing matrix for image compression," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5389–5400, Dec 2015.

[99] W. Chen and M. R. D. Rodrigues, "Dictionary learning with optimized projection design for compressive sensing applications," *IEEE Signal Processing Letters*, vol. 20, no. 10, pp. 992–995, Oct 2013.

[100] H. Bai, G. Li, S. Li, Q. Li, Q. Jiang, and L. Chang, "Alternating optimization of sensing matrix and sparsifying dictionary for compressed sensing," *IEEE Transactions on Signal Processing*, vol. 63, no. 6, pp. 1581–1594, March 2015.

[101] C. F. Caiafa and A. Cichocki, "Computing sparse representations of multidimensional signals using kronecker bases," *Neural Computation*, vol. 25, no. 1, pp. 186–220, Jan 2013.

[102] G. Duan, H. Wang, Z. Liu, J. Deng, and Y. Chen, "K-cpd: Learning of overcomplete dictionaries for tensor sparse coding," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Nov 2012, pp. 493–496.

[103] F. Roemer, G. Del Galdo, and M. Haardt, "Tensor-based algorithms for learning multidimensional separable dictionaries," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 3963–3967.

[104] R. Zhao, Q. Wang, Y. Shen, and J. Li, "Multidimensional dictionary learning algorithm for compressive sensing-based hyperspectral imaging," *Journal of Electronic Imaging*, vol. 25, 2016.

[105] X. Ding, W. Chen, and I. J. Wassell, "Joint sensing matrix and sparsifying dictionary optimization for tensor compressive sensing," *IEEE Transactions on Signal Processing*, vol. 65, no. 14, pp. 3632–3646, July 2017.

[106] E. J. Candès, Y. C. Eldar, D. Needell, and P. Randall, "Compressed sensing with coherent and redundant dictionaries," *Applied and Computational Harmonic Analysis*, vol. 31, no. 1, pp. 59 – 73, 2011.

[107] T. Strohmer and R. W. Heath, "Grassmannian frames with applications to coding and communication," *Applied and Computational Harmonic Analysis*, vol. 14, no. 3, pp. 257 – 275, 2003.

[108] S. Kwon, J. Wang, and B. Shim, "Multipath matching pursuit," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2986–3001, May 2014.

[109] S. Hawe, M. Seibert, and M. Kleinsteuber, "Separable dictionary learning," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, June 2013, pp. 438–445.

[110] D. P. Mandic, S. Kanna, and A. G. Constantinides, "On the intrinsic relationship between the least mean square and kalman filters [lecture notes]," *IEEE Signal Processing Magazine*, vol. 32, no. 6, pp. 117–122, Nov 2015.

[111] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug 2013.

[112] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, June 2010.

[113] N. D. Keni and R. A. Ansari, "Convex optimization based sparse dictionary learning for image compression," in *2017 4th International Conference on Signal Processing and Integrated Networks (SPIN)*, Feb 2017, pp. 584–589.

[114] Y. Rivenson and A. Stern, "Compressed imaging with a separable sensing operator," *IEEE Signal Processing Letters*, vol. 16, no. 6, pp. 449–452, June 2009.

[115] T. Hong and Z. Zhu, "An efficient method for robust projection matrix design," *Signal Processing*, vol. 143, pp. 200 – 210, 2018.

[116] M. Yaghoobi, L. Daudet, and M. E. Davies, "Parametric dictionary design for sparse coding," *IEEE Transactions on Signal Processing*, vol. 57, no. 12, pp. 4800–4810, Dec 2009.

[117] L. Stankovic, D. P. Mandic, M. Dakovic, I. Kisil, E. Sejdic, and A. G. Constantinides, "Understanding the basis of graph signal processing via an intuitive example-driven approach [lecture notes]," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 133–145, Nov 2019.

[118] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Transactions on Signal Processing*, vol. 65, no. 22, pp. 5911–5926, Nov 2017.

[119] B. S. Dees, L. Stankovic, M. Dakovic, A. G. Constantinides, and D. P. Mandic, "A class of doubly stochastic shift operators for random graph signals and their boundedness," 2019.

[120] P. Di Lorenzo, S. Barbarossa, P. Banelli, and S. Sardellitti, "Adaptive least mean squares estimation of graph signals," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 555–568, Dec 2016.

[121] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus, "Adaptive graph signal processing: Algorithms and optimal sampling strategies," *IEEE Transactions on Signal Processing*, vol. 66, no. 13, pp. 3584–3598, July 2018.

[122] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, and Y. Gu, "Time-varying graph signal reconstruction," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 870–883, Sep. 2017.

[123] D. Romero, V. N. Ioannidis, and G. B. Giannakis, "Kernel-based reconstruction of space-time functions on dynamic graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 856–869, Sep. 2017.

[124] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, "Forecasting time series with varma recursions on graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 18, pp. 4870–4885, Sep. 2019.

[125] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, April 2017.

[126] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. Boston: McGraw Hill, 2002.

[127] R. C. de Lamare and R. Sampaio-Neto, "Sparsity-aware adaptive algorithms based on alternating optimization and shrinkage," *IEEE Signal Processing Letters*, vol. 21, no. 2, pp. 225–229, Feb 2014.

[128] Yilun Chen, Y. Gu, and A. O. Hero, "Sparse lms for system identification," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2009, pp. 3125–3128.

[129] O. Taheri and S. A. Vorobyov, "Sparse channel estimation with lp-norm and reweighted l1-norm penalized least mean squares," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 2864–2867.

[130] M. Schmidt, G. Fung, and R. Rosales, "Fast optimization methods for l1 regularization: A comparative study and two new approaches," in *Machine Learning: ECML 2007*, J. N. Kok, J. Koronacki, R. L. d. Mantaras, S. Matwin, D. Mladenič, and A. Skowron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 286–297.

[131] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, May 1995.

[132] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale ℓ_1-regularized least squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, Dec 2007.

[133] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[134] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E*, vol. 83, p. 016107, Jan 2011.

Appendix A

Statement of IEEE Copyright