Composition Structures for System Representation

by

Guolong Su

B.Eng. Electronic Eng., Tsinghua University, China (2011)
S.M. EECS, Massachusetts Institute of Technology (2013)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

Author . . . .
Department of Electrical Engineering and Computer Science
May 19, 2017

Certified by . . . .
Alan V. Oppenheim
Ford Professor of Engineering
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Theses

Composition Structures for System Representation

by

Guolong Su

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

This thesis discusses parameter estimation algorithms for a number of structures for system representation that can be interpreted as different types of composition. We refer to the term composition as the systematic replacement of elements in an object by other object modules, where the objects can be functions that have a single or multiple input variables as well as operators that work on a set of signals of interest. In general, composition structures can be regarded as an important class of constrained parametric representations, which are widely used in signal processing. Different types of composition are considered in this thesis, including multivariate function composition, operator composition that naturally corresponds to cascade systems, and modular composition that we refer to as the replacement of each delay element in a system block diagram with an identical copy of another system module. There are a number of potential advantages of the use of composition structures in signal processing, such as reduction of the total number of independent parameters that achieves representational and computational efficiency, modular structures that benefit hardware implementation, and the ability to form more sophisticated models that can represent significantly larger classes of systems or functions.

The first part of this thesis considers operator composition, which is an alternative interpretation of the class of cascade systems that has been widely studied in signal processing. As an important class of linear time-invariant (LTI) systems, we develop new algorithms to approximate a two-dimensional (2D) finite impulse response (FIR) filter as a cascade of a pair of 2D FIR filters with lower orders, which can gain computational efficiency. For nonlinear systems with a cascade structure, we generalize a two-step parameter estimation algorithm for the Hammerstein model, and propose a generalized all-pole modeling technique with the cascade of multiple nonlinear memoryless functions and LTI subsystems.

The second part of this thesis discusses modular composition, which replaces each delay element in a FIR filter with another subsystem. As an example, we propose the modular Volterra system where the subsystem has the form of the Volterra series. Given statistical information between input and output signals, an algorithm is proposed to estimate the coefficients of the FIR filter and the kernels of the Volterra subsystem, under the assumption that the coefficients of the nonlinear kernels have sufficiently small magnitude.

The third part of this thesis focuses on composition of multivariate functions. In particular, we consider two-level Boolean functions in the conjunctive or disjunctive normal forms, which can be considered as the composition of one-level multivariate Boolean functions that take the logical conjunction (or disjunction) over a subset of binary input variables. We propose new optimization-based approaches for learning a two-level Boolean function from a training dataset for classification purposes, with the joint criteria of accuracy and simplicity of the learned function.

Thesis Supervisor: Alan V. Oppenheim
Title: Ford Professor of Engineering

Acknowledgments

I would like to sincerely express my gratitude to my supervisor Prof. Alan V. Oppenheim for his wonderful guidance, sharp insights, unconventional creativity, and strong mastery of the big picture. It has been my great privilege to work with Al, who truly cares about the comprehensive development and the best interests of his students. Among Al's various contributions to the development of this thesis, I would like to especially thank Al for his intentionally cultivating my independence and overall maturity, while carefully and subtly providing a safety net. I benefited significantly from Al's research style of free association, which tries to connect seemingly unrelated topics but can finally lead to unexpected inspiration. Moreover, I appreciate Al encouraging me to step back from the technical aspects and distill a deeper conceptual understanding for the work that I have done. Having attended Al's lectures for the course *Discrete-time Signal Processing* in multiple semesters as a student and as a teaching assistant, I am impressed and motivated by Al's enthusiasm and devotion to teaching, for which (and for other things) he always aims for excellence and takes the best efforts to further improve, even on the materials that he is so familiar with. Al, thank you for teaching me the way to be a good teacher. In addition, I am very grateful for Al's various warm and voluntary help on non-academic issues.

I am fully grateful to the other members of my thesis committee, Prof. George C. Verghese and Dr. Sefa Demirtas, for their involvement in the development of my research project and their great suggestions in both the committee and individual meetings, which had significant contributions especially in clarifying the concepts and shaping this thesis. Aside from this thesis, I would like to thank Al and George for offering me the opportunity to work on the exercises of their new book *Signals, Systems and Inference*, which became a good motivation for me to gain deeper knowledge about related topics such as Wiener filtering. Sefa, I am so fortunate to have the great and enjoyable collaboration with you on polynomial decomposition algorithms when I was working towards my master's degree, from which some ideas in

this thesis were inspired years later in an unexpected way; beyond our collaboration, you are a kind and supportive mentor, a frank and fun office mate, and a very responsible thesis reader, and I want to sincerely thank you for all of these.

I would like to have my sincere appreciation to Prof. Gregory W. Wornell and Prof. Stefanie S. Jegelka, who kindly granted me with another teaching assistantship opportunity for their course *Inference and Information* in Spring 2015, which turned out an excellent and precious experience for me. In addition to Greg's passion and erudition, I really appreciate his amazing ability in distilling the essence from a complicated concept and illustrating it in a natural and enjoyable way that is easy to grasp. Greg and Stefanie, thank you for showing and training me on how to teach effectively and for deepening my own understanding of the course materials that I feel attractive.

It is my great privilege to be a member of the "academic family", Digital Signal Processing Group (DSPG). I have had abundant interactions with DSPG members that were fun and contributive to this thesis, especially with the following regular attendees at the brainstorming group meeting, both past and present (in alphabetical order): Thomas A. Baran, Petros T. Boufounos, Sefa Demirtas, Dan E. Dudgeon, Xue Feng, Yuantao Gu, Tarek A. Lahlou, Hsin-Yu (Jane) Lai, Pablo Martínez Nuevo, Martin McCormick, Catherine Medlock, Milutin Pajovic, Charlie E. Rohrs, James Ward, Sally P. Wolfe, and Qing Zhuo. The supportive, creative, and harmonic atmosphere in DSPG makes my journey enjoyable and enriching. Beyond this thesis, I benefited considerably from discussions with DSPG members, each of whom has different expertise and is always willing to share. In particular (again in alphabetical order), thanks to Tom for various brainstorming conversations; thanks to Petros for pointing me to existing works on low rank matrix approximation; thanks to Dan for discussions on 2D filter implementation; thanks to Yuantao for your wonderful guidance, friendly mentoring, and sincere discussions during my undergraduate study and your visit to MIT; thanks to Tarek for our so many fun conversations and your initializing the collaboration on the lattice filter problem; thanks to Pablo for being a great office mate and providing opportunities for deep mathematical discussions;

thanks to Milutin for various discussions on statistics. Moreover, I would like to thank Austin D. Collins and Jennifer Tang for bringing fun topics from a different field to the group meetings. I am also grateful to DSPG alumni Andrew I. Russell, Charles K. Sestok, and Wade P. Torres for occasional but helpful discussions on this thesis and their kind help in other aspects. In addition, I want to thank Dr. Fernando A. Mujica, who visited DSPG for a number of times, for pointing me to the indirect learning architecture used in the predistortion linearization technique and for his other valuable help. Special thanks to Laura M. von Bosau, Yugandi S. Ranaweera, and Lesley Rice for their responsible efforts and support to make DSPG function smoothly.

During the Ph.D. program, I was fortunate to have four unforgettable summer internships. I would like to thank all my managers and mentors for offering me these valuable opportunities to explore enjoyable projects and for their great guidance (in chronological order): Dr. Davis Y. Pan at Bose Corporation, Dr. Arthur J. Redfern at Texas Instruments, Dr. Simant Dube at A9.com, and Dr. Dennis Wei (a DSPG alumnus), Dr. Kush R. Varshney, and Dr. Dmitry M. Malioutov at IBM Research. Davis, thank you for taking me to the great trip to Maine. Arthur, thank you for various enjoyable activities such as golf and karting. Simant, thank you for exposing me to the fun field of computer vision. Dennis, Kush, and Dmitry, thank you for introducing me to the attractive topic of Boolean function learning, which later was further developed into a chapter of this thesis, and thank you all for your active involvement and wonderful suggestions even after my internship.

I would like to thank my friends who I fortunately got to know or became familiar with during my stay at Boston: Zhan Su, Ningren (Peter) Han, Jinshuo Zhang, Zhengdong Zhang, Fei Fang, Xun Cai, Qing Liu, Xuhong (Lisa) Zhang, Jie Ding, John Z. Sun, Da Wang, Ying Liu, Li Yu, Qing He, Quan Li, Tianheng Wang, Wenhan Dai, Zhenyu Liu, Jianan Zhang, Omer Tanovic, Anuran Makur, Lucas Nissenbaum, Megan M. Fuller, Ganesh Ajjanagadde, Ronald J. Wilcox, Hangbo Zhao, Xiang Ji, Yuxuan (Cosmi) Lin, Xi Ling, Tian Ming, Haozhe Wang, Xu Zhang, Yi Song, Wenjing Fang, Yanfei Xu, Jianjian Wang, Qiong Ma, Suyang Xu, Linda Ye, Yan Chen, Mingda

Li, Chun Wang Ivan Chan, Xiaowei Cai, Lulu Li, Haoxuan (Jeff) Zheng, Yingxiang Yang, Ruobin Gong, and Yan Miao. Thank all these friends for interesting activities and discussions that make my daily life colorful and enjoyable. In addition, I want to acknowledge these close friends that I had become familiar with far prior to graduate study: Yunhui Tan, Jiannan Lu, Yangqin Fang, Qing Qu, Lanchun Lyu, and Yiwen Wang; since all of us have pursued or are pursuing Ph.D., we had plenty of fun conversations sharing happiness and providing support. A number of my friends, both in and outside Boston, have kindly offered me important help and valuable suggestions regarding to various stages in and after my Ph.D. journey, for which I am really grateful.

I regard myself very lucky to have met my significant other Shengxi Huang, who is sincerely kindhearted, unconventionally humorous, always resourceful, and consistently patient. Shengxi, thank you for all your companionship, love, support, and kindness to me for these past years and for future.

Last but not least, I am deeply indebted to my dear parents. I am grateful for their unboundedly deep love, absolutely supportive understanding, and inexhaustibly amiable encouragement, no matter whether I am a child or a grown-up, no matter whether our geographical distance is near or far. No words can describe my love and gratitude for my parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Parametric representations are widely used in signal processing for modeling both systems and signals. Examples of parametric representations include linear time-invariant filters with rational transfer functions that are parameterized by the coefficients of the numerators and the denominators, bandlimited periodic signals that can be characterized by the fundamental frequency as well as the magnitude and phase of each harmonic, and stochastic signal models with parametric probability distributions. Parameter estimation algorithms are usually needed for these representations to obtain the parameters from empirical observations or statistical information.

For certain classes of parametric models, the number of *independent parameters* is smaller than that of a natural parametrization of the model, which can enable an alternative and more compact parametrization. As an example, a class of two-dimensional (2D) finite impulse response (FIR) filters is the separable filters [3], the transfer functions of which satisfy $H(z_1, z_2) = H_1(z_1) \cdot H_2(z_2)$. If $H_1(z_1)$ and $H_2(z_2)$ are of orders N_1 and N_2, respectively, then the total number of independent parameters[1] is $N_1 + N_2 + 1$. If we consider the impulse response as a natural parametrization of $H(z_1, z_2)$, then it has $(N_1 + 1) \cdot (N_2 + 1)$ parameters, and thus the parametrization with $H_1(z_1)$ and $H_2(z_2)$ is more compact. As an example of signal representation,

[1] In fact, $H_1(z_1)$ has $N_1 + 1$ parameters and $H_2(z_2)$ has $N_2 + 1$ parameters. Since respectively scaling $H_1(z_1)$ and $H_2(z_2)$ with c and $1/c$ leads to the same product, the number of independent parameters is reduced by one.

sparse signals have most elements as zero and can be efficiently characterized by the locations and values of the non-zero elements [4,5]. Moreover, low rank matrices are a widely used model in applications such as recommendation systems [6,7], background modeling for video processing [8, 9], and biosciences [10]; a low rank matrix can be parameterized by the non-zero singular values and the associated singular vectors from the singular value decomposition (SVD), which typically have fewer parameters than the total number of elements in the matrix.

As we can see from these examples, the reduction of independent parameters typically results from additional constraints on the parametric models, such as separability of 2D filters, sparsity of signals, and the low rank property of matrices. These additional constraints introduce dependence among the parameters in the natural representation, and therefore the number of degrees of freedom is reduced. From a geometric perspective, the feasible region of a constrained parametric model can be regarded as a manifold in a higher dimensional space associated with the natural parametrization. In signal processing, the reduction of independent parameters could lead to system implementation with higher efficiency, more compact signal representation, and efficient extraction of key information from high-dimensional data.

As a class of constrained parametric models, composition structures typically have the form of systematic replacement of elements in an object with another object module. By using proper composition structures for system and signal representation, we can achieve the potential advantages of parameter reduction, modularity for hardware implementation, and the ability to form more sophisticated models that can represent significantly larger classes of systems or functions, which we will illustrate in later sections of this chapter.

There are different types of composition with different choices of objects and replacement approaches. As an example, function composition replaces the input variable of a function with the output of another function. It has been well-studied in theoretical and computational mathematics, spanning a wide range of topics such as polynomial composition [11–14], rational function composition [15, 16], and iterated functions that are generated by composing a function with itself multiple times [17,18].

Other types of composition considered in this thesis include operator composition and modular composition, where the latter refers to the replacement of each delay element in a system block diagram with another subsystem.

The remainder of this chapter is organized as follows. Section 1.1 reviews the different types of composition structures and presents a few examples of the signal processing techniques that can be interpreted as a type of composition. After discussing the potential advantages of utilizing composition structures for system representation in Section 1.2, the focus and contributions of this thesis are summarized in Section 1.3. Finally, the outline of this thesis is proposed in Section 1.4.

1.1 Types of Composition Structures

1.1.1 Univariate Function Composition

Function composition generally refers to the application of a function to the result of another function. In particular, for univariate functions $g(\cdot) : A \to B$ and $f(\cdot) : B \to C$, the composition $(f \circ g)(\cdot)$ is a function from the set A to the set C, which satisfies $(f \circ g)(x) \triangleq f(g(x))$ for all $x \in A$. To ensure proper definition, the domain of the outer function $f(\cdot)$ should include the image of the inner function $g(\cdot)$.

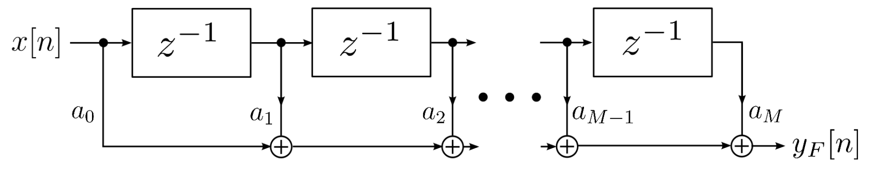
Function composition can be interpreted from the following two equivalent perspectives:

- For any $x \in A$, we obtain $v = g(x)$ and then obtain $y = f(v)$. In other words, we replace the input variable of $f(\cdot)$ with the output of $g(\cdot)$.

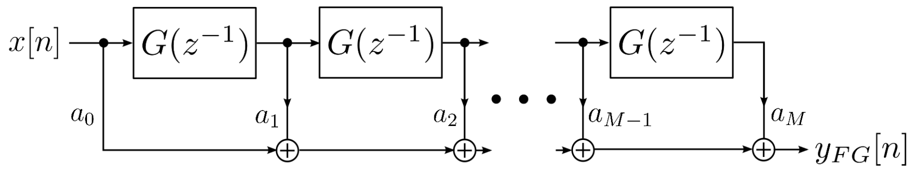
- The function $f(\cdot)$ corresponds to an operator $\mathcal{F}_{\text{lc}}$ that maps the function $g(\cdot)$ to a new function $(f \circ g)(\cdot)$; similarly, the function $g(\cdot)$ corresponds to an operator $\mathcal{G}_{\text{rc}}$ that maps $f(\cdot)$ to $(f \circ g)(\cdot)$. The latter operator $\mathcal{G}_{\text{rc}}$ is linear and named as the composition operator or the *Koopman* operator [19], which has been studied in fields such as nonlinear dynamic systems and flow analysis [20, 21].

The first interpretation will be used in Sections 1.1.2 and 1.1.3, while we will focus on the second interpretation in Section 1.1.4.

Univariate function composition may serve as a convenient description for a number of signal processing methods and system structures. Frequency-warping [22, 23] replaces the frequency variable in the spectrum with a warping function and thus achieves non-uniform spectrum computation with the standard Fast Fourier Transform algorithm. Moreover, time-warping [24, 25] applies a warping function to the time variable and can be used for efficient non-uniform sampling. As an example of system representations with a composition structure, as shown in Figure 1-1, if we replace each delay element[2] in a FIR filter $F(z^{-1})$ with another filter $G(z^{-1})$, then we obtain a *modular filter* with the transfer function as the composition of the transfer functions of the two filters $F(z^{-1})$ and $G(z^{-1})$. The design of such modular filters has been studied in [1, 2]. More detailed review on frequency-warping, time-warping, and modular filter design will be presented in Section 2.1 of this thesis.



(a): FIR filter $F(z^{-1})$



(b): Modular filter $(F \circ G)(z^{-1})$

Figure 1-1: Modular filter with transfer function as $(F \circ G)(z^{-1})$ [1, 2]

1.1.2 Operator Composition

Operator composition has the same core idea as univariate function composition, with the only difference that functions are substituted by operators. In particular, for operators $G\{\cdot\} : \mathbf{A} \to \mathbf{B}$ and $F\{\cdot\} : \mathbf{B} \to \mathbf{C}$, the composition of the two operators

[2]Here each delay element z^{-1} (rather than z itself) is considered as the variable of the transfer functions of FIR filters, in order that the transfer functions become polynomials.

is defined as $(F \circ G)\{\cdot\} \triangleq F\{G\{\cdot\}\}$.

In signal processing, a system can be considered as an operator that works on a set of signals of interest. Therefore, the cascade of two systems that respectively correspond to the operators $G\{\cdot\}$ and $F\{\cdot\}$ can be naturally represented by the operator $(F \circ G)\{\cdot\}$. The systems in the cascade can almost be arbitrary, as long as the system $F\{\cdot\}$ can take the output from the system $G\{\cdot\}$ as an input signal.

Cascade systems have been widely used and studied in signal processing. For linear systems, the cascade implementation of one-dimensional (1D) infinite impulse response (IIR) filters can improve the stability with respect to possible quantization errors in the filter coefficients [26]. For nonlinear systems, the block-oriented models are a useful and wide class of models that typically represents a nonlinear system by the interaction of linear time-invariant (LTI) blocks and nonlinear memoryless blocks [27], and the cascade nonlinear models are an important subclass of these block-oriented models. More detailed review on the block-oriented nonlinear models will be presented in Section 2.2.3 of this thesis.

1.1.3 Multivariate Function Composition

The composition of multivariate functions is a natural generalization of the univariate function composition. For multivariate functions $f(v_1, \cdots, v_R)$ and $g_r(x_1, \cdots, x_d)$ $(1 \leq r \leq R)$, the function $f(g_1(x_1, \cdots, x_d), \cdots, g_R(x_1, \cdots, x_d))$ is referred to as the *generalized composite* of f with $g_1, \cdots, g_R$ [28]. Since there could be multiple inner functions g_r, this multivariate function composition gains additional flexibility over its univariate counterpart.

There are a number of important functions in signal processing that have the form of multivariate composition. As an example, two-level Boolean functions in the disjunctive normal form (DNF, "OR-of-ANDs") or the conjunctive normal form (CNF, "AND-of-ORs") serve as a useful classification model and have wide applications in signal processing [29–31] and machine learning [32–34]. A Boolean function in DNF or CNF is a multivariate function with both the inputs and the output as binary variables, which has the following form of composition:

$$\hat{y} = F(G_1(x_1, \cdots, x_d), \cdots, G_R(x_1, \cdots, x_d)), \tag{1.1}$$

where $(x_1, \cdots, x_d)$ and $\hat{y}$ denote the input and output variables, respectively. For DNF, the functions $F(v_1, \cdots, v_R)$ and $G_r(x_1, \cdots, x_d)$ $(1 \leq r \leq R)$ in (1.1) are defined as

$$F(v_1, \cdots, v_R) = \bigvee_{r=1}^{R} v_r, \quad G_r(x_1, \cdots, x_d) = \bigwedge_{j \in \mathbf{X}_r} x_j, \tag{1.2}$$

where each $\mathbf{X}_r$ $(1 \leq r \leq R)$ is a subset of the index set $\{1, 2, \cdots, d\}$, and the symbols "$\vee$" and "$\wedge$" denote the logical "OR" (i.e. disjunction) and "AND" (i.e. conjunction), respectively. For CNF, we swap the logical "OR" and "AND", and the functions F and G_r $(1 \leq r \leq R)$ are defined similarly as follows.

$$F(v_1, \cdots, v_R) = \bigwedge_{r=1}^{R} v_r, \quad G_r(x_1, \cdots, x_d) = \bigvee_{j \in \mathbf{X}_r} x_j. \tag{1.3}$$

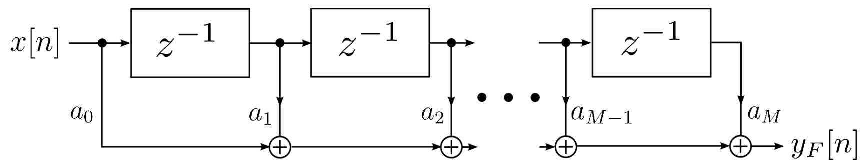
For simplicity, we refer to the individual functions $F(v_1, \cdots, v_R)$ and $G_r(x_1, \cdots, x_d)$ as *one-level* Boolean functions, and thus the two-level functions are composition of the one-level functions.

The two-level Boolean functions in DNF and CNF have a number of benefits. Since any Boolean function can be represented in DNF and CNF[3] [34], these two forms have high model richness and are widely used in applications such as binary classification [34] and digital circuit synthesis [35]. When the two-level Boolean functions are used in real-world classification problems where each input and output variable corresponds to a meaningful feature, the variables selected in each subset $\mathbf{X}_r$ in (1.2) and (1.3) can naturally serve as the *reason* for the predicted output; therefore, a two-level Boolean function model that is learned from a training dataset can be easily understood by the user, which may gain preference over black-box models, especially in fields such as medicine and law where it is important to understand the model [33, 36–38].

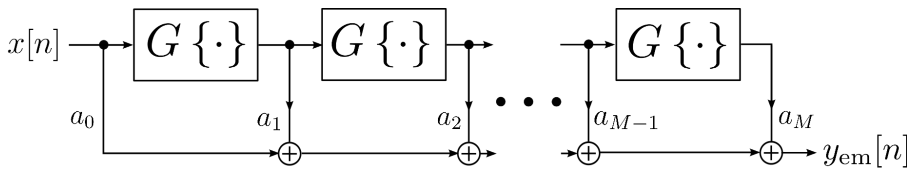
[3]We assume that the negation of each input variable is available.

1.1.4 Modular Composition

The composition forms in previous sections focus on the replacement of the input information of a function or an operator by the output from another function or operator. In contrast, this section considers a generalization of composition by replacing elements in a system block diagram with identical copies of another system module, which we refer to as *modular composition.* In another perspective, the latter system module is *embedded* into the former system. For tractability, we focus on the example of replacing each delay element of a FIR filter $F(z^{-1})$ in the direct form structure [26] with a time-invariant system $G\{\cdot\}$, as shown in Figure 1-2.



(a): FIR filter $F(z^{-1})$ in direct form structure



(b): Modular system

Figure 1-2: Embedding system $G\{\cdot\}$ into the FIR filter $F(z^{-1})$ in the direct form structure.

The modular system in Figure 1-2 (b) is a natural generalization of the modular filter in Figure 1-1 (b), by allowing the system $G\{\cdot\}$ to be a general nonlinear module such as the Volterra series model [39, 40].

Now we consider the mathematical description of the modular composition process. For simplicity, we denote the set of signals of interest as $\mathbf{S}$. The original subsystem $G\{\cdot\}$ is an operator that maps an input signal in $\mathbf{S}$ to an output in $\mathbf{S}$, i.e. $G\{\cdot\} : \mathbf{S} \to \mathbf{S}$. After embedding $G\{\cdot\}$ in $F(z^{-1})$, the modular system is another operator that maps from $\mathbf{S}$ to $\mathbf{S}$. Thus, if we define the set of all operators from $\mathbf{S}$ to

S as

$$\mathscr{U} \triangleq \{H\{\cdot\} : \mathbf{S} \to \mathbf{S}\}, \tag{1.4}$$

then the system $F(z^{-1})$ corresponds to a higher-level[4] operator $\mathcal{F}$ that maps from $\mathscr{U}$ to $\mathscr{U}$. In summary, the modular composition process generally has the following mathematical description: the system within which we embed another module creates a higher-level operator, which has both its input and output as lower-level operators on the set of signals; we refer to this description as *operator mapping*, on which further discussion can be found in Appendix A.

1.2 Potential Benefits of Composition Structures for System Representation

As briefly mentioned before, there are a number of potential advantages of the use of structures that can be interpreted as a type of composition for system representation purposes. In this thesis, we consider structures that have the following potential benefits: parameter reduction, modularity for hardware implementation, and improved model richness, which are illustrated as follows.

- **Parameter Reduction:** If a given signal or system can be represented or approximated by a composition structure that has fewer independent parameters than the original signal or system, then this composition structure achieves representational compression or computational efficiency. As an example, a 2D FIR filter can be represented or approximated by a cascade of a pair of 2D FIR filters that have lower orders. If the cascade system retains the same order as the original 2D filter, then the cascade system typically has fewer independent parameters compared with the original 2D filter [41]. We will discuss the cascade approximation of 2D FIR filters in Chapter 3. Similarly, approximating a univariate polynomial by the composition of two

[4]The operator $G\{\cdot\}$ works on the set of signals while $\mathcal{F}$ works on the set of operators, and therefore $\mathcal{F}$ can be considered as a higher-level operator while $G\{\cdot\}$ is lower-level.

polynomials with lower orders reduces the number of independent parameters and can potentially be applied to the compact representation of 1D signals, which is explored in [2, 42] and will be reviewed in Section 2.1 of this thesis. In addition, multiple layers of composition can achieve compact system structures by reducing the overall complexity. A class of examples is artificial neural networks that can be viewed as function composition of multiple layers, which is an efficient model for compact representations of a dataset and for classification tasks [43, 44]; as suggested by [45–47], reducing the number of layers in a neural network may significantly increase the total number of neurons that are needed to have a reasonable representation.

- **Modularity for Hardware Implementation:** The modular systems in Section 1.1.4 use identical copies of a system module, which thus has a structured representation and achieves modularity. This modular structure can simplify the design and verification for the hardware implementation with very-large-scale integration (VLSI) techniques [2, 48, 49]. As an example, we refer to a *modular Volterra system* as the system obtained by embedding a Volterra series module [39, 40] into a FIR filter, which will be studied in Chapter 5.

- **Improved Model Richness:** Taking the composition of simple parametric models of systems or functions may result in a new model that is able to represent a significantly larger class of systems or functions, i.e. the model richness is improved. As is discussed in Section 1.1.3, the one-level Boolean functions can model only a subset of all Boolean functions; in contrast, the two-level Boolean functions, which are the composition of one-level functions, can represent all Boolean functions if the negation of each input binary variable is available. Chapter 6 of this thesis considers algorithms to learn two-level Boolean functions from a training dataset. As another example for signal representation, the model of a bandlimited signal composed with a time-warping function can express signals that are in a significantly larger class than the bandlimited signals [24], which can enable efficient sampling methods for those signals.

1.3 Focus and Contributions

Parameter estimation is a central problem for the utilization of the benefits of composition structures for system representation. If we aim to use a composition structure to approximate a target system and signal, and if we choose the optimality criterion as the approximation error, then this performance metric is generally a nonlinear and potentially complicated function with respect to the parameters in the structure, which may make efficient parameter estimation challenging.

This thesis focuses on parameter estimation for composition structures. Since an efficient parameter estimation algorithm for general and arbitrary composition structures is unlikely to exist, this thesis proposes algorithms for the following classes of structures that have high importance for signal processing purposes: for the operator composition that corresponds to cascade systems, we consider the cascade approximation of 2D FIR filters and the block-oriented models for nonlinear systems; for modular composition, we propose the modular Volterra system; for multivariate function composition, we focus on algorithms to learn the two-level Boolean functions from a training dataset. We summarize the goals and the main contributions for each class of structures as follows.

- **Operator Composition:** As an example of cascade linear systems, we consider the approximation of a 2D FIR filter by the cascade of a pair of 2D FIR filters that have lower orders, which can achieve computational efficiency for spatial domain implementation. In the transform domain, the cascade approximation becomes approximate bivariate polynomial factorization, for which this thesis introduces new algorithms. Simulation results show that our new algorithm based on the idea of dimensional lifting of the parameter space outperforms the other methods in comparison. In addition, this technique can also be applied to the approximation of a 2D signal by the convolution of a pair of 2D signals with shorter horizontal and vertical lengths, which can result in compact representation of 2D signals.

For cascade structures of nonlinear systems, we consider the block-oriented representations of discrete-time nonlinear systems, where the goal is parameter estimation using statistics or empirical observations of the input and output signals. In particular, we focus on two structures. The first structure is a Hammerstein model [50] that is the cascade of a nonlinear memoryless module followed by a LTI subsystem, where the nonlinear module is a weighted combination over a basis of known functions and the LTI subsystem has no extra constraints. This setup is more general than the LTI subsystems considered in existing literature on the Hammerstein model estimation, which typically are constrained to be FIR filters [51] or filters with rational transfer functions [52]. We generalize the two-step parameter estimation method in [52] from a finite-dimensional to an infinite-dimensional parameter space. The second structure for nonlinear system is used for modeling a black-box nonlinear system by its inverse, where the inverse system is a cascade of multiple nonlinear functions and LTI subsystems. This second structure can be considered as a generalization of the all-pole signal modeling [26] by introducing nonlinear blocks.

- **Modular Composition:** The modular Volterra system is obtained by replacing each delay element in a FIR filter with a Volterra series module. If the Volterra series module has only the linear kernel, then the resulted modular system belongs to the class of modular filters in [1, 2]. In addition to modularity, the incorporation of nonlinear kernels in the Volterra series module has the additional benefit of capturing nonlinear effects and providing more flexibility over modular filters. In this thesis, we consider parameter estimation for the modular Volterra system using the statistical information between the input and output signals. In particular, we focus on the situation where the coefficients of the nonlinear kernels of the Volterra module have sufficiently smaller magnitude compared with those of the linear kernel, i.e. weak nonlinear effects. An estimation algorithm is provided by first obtaining the coefficients of the linear kernel and then the nonlinear kernels, which is shown to be effective by numerical evaluation when the order and the number of states (i.e. the maximum delay) of the system are not high.

- **Multivariate Function Composition:** In contrast to the above systems that have continuous-valued input and output signals, the two-level Boolean functions mentioned in Section 1.1.3 have binary input and output variables. Our goal is to learn two-level Boolean functions in the CNF or DNF [53] for classification purposes, with the joint criteria of accuracy and function simplicity. We propose a unified optimization framework with two formulations, respectively with the accuracy characterized by the 0-1 classification error and a new Hamming distance cost. By exploring the composition structure of the two-level Boolean functions, we develop linear programming relaxation, block coordinate descent, and alternating minimization algorithms for the optimization of the formulations. Numerical experiments show that two-level functions can have considerably higher accuracy than one-level functions, and the algorithms based on the Hamming distance formulation obtain very good tradeoffs between accuracy and simplicity.

In summary, Figure 1-3 shows the framework of this thesis.

1.4 Outline of Thesis

The remainder of this thesis is organized as follows. After a high-level review of the related concepts and background in Chapter 2, we consider each class of systems as mentioned above. For cascade structures that correspond to operator composition, the cascade approximation of 2D FIR filters is presented in Chapter 3, and the block-oriented cascade models for nonlinear systems are studied in Chapter 4. For modular composition, the modular Volterra systems are introduced in Chapter 5. For multivariate function composition, learning two-level Boolean functions in CNF or DNF is considered in Chapter 6. In each chapter, the concrete form and the

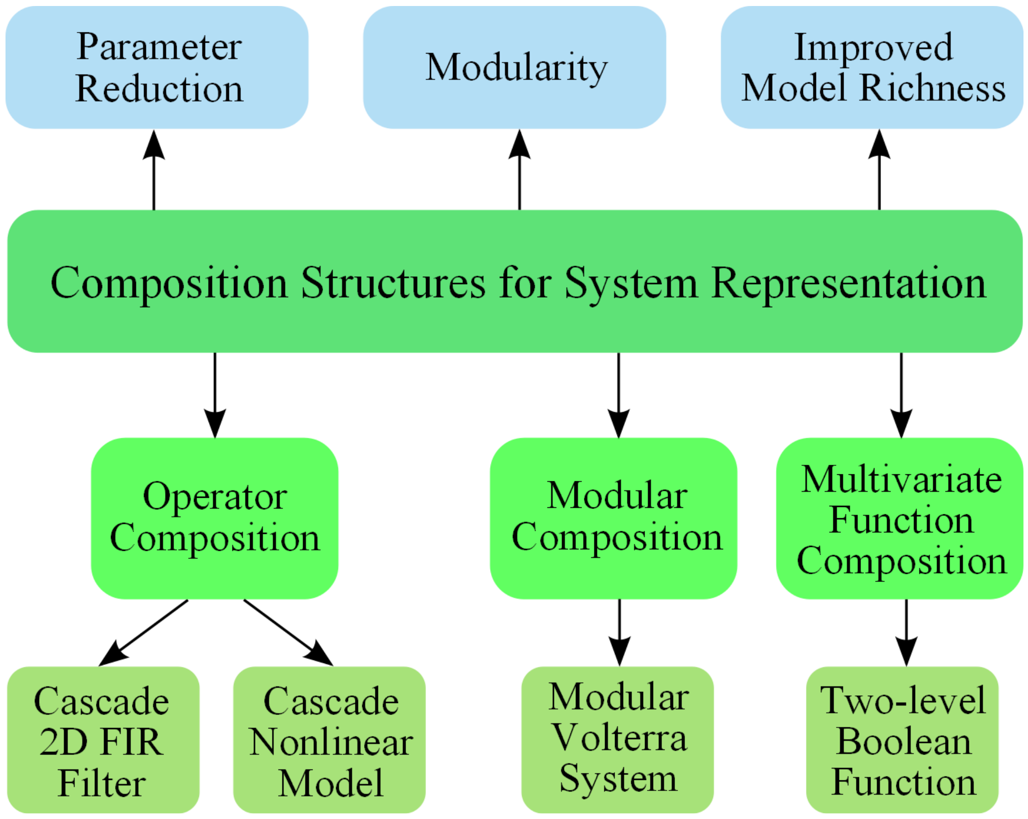


Figure 1-3: Framework of this thesis.

applications of the composition structures are presented, the parameter estimation problem is formulated, related existing work is reviewed, new parameter estimation algorithms are proposed, and simulation results are shown to evaluate the algorithms. Finally, conclusion and future work are provided in Chapter 7.

Chapter 2

General Background

This chapter provides a brief review of fundamental concepts and existing works on a few important topics that are related to this thesis. Some of the reviewed techniques and system structures will be further discussed in later chapters, where a more detailed review will be presented for specific problems. Section 2.1 of this chapter reviews existing signal processing techniques and system structures that can be interpreted as a type of composition. Since both the Volterra series [39] and the block-oriented models [27] for nonlinear systems will be heavily used in later chapters of this thesis, Section 2.2 provides a brief review on these models for nonlinear systems and the associated parameter estimation approaches.

2.1 Systems with a Composition Structure

First, we review the work [2] that is the closest work to this thesis. In [2], Demirtas introduces the first work that systematically studies function composition and decomposition for signal processing purposes, which are applied to the interpretation of existing techniques and the development of new algorithms, from both analysis and synthesis perspectives. The utilization of function composition and decomposition provides the benefits of compact representation and sparsity for signals, as well as modularity and separation of computation for system implementation. In particular, there are three main focuses in [2], namely univariate polynomial decomposition, the design of modular filters by frequency response composition, and the representation of multivariate discrete-valued functions with a composition structure, each of which will be reviewed as below.

- **Univariate Polynomial Decomposition:** Univariate polynomial decomposition generally refers to the determination of a pair of polynomials $f(x)$ and $g(x)$ so that their composition $f(g(x))$ equals or approximates a given polynomial $h(x)$. Approximating a given polynomial $h(x)$ by the composition $f(g(x))$ generally reduces the total number of independent parameters. Since the z-transform of a 1D signal is a polynomial, polynomial decomposition can be used for compact representation of 1D signals. A challenge with polynomial decomposition is the highly nonlinear dependence of the coefficients of $h(x)$ on those of $g(x)$. Practical decomposition algorithms are proposed in [2, 42, 54] and compared with existing works [12–14, 55], for both the exact decomposition where the given polynomial $h(x)$ is guaranteed decomposable and the approximate decomposition where a decomposable polynomial is used to approximate a given indecomposable polynomial. These algorithms can work with either the coefficients or the roots of the given polynomial $h(x)$ as the input information. As shown by [14], univariate polynomial decomposition can be converted to bivariate polynomial factorization, where the latter is related to Chapter 3 of this thesis. In addition to decomposition algorithms, sensitivity of polynomial composition and decomposition is studied in [2,56] by characterizing the robustness of these two processes with respect to small perturbations in the polynomials; the sensitivity is defined as the maximum magnification ratio of the relative perturbation energy of the output polynomial over that of the input polynomial. A method of reducing the sensitivity by equivalent decomposition has been proposed in [2, 56], which improves the robustness for polynomial composition and decomposition.

- **Modular Filters by Frequency Response Composition:** As mentioned in Section 1.1.1, modular filters are obtained by replacing each delay element in an *outer* discrete-time filter with an identical copy of an *inner* filter module; the frequency response of the modular filter is the composition of the transfer function of the outer discrete-time filter and the frequency response of the inner filter module. The modular filter in [1, 2] can be potentially considered as a systematic generalization of the filter sharpening technique [57], which uses multiple copies of a low quality filter to improve the quality of approximation for a given specification. In particular, two choices of the inner filter module are considered in [2]: the first is a linear phase discrete-time FIR filter, and the second is a continuous-time filter. With the optimality criterion as the minimax error in the magnitude response, algorithms are proposed for the modular filter design for both choices above. The modular filter design will be further discussed in Section 5.2.

- **Multivariate Discrete-valued Functions with a Composition Structure:** The third focus of [2] is the composition and decomposition of multivariate discrete-valued functions. Using proper composition can reduce representational complexity and simplify the computation for the marginalization process, where the marginalization process has applications in topics such as nonlinear filtering and graphical models in machine learning and statistical inference. In particular, the work [2] proposes a composition form for multivariate functions by introducing a latent variable that summaries a subset of the input variables. In this framework, a function $f(x_1, \cdots, x_n)$ is represented as $\tilde{f}(x_1, \cdots, x_m, u)$ where the latent variable $u = g(x_{m+1}, \cdots, x_n)$ $(1 \leq m < n)$. If the alphabet size of u is sufficiently small, then representational efficiency is gained since the functions $\tilde{f}$ and g have fewer parameters than the original function f; similarly, computational efficiency is achieved for marginalization. A generalization of the above composition form is proposed in [2], where a low rank approximation is used for a matrix representation of the multivariate function, which may also potentially reduce the number of independent parameters.

In addition to the topics that are explored in [2], there are a number of other techniques that can be interpreted from the perspective of composition structures. Here we list a few examples of such techniques, some of which have already been briefly mentioned in Chapter 1.

- **Time-warping and Efficient Sampling:** As briefly mentioned in Section 1.1.1, replacing the time-variable of a signal $f(t)$ with a function $w(t)$ yields a composed function $g(t) = f(w(t))$, where $w(t)$ is monotonic and can be considered as a time-warping function. If the signal $f(t)$ is non-bandlimited, then a periodic sampling process will lead to information loss and cannot recover $f(t)$ without additional information. However, for certain signals $f(t)$, there may exist a proper time-warping function $w(t)$ such that $f(w(t))$ becomes bandlimited, which thus enables periodic sampling without information loss. The work [24, 25] explores the signal representation technique with a proper time warping before periodic sampling, where the time warping function is also estimated from the input signal and aims to reduce the out-of-band energy before sampling.

- **Frequency Transformation and Filter Design:** A filter design problem can have the specification in a nonlinear scale of the frequency variable; as an example, the audio equalizer may have a logarithm scale specification [23]. If we convert the specification to a linear scale of frequency, then there are potential sharp ripples in the target frequency response where the frequency is compressed. This can lead to a high order of the designed filter and a lack of robustness in implementation. A possible solution to the problem above is the frequency warping technique [23], which approximates the nonlinear scale of frequency axis by replacing the delay elements in a filter with a proper system. This frequency warping process prevents the shape ripples in the target frequency response, leading to lower orders of the designed filters and higher implementation robustness. As briefly mentioned in Section 1.1.1, this idea of frequency warping has also been applied to the non-uniform spectrum computation with the standard Fast Fourier Transform algorithm [22]. In addition to the above warping function that rescales the frequency axis, there are other frequency transformation functions that map the low frequency region to other frequency intervals; therefore, by the composition with a proper frequency transformation function, a prototype lowpass filter can be utilized to design a highpass or a bandpass filter with flexible choices of band-edges [58, 59].

- **Efficient Representation of Nonlinear Systems:** For certain autonomous dynamical systems that have a nonlinear evolution on the state variables, there can be a simplified description of the system by utilizing proper function composition [60]. In particular, we consider an autonomous discrete-time dynamical system with the evolution as $g : \mathbf{V} \to \mathbf{V}$ where $\mathbf{V} \subseteq \mathbb{R}^N$ is the set of all possible state variables. If $\mathbf{x}[n]$ is the state variable at time n, then the system satisfies $\mathbf{x}[n+1] = g(\mathbf{x}[n])$. For any *observable* (i.e. a function of the state variables) $f : \mathbf{V} \to \mathbb{C}$, we consider the Koopman operator $\mathcal{G}$ such that $\mathcal{G}\{f\}(\cdot) = (f \circ g)(\cdot)$, which is a linear operator that forms function composition. If we interpret $f(\mathbf{x}[n])$ as the observation at time n, then $\mathcal{G}$ maps the *function from the current state to current observation* to the *function from the current state to the next observation*. For certain dynamical systems, the evolution of the state variables $\mathbf{x}[n]$ has a simplified description by using a special set of observables, which bypasses the potentially complicated nonlinear function $g(\cdot)$. This simplified description has the following steps [60]. First, we consider the eigenfunctions $f_k(\mathbf{x})$ of the Koopman operator $\mathcal{G}$, namely

$$\mathcal{G}\{f_k\}(\cdot) = \mu_k \cdot f_k(\cdot),\ 1 \leq k \leq K,$$

where μ_k are the singular values, and K can be either finite or infinite. This equation is named the Schröder's equation [61], which along with other similar equations involving function composition has been well-studied in mathematics

[18]. Then, with the assumption that the identical function[1] $\mathbf{I}(\mathbf{x}) = \mathbf{x}$ is in the range of the function space spanned by $f_k(\mathbf{x})$, i.e.

$$\mathbf{x} = \mathbf{I}(\mathbf{x}) = \sum_{k=1}^{K} \mathbf{v}_k \cdot f_k(\mathbf{x}),\ \forall \mathbf{x} \in \mathbf{V},$$

where $\mathbf{v}_k$ $(1 \leq k \leq K)$ are fixed vectors in $\mathbf{V}$, the evolution of the system can be equivalently described in the following expansion

$$g(\mathbf{x}) = \mathcal{G}\left\{\mathbf{I}\right\}(\mathbf{x}) = \mathcal{G}\left\{\sum_{k=1}^{K} \mathbf{v}_k \cdot f_k\right\}(\mathbf{x}) = \sum_{k=1}^{K} \mathbf{v}_k \cdot \mathcal{G}\left\{f_k\right\}(\mathbf{x}) = \sum_{k=1}^{K} \mathbf{v}_k \cdot \mu_k \cdot f_k(\mathbf{x}).$$

This expansion of the state evolution $g(\cdot)$ can be considered as a simplified representation of the system, since the dynamic in the space of the observables $f_k(\cdot)$ $(1 \leq k \leq K)$ is linear; as a result, the state at any time n $(n \geq 0)$ with the initial state $\mathbf{x}[0]$ at time 0 can have the following simple expression

$$\mathbf{x}[n] = g^{[n]}(\mathbf{x}[0]) = \sum_{k=1}^{K} \mathbf{v}_k \cdot \mu_k^n \cdot f_k(\mathbf{x}[0]), \quad n \geq 0,$$

where $g^{[n]}(\cdot)$ denotes the n^{th} iterate of the function $g(\cdot)$. The above expression converts the nonlinear and potentially complicated evolution of the state variables into a weighted combination of the fixed functions $f_k(\cdot)$, where the weights has an exponential relationship with the time. In the study of the Koopman operator, the three elements μ_k, $f_k(\cdot)$, and $\mathbf{v}_k$ have the terminologies of Koopman eigenvalues, eigenfunctions, and modes, respectively [60]. The estimation of these three elements is the critical step for utilizing the above efficient representation of nonlinear systems. With various setup of available information, there have been a number of works on the estimation of the three elements or a subset of them, such as the generalized Laplace analysis [62,63] and the dynamic mode decomposition with its extensions [60,64,65]. In addition to

[1]Since the identical function has a vector output, we can consider each scalar element of the output and then stack them into a vector.

discrete-time systems, this representation technique with the Koopman operator has also been generalized to continuous-time systems [62, 66].

2.2 Models for Nonlinear Systems

Since nonlinear systems are defined by the lack of linearity, there is no unified parametric description for a general nonlinear system. In contrast, various models for different classes of nonlinear systems have been proposed. This section reviews three widely used models, namely the Volterra series [39], the Wiener series [67], and the block-oriented models [27].

2.2.1 Volterra Series Model

The Volterra series model [39, 40, 68, 69] is applicable to represent a wide class of nonlinear systems. For a discrete-time time-invariant system, the Volterra series model represents each output sample $y[n]$ as a series expansion of the input samples $x[n]$:

$$y[n] = H\left\{x[n]\right\} = \sum_{k=0}^{\infty} H^{(k)}\left\{x[n]\right\}, \tag{2.1}$$

where $H^{(k)}\left\{\cdot\right\}$ is the k^{th}-order operator of the Volterra system, which satisfies

$$H^{(k)}\left\{x[n]\right\} = \sum_{i_1, i_2, \ldots i_k} h^{(k)}[i_1, i_2, \ldots, i_k] \cdot x[n-i_1] \cdot x[n-i_2] \cdots x[n-i_k], \tag{2.2}$$

where the fixed discrete-time function $h^{(k)}[i_1, i_2, \ldots, i_k]$ is referred to as the k^{th}-order Volterra *kernel*. If the system is causal, the kernels $h^{(k)}[i_1, i_2, \ldots, i_k]$ are nonzero only if all $i_q \geq 0$ $(1 \leq q \leq k)$.

There are multiple perspectives to interpret the Volterra series model. If the system is memoryless, then the Volterra series recovers the Taylor series; therefore, the Volterra series could be regarded as a generalization of the Taylor series with memory effects. While the Taylor series can represent a wide class of functions, the Volterra series can be considered as a representation of operators that map the input signal to the output signal. The Taylor series may be inefficient and have slow convergence if there is discontinuity in the function or its derivatives; similarly, the Volterra series is more effective to model systems if the input-output relationship is smooth and if the nonlinear properties can be effectively captured in low order terms.

The k^{th}-order operator $H^{(k)}\{\cdot\}$ in the Volterra series model is k^{th}-order homogeneous with respect to scaling of the input signal. In other words, if we replace $x[n]$ by $c \cdot x[n]$, then the output term satisfies $H^{(k)}\{c \cdot x[n]\} = c^k \cdot H^{(k)}\{x[n]\}$. The Volterra series model can be regarded as a generalization of the linear system; for a linear system, the first-order Volterra kernel is the same as the impulse response of the system, and kernels of the other orders are all zero.

From another perspective with a dimensionally lifted signal space, the Volterra series represents the output of the nonlinear system as the summation across all the orders of the diagonal results from the multi-dimensional linear filtering of the kernels and the outer-products of the input signal. More specifically, we can construct the outer-products of the input signal in the k-dimensional space as

$$\tilde{x}^{(k)}[n_1, n_2, \dots, n_k] \triangleq \prod_{q=1}^{k} x[n_q]. \tag{2.3}$$

If we filter this signal with the k^{th}-order kernel $h^{(k)}[i_1, i_2, \dots, i_k]$ and then take the diagonal elements, we obtain the term $H^{(k)}\{x[n]\}$ in (2.2). Finally, as shown by (2.1), the output signal $y[n]$ is the summation of $H^{(k)}\{x[n]\}$ across all orders $k \geq 0$. From this perspective, the kernel $h^{(k)}[i_1, i_2, \dots, i_k]$ could be regarded as a generalized impulse response in the k-dimensional space for the Volterra system. Moreover, it is clear that the output signal is linearly dependent on the kernels $h^{(k)}[i_1, i_2, \dots, i_k]$.

By constraining both a finite order and a finite delay (i.e. a finite number of state variables), the Volterra model becomes simplified and more practical, which can be expressed as

$$\hat{y}[n] = \sum_{k=0}^{K} \sum_{i_1=0}^{N} \sum_{i_2=0}^{N} \cdots \sum_{i_k=0}^{N} h^{(k)}[i_1, i_2, \dots, i_k] \cdot x[n-i_1] \cdot x[n-i_2] \cdots x[n-i_k], \tag{2.4}$$

where K and N are the highest order and the maximum time delay (i.e. the number of state variables), respectively. By increasing K and N in (2.4), this truncated Volterra series model can approximate a nonlinear system $y[n] = H_{\mathrm{NL}}\{x[n]\}$ with any specified accuracy, as long as the nonlinear system satisfies the following constraints [70]:

- The operator $H_{\mathrm{NL}}\{\cdot\}$ is causal, time-invariant, continuous, and has fading memory as defined in [70];
- The input signal $x[n]$ is upper and lower bounded.

Despite its high model richness, a disadvantage of the Volterra model results from the large number of parameters in the kernels. If we fix the maximum delay of the system, then the number of parameters in the k^{th}-order kernel is exponential with respect to k. Therefore, these kernels may require sufficiently long observed signals in order to avoid overfitting for the parameter estimation.

The Volterra series model can also be applied to representing continuous-time nonlinear systems. For a time-invariant system, the output signal $y(t)$ can be expressed in terms of the input signal $x(t)$ and the Volterra kernels $h^{(k)}(t_1, t_2, \ldots, t_k)$ in the form below:

$$\begin{aligned} y(t) &= \sum_{k=0}^{\infty} H^{(k)}\{x(t)\}, \text{ where} \\ H^{(k)}\{x(t)\} &= \int \cdots \int \Big(h^{(k)}(t_1, t_2, \ldots, t_k) \cdot x(t-t_1) \cdot x(t-t_2) \cdots x(t-t_k) \Big) \mathrm{d}t_1 \mathrm{d}t_2 \cdots \mathrm{d}t_k. \end{aligned}$$

We can see that the summations in (2.2) for discrete-time systems are replaced by the integrations in the equation above for continuous-time systems.

A number of techniques have been developed to estimate the kernels in both the discrete-time and continuous-time Volterra models, with the tools of the higher-order correlations, orthogonal expansions, and frequency domain methods [71]. Here is a brief review on a few existing kernel estimation techniques. In addition to the following techniques, the Volterra series model can be reorganized into the Wiener series model [67] that is reviewed in Section 2.2.2, for which there are other kernel estimation approaches.

For a discrete-time Volterra system with a finite order and a finite delay, the output signal has linear dependence on the kernels. As a result, if the higher-order statistics [72] of the input and output signals are available, then the minimization of the power of the estimation error in the output can be formulated as a linear regression problem, and thus the kernels can be estimated by the least squares solution. Despite the simplicity of this approach, it may require heavy computation when the order or the delay is big. Generally, the kernels of different orders in the Volterra series are correlated with each other, and therefore all of the kernels need to be estimated simultaneously by solving a set of high-dimensional linear equations.

For continuous-time Volterra systems, the minimization of the mean squared error in the output involves integration equations that are generally challenging to solve directly. A special situation where the integration equations have a straightforward solution is Volterra series with kernels only up to the second order and with white Gaussian input signals [71]. An approach for general continuous-time Volterra kernel estimation is to expand the kernels on an orthogonal function basis and then estimate the expansion coefficients [67,71], where the latter step can be achieved by approaches such as gradient-based techniques [73, 74] and pattern recognition methods [75]. As an alternative approach for kernel estimation, if we consider the Laplace or Fourier transform of the kernels, then the associated multi-dimensional transfer functions can be estimated with the higher-order spectra between the input and output signals [76,77], and efficient transform algorithms can be applied to reduce the computational complexity [78].

2.2.2 Wiener Series Model

The kernels of the Volterra series can be reorganized in order to achieve mutual orthogonality. Here the orthogonality is in the statistical sense and considers the cross-correlation between different terms in the series expansion of the output signal, where the input signal is a stochastic process with a specified probability distribution and autocorrelation properties. As a famous example, the Wiener series [67, 69, 79] has the kth term[2] $W^{(k)}\{\cdot\}$ as a linear combination of kernels with orders no higher than k. If the input signal $x[n]$ is a white Gaussian process with mean zero and variance σ_x^2, then the kth Wiener term $W^{(k)}\{\cdot\}$ is orthogonal to *any* homogeneous Volterra kernel $H^{(m)}\{\cdot\}$ where $m < k$, namely

$$\mathbb{E}\left\{W^{(k)}\{x[n]\} \cdot H^{(m)}\{x[n]\}\right\} = 0, \quad \text{for } m < k \text{ and white Gaussian process } x[n]. \tag{2.5}$$

Since $W^{(m)}\{x[n]\}$ is a linear combination of kernels with orders no higher than m, this orthogonality property (2.5) automatically guarantees that two Wiener terms with different orders are orthogonal. For discrete-time systems, the Wiener terms of the lowest few orders have the following forms:

$$\begin{aligned}
W^{(0)}\{x[n]\} &= w^{(0)}, \\
W^{(1)}\{x[n]\} &= \sum_{i_1} w^{(1)}[i_1] \cdot x[n-i_1], \\
W^{(2)}\{x[n]\} &= \sum_{i_1,i_2} w^{(2)}[i_1,i_2] \cdot x[n-i_1] \cdot x[n-i_2] - \sigma_x^2 \cdot \sum_{i_1} w^{(2)}[i_1,i_1], \\
W^{(3)}\{x[n]\} &= \sum_{i_1,i_2,i_3} w^{(3)}[i_1,i_2,i_3] \cdot x[n-i_1] \cdot x[n-i_2] \cdot x[n-i_3] \\
&\quad -3 \cdot \sigma_x^2 \cdot \sum_{i_1,i_2} x[n-i_1] \cdot w^{(3)}[i_1,i_2,i_2],
\end{aligned}$$

where $w^{(k)}[i_1,\cdots,i_k]$ is named as the kth-order Wiener kernel [67, 79] and we assume that each kernel is a symmetric function of its arguments, i.e. swapping any two arguments does not change the value of the kernel. From these examples, it can be observed that the kth Wiener term $W^{(k)}\{\cdot\}$ includes the kth-order Wiener kernel $w^{(k)}[i_1,\cdots,i_k]$ as well as lower order kernels that are obtained by systematically marginalizing $w^{(k)}[i_1,\cdots,i_k]$ into dimension m, where m is smaller than k and has the same parity as k. For a general Wiener term $W^{(k)}\{\cdot\}$, the coefficients can be obtained from the Hermite polynomials [67, 79]. By replacing the summations with integrations, the Wiener series can also represent continuous-time nonlinear systems.

Since the symmetry of the Gaussian distribution ensures that all odd order

[2]The terms $W^{(k)}\{\cdot\}$ are referred to as the Wiener G-functionals [67, 79].

moments are zero, the orthogonality property (2.5) holds if m and k have different parity. When m and k have the same parity, the orthogonality property can be shown with the properties of the high order moments of Gaussian variables. As an example, we show the orthogonality property (2.5) for the situation when $k = 3$ and $m = 1$:

$$\begin{aligned}
& \mathbb{E}\left\{W^{(3)}\{x[n]\} \cdot H^{(1)}\{x[n]\}\right\} \\
= & \sum_{j_1} w^{(3)}[j_1, j_1, j_1] \cdot h^{(1)}[j_1] \cdot \mathbb{E}\left\{(x[n-j_1])^4\right\} \\
& + \sum_{i_2 \neq j_1} \left(w^{(3)}[i_2, i_2, j_1] + w^{(3)}[i_2, j_1, i_2] + w^{(3)}[j_1, i_2, i_2]\right) \cdot h^{(1)}[j_1] \cdot \mathbb{E}\left\{(x[n-i_2])^2\right\} \cdot \mathbb{E}\left\{(x[n-j_1])^2\right\} \\
& -3 \cdot \sigma_x^2 \cdot \sum_{i_1, i_2} w^{(3)}[i_1, i_2, i_2] \cdot h^{(1)}[i_1] \cdot \mathbb{E}\left\{(x[n-i_1])^2\right\} \\
= & \sum_{j_1} w^{(3)}[j_1, j_1, j_1] \cdot h^{(1)}[j_1] \cdot 3 \cdot \sigma_x^4 + 3 \cdot \sum_{i_2 \neq j_1} w^{(3)}[j_1, i_2, i_2] \cdot h^{(1)}[j_1] \cdot \sigma_x^4 - 3 \cdot \sum_{i_1, i_2} w^{(3)}[i_1, i_2, i_2] \cdot h^{(1)}[i_1] \cdot \sigma_x^4 \\
= & \ 0.
\end{aligned}$$

where we apply the property $\mathbb{E}\{x^4\} = 3 \cdot (\mathbb{E}\{x^2\})^2$ for a Gaussian variable x with zero mean.

With mutual orthogonality, each kernel can be estimated independently. Therefore, if we increase the order of the Wiener series, the lower order terms remain the same and do not need to be updated; however, we should notice that in general the m^{th}-order Wiener kernel from $W^{(m)}\{\cdot\}$ does not fully determine the m^{th}-order homogeneous kernel of the nonlinear system, since the term $W^{(k)}\{\cdot\}$ with $k > m$ can contribute to the m^{th}-order homogeneous kernel. An approach for the estimation of the Wiener kernels is to further expand each kernel onto an orthogonal function basis and then to estimate the expansion coefficients [67, 69]. As an example, if we use the multi-dimensional Laguerre polynomials as the orthogonal basis for a continuous-time Wiener series model, then the estimation for each expansion coefficient has the following steps [67]. First, an auxiliary nonlinear system is created with the kernel as the Wiener G-functional applied to a function in the orthogonal basis. Then, the input signal is processed with this auxiliary nonlinear system. Finally, the expansion coefficient is the correlation between the output of this auxiliary system and the true system output. For this approach, the synthesis of the auxiliary systems may require excessive resources.

A widely-used approach for the Wiener kernel estimation is the Lee-Schetzen method and its variants [79–82], which uses the cross-correlation between the true system output $y[n]$ and the delayed input signals. If the true system exactly satisfies the Wiener series model, then as is shown in [79], the coefficients of the k^{th}-order Wiener kernel can be estimated as follows

$$w^{(k)}[i_1, \cdots, i_k] = \frac{1}{k! \cdot \sigma_x^{2k}} \cdot \mathbb{E}\left\{ \left(y[n] - \sum_{m=0}^{k-1} W^{(m)} \{x[n]\} \right) \cdot x[n-i_1] \cdots x[n-i_k] \right\}.$$

In addition, if the indices $i_1, i_2, \cdots, i_k$ are mutually different, then the result above can be further simplified as [79]

$$w^{(k)}[i_1, \cdots, i_k] = \frac{1}{k! \cdot \sigma_x^{2k}} \cdot \mathbb{E}\left\{ y[n] \cdot x[n-i_1] \cdots x[n-i_k] \right\}.$$

More rigorous analysis and deeper discussion about the applicability of the Lee-Schetzen approach can be found in [80, 81].

2.2.3 Block-oriented Models

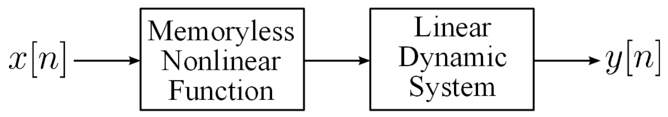
The block-oriented models [27, 50, 67] represent a nonlinear system by the interaction between two types of blocks, namely the memoryless nonlinear functions and the linear time-invariant (LTI) systems. Compared with the Volterra series model, the block-oriented models typically compromise model richness to reduce the total number of independent parameters, which potentially simplifies the process of system identification. Another potential advantage of the block-oriented models is that each block in the structure may explicitly correspond to a physical property of the device or system that is modeled [27], which can result in improved model accuracy and physical interpretability.

As the most fundamental structures in this model class, the Hammerstein model [50] as shown in Figure 2-1 (a) is the cascade of a nonlinear memoryless block followed by a LTI subsystem, and the Wiener model [67] as shown in Figure 2-1 (b) is the cascade of the same blocks in the reverse order. Although relatively simple, the Hammerstein and Wiener models are able to accurately capture the nonlinear properties and dynamics in systems such as power amplifiers [83, 84], chemical processes [85,86], and biological systems [87].

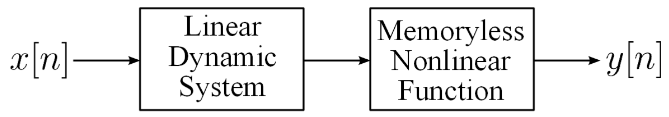
If the nonlinear functions in the Hammerstein or Wiener models are polynomials, then the associated Hammerstein or Wiener models can also be represented by Volterra series. However, there are nonlinear systems that can be represented by Volterra series but not by the Hammerstein or Wiener models.

Using these two fundamental models, more general structures include the Hammerstein-Wiener model in Figure 2-1 (c) that has a LTI system between two memoryless nonlinear functions, and the Wiener-Hammerstein structure in Figure 2-1 (d) with a nonlinear function between two LTI systems. In addition, parallel structures where each branch is one of the models above have also been proposed [88,89]. In particular, any discrete-time Volterra or Wiener series model with a finite delay and a finite order can be represented as the General Wiener Model [67, 69], which is a cascade of three blocks: the first block is a single-input multi-output linear system with memory, the second block is a multi-input multi-output nonlinear memoryless system with adders and multipliers, and the third block is a multi-input single-output system where the output is a weighted combination of its inputs.

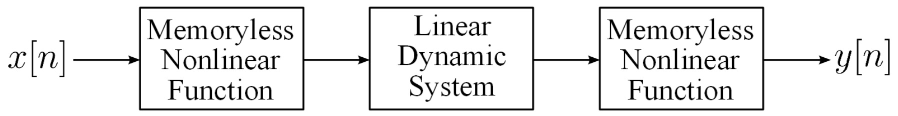
Parameter estimation for the block-oriented models has attracted considerable interest in nonlinear signal processing, and various branches of methods under different conditions have been proposed [27]. The first branch utilizes the time domain stochastic properties of the input signal, such as the joint Gaussian distribution or the *invariance property*, which disentangle the estimation of the nonlinear function and the LTI subsystem [90, 91]. The second branch that is based on the frequency domain probes the Hammerstein system by a pure sinusoid input signal, where the frequency of the signal is swept to fully characterize the system [92]. The third branch applies inference and machine learning techniques, such as the maximum likelihood estimation [93] and the support vector machines [94, 95]. The fourth branch includes blind methods for parameter estimation, where the output signal is observed but not the input [96, 97].



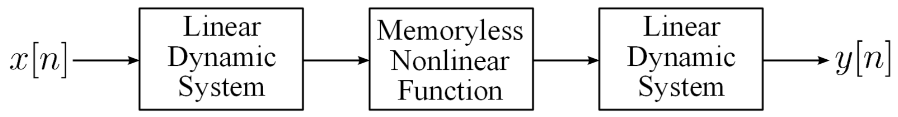
(a) Hammerstein model



(b) Wiener model



(c) Hammerstein-Wiener model



(d) Wiener-Hammerstein model

Figure 2-1: Fundamental block-oriented models for nonlinear systems.

Chapter 3

Cascade Approximation of Two-dimensional FIR Filters

The cascade structure for filter implementation has been important and well-studied for both one-dimensional (1D) and two-dimensional (2D) filters [26,41], which can be described by the operator composition in Section 1.1.2. This chapter considers the approximation of a 2D finite impulse response (FIR) filter by the cascade of a pair of 2D FIR filters of lower orders, which can reduce the total number of independent parameters and lead to computational efficiency for spatial domain implementation [41]. In addition, if we consider a 2D signal as the impulse response of a 2D filter, then the cascade approximation of a 2D filter is equivalent to the approximation of a 2D signal by the convolution of a pair of 2D signals with shorter horizontal and vertical lengths, which can achieve representational compression for 2D signals. Since the transfer function of a 2D FIR filter is a bivariate polynomial, the cascade approximation of a 2D FIR filter corresponds to approximate bivariate polynomial factorization, which is the focus of this chapter.

Bivariate polynomial factorization has fundamental differences from univariate polynomial factorization. Every univariate polynomial is guaranteed factorizable into degree-one factors with complex coefficients; in contrast, almost all bivariate polynomials are not exactly factorizable into polynomials with strictly lower degrees, which shows the importance of approximate factorization. If we ensure that the order[1] of the cascade system equals that of the original filter, then the cascade form of a 1D filter retains the same number of independent parameters as the original 1D filter, while a cascade representation of a 2D filter can reduce the total number of independent parameters.

The main contribution of this chapter includes the proposal of two new algorithms for bivariate polynomial factorization, which we refer to as the zero-sum mixed integer programming and the lifted alternating minimization. In addition, the robustness of bivariate polynomial factorization is characterized with respect to perturbations in the factors. The content of this chapter is as follows. Section 3.1 motivates this work by showing the cascade representation yields parameter reduction for a 2D FIR filter, and then the main problem is formulated in Section 3.2. After reviewing existing algorithms for bivariate polynomial factorization in Section 3.3, Section 3.4 proposes and evaluates the two new algorithms. A method is then proposed in Section 3.5 in order to retain certain desirable symmetric properties of the 2D filters in the factorization, and the sensitivity of the factorization process is studied in Section 3.6 in order to characterize the robustness. Section 3.7 concludes this chapter and proposes future work.

3.1 Motivation

We denote the transfer function of a 2D causal FIR filter with order (P, Q) as

$$H(z_1, z_2) = \sum_{k_1=0}^{P} \sum_{k_2=0}^{Q} h[k_1, k_2] \cdot z_1^{-k_1} \cdot z_2^{-k_2},$$

which has $(P+1) \cdot (Q+1)$ parameters in total.

If $H(z_1, z_2)$ could be exactly represented or reasonably approximated by the cascade of a pair of 2D FIR filters with transfer functions as $F(z_1, z_2)$ and $G(z_1, z_2)$, i.e. $H(z_1, z_2) \approx F(z_1, z_2) \cdot G(z_1, z_2)$, then for the cascade representation where $F(z_1, z_2)$ and $G(z_1, z_2)$ are in tandem, the total number of independent parameters is

[1]For a 2D filter, we consider the order for each dimension.

$(P_F+1)(Q_F+1)+(P_G+1)(Q_G+1)$ where (P_F, Q_F) and (P_G, Q_G) are respectively the orders of the two subfilters.

If the order of the cascade approximation is the same as that of the original filter, i.e. $P = P_F + P_G$ and $Q = Q_F + Q_G$, then the difference of the total number of parameters between the original and cascade representations is

$$\begin{aligned} & (P+1)(Q+1) - ((P_F+1)(Q_F+1) + (P_G+1)(Q_G+1)) \\ = \; & (P_F+P_G+1)(Q_F+Q_G+1) - (P_F+1)(Q_F+1) - (P_G+1)(Q_G+1) \\ = \; & P_F \cdot Q_G + P_G \cdot Q_F - 1, \end{aligned} \tag{3.1}$$

which we can show is a positive number if $P \geq 2$, $Q \geq 2$, and neither of $F(z_1, z_2)$ and $G(z_1, z_2)$ is a constant (i.e. the cascade is non-trivial). Consequently, a non-trivial cascade approximation with $P \geq 2$ and $Q \geq 2$ has a smaller total number of parameters than the original 2D filter and achieves computational efficiency for spatial domain implementation [41]. To explore this computational efficiency, it is meaningful to develop algorithms to obtain the factors $F(z_1, z_2)$ and $G(z_1, z_2)$, the product of which is close in some appropriate sense to the target bivariate polynomial $H(z_1, z_2)$. In addition, as implied in (3.1), the computational saving of the cascade approximation depends on the orders of the two subfilters in the cascade, and thus it would be desirable for the polynomial factorization algorithm to have control over the degrees of the two factors.

A related approach to reduce the computational complexity of 2D filters is to design *separable* filters, which generally satisfy $H(z_1, z_2) = H_1(z_1) \cdot H_2(z_2)$ [3]. Such separability can be regarded as a special case of bivariate polynomial factorization with the additional constraints $(P_F, Q_F) = (P, 0)$ and $(P_G, Q_G) = (0, Q)$, i.e. each factor depends on only a single variable. A 2D FIR filter can be implemented in a parallel form where each branch is a separable filter [3]. There are other filter structures related to the separable filters above, such as 2D polar separable filters [98] and 2D fan filters [99, 100]. In this chapter, we focus on the general factorization of bivariate polynomials instead of the separable form.

3.2 Problem Formulation

This section formulates the main problem and defines the notations in this chapter. The main goal is to exactly or approximately factorize the bivariate polynomial $H(x,y)$ into two factors $F(x,y)$ and $G(x,y)$; in other words, the coefficients of the polynomials $F(x,y)$ and $G(x,y)$ are the parameters to be estimated. The degree of the input polynomial is $\deg(H) = (P,Q)$, which means that P is the degree of $H(x,y)$ when we consider x as the main variable while y as a parameter, and Q is the degree when y is the main variable.

If we choose the ℓ_2-norm of the approximation error as the criterion, then the main goal of this chapter can be formulated as

$$\min_{F(x,y),\ G(x,y)} \quad \|H(x,y) - F(x,y)\cdot G(x,y)\|_2\,, \tag{3.2}$$

$$\text{s.t.} \quad \deg(F) = (P_F, Q_F),\ \deg(G) = (P_G, Q_G), \tag{3.3}$$

$$P_F + P_G = P,\ Q_F + Q_G = Q, \tag{3.4}$$

where the degrees of the two factors (P_F, Q_F) and (P_G, Q_G) are fully specified. In addition, the ℓ_2-norm in (3.2) of a polynomial $E(x,y) = \sum_{i,j} e_{i,j}x^i y^j$ is defined as $\|E(x,y)\|_2 = \sqrt{\sum_{i,j} e_{i,j}^2}$.

3.3 Review of Existing Work

Bivariate polynomial factorization has drawn considerable attention in computational and symbolic mathematics [101–107]. Most existing approaches focus on exact factorization of bivariate polynomials that are known to be factorizable; approximate factorization algorithms for non-factorizable polynomials seem to work only if the input polynomials are close to factorizable ones, and they generally lack useful analysis or guarantees on the performance.

For the purpose of the cascade representation of 2D filters, we consider only algorithms that work with real-valued (rather than integer, rational, or complex) coefficients, and in particular focus on four main branches of approaches.

The first branch applies alternating minimization to both factors [102], which we refer to as the *direct alternating minimization algorithm*. This algorithm takes iterations, each of which consists of two steps: optimization over $F(x,y)$ with $G(x,y)$ fixed, and vice versa. Since the product $H(x,y)$ has a bilinear relationship[2] with the factors $F(x,y)$ and $G(x,y)$, if we consider the ℓ_2-norm of the approximation error as the criterion, then the optimization of a factor with the other one fixed becomes a linear least squares problem and has a straight-forward solution. An advantage of this simple algorithm is the direct control over the degrees of the factors, which can enable a flexible tradeoff between computational complexity of the associated filter implementation and approximation accuracy. However, if we consider the optimization over both factors, the formulation generally is non-convex and locally optimal solutions are likely obtained by this algorithm. This algorithm is evaluated in Section 3.4.3.

The second branch converts the factorization of bivariate polynomials into rank deficient matrix approximation [104, 106, 107]. Specifically, as is shown in [104, 106], a bivariate polynomial is exactly factorizable if and only if an associated partial different equation (PDE) has a non-trivial solution; the solution to this PDE further corresponds to the null space of a structured matrix with elements related to the coefficients of the bivariate polynomial, where the matrix is referred to as the Ruppert matrix [104]. Therefore, the rank deficiency status of the Ruppert matrix determines whether the bivariate polynomial is exactly factorizable; as is proposed in [106, 107], the null space of the Ruppert matrix can be used to obtain the factors of the polynomial. In addition, approximate bivariate polynomial factorization corresponds to a rank deficient approximation of the structured Ruppert matrix, which can be solved by algorithms such as structured total least squares [108, 109] or Riemann singular value decomposition [110]. Although this approach has profound theoretical background, it does not seem to have a sufficiently flexible control over the degrees of

[2]If we consider the bivariate polynomials as z-transforms of 2D signals, then the multiplication of the two polynomials becomes the z-transform of the convolution of the two signals. The bilinear property of the convolution shows the bilinear dependence of the product $H(x,y)$ on the factors $F(x,y)$ and $G(x,y)$.

the factors that would be important for the computational efficiency of the cascade implementation of 2D filters, as is discussed in Section 3.1. As a related application of the Ruppert matrix in signal processing, univariate polynomial decomposition has been explored in [2, 42] with algorithms utilizing the Ruppert matrix.

The third branch of algorithms tracks the root contour of the bivariate polynomial, where one variable is considered as the main variable while the other is a parameter fixed at a few constant values [105]. If $H(x, y) = F(x, y)\cdot G(x, y)$ where x is considered as a variable and y is fixed at a constant value, then the roots of $H(\cdot, y)$ are the union of the roots of $F(\cdot, y)$ and $G(\cdot, y)$. For simplicity, we suppose that all roots at this fixed y are single roots, which guarantees that $F(\cdot, y)$ and $G(\cdot, y)$ do not share a common root. For a root of $H(\cdot, y)$ that is also a root of $F(\cdot, y)$, if we continuously perturb the value of y, then the corresponding root is also perturbed and forms a contour; within sufficiently small perturbations of y, the root contour of x entirely is a root contour of $F(\cdot, y)$. Each obtained root pair (x_r, y_r) satisfies $F(x_r, y_r) = 0$, which forms a linear equation with respect to the coefficients of $F(\cdot, \cdot)$. With the root contour, sufficient linear equations can be established and thus the coefficients of $F(x, y)$ can be solved. In the last step, the other factor $G(x, y)$ can be obtained by a least squares solution with $F(x, y)$ available.

The fourth branch of approaches uses the zero-sum property of the roots of bivariate polynomials, which again considers one variable as the main variable and the other as a parameter [101, 103]. Similar to the previous approach, this method uses the fact that the roots of $H(x, y)$ with respect to x are implicit functions[3] of y, which can be referred to as root functions $x = \phi_i(y)$ $(1 \leq i \leq \deg_x(H)$ where $\deg_x(H)$ is the degree of $H(x, y)$ with respect to $x)$. With the root functions, bivariate polynomial factorization is converted to determining a partition among the root functions, so that the multiplication[4] of the factors $(x - \phi_i(y))$ with the root functions $\phi_i(y)$ in each partition forms a polynomial factor. As is shown in [103], certain power summation of

[3]Again, we suppose all roots are single roots in the discussion. If there are multiple roots, then choosing a different value of y can avoid this issue.

[4]More precisely, an extra polynomial of y may also be needed in this multiplication in addition to the factors $(x - \phi_i(y))$, the details for which are discussed in Section 3.4.1.

the root functions in each partition should also be polynomials, which is guaranteed to have all coefficients as 0 for orders above certain threshold (i.e. the "zero-sum" property). Finally, determining the partition of the root functions that satisfies the zero-sum property obtains the factors for the original bivariate polynomial. Since this method can have control over the degrees of factors, we use the idea of zero-sum property and further propose a new factorization algorithm in Section 3.4.1.

3.4 Algorithms for Bivariate Polynomial Factorization

This section proposes two new algorithms for bivariate polynomial factorization, which we refer to as the zero-sum mixed integer programming algorithm in Section 3.4.1 and the lifted alternating minimization algorithm in Section 3.4.2. The numerical evaluation for these algorithms is in Section 3.4.3.

3.4.1 Zero-sum Mixed Integer Programming Algorithm

This section introduces a mixed integer programming (MIP) algorithm based on the zero-sum property that is introduced in [101]. Instead of working with the coefficients directly, this algorithm considers the root functions of the bivariate polynomials. Before describing the algorithm, we first review the zero-sum property of the root functions associated with a bivariate polynomial [101]. A bivariate polynomial $H(x, y)$ can be expressed in the form of a univariate polynomial of x where the *coefficients* are polynomials of the parameter y, namely

$$H(x, y) = \sum_{i=0}^{P} h_i(y) \cdot x^i, \tag{3.5}$$

where $P = \deg_x(H)$ is the degree of $H(x, y)$ in x. If we fix a value of the parameter y and factorize with respect to x, then we have P roots[5]. Perturbation of the value of

[5]For simplicity, we assume that all the P roots of x are single roots, otherwise we choose a different value of y.

y continuously changes the associated roots of x, and thus we can denote continuous functions $\tilde{\phi}_i(y)$ $(1 \le i \le P)$ as the i^{th} root of the univariate polynomial in x that depends on the value of y. As a result, the bivariate polynomial $H(x, y)$ can be expressed as

$$H(x,y) = h_P(y) \cdot \prod_{i=1}^{P} (x - \tilde{\phi}_i(y)). \tag{3.6}$$

By the implicit function theorem [111], it can be shown that the functions $\tilde{\phi}_i(y)$ are smooth at values of y where $H(x, y)$ has only single roots in x. Typically, these root functions are not necessarily polynomials and may have complicated expressions, however, some combination of these functions are guaranteed polynomials as will be discussed below [103].

By Vieta's theorem that states the relationship between the coefficients and the roots of a polynomial [112], the J^{th} elementary symmetric function of the root functions $\tilde{\phi}_i(y)$ satisfies

$$\sum_{k_1 < k_2 < \cdots < k_J} \prod_{j=1}^{J} \tilde{\phi}_{k_j}(y) = (-1)^J \cdot \frac{h_{P-J}(y)}{h_P(y)}, \quad 1 \le J \le P. \tag{3.7}$$

Furthermore, by Newton's identities [113] that relate the elementary symmetric functions to the power summations, the power summations of the root functions satisfy

$$\sum_{i=1}^{P} \left(\tilde{\phi}_i(y)\right)^J = \frac{\tilde{h}_J(y)}{(h_P(y))^J}, \quad 1 \le J \le P, \tag{3.8}$$

where $\tilde{h}_J(y)$ are polynomials of degrees[6] no higher than QJ where $Q = \deg_y(H)$. The coefficients of $\tilde{h}_J(y)$ depend on $h_{P-j}(y)$ for $0 \le j \le J$. Finally, we see that

$$(h_P(y))^J \cdot \left(\sum_{i=1}^{P} \left(\tilde{\phi}_i(y)\right)^J \right) = \tilde{h}_J(y), \quad 1 \le J \le P, \tag{3.9}$$

which is a polynomial in y with degree no higher than QJ, despite the possibly complicated expressions of the root functions $\tilde{\phi}_i(y)$.

[6]A tighter upper bound for the degrees of $\tilde{h}_J(y)$ is in [101].

To avoid directly using the complicated root functions, it is possible to replace $\tilde{\phi}_i(y)$ in (3.9) by its truncated Taylor expansion up to order R, which is denoted by $\phi_i(y)$. Then, the coefficient of y^r remains 0 in the summation in (3.9) for $QJ \le r \le R$, since the truncation of Taylor series affects only the terms with orders higher than R.

Next, we consider the factorization $H(x, y) = F(x, y) \cdot G(x, y)$. If all roots of $H(x, y)$ are single roots at the fixed y, then the union of the root functions of $F(x, y)$ and $G(x, y)$ forms the root functions of $H(x, y)$. With a similar derivation as above, for the left-hand-side of (3.9), if we take the summation over only the root functions that correspond to $F(x, y)$ (or $G(x, y)$), then the coefficient of the term y^r in this summation still remains 0 for $QJ \le r \le R$; we can notice that the range of J in (3.9) becomes $1 \le J \le P_F$ for $F(x, y)$ (and $1 \le J \le P_G$ for $G(x, y)$), where P_F and P_G are the degrees with respect to x for $F(x, y)$ and $G(x, y)$, respectively. The statement above is the zero-sum property that is mentioned in [101], and we have completed the review of this property.

With the zero-sum property, the problem of factorization is converted to the determination of a subset of the root functions, where the subset satisfies the zero-sum property. If $H(x, y)$ is approximately factorizable, then the coefficients of terms within a certain range of orders of y in the summation (3.9) should be close to 0. In order to determine the subset of the root functions, we propose a new mixed integer program formulation. Binary variables b_i ($1 \le i \le P$) are used to indicate whether the i^{th} root function of $H(x, y)$ is also a root function of $F(x, y)$ (when $b_i = 0$) or $G(x, y)$ (when $b_i = 1$). A possible choice of the objective function is the absolute summation of the coefficients in the left-hand-side of (3.9) over the range of orders of y where the coefficients should be 0 (if $H(x, y)$ is exactly factorizable), for both $F(x, y)$ and $G(x, y)$. Finally, the following MIP formulation is introduced.

$$\min_{b_j, \psi^{(F)}_{J,k}, \psi^{(G)}_{J,k}} \quad \sum_{J=1}^{P_F} \sum_{k \in \mathcal{R}_J} \left| \psi^{(F)}_{J,k} \right| + \sum_{J=1}^{P_G} \sum_{k \in \mathcal{R}_J} \left| \psi^{(G)}_{J,k} \right| \tag{3.10}$$

$$\text{s. t.} \quad \sum_{i=1}^{P} b_i = P_G, \tag{3.11}$$

$$\psi_{J,k}^{(F)} = \text{coeff}\left[(h_P(y))^J \cdot \left(\sum_{i=1}^{P} (1-b_i) \cdot (\phi_i(y))^J \right),\ y^k \right],\ \text{for } k \in \mathcal{R}_J,\ 1 \le J \le P_F, \tag{3.12}$$

$$\psi_{J,k}^{(G)} = \text{coeff}\left[(h_P(y))^J \cdot \left(\sum_{i=1}^{P} b_i \cdot (\phi_i(y))^J \right),\ y^k \right],\quad \text{for } k \in \mathcal{R}_J,\ 1 \le J \le P_G, \tag{3.13}$$

$$b_i \in \{0,1\}, \quad \text{for } i = 1, 2, ..., P. \tag{3.14}$$

For a given J, the set $\mathcal{R}_J$ in the formulation above denotes the range of orders of y in the left-hand-side of (3.9) (i.e. the J^{th} power summation), for which the coefficients should be 0 if $H(x,y)$ is exactly factorizable; the largest element in $\mathcal{R}_J$ depends on the highest order in the truncated Taylor series $\phi_i(y)$. As a result, the cost in (3.10) is the total residue coefficients in the power summations for both factors $F(x,y)$ and $G(x,y)$. The constraint (3.11) controls the degrees of both factors. Constraints (3.12) and (3.13) define the actual coefficients in the power summations of the root functions for $F(x,y)$ and $G(x,y)$, respectively, where the notation "coeff $\left[h(y),\ y^k\right]$" denotes the coefficient of the term y^k in the polynomial $h(y)$. In practice, $\psi_{J,k}^{(F)}$ and $\psi_{J,k}^{(G)}$ can be complex numbers, and the absolute values in (3.10) is interpreted as the absolute summation of the real and the imaginary parts; in addition, the coefficients in (3.12) and (3.13) can be weighted in the cost function to avoid the cost function favoring higher order terms that typically have coefficients with larger magnitude, which may improve robustness.

The solution to the MIP formulation in (3.10) provides a partition of the root functions that correspond to $F(x,y)$ and $G(x,y)$, and additional steps are necessary to convert the partition of the root functions to the coefficients of $F(x,y)$ and $G(x,y)$. First, we consider the simple case where the *leading polynomial* $h_P(y) = h_P$ is a constant and independent of y. For this case, the coefficients of the factors are already available from the partition of the root functions, which are naturally obtained by multiplying the terms $(x - \phi_i(y))$ that correspond to $F(x,y)$ and $G(x,y)$, respectively. Since the Taylor expansions $\phi_i(y)$ typically have high orders, we need to truncate the product of $(x - \phi_i(y))$ with a specified degree of y. The scaling h_P can be applied to either of the factors and does not influence the approximation performance.

For the more challenging situation where the leading polynomial $h_P(y)$ is not a constant, it is less straight-forward to directly obtain the coefficients of the factors from the partition of the root functions. In this case, the polynomial $h_P(y)$ can be approximately factorized into two parts, each of which is incorporated into $F(x, y)$ or $G(x, y)$. In other words, we approximately factorize $h_P(y) \approx h_P^{(F)}(y) \cdot h_P^{(G)}(y)$, and then compute

$$\tilde{F}(x, y) = h_P^{(F)}(y) \cdot \prod_{i:b_i=0} (x - \phi_i(y)), \tag{3.15}$$

$$\tilde{G}(x, y) = h_P^{(G)}(y) \cdot \prod_{i:b_i=1} (x - \phi_i(y)). \tag{3.16}$$

Since the Taylor expansions $\phi_i(y)$ have higher orders than the final factors, the final step is again to truncate $\tilde{F}(x, y)$ and $\tilde{G}(x, y)$, in which different factorization of $h_P(y)$ may have different approximation performance. Typically, the optimal factorization of $h_P(y)$ may be challenging to obtain; since the total number of the factorization pairs $h_P^{(F)}(y)$ and $h_P^{(G)}(y)$ increases exponentially with the degree of $h_P(y)$, it is generally inefficient to sweep over all possibilities of factorization. As a heuristic approach, an alternating minimization is applied to the coefficients of $h_P^{(F)}(y)$ and $h_P^{(G)}(y)$, which aims to minimize the ℓ_2-norm of error between $H(x, y)$ and the factorizable approximation.

The zero-sum mixed integer programming algorithm is summarized as follows.

Zero-sum Mixed Integer Programming Algorithm

(1) Fix a value $y = y_0$ at which $H(x, y_0)$ has single roots only.

(2) Factorize $H(x, y_0)$ and obtain the Taylor expansions $\phi_i(y)$ at each root of $H(x, y_0)$ by properly matching the coefficients in $H(x, y)$ and in the product $h_P(y) \cdot \prod_{i=1}^{P} (x - \phi_i(y))$.

(3) Formulate and solve the MIP (3.10).

(4) If the leading polynomial $h_P(y)$ is not a constant, then determine

$h_P^{(F)}(y)$ and $h_P^{(G)}(y)$ by alternating minimization.

(5) The factors $F(x, y)$ and $G(x, y)$ are obtained by truncating $\tilde{F}(x, y)$ and $\tilde{G}(x, y)$ in (3.15) and (3.16) up to the specified degrees of y.

A potential challenge with this algorithm is the computation of the Taylor expansion $\phi_i(y)$ that is also used in [101]; for certain polynomials, the coefficients of the Taylor series may expand with the orders, which reduces the robustness of our new MIP algorithm. In addition, the current algorithm uses only a single value of y, which may also reduce the robustness of the algorithm. An approach is proposed in [101] to utilize lower-order Taylor series at multiple nearby values of y to gain robustness, and a possible direction for future work is to incorporate this idea in our new MIP formulation.

3.4.2 Lifted Alternating Minimization Algorithm

Based on the idea of simplifying constraints via lifting the problem to a higher dimensional parameter space, this section introduces an alternating minimization algorithm in a dimensionally lifted space, which works on the coefficients of the polynomial factors. If we list the coefficients of the factors $F(x, y)$ and $G(x, y)$ into column vectors $\mathbf{f}$ and $\mathbf{g}$, respectively, then the matrix

$$\mathbf{M} = \mathbf{f} \cdot \mathbf{g}^{\mathrm{T}} \tag{3.17}$$

contains the coefficients of individual terms in the multiplication $F(x, y) \cdot G(x, y)$ without combining similar terms (i.e. terms with the same orders). The matrix $\mathbf{M}$ in (3.17) is an outer product and therefore has rank one. In order to combine the similar terms after multiplication, we denote the index sets

$$\mathcal{C}(i, j) = \big\{(a, b) : \ M_{a,b} \text{ corresponds to } x^i y^j \text{ in } F(x, y) \cdot G(x, y)\big\} \tag{3.18}$$

that contains all the indices of the elements in the matrix $\mathbf{M}$ that contribute to the term $x^i y^j$ in the product $F(x, y) \cdot G(x, y)$. Since the degrees of the factors satisfy (3.4), we know that the index sets $\mathcal{C}(i,j)$ with $(0,0) \leq (i,j) \leq (P,Q)$ form a partition of the indices (a,b) of all the elements in $\mathbf{M}$. If we denote

$$H(x,y) = \sum_{i=0}^{P} \sum_{j=0}^{Q} h_{i,j} \cdot x^i y^j,$$

then an exact factorization matches the coefficients, i.e.

$$h_{i,j} = \sum_{(a,b)\in\mathcal{C}(i,j)} M_{a,b}, \text{ for all } (0,0) \leq (i,j) \leq (P,Q). \tag{3.19}$$

As a result, for a factorizable polynomial $H(x,y)$, its factors correspond to a matrix $\mathbf{M}$ that is in the intersection of the set of rank-one matrices and the linear space with constraints (3.19). For simplicity of description, the two sets of matrices above are denoted as follows,

$$\mathcal{U}_{\text{rank}} = \left\{\mathbf{M} \in \mathbb{R}^{\dim(\mathbf{f})\times\dim(\mathbf{g})} : \mathbf{M} \text{ has rank } 1\right\}, \tag{3.20}$$

$$\mathcal{U}_{\text{coeff}} = \left\{\mathbf{M} \in \mathbb{R}^{\dim(\mathbf{f})\times\dim(\mathbf{g})} : \sum_{(a,b)\in\mathcal{C}(i,j)} M_{a,b} = h_{i,j} \text{ for } (0,0) \leq (i,j) \leq (P,Q)\right\}, \tag{3.21}$$

where $\dim(\mathbf{f})$ and $\dim(\mathbf{g})$ are the dimensions of the vectors $\mathbf{f}$ and $\mathbf{g}$ in (3.17), respectively.

For a general polynomial $H(x,y)$ that is not guaranteed factorizable, the formulation (3.2) in the *dimensionally lifted*[7] parameter space of the matrix $\mathbf{M} \in \mathbb{R}^{\dim(\mathbf{f})\times\dim(\mathbf{g})}$ becomes

$$\min_{\mathbf{M} \in \mathcal{U}_{\text{rank}}} \quad V_{\text{poly}}(\mathbf{M}) \triangleq \sum_{i=0}^{P} \sum_{j=0}^{Q} \left(h_{i,j} - \sum_{(a,b)\in\mathcal{C}(i,j)} M_{a,b} \right)^2. \tag{3.22}$$

[7]The total number of parameters of the matrix $\mathbf{M}$ is $\dim(\mathbf{f}) \times \dim(\mathbf{g})$, while the two polynomials $F(x,y)$ and $G(x,y)$ have a total of $\dim(\mathbf{f}) + \dim(\mathbf{g})$ coefficients. We can see that the matrix $\mathbf{M}$ is in a higher dimensional space than the space of the coefficients in $F(x,y)$ and $G(x,y)$, and we refer to the former as a dimensionally lifted space.

If we have the optimal matrix $\mathbf{M}$, then the two vectors $\mathbf{f}$ and $\mathbf{g}$ in (3.17) are directly applicable for the construction of the factors $F(x,y)$ and $G(x,y)$. Since (3.22) has a low rank constraint, the existing algorithms on low rank approximation can be applied to this formulation. As a few examples of these algorithms, the nuclear norm minimization [114], the singular value projection (SVP) [115], and the atomic decomposition for minimum rank approximation [116] are briefly reviewed in Appendix B.

From an alternative perspective, it is possible to avoid solving for (3.22) directly, but instead we aim to determine the pair of the closest elements between the set of rank-one matrices $\mathcal{U}_{\text{rank}}$ and the linear space $\mathcal{U}_{\text{coeff}}$ that exactly matches the coefficients of $H(x,y)$. Thus, we propose the following formulation (3.23), where the Frobenius norm[8] is chosen as the criterion for mathematical tractability,

$$\min_{\widetilde{\mathbf{M}} \,\in\, \mathcal{U}_{\text{rank}},\ \widehat{\mathbf{M}} \,\in\, \mathcal{U}_{\text{coeff}}} \quad V_{\text{Fro}}(\widetilde{\mathbf{M}},\ \widehat{\mathbf{M}}) \triangleq \left\| \widehat{\mathbf{M}} - \widetilde{\mathbf{M}} \right\|_{\text{F}}^{2}. \tag{3.23}$$

Relationship between the Formulations (3.22) and (3.23)

Before developing the algorithm to solve (3.23), we first discuss the relationship between the two formulations (3.22) and (3.23). Generally, the optimal solutions and the associated minimal objective values of these two formulations are not identical. However, there is a mutual bound between the minimal objective values in the formulations (3.22) and (3.23), which will be established in this section.

First, we introduce a theorem that obtains the optimal matrix $\widehat{\mathbf{M}}$ in the linear space $\mathcal{U}_{\text{coeff}}$ that minimizes (3.23), where the rank-one matrix $\widetilde{\mathbf{M}}$ is fixed. The proof for Theorem 3.1 is in Appendix C.

Theorem 3.1. *For a fixed matrix* $\widetilde{\mathbf{M}} \in \mathcal{U}_{\text{rank}}$*, we denote the matrix in* $\mathcal{U}_{\text{coeff}}$ *that minimizes (3.23) as*

$$\widehat{\mathbf{M}}^{\text{proj}} = \operatorname*{arg\,min}_{\widehat{\mathbf{M}} \,\in\, \mathcal{U}_{\text{coeff}}} \quad V_{\text{Fro}}(\widetilde{\mathbf{M}},\ \widehat{\mathbf{M}}). \tag{3.24}$$

[8]The Frobenius norm of a matrix $\mathbf{S}$ is $\|\mathbf{S}\|_{\text{F}} = \sqrt{\sum_a \sum_b S_{a,b}^2}$.

This optimal matrix $\widehat{\mathbf{M}}^{\text{proj}}$ has the closed form expression as below,

$$\widehat{M}^{\text{proj}}_{a,b} = \widetilde{M}_{a,b} + \frac{R(i,j)}{|\mathcal{C}(i,j)|}, \text{ for } (a,b) \in \mathcal{C}(i,j), \text{ and } (0,0) \leq (i,j) \leq (P,Q), \tag{3.25}$$

where

$$R(i,j) = h_{i,j} - \sum_{(a',b') \in \mathcal{C}(i,j)} \widetilde{M}_{a',b'}. \tag{3.26}$$

Theorem 3.1 obtains the projection of a matrix[9] onto the linear space $\mathcal{U}_{\text{coeff}}$ with respect to the Frobenius norm, which will later be used in our algorithm. The result (3.25) has a nice interpretation: we can obtain the projection of a matrix $\widetilde{\mathbf{M}}$ onto $\mathcal{U}_{\text{coeff}}$ in two steps – first in (3.26) we obtain the difference of the coefficients of the term $x^i y^j$ between the input polynomial $H(x,y)$ and the polynomial associated with the matrix $\widetilde{\mathbf{M}}$, and then in (3.25) we equally distribute this difference onto all the elements of the matrix $\widetilde{\mathbf{M}}$ that contribute to $x^i y^j$ (for each $(0,0) \leq (i,j) \leq (P,Q)$).

From another perspective, if we list the elements of the matrix $\widetilde{\mathbf{M}}$ as a vector $\widetilde{\mathbf{m}}$, then the Frobenius norm in the matrix space of $\widetilde{\mathbf{M}}$ becomes the ℓ_2-norm in the vector space of $\widetilde{\mathbf{m}}$. The formulation (3.24) becomes the projection of a vector $\widetilde{\mathbf{m}}$ onto a linear space with respect to the ℓ_2-norm, for which an alternative approach is to use the Moore-Penrose pseudo-inverse [117, 118] of an associated matrix $\mathbf{A}$. This matrix $\mathbf{A}$ maps from the vector space of $\widetilde{\mathbf{m}}$ to the space of vectors $\mathbf{h}_{\mathbf{M}}$ with elements as coefficients of the bivariate polynomial that corresponds to $\widetilde{\mathbf{M}}$ (i.e. $\mathbf{h}_{\mathbf{M}} = \mathbf{A} \cdot \widetilde{\mathbf{m}}$). Our result (3.25) implicitly solves this pseudo-inverse problem without constructing $\widetilde{\mathbf{m}}$ or $\mathbf{A}$, and in addition it has a nice interpretation as mentioned above.

With Theorem 3.1, we can have the following bounds between the objective functions in (3.22) and (3.23), for a fixed rank-one matrix and for the global optimal objective values. The proof is again in Appendix C.

Theorem 3.2. *For a fixed matrix $\widetilde{\mathbf{M}} \in \mathcal{U}_{\text{rank}}$, the objective values in (3.22) and*

[9]Actually, for Theorem 3.1 to hold, the matrix $\widetilde{\mathbf{M}}$ does not need to be rank-one and can be a general matrix of the same size.

(3.23) have the bounds below,

$$\left(\min_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot V_{\text{Fro}}(\widetilde{\mathbf{M}},\ \widehat{\mathbf{M}}^{\text{proj}}) \ \le\ V_{\text{poly}}(\widetilde{\mathbf{M}}) \ \le\ \left(\max_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot V_{\text{Fro}}(\widetilde{\mathbf{M}},\ \widehat{\mathbf{M}}^{\text{proj}}), \tag{3.27}$$

where $|\mathcal{C}(i,j)|$ is the cardinality of the index set $\mathcal{C}(i,j)$; $\widehat{\mathbf{M}}^{\text{proj}}$ is defined in (3.24) and associated with the rank-one matrix $\widetilde{\mathbf{M}}$. Furthermore, if we denote the minimal objective values of the formulations (3.22) and (3.23) as $V_{\text{poly}}^{\text{opt}}$ and $V_{\text{Fro}}^{\text{opt}}$, respectively, then these two minimal values are mutually bounded as below:

$$\left(\min_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot V_{\text{Fro}}^{\text{opt}} \ \le\ V_{\text{poly}}^{\text{opt}} \ \le\ \left(\max_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot V_{\text{Fro}}^{\text{opt}}. \tag{3.28}$$

Theorem 3.2 implies that if the dynamic range of $|\mathcal{C}(i,j)|$ is small, then the two formulations (3.22) and (3.23) are mutually close in terms of the objective values. In the special situation where $|\mathcal{C}(i,j)| = 1$ for all $(0,0) \le (i,j) \le (P,Q)$, Theorem 3.2 shows that the two formulations are identical, which can also be seen from the definition of the objective functions in (3.22) and (3.23).

Algorithm for the Formulation (3.23)

The formulation (3.23) is generally challenging to solve due to the non-convexity of the rank-one matrix set $\mathcal{U}_{\text{rank}}$. A possible method to solve (3.23) is an iterative alternating minimization algorithm, where each iteration has two projections between $\mathcal{U}_{\text{rank}}$ and $\mathcal{U}_{\text{coeff}}$ with respect to the Frobenius norm. One of the projections is from the current matrix towards the set of rank-one matrices $\mathcal{U}_{\text{rank}}$. As shown by the Eckart-Young-Mirsky theorem [119], the rank-one approximation that is closest in the Frobenius norm to the current matrix is achieved by performing the singular value decomposition (SVD) and taking only the largest singular value with its corresponding singular vectors. The other projection is towards the linear space of matrices $\mathcal{U}_{\text{coeff}}$, the result for which actually has been provided in Theorem 3.1. As a result, both projections in an iteration are exactly solved, and the algorithm is summarized as follows.

```
LIFTED ALTERNATING MINIMIZATION ALGORITHM
(1) Initialize $\widetilde{\mathbf{M}}^{(0)} = \mathbf{0}$. Let $k = 1$.
(2) In the $k^{\text{th}}$ iteration, perform the following two projections:
(3)     Obtain the projection onto the linear space $\mathcal{U}_{\text{coeff}}$:
            Compute $R^{(k)}(i,j)$ as in (3.26) with the current matrix $\widetilde{\mathbf{M}}^{(k-1)}$,
            and then compute the projection $\widehat{\mathbf{M}}^{(k)}$ by (3.25).
(4)     Obtain the projection onto the set of rank-one matrices $\mathcal{U}_{\text{rank}}$:
            Compute the singular value decomposition of $\widehat{\mathbf{M}}^{(k)}$, and construct
            the projection as $\widetilde{\mathbf{M}}^{(k)} = \mathbf{u}_1^{(k)} \cdot \sigma_1^{(k)} \cdot \left(\mathbf{v}_1^{(k)}\right)^{\mathrm{T}}$, in which $\sigma_1^{(k)}$ is the
            largest singular value of $\widehat{\mathbf{M}}^{(k)}$, and $\mathbf{u}_1^{(k)}$ and $\mathbf{v}_1^{(k)}$ are the
            corresponding left and right singular vectors, respectively.
(5)     $k \leftarrow k + 1$.
(6) Iterate until the results between consecutive iterations have difference
    smaller than a threshold, or a threshold on the iteration steps is reached.
(7) Let $\mathbf{f} = \mathbf{u}_1^{(k)}$ and $\mathbf{g} = \sigma_1^{(k)} \cdot \mathbf{v}_1^{(k)}$, and obtain $F(x,y)$ and $G(x,y)$.
```

We have a few remarks on this lifted alternating minimization algorithm. First, since each projection is theoretically precise, the Frobenius norm in the objective (3.23) is monotonically non-increasing with the iteration step. Second, since the rank-one matrix forms a non-convex set, there is generally no guarantee for this algorithm to converge to the global optimum; the same as general alternating minimization algorithms, the final result depends on the initialization of the matrix, and an initial value $\widetilde{\mathbf{M}}^{(0)}$ that is closer to the global optimum is typically beneficial for better approximation. Third, the objective function (3.23) defined between matrices in the dimensionally lifted space is related while different from the squared ℓ_2-norm of the polynomial approximation error in (3.22), which implies that there could be other cost functions for the alternating minimization algorithm with higher faithfulness to the criterion (3.22) that we generally care more.

Finally, we want to point out a special situation where the lifted alternating minimization algorithm could be non-iterative and be guaranteed optimality with the criterion (3.23) (and also with (3.22)). If *all* the index sets $\mathcal{C}(i,j)$ have a single element[10], i.e. $|\mathcal{C}(i,j)| = 1$ for all (i,j), then the formulation (3.23) (and (3.22)) is significantly simplified and has a closed form solution. Under this assumption, the goal becomes to obtain the rank-one approximation for the matrix $\mathbf{H} = [h_{i,j}]$ that is closest in the Frobenius norm. The Eckart-Young-Mirsky theorem [119] shows that the optimal solution in this situation is the product of the largest singular value of $\mathbf{H}$ and the associated singular vectors. In fact, we can show that the steps (1)-(4) in the lifted alternating minimization algorithm exactly obtain the optimal solution in this situation, and therefore no more iterations are needed.

The condition $|\mathcal{C}(i,j)| = 1$ for all (i,j) implies that every term in $F(x,y) \cdot G(x,y)$ (without combining the similar terms) has a unique order among all terms in the product. When the degrees of the two factors satisfy $\deg(F) = (P,0)$ and $\deg(G) = (0,Q)$, i.e. a separable factorization with one factor depending only on x and the other only on y, the condition $|\mathcal{C}(i,j)| = 1$ is guaranteed; in this special situation, our algorithm is essentially equivalent to the SVD approach for the design of separable 2D FIR filters [3]. The separable factorization is not the only situation with $|\mathcal{C}(i,j)| = 1$; for non-separable factorization, the condition $|\mathcal{C}(i,j)| = 1$ (for all (i,j)) typically requires that some terms are guaranteed missing in the two factors, in order to ensure that the product $F(x,y) \cdot G(x,y)$ does not have two terms with exactly the same order. In this case, prior knowledge on which terms are missing in the two factors is needed to simplify the lifted alternating minimization algorithm.

3.4.3 Numerical Evaluation

Setup

This section evaluates the algorithms by factorizing synthetic bivariate polynomials. Each synthetic polynomial $H_{\text{input}}(x,y)$ is obtained by an exactly factorizable poly-

[10]Here we can consider only the values (i,j), for which the term $x^i y^j$ corresponds to at least an element in the matrix $\mathbf{M} = \mathbf{f} \cdot \mathbf{g}^{\mathrm{T}}$.

nomial with two randomly generated factors plus random noise, i.e. $H_{\text{input}}(x, y) = F_{\text{input}}(x, y) \cdot G_{\text{input}}(x, y) + H_{\text{noise}}(x, y)$; the coefficients of both the factors and the additive noise follow independent and zero-mean Gaussian distribution.

We define the input signal-to-noise-ratio (SNR) as the energy[11] of the factorizable polynomial $F_{\text{input}}(x, y) \cdot G_{\text{input}}(x, y)$ over the energy of the additive noise. Three values of input SNR are used in simulation, namely 40dB, 60dB, and noiseless.

The algorithms in our comparison and their abbreviations in the figures are as follows:

- the direct alternating minimization algorithm between the two polynomial factors [102] ("Direct Alter Min"), which is reviewed in Section 3.3,
- the zero-sum MIP algorithm in Section 3.4.1 ("Zero-sum MIP"),
- the nuclear norm minimization algorithm [114] for low rank matrix approximation ("Lifted Nuc Norm"),
- the singular value projection algorithm [115] for low rank matrix approximation ("Lifted SVP"),
- the atomic decomposition for minimum rank approximation algorithm [116] for low rank matrix approximation ("Lifted ADMiRA"),
- the lifted alternating minimization algorithm proposed in Section 3.4.2 ("Lifted Alter Min").

The last four algorithms in the list above are all for low rank matrix approximation, which are applied to the dimensionally lifted parameter space with the formulations proposed in Section 3.4.2. The nuclear norm minimization [114], the singular value projection [115], and the atomic decomposition for minimum rank approximation [116] are briefly reviewed in Appendix B.

[11] We refer to the energy of a polynomial as the squared ℓ_2-norm of the vector that contains all the coefficients in the polynomial.

The parameter settings for the algorithms above are as follows. For the “Direct Alter Min” algorithms, the upper limit on the iteration steps is 500. For the “Zero-sum MIP” algorithm, the MIP formulation in (3.10) is solved by IBM CPLEX version 12 [120] on a computer with 8.0GB RAM and Intel i7-4700MQ CPU @ 2.40GHz and 2.39GHz, with the time limit for each MIP as 60 seconds. For the “Lifted Nuc Norm” algorithm, the weighting parameter λ in formulation (B.4) in Appendix B.1 is set as 0.1. For the “Lifted SVP” algorithm in Appendix B.2, we use a small step size $\mu = 0.01$ in order to reduce the likelihood of divergence, and we set the upper limit on the iteration steps as high as 50000 to counteract the slow convergence caused by the small step size. In addition, we terminate the “Lifted SVP” algorithm if the rank-one matrices obtained in two consecutive iterations (denoted as $\mathbf{M}^{(k-1)}$ and $\mathbf{M}^{(k)}$) are sufficiently close; without loss of generality, we set this termination condition as $\left\|\mathbf{M}^{(k)} - \mathbf{M}^{(k-1)}\right\|_{\mathrm{F}}^{2} \leq 10^{-12} \cdot \left\|\mathbf{M}^{(k)}\right\|_{\mathrm{F}}^{2}$ where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm of a matrix. For the “Lifted ADMiRA” and “Lifted Alter Min” algorithms, we set the upper limit on the iteration steps as 500.

We apply each algorithm to each input bivariate polynomial $H_{\text{input}}(x, y)$, obtain the two approximate factors $F_{\text{output}}(x, y)$ and $G_{\text{output}}(x, y)$, compute their product as $H_{\text{output}}(x, y)$, and then measure the approximation error between the input polynomial $H_{\text{input}}(x, y)$ and the output result $H_{\text{output}}(x, y)$. In particular, if the output SNR (i.e. the energy ratio between the input polynomial and the approximation error) is no less than a threshold, then we consider the approximate factorization is successful. As an example, we set this threshold as 20dB in the simulation.

There are in total four degrees of the two polynomial factors, namely $\deg_x(F_{\text{input}})$, $\deg_y(F_{\text{input}})$, $\deg_x(G_{\text{input}})$, and $\deg_y(G_{\text{input}})$. In the experiments, we use the following options for the parameter setting.

- **Option (i)**: Fix $\deg_x(F_{\text{input}}) = \deg_y(G_{\text{input}}) = 5$.
  Keep $\deg_y(F_{\text{input}}) = \deg_x(G_{\text{input}})$, and sweep[12] $\deg_y(F_{\text{input}})$ from 1 to 15.

- **Option (ii)**: Fix $\deg_x(G_{\text{input}}) = 4$ and $\deg_y(G_{\text{input}}) = 6$.

[12]Because of the relatively heavy computational resources that are needed by the zero-sum MIP approach, we sweep only $1 \leq \deg_y(F_{\text{input}}) \leq 13$ for this algorithm.

Keep $\deg_x(F_{\text{input}}) + \deg_y(F_{\text{input}}) = 10$, and sweep $\deg_x(F_{\text{input}})$ from 0 to 10.

Parameter setting option (i) tests the performance of the algorithms with respect to the increase of the polynomial degrees, while option (ii) shows the performance when the factors have fixed total degrees. For each degree and each value of input SNR, 100 examples of polynomials $H_{\text{input}}(x, y)$ are generated, and we record the percentage of successful factorizations for each algorithm. The results for option (i) and (ii) are shown in Figures 3-1 and 3-2, respectively.

Observations

We can have the following observations from the simulation results.

First, the lifted alternating minimization approach has the highest success rates compared with the other algorithms in our simulation, followed by the lifted SVP algorithm. These two algorithms generally outperform the other algorithms by a large margin. Compared with the direct alternating minimization algorithm, the dimensionally lifting step in the lifted alternating minimization and lifted SVP algorithms could simplify the problem and possibly reduce the extent to which the iterations are trapped by local minimums, which leads to overall improved performance.

Second, by a comparison of the performance with different input SNR, the zero-sum MIP algorithm seems the most sensitive to additional noise among all the algorithms in simulation. In fact, for noiseless input polynomials, the zero-sum MIP can have higher success rates than the direct alternating minimization and the lifted nuclear norm minimization algorithms, but it has relatively lower success rates when the SNR is low. The zero-sum MIP method uses Taylor series of the root functions that may be sensitive to the noise in the coefficients of the polynomials, and thus lower SNR significantly reduces the success rates of this algorithm. The performance of the other algorithms does not show much difference between different SNR levels.

Third, for the lifted alternating minimization and the lifted SVP algorithms, the success rate seems to be reduced if the two factors have exactly the same (or very similar) degrees, i.e. $\deg_x(F_{\text{input}}) = \deg_x(G_{\text{input}})$ and $\deg_y(F_{\text{input}}) = \deg_y(G_{\text{input}})$.

This observation is supported by the troughs around $\deg_y(F_{\text{input}}) = \deg_x(G_{\text{input}}) = 5$ in Figure 3-1 and around $\deg_x(F_{\text{input}}) = 4$ in Figure 3-2, at which the two polynomial factors F_{input} and G_{input} have the same degrees in the respective settings. In addition, this phenomenon happens even if the input polynomials are noiseless. Although the exact reason for this phenomenon is not clear, we have some possible conjectures. One possible explanation is the non-uniqueness of the optimal solution to the formulations (3.23) and (3.22) when the two factors have the same degrees[13] but different coefficients (up to a scaling). Suppose the input polynomial is exactly factorizable; if the two factors have the same degrees, then swapping their coefficients retains their product but transposes the corresponding matrix $\widetilde{\mathbf{M}}$. If the coefficients of the two factors are not identical (up to a scaling), then both $\widetilde{\mathbf{M}}$ and $\widetilde{\mathbf{M}}^{\mathrm{T}}$ (with the corresponding $\widehat{\mathbf{M}}^{\text{proj}}$ and $\left(\widehat{\mathbf{M}}^{\text{proj}}\right)^{\mathrm{T}}$) are optimal solutions to the formulations (3.23) and (3.22). In this situation, the formulations (3.23) and (3.22) have multiple optimal solutions, which may result in a complex geometric structure in the solution space and make the algorithms less effective. For the lifted alternating minimization algorithm, we have another possible explanation for the reduced performance when factors have identical degrees, which relates to the cardinality of the index sets $\mathcal{C}(i,j)$ defined in (3.18). As discussed in Section 3.4.2, if all the index sets satisfy $|\mathcal{C}(i,j)| = 1$, then the formulation (3.23) becomes easily solvable with SVD. Therefore, the cardinality $|\mathcal{C}(i,j)|$ could possibly be related to the difficulty of solving (3.23). With the parameter setting option (ii), the maximum cardinality $\max_{i,j} |\mathcal{C}(i,j)|$ happens to achieve the highest value when $\deg_x(F_{\text{input}}) = 4$ and decreases as $\deg_x(F_{\text{input}})$ is away from 4. We can observe that the success rates of the lifted alternating minimization algorithm in Figure 3-2 happen to decrease as the increase of $\max_{i,j} |\mathcal{C}(i,j)|$, although we have not established a causal relationship between the former and the latter.

Fourth, as shown by Figure 3-1, polynomials with higher degrees are generally more challenging to approximately factorize than those with lower degrees, which may result from the fact that higher degree polynomials can have higher reduction

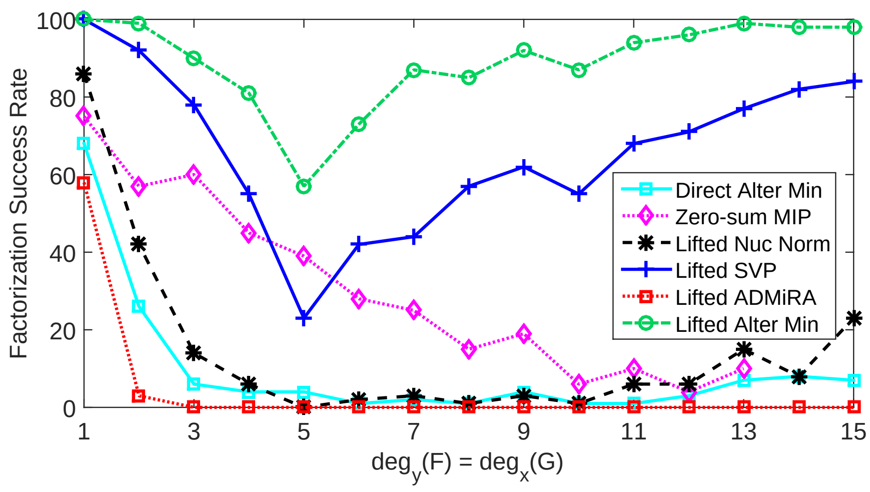
[13]When the factors have different degrees, we have not shown whether the optimal solution is unique; however, the two factors cannot be swapped as in the following argument.

in the total number of parameters from the original polynomial to the factorized representation. As a result, the successful factorization rates of the direct alternating minimization, the zero-sum MIP, the lifted nuclear norm minimization, and the lifted atomic decomposition for minimum rank approximation algorithms generally decrease with higher degrees of the polynomials. The success rates of the lifted alternating minimization and the lifted SVP algorithms do not show this monotonic trend in the current parameter setup, which is possibly due to the following reasons. One reason is the reduced success rate for these two algorithms when the two factors have identical degrees, which happens at $\deg_y(F_{\text{input}}) = \deg_x(G_{\text{input}}) = 5$ in Figure 3-1. The other possible reason is the current parameter setting and the choice of the criterion for a successful factorization (i.e. no less than 20dB SNR in the output product); as an example, if we reduce the upper limit of the iteration steps from 500 to 100 for the lifted alternating minimization algorithm, then its success rates generally decrease with the degrees of the polynomials (in the range $\deg_y(F_{\text{input}}) > 5$), which seems to imply that the polynomials with higher degrees need more iterations for the algorithm to converge.

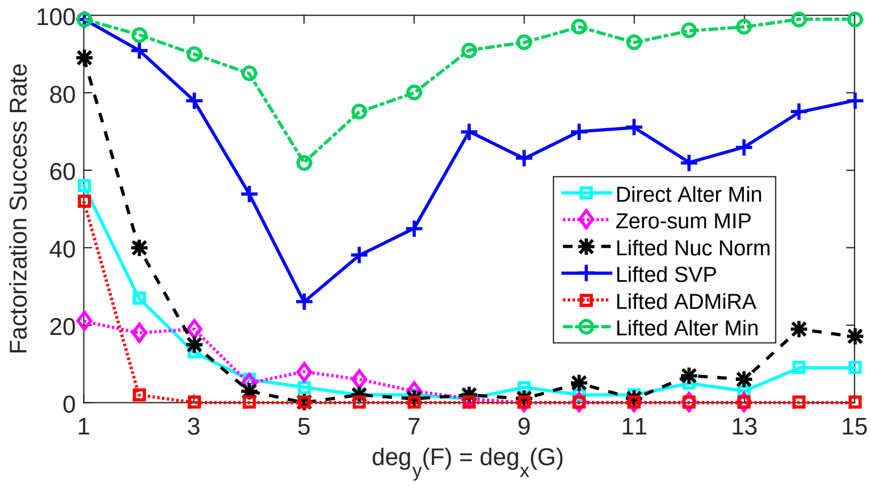
Fifth, when the parameter setting option (ii) is used, the results in Figure 3-2 seem to suggest that the zero-sum MIP algorithm has better performance when $\deg_x(F_{\text{input}})$ is low, which could result from the fact that our algorithm considers x as the main variable and y as the parameter; when $\deg_x(G_{\text{input}}) = 4$ is fixed, a larger $\deg_x(F_{\text{input}})$ leads to a larger $\deg_x(H_{\text{input}})$, and thus the factorization and the root functions of $H_{\text{input}}(\cdot, y)$ can be more difficult to obtain. In contrast, all the other algorithms seem to have better performance when either $\deg_x(F_{\text{input}})$ or $\deg_y(F_{\text{input}})$ is close to 0 in Figure 3-2.

3.5 Factorization with Symmetric Properties

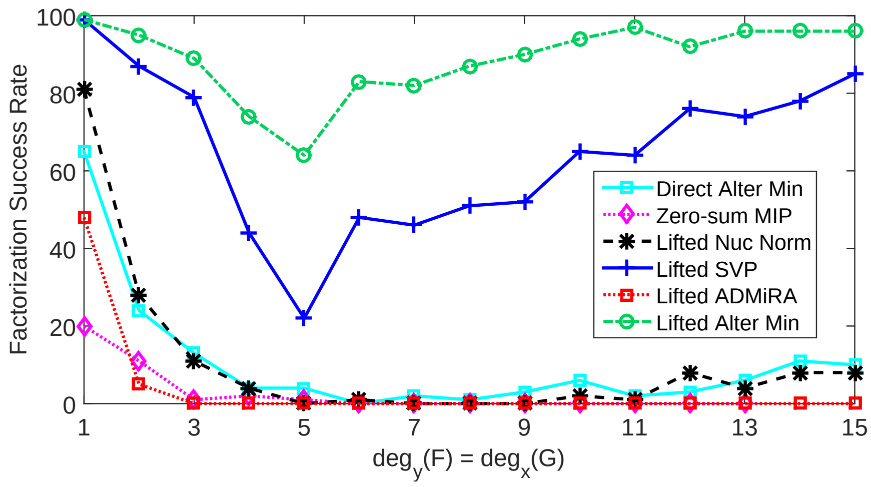
In contrast to general bivariate polynomials, 2D FIR filters can have additional symmetric properties that are desirable for certain signal processing purposes. Therefore, it is meaningful to retain these properties in the factorizable approximation, i.e. the



(a) Noiseless input

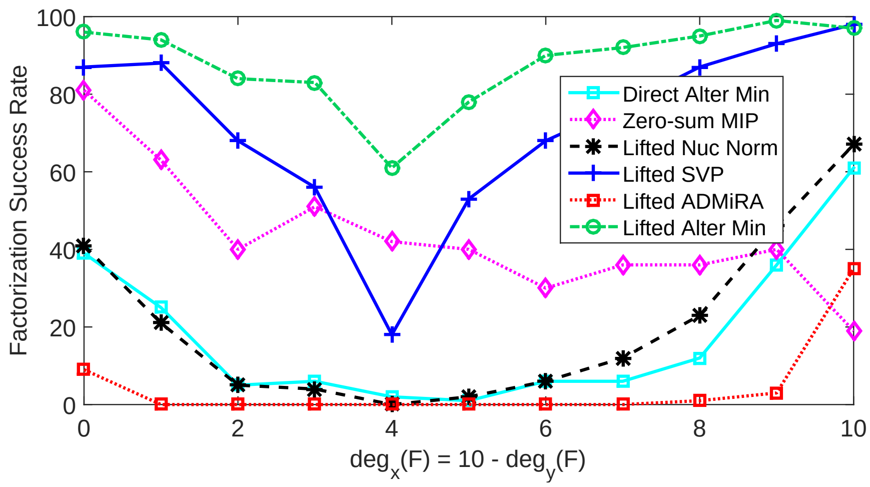


(b) Input SNR= 60dB

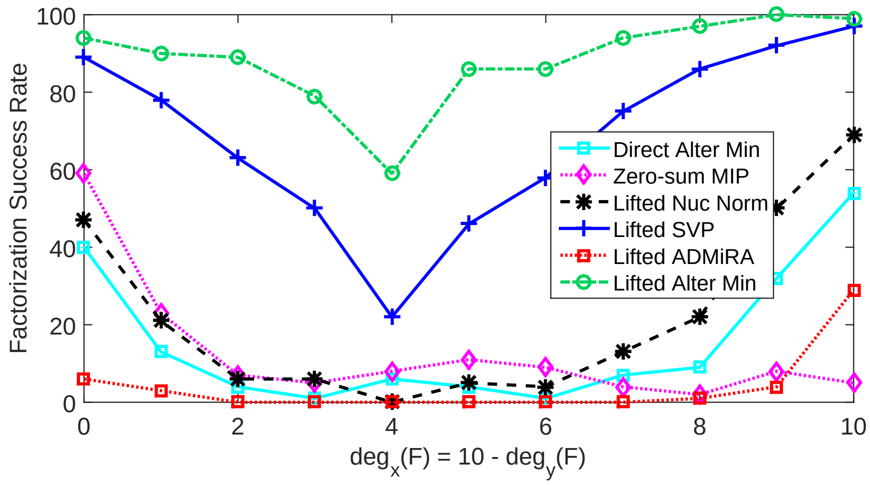


(c) Input SNR= 40dB

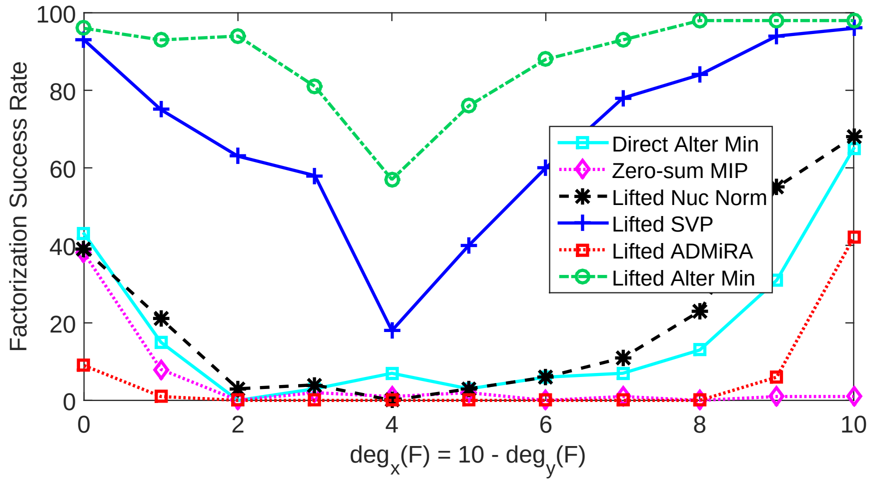
Figure 3-1: Comparison of the factorization algorithms with different input SNR. Parameters are set as option (i): fix $\deg_x(F_{\text{input}}) = \deg_y(G_{\text{input}}) = 5$, and sweep $\deg_y(F_{\text{input}}) = \deg_x(G_{\text{input}})$ from 1 to 15; for the zero-sum MIP algorithm, the results for $\deg_y(F_{\text{input}})$ at 14 and 15 are unavailable because of over-heavy computational resource requirement.



(a) Noiseless input



(b) Input SNR= 60dB



(c) Input SNR= 40dB

Figure 3-2: Comparison of the factorization algorithms with different input SNR. Parameters are set as option (ii): fix $\deg(G_{\text{input}}) = (4, 6)$, and sweep $\deg_x(F_{\text{input}}) = 10 - \deg_y(F_{\text{input}})$ from 0 to 10.

cascade approximation for the filter. A possible approach for this goal is to ensure that each of the two obtained factors retains these symmetric properties, which can be more restrictive but may have a higher tractability.

This section considers a method for retaining certain symmetric properties in the factors by applying a proper change of variables. In particular, we discuss the following types of symmetric properties.

- **(a)** $H(x, y) = H(y, x)$**.** The 2D filter with such a transfer function is symmetric with respect to the line $y = x$. In this case, if we use the change of variables $r = x \cdot y$ and $s = x + y$, then the bivariate polynomial with the property $H(x, y) = H(y, x)$ is guaranteed to be a finite degree bivariate polynomial $\tilde{H}(r, s)$ with the new variables[14]. In addition, an arbitrary bivariate polynomial with the variables r and s is guaranteed to have the symmetry $H(x, y) = H(y, x)$ when expressed with x and y. As a result, we can perform factorization on $\tilde{H}(r, s)$ without considering the symmetric property, and then changing the variables back to x and y obtains the factors that satisfy the above symmetric property.

- **(b)** $H(x, y) = H(x^{-1}, y)$**.** In this type of symmetry, we allow each term in the bivariate polynomial $H(x, y)$ to have either positive or negative[15] powers of x and y. A polynomial with $H(x, y) = H(x^{-1}, y)$ remains the same after the spatial reversal along the x axis. If we use the change of variables $r = x + x^{-1}$ and $s = y$, then the polynomial with the symmetry $H(x, y) = H(x^{-1}, y)$ is guaranteed expressible as a polynomial in r and s. In addition, any polynomials in r and s have symmetry $H(x, y) = H(x^{-1}, y)$ after the inverse change of variables. As a result, factorization can be performed on the polynomial expressed with r and s in order to retain the above symmetric property.

- **(c)** $H(x, y) = H(x^{-1}, y) = H(x, y^{-1}) = H(x^{-1}, y^{-1})$**.** This is the generalization of item (b) for both x and y axes. As a result, we can use the change of

[14]It can be shown by the method of mathematical induction.

[15]The corresponding filters with such transfer functions are generally noncausal, however, a proper shift in the spatial domain will make them causal and therefore such an assumption is reasonable.

variables $r = x + x^{-1}$ and $s = y + y^{-1}$, and the polynomial with the symmetry $H(x, y) = H(x^{-1}, y) = H(x, y^{-1}) = H(x^{-1}, y^{-1})$ is guaranteed expressible as a polynomial in r and s.

3.6 Sensitivity Analysis

This section evaluates the robustness of bivariate polynomial multiplication and factorization from the perspective of sensitivity[16]. We consider the multiplication process first. For a factorizable polynomial $H(x, y) = F(x, y) \cdot G(x, y)$, if the factor $F(x, y)$ has a slight perturbation while $G(x, y)$ remains the same, then the product $H(x, y)$ also has a perturbation; however, the energy of the perturbation in $F(x, y)$ and in $H(x, y)$ is generally different. To characterize the robustness with respect to perturbations, we define the multiplication sensitivity as the maximum expansion ratio of the normalized perturbation energy from $F(x, y)$ to $H(x, y)$; if we organize the coefficients of the polynomials $F(x, y)$ and $H(x, y)$ into vectors $\mathbf{f}$ and $\mathbf{h}$, respectively, then the sensitivity becomes

$$S_{F \to H} = \max_{\mathbf{\Delta f}} \frac{\|\mathbf{\Delta h}\|_2^2 / \|\mathbf{h}\|_2^2}{\|\mathbf{\Delta f}\|_2^2 / \|\mathbf{f}\|_2^2}, \tag{3.29}$$

where $\mathbf{\Delta f}$ and $\mathbf{\Delta h}$ correspond to the perturbations in $F(x, y)$ and $H(x, y)$, respectively, and $\|\cdot\|_2$ denotes the ℓ_2-norm of a vector. The value of $S_{F \to H}$ indicates the robustness in the multiplication process with a fixed polynomial $G(x, y)$; lower value of the sensitivity indicates higher robustness. Due to the symmetry between $F(x, y)$ and $G(x, y)$, the sensitivity $S_{G \to H}$ can be defined in the same approach.

Similarly, sensitivity can be defined for the factorization process to characterize the robustness from the product to its factors, i.e. the worst magnification of normalized energy from a perturbation in the product to that in the factors. However, a challenge is that a perturbation in the product may generally result in a non-factorizable polynomial or a polynomial factorizable into different degrees, which

[16]As a related topic, the sensitivity of univariate polynomial composition and decomposition is studied in [56].

makes the perturbation of the factors undefined. Therefore, this section considers only the perturbation in the product that retains the degrees of the factors. With this constraint, the sensitivity from the product to the factor becomes

$$S_{H\to F} = \max_{\mathbf{\Delta f}} \frac{\|\mathbf{\Delta f}\|_2^2/\|\mathbf{f}\|_2^2}{\|\mathbf{\Delta h}\|_2^2/\|\mathbf{h}\|_2^2} = \left(\min_{\mathbf{\Delta f}} \frac{\|\mathbf{\Delta h}\|_2^2/\|\mathbf{h}\|_2^2}{\|\mathbf{\Delta f}\|_2^2/\|\mathbf{f}\|_2^2}\right)^{-1}. \tag{3.30}$$

Similar with the multiplication process, lower factorization sensitivity $S_{H\to F}$ corresponds to higher robustness of the factors when their product is perturbed. The sensitivity $S_{H\to G}$ can be defined similarly.

The bilinear relationship of $H(x,y)$ with $F(x,y)$ and $G(x,y)$ implies the following matrix representation for polynomial multiplication,

$$\mathbf{h} = \mathbf{G}\cdot\mathbf{f}, \tag{3.31}$$

where the matrix $\mathbf{G}$ summarizes the linear dependence of $\mathbf{h}$ on $\mathbf{f}$ with a fixed polynomial $G(x,y)$. Similarly, the perturbations in $F(x,y)$ and in $H(x,y)$ satisfy $\mathbf{\Delta h} = \mathbf{G}\cdot\mathbf{\Delta f}$. Thus, the norm ratio between the two perturbations has the upper and lower bounds as

$$\sigma_{\mathbf{G},\min} \le \frac{\|\mathbf{\Delta h}\|_2}{\|\mathbf{\Delta f}\|_2} \le \sigma_{\mathbf{G},\max}, \tag{3.32}$$

where $\sigma_{\mathbf{G},\min}$ and $\sigma_{\mathbf{G},\max}$ are the minimal and maximal singular values of the matrix $\mathbf{G}$, respectively. Both the bounds in (3.32) are tight and achieved when $\mathbf{\Delta f}$ (up to a scaling factor) is the singular vector of $\mathbf{G}$ associated with the maximal or the minimal singular value, respectively.

The bounds in (3.32) simplify the sensitivities in (3.29) and (3.30), namely

$$S_{F\to H} = \sigma^2_{\mathbf{G},\max}\cdot\frac{\|\mathbf{f}\|_2^2}{\|\mathbf{h}\|_2^2}, \text{ and } S_{H\to F} = \frac{1}{\sigma^2_{\mathbf{G},\min}}\cdot\frac{\|\mathbf{h}\|_2^2}{\|\mathbf{f}\|_2^2}.$$

A natural bound for the two sensitivities above with a fixed polynomial $G(x,y)$ is

$$S_{F\to H} \le \frac{\sigma^2_{\mathbf{G},\max}}{\sigma^2_{\mathbf{G},\min}}, \quad S_{H\to F} \le \frac{\sigma^2_{\mathbf{G},\max}}{\sigma^2_{\mathbf{G},\min}},$$

which follows from the relationship (3.31).

Finally, we show empirical evaluation of the sensitivities in both the multiplication and the factorization processes. We use randomly generated polynomials $F(x, y)$ and $G(x, y)$, where each coefficient satisfies the standard Gaussian distribution. For simplicity, we use the same degrees for both x and y and for both factors $F(x, y)$ and $G(x, y)$, i.e. $\deg_x(F) = \deg_y(F) = \deg_x(G) = \deg_y(G)$. The degrees are swept from 1 to 15; at each degree, 100 pairs of factors are generated and the sensitivities are evaluated.

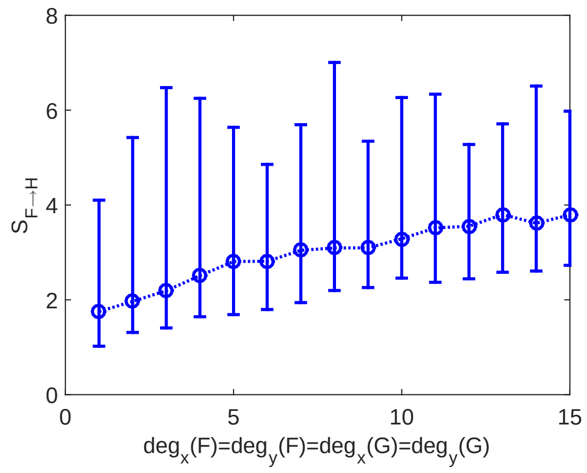
Figure 3-3 shows the median sensitivity among the 100 examples at each degree, with the vertical bars denoting the minimal and maximal sensitivity achieved. We can see that the sensitivity generally increases with the degrees of polynomials at a moderate rate, which seems to suggest that bivariate polynomial multiplication and factorization may have reasonable robustness with respect to perturbations[17]. In addition, the factorization sensitivity is generally higher than the multiplication sensitivity, which may imply that factorization is less robust with respect to input noise.

3.7 Chapter Conclusion and Future Work

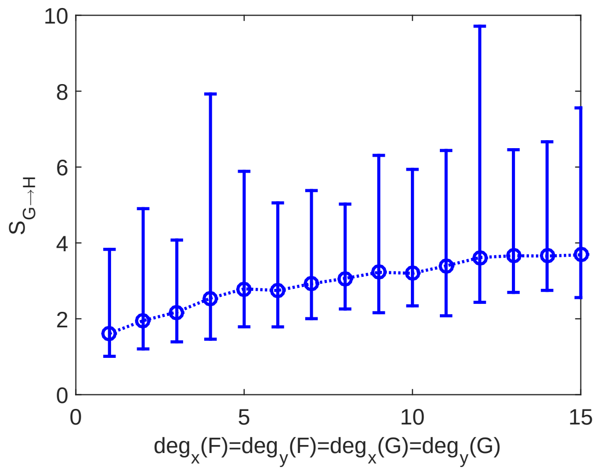
This chapter considers the cascade approximation of 2D FIR filters, which is an example of the linear systems that can be described by the operator composition in Section 1.1.2. This cascade structure gains computational efficiency by the reduction of the total number of independent parameters. Equivalently, the same technique can potentially be applied to the convolutional approximation of 2D signals that achieves representational compression.

The cascade approximation of 2D FIR filters becomes the approximate bivariate polynomial factorization in the transform domain, which this chapter focuses on. Two new factorization approaches are proposed, namely the zero-sum MIP and

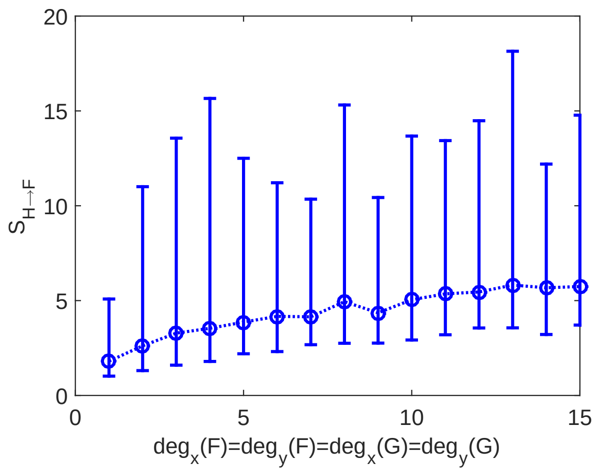
[17]Again, for factorization, we assume that the perturbed polynomial is factorizable and the degrees of factors remain the same.



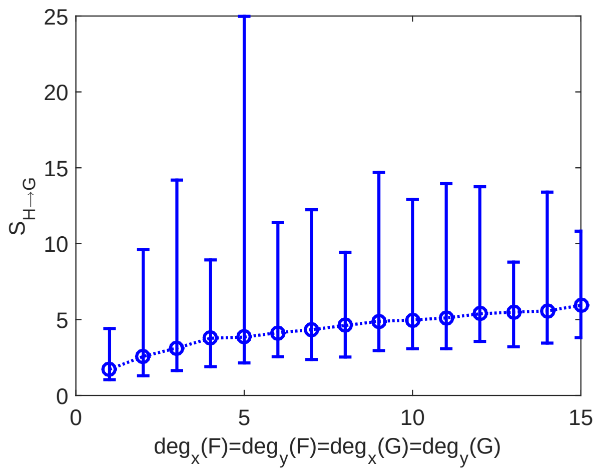
(a): Sensitivity $S_{F\to H}$



(b): Sensitivity $S_{G\to H}$



(c): Sensitivity $S_{H\to F}$



(d): Sensitivity $S_{H\to G}$

Figure 3-3: Sensitivities v.s. degrees of polynomials.

the lifted alternating minimization algorithms. Numerical simulations compare these approaches with baseline algorithms for low rank matrix approximation and another approach that performs alternating minimization of error directly on the two factors. A method is proposed to keep certain symmetric properties in the factorizable approximation, and the sensitivity of bivariate polynomial multiplication and factorization is characterized and evaluated.

Simulation results suggest that the lifted alternating minimization algorithm noticeably outperforms the other factorization algorithms, followed by the lifted SVP algorithm. The zero-sum MIP algorithm is sensitive to additional noise in the input polynomial while the other algorithms have relatively low sensitivity. Empirically, the lifted alternating minimization and the lifted SVP algorithms have higher success rates if the degrees of the two factors are not very close; if the two factors have the same degrees, then there are typically multiple optimal solutions to the formulation in the dimensionally lifted parameter space, possibly resulting in more complicated geometric structure of the solution space and lower efficiency of these algorithms.

There are a few directions that have potential for future work. First, there are approximation criteria for the 2D FIR design other than the ℓ_2-norm of error, e.g. the minimax error among all frequencies, with which new factorization algorithm can be explored. Second, for the zero-sum MIP algorithm, it is interesting to incorporate an idea in [101], which utilizes lower-order Taylor series at multiple points, aiming for potential improvement of robustness. Third, we would like to consider alternative cost functions in the lifted alternating minimization algorithm, which may still be tractable and more faithful to the approximation error in the polynomial coefficients. Fourth, we would like to have theoretical analysis on the lifted alternating minimization algorithm, with possible topics including the uniqueness of solution in the dimensionally lifted parameter space (when the two factors do not have the same degrees) and the convergence rate of the algorithm. Fifth, this chapter considers only the approximation of a bivariate polynomial with a single factorizable polynomial, and it would be interesting to consider the generalized situation where a summation of multiple factorizable polynomials is used in the approximation, which is related to the parallel form for 2D filter representation with each branch as a separable filter [3].

Chapter 4

Cascade Representation of Nonlinear Systems

This chapter continues the discussion on systems that can be interpreted as the operator composition in Section 1.1.2, and shifts the focus from linear systems to block-oriented representations of nonlinear systems; as is reviewed in Section 2.2.3, the block-oriented representations [27, 71] typically model a nonlinear system with an interaction of memoryless nonlinear functions and linear time-invariant (LTI) subsystems. We propose parameter estimation approaches for two discrete-time block-oriented nonlinear models, with the available information as the statistics or empirical observations of the input and output signals. The first block-oriented nonlinear model in Section 4.1 is a Hammerstein model [50] that is the cascade of a nonlinear memoryless module followed by a LTI subsystem, where the nonlinear module is a weighted combination over a basis of known functions and the LTI subsystem has no extra constraints. This setup is more general than the LTI subsystems considered in most existing works on the Hammerstein model estimation, which typically are constrained to be FIR filters [51] or filters with rational transfer functions [52]. A generalization of the two-step parameter estimation method in [52] is proposed and the optimal solution is derived with the respective criterion that is used in each of the two steps. For the second block-oriented nonlinear system in Section 4.2, we aim for modeling a black-box nonlinear system by its inverse, where

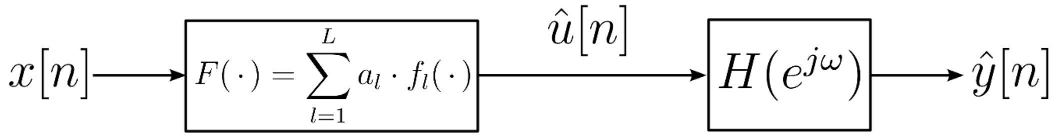


Figure 4-1: A Hammerstein model with the nonlinear module as a weighted combination of given functions.

the inverse system is a cascade of multiple nonlinear functions and LTI subsystems. This structure can be considered as a generalization of the all-pole signal modeling [26] by introducing nonlinear blocks. Parameter estimation algorithms are proposed for each block in the cascade.

4.1 Hammerstein Model with a General LTI Subsystem

4.1.1 Problem Formulation

In this section, the aim is to model a pair of input signal $x[n]$ and output signal $y[n]$ with a Hammerstein structure as shown in Figure 4-1. As mentioned above, a Hammerstein model is a cascade of two blocks, where the first block is a memoryless nonlinear function $F(\cdot)$ and the second is a LTI subsystem $H(e^{j\omega})$ with impulse response $h[n]$. For simplicity, we assume that the memoryless nonlinear module is a weighted combination of given functions, i.e.

$$F(x) = \sum_{l=1}^{L} a_l \cdot f_l(x), \tag{4.1}$$

where $f_l(\cdot)$ are the known functions and a_l are the weights that will be estimated. With the input signal as $x[n]$, the estimated output signal from the model in Figure 4-1 becomes

$$\hat{y}[n] = h[n] * \hat{u}[n] = h[n] * F(x[n]). \tag{4.2}$$

This section considers the estimation of the weights a_l in the nonlinear function and the impulse response $h[n]$ of the LTI subsystem, with the goal of making the estimated output signal $\hat{y}[n]$ close to the true output $y[n]$. The input $x[n]$, the output $y[n]$, and all $f_l(x[n])$ with $1 \le l \le L$ are assumed to be wide-sense stationary stochastic signals. Without loss of generality, we assume that the mean of all the mentioned signals is zero, i.e.

$$\mathbb{E}\left\{x[n]\right\} = \mathbb{E}\left\{y[n]\right\} = \mathbb{E}\left\{f_l(x[n])\right\} = 0, \text{ for all } 1 \le l \le L. \tag{4.3}$$

In addition, we assume that all auto-correlation and cross-correlation functions among $y[n]$ and $f_l(x[n])$ $(1 \le l \le L)$ are known, which are denoted as follows,

$$\begin{aligned} R_{y,f_l}[m] &= \mathbb{E}\left\{y[n+m] \cdot f_l^*(x[n])\right\}, \quad 1 \le l \le L, && (4.4)\\ R_{f_l,f_k}[m] &= \mathbb{E}\left\{f_l(x[n+m]) \cdot f_k^*(x[n])\right\}, \quad 1 \le l \le L,\ 1 \le k \le L, && (4.5)\end{aligned}$$

in which the superscript "*" denotes the complex conjugate. The corresponding power spectral density functions of $R_{y,f_l}[m]$ and $R_{f_l,f_k}[m]$ are denoted as $S_{y,f_l}(e^{j\omega})$ and $S_{f_l,f_k}(e^{j\omega})$, respectively. For simplicity, we further assume that all power spectral density functions are continuous, and the matrix

$$\mathbf{S}(e^{j\omega}) \triangleq \begin{bmatrix} S_{f_1,f_1}(e^{j\omega}) & \cdots & S_{f_1,f_L}(e^{j\omega}) \\ \vdots & \ddots & \vdots \\ S_{f_L,f_1}(e^{j\omega}) & \cdots & S_{f_L,f_L}(e^{j\omega}) \end{bmatrix} \tag{4.6}$$

has full rank at each value of ω over the interval $[-\pi, \pi]$.

A possible criterion for the difference between $y[n]$ and $\hat{y}[n]$ is the mean squared error, namely $\mathbb{E}\left\{|y[n] - \hat{y}[n]|^2\right\}$. However, the direct minimization of this cost function is challenging, since it is nonlinear with respect to the variables a_l and $h[n]$ due to the multiplicative terms between them.

To tackle this challenge, we consider a two-step parameter estimation algorithm, which is an extension of the method in [52]: in the first step, we obtain the optimal parameters in a *dimensionally lifted* space that corresponds to a more flexible system, where the objective function is the mean squared error of the output signal; in the second step, we determine the projection of the optimal parameters in the dimensionally lifted space (i.e. of the more flexible system) onto the subset that is consistent with the Hammerstein structure in Figure 4-1, where the approximation criterion is the difference of the parameters in the dimensionally lifted space. The details of this algorithm are in Section 4.1.3.

4.1.2 Relation to Existing Work

The closest work to our algorithm is [52], which proposes an *over-parametrization* method for certain Hammerstein structures. However, we would like to point out that there are noticeable differences between our work and the work [52]. First, the work [52] focuses on LTI subsystem with a rational transfer function, and consequently the parameter space has a finite dimension. Although LTI systems with rational transfer functions have many advantages including simple implementation, however, there are important LTI systems that do not have a rational transfer function, such as the ideal low-pass filters. In contrast to [52], we consider an arbitrary LTI subsystem and thus generalize the over-parametrization approach into systems with an infinite dimensional parameter space, which is less restrictive and does not require the choice of order of the rational transfer function. As a second difference between our work and [52], time-domain parameter estimation is used in [52], while our approach considers the over-parametrization in the frequency domain. More review on the parameter estimation for general block-oriented nonlinear models can be found in Section 2.2.3, which is less directly related to this section.

4.1.3 Two-step Parameter Estimation Algorithm

This algorithm has two steps with two different optimization criteria. In the first step, we consider a system with L parallel branches, where the l^{th} branch has a filter $H_l(e^{j\omega})$ with the input as $\tilde{u}_l[n] \triangleq f_l(x[n])$, which is shown in Figure 4-2. This

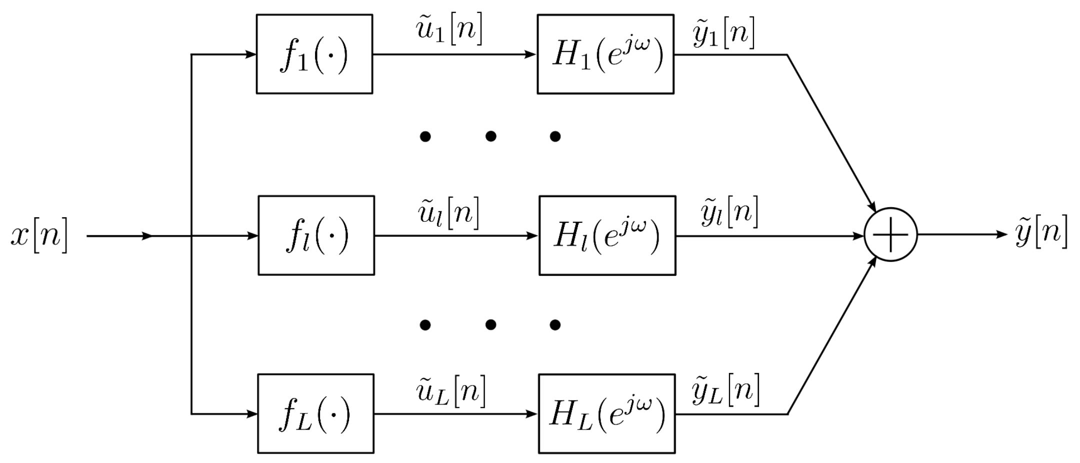


Figure 4-2: A parallel Hammerstein model with a dimensionally lifted parameter space.

system in Figure 4-2 can be regarded as to have a dimensionally lifted space[1] of the parameters compared with the system in Figure 4-1, since the latter is the special situation of the former with the additional constraints

$$H_l(e^{j\omega}) = a_l \cdot H(e^{j\omega}),\ 1 \le l \le L. \tag{4.7}$$

The criterion in this first step is to minimize the mean squared error between $y[n]$ and the estimated output $\tilde{y}[n]$ in Figure 4-2, as is formulated below:

$$\min_{H_1(e^{j\omega}),\dots,H_L(e^{j\omega})} \quad \mathbb{E}\left\{|y[n]-\tilde{y}[n]|^2\right\}. \tag{4.8}$$

In the dimensionally lifted parameter space, the formulation (4.8) is a linear least squares problem and the solution is the multi-input single-output Wiener filter [121, 122]. As presented in Appendix D, if the matrix $\mathbf{S}(e^{j\omega})$ in (4.6) has full rank at each

[1]Precisely speaking, both systems in Figure 4-1 and Figure 4-2 have countably infinite parameters; however, since the former system is a constrained form of the latter and the additional constraints (4.7) introduce dependence among the parameters, it is convenient to consider the dimension of the latter system to be higher than the former, in order to emphasize the additional flexibility of the latter system.

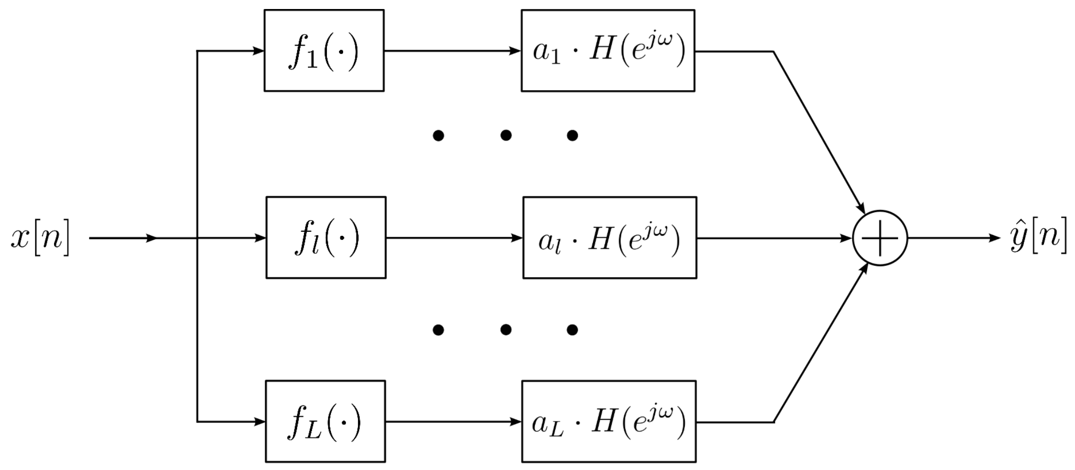

$x[n]$
$f_1(\cdot)$
$f_l(\cdot)$
$f_L(\cdot)$
$a_1 \cdot H(e^{j\omega})$
$a_l \cdot H(e^{j\omega})$
$a_L \cdot H(e^{j\omega})$
$\hat{y}[n]$

Figure 4-3: An equivalent block diagram for the system in Figure 4-1.

$-\pi \leq \omega < \pi$, then the optimal solution to (4.8) is [121, 122]

$$\begin{bmatrix} H_1^{\text{opt}}(e^{j\omega}) \\ \vdots \\ H_L^{\text{opt}}(e^{j\omega}) \end{bmatrix} = \left(\mathbf{S}^*(e^{j\omega})\right)^{-1} \cdot \begin{bmatrix} S_{y,f_1}(e^{j\omega}) \\ \vdots \\ S_{y,f_L}(e^{j\omega}) \end{bmatrix}. \tag{4.9}$$

where the superscript "*" again denotes the complex conjugate; the power spectral density functions $S_{y,f_l}(e^{j\omega})$ are associated with the correlation functions in (4.4).

In the second step of the parameter estimation algorithm, the goal is to approximate the multi-branch system in Figure 4-2 with the simpler system in Figure 4-1. Since the system in Figure 4-1 is equivalently represented as Figure 4-3, a possible optimization criterion for this step is the total squared difference in the LTI blocks in each branch between Figure 4-2 and Figure 4-3, which is formulated as follows

$$\min_{a_1,\ldots,a_L,H(e^{j\omega})} \sum_{l=1}^{L} \left\| H_l^{\text{opt}}(e^{j\omega}) - a_l \cdot H(e^{j\omega}) \right\|^2, \tag{4.10}$$

where the squared difference between two spectrums is defined as

$$\begin{aligned} \left\| H_l^{\text{opt}}(e^{j\omega}) - a_l \cdot H(e^{j\omega}) \right\|^2 &\triangleq \int_{-\pi}^{\pi} \left| H_l^{\text{opt}}(e^{j\omega}) - a_l \cdot H(e^{j\omega}) \right|^2 \mathrm{d}\omega \qquad (4.11) \\ &= 2\pi \cdot \sum_{n=-\infty}^{\infty} \left| h_l^{\text{opt}}[n] - a_l \cdot h[n] \right|^2, \end{aligned}$$

where $h_l^{\text{opt}}[n]$ is the impulse response of $H_l^{\text{opt}}(e^{j\omega})$, and the last step applies Parseval's theorem.

The formulation (4.10) still involves multiplicative terms between a_l and $H(e^{j\omega})$, which could be a challenge for optimization approaches such as gradient descent. Furthermore, the parameter space of the variables in (4.10) has an infinite dimension. However, there is a nice generalization of the singular value decomposition (SVD) technique from finite-dimensional to infinite-dimensional matrices, which can be utilized to solve (4.10). First, we define the following *quasimatrix*[2] $\mathbf{M} \in \mathbb{C}^{[-\pi,\pi]\times L}$

$$\mathbf{M} = \begin{bmatrix} | & & | \\ H_1^{\text{opt}}(e^{j\omega}) & \cdots & H_L^{\text{opt}}(e^{j\omega}) \\ | & & | \end{bmatrix}. \tag{4.12}$$

Then, the problem (4.10) becomes the generalized rank-one approximation of the quasimatrix $\mathbf{M}$ with the generalized Frobenius norm[3].

As a continuous analogue of the singular value decomposition of regular matrices, if each column of a quasimatrix is a continuous function, then this quasimatrix can also have a SVD factorization as stated by the theorem below [124–126].

Theorem 4.1. [124–126] *A quasimatrix $\mathbf{M} = [M_1(\omega), \cdots, M_L(\omega)]$ with each column as a continuous function over the interval $\omega \in [\omega_a, \omega_b]$ has the singular value decomposition as below*

$$\mathbf{M} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^\dagger, \tag{4.13}$$

where $\mathbf{U} \triangleq [U_1(\omega), \cdots, U_L(\omega)]$ is an $[\omega_a, \omega_b] \times L$ quasimatrix with orthonormal columns[4]*, $\mathbf{\Sigma}$ is an $L \times L$ real-valued diagonal matrix with diagonal elements $\sigma_1 \geq$*

[2]A quasimatrix [123] generally is a bivariate function where the first variable takes values from an interval and the second has a finite alphabet, i.e. each column of a quasimatrix is a function of a continuous-valued variable. A quasimatrix can also be regarded as an infinite-dimensional continuous analogue of a regular matrix.

[3]In the Frobenius norm, we replace the summation over indices of regular matrices by the integration over the continuous-valued variable of quasimatrices; for the matrix $\mathbf{M}$ in (4.12), its Frobenius norm is $\|\mathbf{M}\|_{\text{F}} = \sqrt{\sum_l \int |H_l^{\text{opt}}(\omega)|^2 \mathrm{d}\omega}$.

[4]In this section, we define the inner product between continuous functions over the interval $[\omega_a, \omega_b]$ as $\langle H_1(\omega), H_2(\omega)\rangle \triangleq \int_{\omega_a}^{\omega_b} H_1^*(\omega) \cdot H_2(\omega) \mathrm{d}\omega$.

$\sigma_2 \geq \cdots \geq \sigma_L \geq 0$, $\mathbf{V} \triangleq [\mathbf{v}_1, \cdots, \mathbf{v}_L]$ is an $L \times L$ unitary matrix, and the superscript "$\dagger$" denotes the conjugate transpose.

In addition, the generalized SVD for a quasimatrix provides its low-rank approximation that is optimal in the Frobenius norm, as shown by [126]:

Theorem 4.2. [126] *If the quasimatrix $\mathbf{M}$ has each column as a continuous function and has the SVD as in (4.13), then the partial summation*

$$\mathbf{P}_k \triangleq \sum_{m=1}^{k} \sigma_m \cdot U_m(\omega) \cdot \mathbf{v}_m^\dagger \tag{4.14}$$

is the optimal rank-k approximation to the quasimatrix $\mathbf{M}$ that minimizes the Frobenius norm of error $\|\mathbf{M} - \mathbf{P}_k\|_{\mathrm{F}}$.

Now we apply the above results on quasimatrices to the problem (4.10). Since we assume that all power spectral density functions are continuous, the quasimatrix $\mathbf{M}$ in (4.12) has each column as a continuous function. As a result, the optimal parameters $a_1, \cdots, a_L$ and $H(e^{j\omega})$ in (4.10) may be obtained by the generalized SVD of $\mathbf{M}$ and taking the singular vectors associated with the largest singular value, i.e.

$$\begin{aligned} a_l^{\text{opt}} &= v_{l,1}^*, \quad 1 \leq l \leq L, & (4.15)\\ H^{\text{opt}}(e^{j\omega}) &= \sigma_1 \cdot U_1(\omega), \quad -\pi \leq \omega \leq \pi, & (4.16) \end{aligned}$$

where $v_{l,1}$ is the l^{th} element of the column vector $\mathbf{v}_1$ from the matrix $\mathbf{V}$ in (4.13).

The above approach for the optimal parameters in (4.15) and (4.16) requires a direct computation of the SVD of the infinite-dimensional quasimatrix $\mathbf{M}$, which might be a conceptual or computational challenge. To simplify the solution, we can have an alternative approach to obtain the optimal parameters without explicitly computing the SVD of the quasimatrix, as we will show below. If we consider the following product

$$\mathbf{M}^{\dagger} \cdot \mathbf{M} = \begin{bmatrix} \langle H_1^{\text{opt}}(e^{j\omega}), H_1^{\text{opt}}(e^{j\omega}) \rangle & \cdots & \langle H_1^{\text{opt}}(e^{j\omega}), H_L^{\text{opt}}(e^{j\omega}) \rangle \\ \vdots & \ddots & \vdots \\ \langle H_L^{\text{opt}}(e^{j\omega}), H_1^{\text{opt}}(e^{j\omega}) \rangle & \cdots & \langle H_L^{\text{opt}}(e^{j\omega}), H_L^{\text{opt}}(e^{j\omega}) \rangle \end{bmatrix}, \tag{4.17}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product, then applying the SVD (4.13) in (4.17) leads to

$$\mathbf{M}^{\dagger} \cdot \mathbf{M} = \mathbf{V} \cdot \mathbf{\Sigma} \cdot \mathbf{U}^{\dagger} \cdot \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^{\dagger} = \mathbf{V} \cdot \mathbf{\Sigma}^2 \cdot \mathbf{V}^{\dagger} \tag{4.18}$$

where the last step uses the fact that $\mathbf{U}$ is orthonormal. Therefore, $\mathbf{V} \cdot \mathbf{\Sigma}^2 \cdot \mathbf{V}^{\dagger}$ is the SVD representation of the matrix $\big(\mathbf{M}^{\dagger} \cdot \mathbf{M}\big)$, which has the finite dimension of $L \times L$. As implied by (4.15), the optimal parameters a_l^{opt} are available from the SVD of the finite dimensional matrix $\big(\mathbf{M}^{\dagger} \cdot \mathbf{M}\big)$.

The optimal LTI system $H^{\text{opt}}(e^{j\omega})$ can then be directly obtained after a_l^{opt} are available. From (4.10) and (4.11), the optimal LTI system with fixed parameters a_l^{opt} satisfies

$$\begin{aligned} H^{\text{opt}}(e^{j\omega}) &= \arg\min_{H(e^{j\omega})} \sum_{l=1}^{L} |H_l^{\text{opt}}(e^{j\omega}) - a_l^{\text{opt}} \cdot H(e^{j\omega})|^2 \\ &= \arg\min_{H(e^{j\omega})} \left(\sum_{l=1}^{L} |a_l^{\text{opt}}|^2 \right) \cdot \left| H(e^{j\omega}) - \frac{\sum_{l=1}^{L} (a_l^{\text{opt}})^* \cdot H_l^{\text{opt}}(e^{j\omega})}{\sum_{l=1}^{L} |a_l^{\text{opt}}|^2} \right|^2 \\ &\qquad + \sum_{l=1}^{L} |H_l^{\text{opt}}(e^{j\omega})|^2 - \frac{|\sum_{l=1}^{L} (a_l^{\text{opt}})^* \cdot H_l^{\text{opt}}(e^{j\omega})|^2}{\sum_{l=1}^{L} |a_l^{\text{opt}}|^2} \\ &= \frac{\sum_{l=1}^{L} (a_l^{\text{opt}})^* \cdot H_l^{\text{opt}}(e^{j\omega})}{\sum_{l=1}^{L} |a_l^{\text{opt}}|^2} \\ &= \sum_{l=1}^{L} v_{l,1} \cdot H_l^{\text{opt}}(e^{j\omega}), \end{aligned} \tag{4.19}$$

in which the last step uses (4.15) as well as the fact that $\|\mathbf{v}_1\| = 1$ for the unitary matrix $\mathbf{V}$. In this alternative approach to obtain a_l^{opt} and $H^{\text{opt}}(e^{j\omega})$, we avoid explicitly computing the SVD of the infinite-dimensional quasimatrix $\mathbf{M}$.

In summary, the two-step parameter estimation algorithm for the Hammerstein system in Figure 4-1 is as follows.

Parameter Estimation for a Hammerstein System in Figure 4-1

(1) First step: with the goal as (4.8), compute the optimal solution in the dimensionally lifted space according to (4.9).

(2) Second step: The goal is (4.10).

(2.1) Compute $\left(\mathbf{M}^\dagger \cdot \mathbf{M}\right)$ in (4.17).

(2.2) Compute the SVD of the matrix $\left(\mathbf{M}^\dagger \cdot \mathbf{M}\right) = \mathbf{V} \cdot \mathbf{\Sigma}^2 \cdot \mathbf{V}^\dagger$.

(2.3) The optimal parameters a_l^{opt} are provided in (4.15), where $v_{l,1}$ $(1 \leq l \leq L)$ are the elements from the matrix $\mathbf{V}$.

(2.4) The optimal LTI system $H^{\text{opt}}(e^{j\omega})$ is given in (4.19).

As a final comment, if all signals $x[n]$, $f_l(x[n])$, and $y[n]$ are real-valued, then the symmetry properties of the Fourier transform show that the impulse responses $h_l^{\text{opt}}[n]$ in the first step of the parameter estimation procedure are real-valued. In the second step, $\left(\mathbf{M}^\dagger \cdot \mathbf{M}\right)$ is a real matrix, and thus the parameters a_l^{opt} from SVD are real-valued. Finally, the impulse response $h^{\text{opt}}[n]$ of the LTI subsystem is guaranteed real, which is consistent with our intuition.

4.2 Modeling a System by its Inverse with a Cascade Structure

This section considers modeling an invertible black-box nonlinear system by its inverse, where the inverse system is modeled by a cascade of multiple nonlinear functions and LTI subsystems. Inspired by the all-pole modeling [26] for which the estimation error formulated with the inverse system is easier to optimize than that with the original system, we consider the nonlinear systems whose inverse has simpler structures or can enable potentially easier parameter estimation. For simplicity, we focus on nonlinear systems whose inverse is a cascade of invertible blocks, each of which is either a memoryless monotonic polynomial or a FIR filter. Figure 4-4 shows the systems that we are interested in as well as the notations.

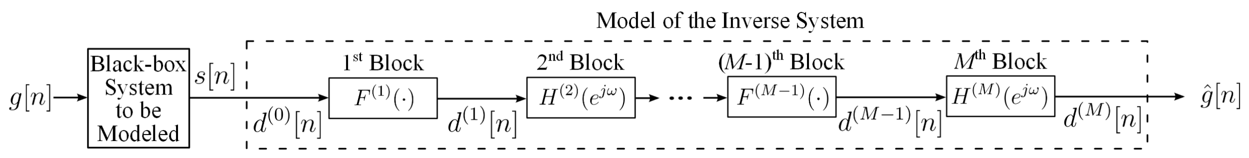


Figure 4-4: A block-oriented cascade model for the inverse system.

There are in total M subsystems in the model for the inverse system, and $d^{(m)}[n]$ denotes the output signal of the m^{th} subsystem. Memoryless monotonic polynomials $F^{(m)}(\cdot)$ and FIR filters $H^{(m)}(e^{j\omega})$ alternate in the cascade[5], and we assume the first subsystem in the modeled inverse is a memoryless polynomial in Figure 4-4. For simplicity, we assume that the memoryless polynomials have only odd-order terms with non-negative coefficients to ensure monotonicity, i.e.

$$F^{(m)}(x) \triangleq \sum_{l=1}^{L} \beta_{2l-1}^{(m)} \cdot x^{2l-1}, \quad \beta_{2l-1}^{(m)} \geq 0, \tag{4.20}$$

where $(2L-1)$ is the degree of the polynomial. The FIR filter $H^{(m)}(e^{j\omega})$ has order N and impulse response $h^{(m)}[n]$ $(0 \leq n \leq N)$. All FIR filters are assumed to be minimum phase, since the inverse filter of each FIR will be used to model the unknown black-box system in Figure 4-4. The input and output signals of the original black-box nonlinear system are denoted as $g[n]$ and $s[n]$, respectively. The output signal of the modeled inverse system is $\hat{g}[n]$. All signals in Figure 4-4 are assumed to be wide-sense stationary stochastic signals.

The goal of this section is to estimate the coefficients $\beta_l^{(m)}$ and the impulse responses $h^{(m)}[n]$ that minimize[6] $\mathbb{E}\{|e[n]|^2\}$, where $e[n] \triangleq g[n] - \hat{g}[n]$ denotes the residue error. If the parameter estimation could be performed for each subsystem in the inverse system in Figure 4-4, then the original black-box system can be modeled by the cascade of the inverse of each subsystem in the reverse order, i.e. the cascade of the filter with transfer function $1/H^{(M)}(e^{j\omega})$, the inverse function of $F^{(M-1)}(\cdot)$, ...,

[5]If two consecutive subsystems are of the same type (polynomials or FIR filters), then we can combine them into a single subsystem of this type.

[6]In this thesis, we assume that all stochastic processes involved are ergodic, so that we do not explicitly distinguish between the ensemble average (i.e. average over the probability space) and the time average.

and the inverse function of $F^{(1)}(\cdot)$.

For each individual subsystem, sufficient statistical information between its input and output signals typically enables simple estimation of its parameters; as an example, the optimal solution is available with the criterion as the minimal mean squared error in the output signal of the subsystem under consideration, since the output depends linearly on the parameters with a given input signal. However, the intermediate signals $d^{(m)}[n]$ $(1 \leq m \leq M-1)$ are usually not available in reality; since the cascade of multiple subsystems results in nonlinear dependence of the output signal $\hat{g}[n]$ on the parameters, the estimation is challenging when only $s[n]$ and $g[n]$ are known.

There are a few parameter settings which result in simpler solution compared with the general formulation. First, if there is only a single FIR block in the inverse system in Figure 4-4, then our formulation becomes the standard all-pole modeling problem to which there is well-studied solution [26]. Second, if there are no more than three subsystems in the cascade, then the inverse system belongs to the Hammerstein, the Wiener, the Hammerstein-Wiener, or the Wiener-Hammerstein models, all of which have been studied for parameter estimation [27]. Compared with the special parameter settings above, a general system in Figure 4-4 has higher flexibility and thus may model a larger class of nonlinear systems, which therefore is worth exploration. However, to the best of the author's knowledge, cascade models with more than three subsystems have been seldom studied; some existing work assumes that certain intermediate signals in addition to the input and output signals are also available for parameter estimation [127, Chapter 2.5], which is not always a practical assumption and thus limits the application of the associated algorithms.

A topic related to our problem is the inversion of a Volterra series model [69], where the goal is to design an inverse system so that the cascade of the original and the inverse systems approximates the identity system. There are two main branches for Volterra series inversion, namely the p^{th}-order inverse [128] and the indirect learning method [84, 129]. The p^{th}-order inverse aims to nullify all the kernels from the 2^{nd}-order to the p^{th}-order in the cascade of the two systems, which typically assumes the knowledge of the Volterra kernels of the original system and has a different criterion as our problem. The indirect learning method aims to solve the pre-distortion problem (i.e. the original system follows the designed inverse system in the cascade); typically, the core idea is replacing the pre-distortion task with a post-distortion problem (i.e. the cascade in the reverse order) and applying adaptive filtering techniques to the latter problem. In contrast to the indirect learning approach, our problem focuses on the system structure in Figure 4-4 that is already formulated with the post-distortion setup; in addition, we use a special cascade structure for the inverse system instead of a general Volterra model.

In Section 4.2.1, three algorithms are presented for the parameter estimation of the cascade model. The first algorithm solves the Yule-Walker equations [130, 131] to estimate the FIR subsystems and formulates quadratic programs to determine the monotonic polynomial blocks. The second algorithm is a block coordinate descent algorithm that updates the parameters in a single subsystem with those in the other subsystems fixed. The third approach is the gradient descent approach for nonlinear optimization [132]. These algorithms will be evaluated and compared in Section 4.2.2.

4.2.1 Parameter Estimation Algorithms

Parameter Estimation using the Yule-Walker Equations

This algorithm performs parameter estimation for the m^{th} subsystem from $m = 1$ to $m = M$ in the cascade inverse model in Figure 4-4. In this sequential estimation process, each FIR filter is obtained by the solution to the Yule-Walker equations [130, 131], and each monotonic polynomial block $F^{(m)}(\cdot)$ is solved by a quadratic program that minimizes $\mathbb{E}\left\{|g[n] - d^{(m)}[n]|^2\right\}$.

The Yule-Walker equations are used in the all-pole modeling formulation [26], which does not involve the nonlinear subsystems that are used in our problem. The solution to the Yule-Walker equation is guaranteed to be minimum phase [26], which is an advantage in our setup since inverting the modeled inverse system requires that the FIR subsystems be minimum phase.

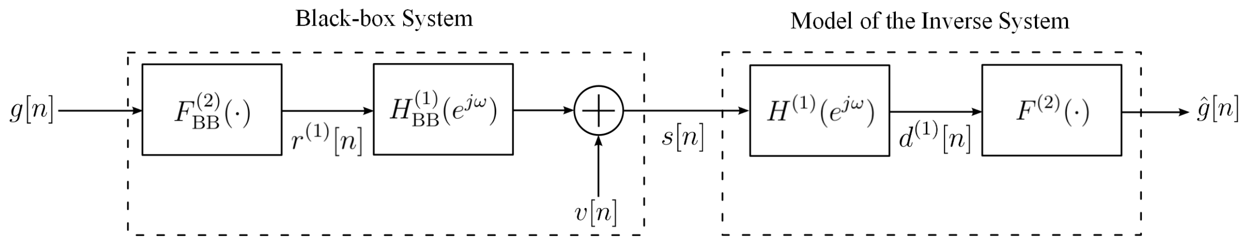


Figure 4-5: A special black-box system for which all-pole modeling can be desirable.

We first consider a special situation in which the solution to the Yule-Walker equations has potentially desirable properties. In this special situation, the original black-box system in Figure 4-4 is indeed a cascade of a memoryless monotonic nonlinear function $F^{(2)}_{\text{BB}}(\cdot)$ and a causal filter $H^{(1)}_{\text{BB}}(e^{j\omega})$ plus an independent zero-mean observation noise $v[n]$, as shown by Figure 4-5. Without loss of generality, we assume that the impulse responses of $H^{(1)}_{\text{BB}}(e^{j\omega})$ and $H^{(1)}(e^{j\omega})$ satisfy $h^{(1)}_{\text{BB}}[0] = h^{(1)}[0] = 1$. Using this setup, a possible criterion[7] for the estimation of the FIR filter $H^{(1)}(e^{j\omega})$ is

$$\begin{aligned} \min_{h^{(1)}[n]} \quad & \mathbb{E}\left\{|d^{(1)}[n] - r^{(1)}[n]|^2\right\} \\ \text{s.t.} \quad & h^{(1)}[0] = 1 \end{aligned} \tag{4.21}$$

where $r^{(1)}[n]$ is the signal in the black-box system between the nonlinear function and the causal filter.

If the signal $r^{(1)}[n]$ is white and has zero-mean, then the objective function in (4.21) is equivalent as the minimization of the signal power $\mathbb{E}\left\{|d^{(1)}[n]|^2\right\}$ [26]. Specifically, the function in (4.21) can be simplified as

$$\begin{aligned} & \mathbb{E}\left\{|d^{(1)}[n] - r^{(1)}[n]|^2\right\} \\ = \ & \mathbb{E}\left\{|d^{(1)}[n]|^2\right\} + \mathbb{E}\left\{|r^{(1)}[n]|^2\right\} - 2 \cdot \mathbb{E}\left\{d^{(1)}[n] \cdot r^{(1)}[n]\right\} \\ = \ & \mathbb{E}\left\{|d^{(1)}[n]|^2\right\} + \mathbb{E}\left\{|r^{(1)}[n]|^2\right\} - 2 \cdot \mathbb{E}\left\{s[n] \cdot r^{(1)}[n]\right\} && (4.22) \\ = \ & \mathbb{E}\left\{|d^{(1)}[n]|^2\right\} + \mathbb{E}\left\{|r^{(1)}[n]|^2\right\} - 2 \cdot \mathbb{E}\left\{|r^{(1)}[n]|^2\right\} && (4.23) \\ = \ & \mathbb{E}\left\{|d^{(1)}[n]|^2\right\} - \mathbb{E}\left\{|r^{(1)}[n]|^2\right\}, && (4.24) \end{aligned}$$

[7]The criterion (4.21) can be different from the mean squared error in $g[n]$, i.e. $\mathbb{E}\left\{|g[n] - \hat{g}[n]|^2\right\}$.

where the step (4.22) uses the constraint $h^{(1)}[0] = 1$ as well as $\mathbb{E}\left\{s[n-m] \cdot r^{(1)}[n]\right\} = 0$ (for $m \geq 1$) which results from the assumption that $r^{(1)}[n]$ is a white sequence and uncorrelated with the zero-mean observation noise $v[n]$; these properties on correlation and the constraint $h^{(1)}_{\mathrm{BB}}[0] = 1$ are applied similarly in the step (4.23).

Since $\mathbb{E}\left\{|r^{(1)}[n]|^2\right\}$ in (4.24) is independent of the FIR filter $H^{(1)}(e^{j\omega})$, the objective function in (4.21) is the same as the minimization of the signal power $\mathbb{E}\left\{|d^{(1)}[n]|^2\right\}$; by the same derivation as the all-pole modeling, the optimal parameters are the solution to the Yule-Walker equations with the auto-correlation function of the signal $s[n]$.

In terms of the signal $g[n]$ that we have access to, a sufficient condition to ensure $r^{(1)}[n]$ be a white and zero-mean sequence is: (i) $g[n]$ is an i.i.d. sequence, (ii) the probability density function of $g[n]$ is symmetric with respect to the origin, and (iii) the nonlinear function $F^{(2)}_{\mathrm{BB}}(\cdot)$ is an odd function. If $F^{(2)}_{\mathrm{BB}}(\cdot)$ is the inverse function of the class of the parametric functions in (4.20), then it is an odd function and (iii) is satisfied.

For the system in Figure 4-5, after obtaining the system $H^{(1)}(e^{j\omega})$ from the Yule-Walker equations, the parameter estimation for the system $F^{(2)}(\cdot)$ can be performed by minimizing the difference between the model output $\hat{g}[n]$ and the true input signal to the black-box system $g[n]$, with the constraints $\beta^{(2)}_{2l-1} \geq 0$ where the parameters $\beta^{(2)}_{2l-1}$ are defined in (4.20). Since an estimate of the input signal $d^{(1)}[n]$ of the system $F^{(2)}(\cdot)$ is available, this formulation is a quadratic programming problem[8], for which there are various solvers available [133, 134]. As a result, we have established a parameter estimation algorithm for the system in Figure 4-5 under the condition that $r^{(1)}[n]$ is a white and zero-mean sequence, for which the solution to the Yule-Walker equations can potentially be justified since it minimizes the cost in (4.21).

For a system more general than that in Figure 4-5, the above method can also be used for parameter estimation, with FIR filters and monotonic polynomial blocks obtained by the solution to the Yule-Walker equations and associated quadratic programs, respectively. However, for a general system, we are not aware of potential

[8]It is not the standard linear least squares problem due to the constraints $\beta^{(2)}_{2l-1} \geq 0$.

justification that is similar to the formulation (4.21).

Block Coordinate Descent Algorithm

As mentioned previously, knowledge of all the intermediate signals $d^{(m)}[n]$ $(1 \leq m \leq M-1)$ would simplify the parameter estimation problem for all subsystems in Figure 4-4. In fact, if all intermediate signals are known and the criterion is set as the mean squared error in the output signal of each subsystem, then the formulation becomes a constrained quadratic program. Although the true values of the intermediate signals are unavailable, we could use an estimation of those signals in the parameter estimation process.

A possible iterative approach for parameter estimation is to update the parameters in a single subsystem while fixing the current parameters in all other subsystems, i.e. the block coordinate descent method if the parameters in a single subsystem is regarded as a block of coordinates. In an iteration to update the parameters in the m^{th} subsystem in Figure 4-4, its input signal $d^{(m-1)}[n]$ can be estimated by passing $s[n]$ through the $(m-1)$ preceding subsystems with their respective current parameters; an estimate for its output signal $d^{(m)}[n]$ is available by passing the signal $g[n]$ through the inverse of each subsystem following the m^{th} subsystem, with the reverse order as in the cascade in Figure 4-4. With the estimated input and output signals of the m^{th} subsystem, the problem is simplified as the parameter estimation for a single subsystem, which can be solved as follows.

As discussed in the method above that is based on the Yule-Walker equations, for a nonlinear function $F^{(m)}(\cdot)$, the minimization of the mean squared error in its output signal $d^{(m)}[n]$ is formulated as a linearly constrained quadratic program when both its input and output signals are available, for which we can apply optimization solvers [133, 134]. However, for a FIR subsystem $H^{(m)}(e^{j\omega})$, a direct minimization of the mean squared error in its output signal (i.e. the least squares solution) may result in a potentially non-minimum-phase FIR filter, which does not have a stable and causal inverse system and thus is improper for our purpose of modeling the black-box system in Figure 4-4. Furthermore, solving the least squares formulation with the minimum-phase constraint is generally a non-convex optimization problem and seems challenging. Therefore, the following heuristic is used: for a zero of the FIR filter from the least squares solution that is outside the unit circle, we replace it with its conjugate reciprocal. By this replacement procedure, the obtained FIR filter are guaranteed minimum phase and the magnitude response remains the same as the FIR from the least squares solution.

This block coordinate descent algorithm makes use of the special cascade structure of the system in Figure 4-4; however, since different iterations have essentially different objectives to minimize (i.e. the energy of the output signal error of the subsystem to be updated), we are unaware of convergence guarantees of this algorithm. In addition, this algorithm considers locally optimal points, and thus the result is not necessarily the globally optimal parameters.

Gradient Descent Approach

The gradient descent approach for nonlinear optimization [132] can be applied to our parameter estimation problem with the objective function chosen as $\mathbb{E}\left\{|g[n]-\hat{g}[n]|^2\right\}$. Due to the cascade structure of the system in Figure 4-4, the output signal $\hat{g}[n]$ can be expressed as the following function composition in terms of the parameters of each subsystem

$$\hat{g}[n] = h^{(M)}[n] * F^{(M-1)}\left(h^{(M-2)}[n] * F^{(M-3)}\left(\cdots\ h^{(2)}[n] * F^{(1)}\left(s[n]\right)\ \cdots\right)\right),$$

where "$*$" denotes convolution. From this function composition structure, the gradient with respect to the parameters of each subsystem can be efficiently computed by repeatedly applying the chain rule of the derivatives, which has also be utilized in the back-propagation technique for training a neural network model [135].

This algorithm has the benefit of a unified objective function for optimization. With a sufficiently small step size for each gradient descent iteration, this method generally converges to a local minimum. On the other hand, a small step size may reduce the speed of convergence. Furthermore, this algorithm may also require additional steps to ensure that the estimated FIR filters are minimal-phase and the nonlinear functions have all coefficients satisfying $\beta_{2l-1}^{(m)} \geq 0$, in order to guarantee system invertibility.

4.2.2 Numerical Evaluation

This section evaluates the three algorithms above by using synthetic signals and systems, with the following parameter settings. Synthetic signals $s[n]$ are generated by passing a white Gaussian signal $g[n]$ through a *black-box* system, whose inverse system is a cascade of nonlinear functions $F^{(m)}(\cdot)$ and FIR filters $H^{(m)}(e^{j\omega})$ as in Figure 4-4. In addition to noiseless synthetic signal, we also consider noisy signals by adding independent noise on $s[n]$ with SNR at 40dB.

For data generation, the cascade inverse system in Figure 4-4 has the following parameter settings. The total number of subsystems M takes the values 2, 4, and 6. The first subsystem in the cascade inverse can either be a memoryless nonlinear function or a FIR filter (denoted by "Poly" and "FIR" in the "First System" column in Table 4.1, respectively). The order of each FIR filter is either $N = 1$ or $N = 2$ (i.e. length of its impulse response is 2 or 3), and the roots of each FIR are uniformly distributed in the circle centered at the origin with radius 0.9 in the z-transform domain. The variable L in (4.20) takes the values of 2 and 3, and the coefficients $\beta_l^{(m)}$ follow the Gaussian distribution (on non-negative values only). For each parameter setting, a total of 50 test trails are generated by creating 10 examples of the systems in Figure 4-4 and applying 5 randomly generated input signals $g[n]$ to each system.

The aim is to determine the parameters in each block of the cascade inverse system, in order that the output signal $\hat{g}[n]$ of the cascade inverse with input signal $s[n]$ is close to the true synthetic signal $g[n]$. In particular, we consider the parameter estimation is successful if the SNR between the signal $g[n]$ and the error signal $(g[n] - \hat{g}[n])$ is larger than or equal to 20dB. The block coordinate descent algorithm has the maximum iteration as $100M$ steps[9]; the gradient descent algorithm has the step size

[9]Each iteration updates a single subsystem in the block coordinate descent algorithm, and $100M$ steps of iteration update every subsystem 100 times.

as 2×10^{-5} and the maximum iteration[10] as 1000 steps.

The percentage of successful parameter estimation among the 50 trails at each parameter setting is recorded and displayed in Table 4.1. The columns with "Alg: Y-W", "Alg: BCD", and "Alg: GD" respectively correspond to the percentage of successful parameter estimation of the method based on the Yule-Walker equations, the block coordinate descent, and the gradient descent algorithms; for the two percentage values in each grid of these three columns, the first is the value for noiseless signal $s[n]$ and the second for noisy signal $s[n]$ with 40dB SNR.

We can have the following observations from Table 4.1. First, as the increase of the total number of subsystems in the cascade and the orders of each subsystem, the challenge of parameter estimation increases. Second, the gradient descent algorithm has the highest overall empirical performance among the three algorithms when the cascade system has more than two subsystems, possibly due to the unified objective function that is used in the optimization process; however, when the cascade has two subsystems, the block coordinate descent algorithm seems slightly superior than the gradient descent algorithm. Third, comparing the performance of the approach based on the Yule-Walker equations with different parameter settings, this method has the best performance for situations where the inverse system is a Wiener model (i.e. $M = 2$, and the first system is "FIR"), possibly due to the justification in Section 4.2.1. In contrast, for other settings the above justification does not apply and thus the success rate substantially reduces.

4.3 Chapter Conclusion and Future Work

This chapter considers two nonlinear models that can be interpreted as operator composition in Section 1.1.2. Both models are block-oriented representations for nonlinear systems by a cascade of memoryless nonlinear functions and LTI subsystems, and parameter estimation algorithms are developed for each of these

[10]Each iteration in the gradient descent approach is much more efficient than that in the block coordinate descent algorithm, so the comparison is fair with more iterations allowed in the gradient descent approach.

Table 4.1: Success percentage of the parameter estimation algorithms for cascade inverse systems in Figure 4-4. The two percentage values in each grid of the last three columns correspond to noiseless signal $s[n]$ and noisy signal $s[n]$ with 40dB SNR, respectively.

M	First System	N	L	Alg: Y-W	Alg: BCD	Alg: GD
2	FIR	1	2	94 / 94	100 / 100	100 / 100
2	FIR	1	3	92 / 92	100 / 100	100 / 100
2	FIR	2	2	88 / 86	100 / 100	90 / 88
2	FIR	2	3	76 / 66	100 / 90	86 / 88
2	Poly	1	2	40 / 40	100 / 100	100 / 100
2	Poly	1	3	30 / 30	100 / 100	100 / 92
2	Poly	2	2	10 / 10	100 / 80	90 / 70
2	Poly	2	3	20 / 20	100 / 88	100 / 80
4	FIR	1	2	16 / 14	18 / 8	74 / 72
4	FIR	1	3	0 / 0	24 / 20	44 / 40
4	FIR	2	2	0 / 0	20 / 20	38 / 28
4	FIR	2	3	0 / 0	10 / 0	2 / 0
4	Poly	1	2	22 / 16	50 / 50	90 / 56
4	Poly	1	3	30 / 24	80 / 60	60 / 50
4	Poly	2	2	8 / 6	30 / 24	70 / 48
4	Poly	2	3	0 / 0	10 / 0	58 / 26
6	FIR	1	2	2 / 0	10 / 8	40 / 34
6	FIR	1	3	0 / 0	0 / 0	4 / 0
6	FIR	2	2	0 / 0	0 / 0	0 / 0
6	FIR	2	3	0 / 0	0 / 0	0 / 0
6	Poly	1	2	0 / 0	10 / 10	20 / 20
6	Poly	1	3	4 / 2	10 / 10	18 / 14
6	Poly	2	2	0 / 0	0 / 0	22 / 20
6	Poly	2	3	0 / 0	0 / 0	0 / 0

models.

The first model belongs to the Hammerstein structure, in which the nonlinear function is constrained to be a weighted combination of known functions and the linear subsystem is a general filter. We generalize the two-step dimensionally lifting parameter estimation procedure for this problem, where the parameter space is infinite dimensional. The first step obtains the parameters in a modified system with a dimensionally lifted parameter space, and the goal is to minimize the mean squared error of the output signal. The second step projects the parameters of the modified system to those of the original Hammerstein system with the criterion as the difference in the parameter space. The theoretical optimal solution for the respective criterion in each of the two steps is derived under mild conditions; although the second step involves a generalized SVD of an infinite dimensional quasimatrix, it is shown that the SVD can be equivalently performed with a regular matrix of finite dimensions. There are a few directions for future work. Since the second step in the parameter estimation algorithm uses a different criterion (i.e. the difference in the parameter space) from the squared error of the output signal, it is interesting to identify conditions under which the criterion in the parameter space is equivalent to or a reasonable approximation for the output squared error. It is also meaningful to consider other criteria that are different from the squared error for parameter estimation.

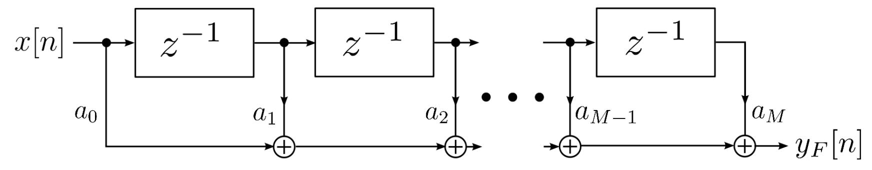
The second model in this chapter considers modeling a nonlinear system by its inverse, where the model for the inverse system has the structure of a cascade of multiple subsystems with alternated nonlinear memoryless functions and FIR filters. This may be considered as a generalization of the all-pole modeling technique, and the introduction of nonlinear functions makes the model more flexible while the parameter estimation generally becomes more challenging. Three methods are presented, namely the algorithm based on the Yule-Walker equations, the block coordinate descent approach, and the gradient descent method. The approach based on the Yule-Walker equations is justified for a special situation, where the cascade inverse system belongs to the Wiener model and the signals satisfy various symmetry properties. The other two approaches may converge to locally optimal solutions. From simulation results, the gradient descent method overall outperforms the other two approaches; the parameter estimation task becomes challenging with the increase of the total number of blocks in the cascade and the order of each subsystem, which leaves room for further improvement. Future work include alternative methods for the estimation of FIR filters to ensure the minimum-phase property, as well as parameter estimation with a different criterion. The proper choice of the step size in the gradient descent method also worths further consideration.

Chapter 5

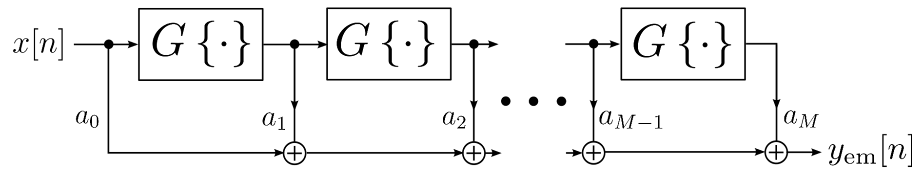
Modular Volterra System

Following the discussion on nonlinear systems in the previous chapter, this chapter focuses on the *modular Volterra system* that refers to the system obtained by replacing each delay element in a FIR filter $F(z)$ with an identical copy of a Volterra series module $G\{\cdot\}$, where the Volterra series model is reviewed in Section 2.2.1. This modular Volterra system corresponds to modular composition as is discussed in Section 1.1.4. When each Volterra module $G\{\cdot\}$ has only the linear kernel, the modular Volterra system becomes the modular FIR filter that is systematically studied in [1,2]. As an advantage of this modular Volterra structure, it achieves system modularity that enables simpler design and verification for hardware implementation with very-large-scale integration (VLSI) techniques [2, 48, 49].

In this chapter, we consider parameter estimation for a modular Volterra system, using statistical information between the input and output signals. As will be discussed later, the output signal of the modular system can potentially have a complicated nonlinear dependence on the kernels, which makes the parameter estimation challenging in the general setting. The Volterra series module $G\{\cdot\}$ is a superposition of a linear operator and nonlinear operators; for simplification, we assume that the magnitude of the coefficients of the nonlinear kernels is sufficiently smaller than that of the linear kernel. This assumption enables an efficient two-step algorithm to approximately estimate the parameters, where the first step estimates the coefficients of the linear kernel of $G\{\cdot\}$ and the FIR filter $F(z)$, and the second step obtains the coefficients of the nonlinear kernels of $G\{\cdot\}$.



(a): FIR filter $F(z)$



(b): Modular Volterra system $H\{\cdot\}$

Figure 5-1: Modular Volterra system with $F(z)$ and $G\{\cdot\}$.

5.1 Problem Definition

We focus on the modular Volterra system $H\{\cdot\}$ as shown in Figure 5-1 (b). In this system, the FIR filter in Figure 5-1 (a) has the transfer function

$$F(z) = \sum_{m=0}^{M} a_m \cdot z^{-m}$$

and is implemented in the direct form structure. The k^{th}-order kernels of the Volterra module $G\{\cdot\}$ and the modular system $H\{\cdot\}$ are denoted as $g^{(k)}[i_1, \ldots, i_k]$ and $h^{(k)}[i_1, \ldots, i_k]$, respectively. We assume that the highest order of the kernels of $G\{\cdot\}$ is K; each kernel in $G\{\cdot\}$ is causal and has the maximum time delay (i.e. the number of state variables) of N. For simplicity, we further assume that the system module $G\{\cdot\}$ has the constant offset as $G^{(0)} = 0$, and therefore its lowest-order kernel is the linear kernel.

The goal of this chapter is to estimate the coefficients of the kernels of $G\{\cdot\}$ and a_i $(0 \leq i \leq M)$ of the FIR filter $F(z)$, using the statistical information between the input signal $x[n]$ and the output signal $y[n]$; as an example, the statistics can be in the form of higher-order correlation functions or equivalently higher-order spectra [72,136]. For simplicity, we assume that all involved stochastic signals are stationary in terms of the higher-order statistics.

A general Volterra module $G\{\cdot\}$ may result in complicated kernels of the modular system $H\{\cdot\}$. As an example, the cascade of an order-P and an order-Q Volterra kernels becomes a Volterra kernel with order-(PQ). Therefore, a kernel of a cascade Volterra system may have a complicated relationship with the lower-order kernels of each individual subsystem in the cascade. Since the modular Volterra system involves a cascade of Volterra modules, the kernels of $H\{\cdot\}$ typically have complicated dependence on those of $G\{\cdot\}$, which may make parameter estimation for $G\{\cdot\}$ challenging.

To avoid the mentioned complication for parameter estimation, a possible method is to assume that the coefficients of the nonlinear kernels of the module $G\{\cdot\}$ have sufficiently smaller magnitude compared with those of the linear kernel; mathematically, the coefficients of the nonlinear kernels satisfy

$$|g^{(k)}[i_1,\ldots,i_k]| \le \varepsilon \cdot \left(\max_{0\le i\le N} |g^{(1)}[i]|\right), \quad \text{for all } i_1,\ldots,i_k,\ k\ge 2, \tag{5.1}$$

where $\varepsilon \ll 1$. In other words, if the linear kernel has coefficients $g^{(1)}[i] = O(1)$, then the coefficients of the nonlinear kernels satisfy $g^{(k)}[i_1,\ldots,i_k] = O(\varepsilon)$, which has an equivalent expression as

$$g^{(k)}[i_1,\ldots,i_k] = \varepsilon \cdot g^{(k)}_{\mathrm{nrm}}[i_1,\ldots,i_k],$$

where we refer to $g^{(k)}_{\mathrm{nrm}}[i_1,\ldots,i_k]$ as the *normalized coefficients* of the nonlinear kernels, which satisfy $g^{(k)}_{\mathrm{nrm}}[i_1,\ldots,i_k] = O(1)$.

We further assume that all signals in the entire modular Volterra system are uniformly bounded, and this bound is sufficiently small[1]; moreover, the input signal $x[n]$ is assumed not in the null space of the linear operator of the Volterra module

[1]Precisely, the bound on the signal magnitude is assumed to guarantee that the terms $r_q(\cdot)$ in the later Equation (5.2) satisfy $r_q(\cdot) \ll 1/\varepsilon$.

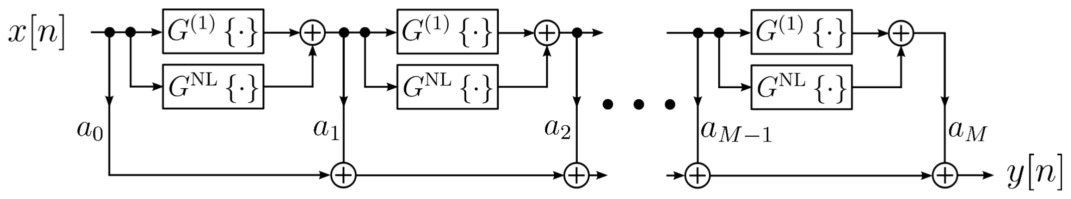


Figure 5-2: Modular Volterra system with linear and nonlinear branches for each module.

$G\{\cdot\}$, which is denoted as $G^{(1)}\{\cdot\}$.

Under the above assumptions, the output signal $y[n]$ of the modular Volterra system can be expanded as the following power series with respect to ε,

$$y[n] = \sum_{q=0}^{\infty} r_q\left(x[n-l], a_i, g^{(1)}[i], g^{(k)}_{\mathrm{nrm}}[i_1, \dots, i_k]\right) \cdot \varepsilon^q, \tag{5.2}$$

where the coefficients $r_q(\cdot)$ of this power series are functions of the input signal $x[n-l]$ $(0 \le l \le NM)$, the coefficients of the FIR filter $F(z)$ and the linear kernel of $G\{\cdot\}$, and the normalized coefficients $g^{(k)}_{\mathrm{nrm}}[i_1, \dots, i_k]$ of the nonlinear kernels of $G\{\cdot\}$. In this power series expansion, $r_0(\cdot)$ is the output signal of the modular Volterra system when each Volterra module has only the linear operator $G^{(1)}\{\cdot\}$. If we fully expand the functions $r_q(\cdot)$ into summation of the product terms of the input signal and the coefficients in $F(z)$ and $G\{\cdot\}$, then each term in $r_q(\cdot)$ has exactly q factors that belong to the normalized coefficients $g^{(k)}_{\mathrm{nrm}}[i_1, \dots, i_k]$. Consequently, the truncated approximation below

$$y[n] \approx \sum_{q=0}^{1} r_q\left(x[n-l], a_i, g^{(1)}[i], g^{(k)}_{\mathrm{nrm}}[i_1, \dots, i_k]\right) \cdot \varepsilon^q \tag{5.3}$$

corresponds to the terms in the output signal that involve at most a single coefficient of the nonlinear kernels of $G\{\cdot\}$.

As is shown in Figure 5-2, each Volterra module $G\{\cdot\}$ can be represented as a parallel system with two branches, one with the linear operator $G^{(1)}\{\cdot\}$ and the other with the nonlinear operator $G^{\mathrm{NL}}\{\cdot\}$. Intuitively, for the system in Figure 5-2, the approximation (5.3) consists of the terms generated by "passing"[2] the input signal through at most one nonlinear operator. Due to the sufficiently small magnitude of the coefficients of the nonlinear kernels, if the input signal passes through at least two nonlinear operators, then the corresponding term in the output signal involves the multiplication of at least two small coefficients and thus can be neglected. The approximation in (5.3) will be used for our parameter estimation method.

5.2 Review of Related Work

The most relevant work of this chapter is the modular filter that is systematically studied in [1, 2], which is briefly reviewed in Section 2.1 of this thesis. The modular filter replaces every delay element in a FIR filter by another filter, where the inner filter module can be either a discrete-time or continuous-time filter. The filter design problem is considered in [1, 2] with the minimax criterion. For discrete-time linear-phase FIR modular filter, the First Algorithm of Remez [137] is applied to obtain the optimal minimax approximation. It is shown that the modular filter outperforms the direct form filter if the total number of degrees of freedom is the same between the two structures. If the inner module is a continuous-time filter, then there is an additional challenge that the phase response is generally unavailable in the specification of the filter design problem; therefore, an alternating minimization algorithm is proposed which alternates between the coefficients and the target phase response of the modular filter. The minimax error is shown to reduce in each iteration, although the theoretical guarantee for convergence to the globally optimal solution remains unknown.

With each module as a FIR filter, the modular filter design problem in the z-transform domain is also related to univariate polynomial decomposition, which refers to the determination of polynomials $f(z)$ and $g(z)$ so that their composition $f(g(z))$ equals or approximates a given polynomial $h(z)$. In addition to modular filter design,

[2]Since a nonlinear operator introduces multiplication among its input signal samples, it may be not fully clear to define the branches in Figure 5-2 that the input signal "passes" through to yield a specific term in the output signal. However, a more precise analysis and formulation will be provided in Section 5.3.2.

polynomial decomposition can also be applied to efficient signal representation and is explored in [2]. There are a number of algorithms for polynomial decomposition, some of which work with the coefficients of the polynomial [13, 14, 42] and others work with the roots [54, 55]. As an example of efficient approximate polynomial decomposition algorithms that work with the coefficients, the method in [13] takes iterations, each of which has two steps: optimization over $f(z)$ with the current $g(z)$, and approximate least squares solution for $g(z)$ to a truncated Taylor series expansion of the cost function with the current $f(z)$. In addition, this algorithm initializes the two component polynomials by using another polynomial decomposition method [12], which matches the coefficients of the terms of $h(z)$ with the highest degrees and obtains the coefficients of $g(z)$ sequentially from higher to lower degrees.

Prior to [2], there are a number of methodologies that implicitly utilize systems with modular structures. As an example, the filter sharpening method [57] utilizes special linear combinations of the cascade of a fixed filter to enhance the approximation performance with respect to a specified frequency response.

In another thread, the Volterra series model has been widely used in modeling nonlinear systems and a number of parameter estimation techniques have been proposed [39,40,69], which have been briefly reviewed in Section 2.2. However, these techniques do not consider the modular structure in this chapter and thus are not directly applicable.

5.3 Parameter Estimation Algorithm

We propose a parameter estimation algorithm in this section under the weak nonlinearity assumption, i.e. the coefficients of the nonlinear kernels of $G\{\cdot\}$ have sufficiently smaller magnitude compared with those of the linear kernel. The algorithm has two steps: the first step obtains the linear kernel coefficients $g^{(1)}[i_1]$ and the coefficients of $F(z)$, and the second step estimates the nonlinear kernel coefficients $g^{(k)}[i_1, \ldots, i_k]$ for $k \geq 2$.

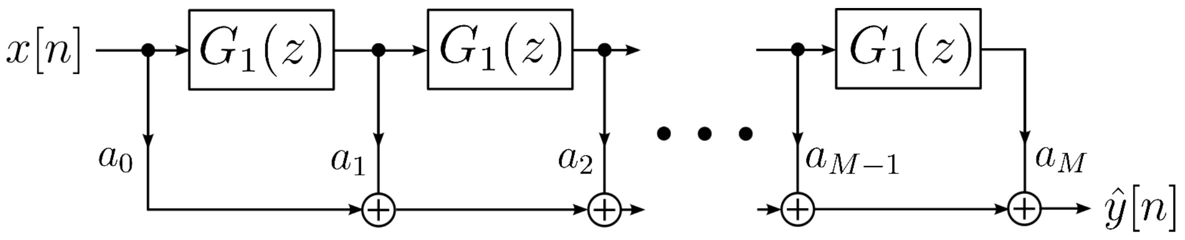
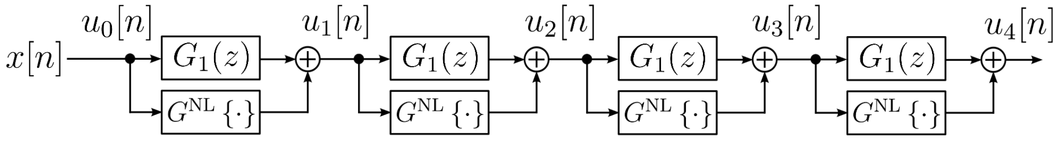


Figure 5-3: Simplified modular system with the linear kernel only.

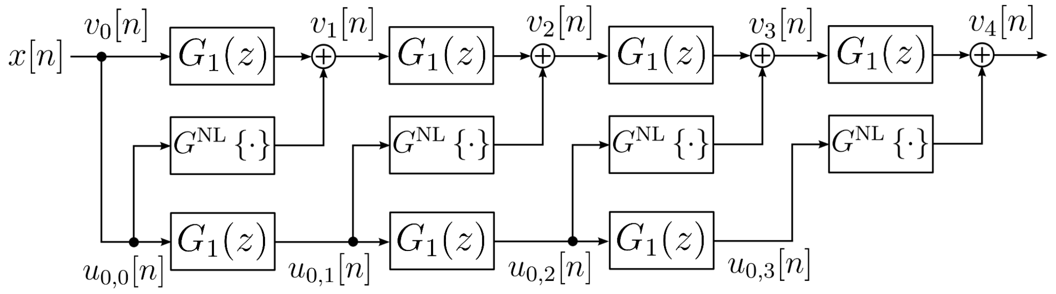
5.3.1 First Step: Linear Kernel Estimation

In this step, we temporarily neglect the nonlinear kernels in each Volterra module, and the modular system is simplified as in Figure 5-3.

The simplified modular system becomes a modular FIR filter with the transfer function as the polynomial composition $F(G_1(z))$, where $G_1(z)$ is the z-transform of the linear operator $G^{(1)}\{\cdot\}$. Since the orders of the filters $F(z)$ and $G_1(z)$ are M and N, respectively, the composed polynomial $F(G_1(z))$ has the degree of MN. The goal in this step becomes approximate univariate polynomial decomposition to obtain $F(z)$ and $G_1(z)$, with given statistical information between the input and output signals. A possible performance criterion is to minimize the mean squared output error $\mathbb{E}\left\{|y[n]-\hat{y}[n]|^2\right\}$, where $\hat{y}[n]$ is the output signal of the system in Figure 5-3. Since the mean squared error still has a nonlinear dependence on the coefficients of $G_1(z)$, a direct estimation of the parameters may remain challenging. A possible algorithm is to take iterations, each of which performs alternate approximate optimization for $F(z)$ (or $G_1(z)$) with the current $G_1(z)$ (or $F(z)$), which is similar to the approach in [13]. Alternatively, we can directly apply univariate polynomial decomposition algorithms as follows: first, we obtain a FIR filter $\hat{H}_1(z)$ with order-(MN) using the second-order correlation functions between the input and output signals, which minimizes the mean squared error in the output; then, approximate polynomial decomposition is performed on $\hat{H}_1(z)$ to obtain the *component* polynomials $F(z)$ and $G_1(z)$ such that $\hat{H}_1(z) \approx F(G_1(z))$. As is reviewed in Section 5.2, there are various polynomial decomposition methods available to obtain the components $F(z)$ and $G_1(z)$ from $\hat{H}_1(z)$; as an example, we can choose the algorithm in [13], which will be used in the



(a): Original cascade Volterra system with $M = 4$ modules



(b): Approximation of the cascade Volterra system

Figure 5-4: Approximation of a cascade Volterra system under the weak nonlinearity assumption. The total number of modules is $M = 4$.

simulation later.

5.3.2 Second Step: Nonlinear Kernel Estimation

In this step, the coefficients of the nonlinear kernels of the Volterra module $G\{\cdot\}$ are estimated, with the linear kernel $g^{(1)}[i_1]$ and the coefficients of $F(z)$ that are available from the previous step.

We first simplify the cascade of the Volterra module $G\{\cdot\}$, under the weak nonlinearity assumption. Similar to the Taylor expansion (5.2) in terms of ε defined in (5.1), for the cascade system[3] in Figure 5-4 (a), if we denote the $O(1)$ component of the signals $u_m[n]$ $(0 \leq m \leq M)$ as $u_{0,m}[n]$, then it can be observed that $u_{0,m}[n]$ is the output signal by filtering $u_{0,m-1}[n]$ with $G_1(z)$, i.e.

$$u_{0,m}[n] = G^{(1)}\{u_{0,m-1}[n]\},$$

where $u_{0,0}[n] = x[n]$. We then consider the $O(\varepsilon)$ component of $u_m[n]$, which is denoted

[3]To simplify, we use a cascade of 4 modules as an example in Figure 5-4.

as $u_{1,m}[n]$. Since

$$G^{\text{NL}}\left\{u_{m-1}[n]\right\} = G^{\text{NL}}\left\{u_{0,m-1}[n] + O(\varepsilon)\right\} = G^{\text{NL}}\left\{u_{0,m-1}[n]\right\} + O(\varepsilon^2),$$

where we use the fact that the coefficients of the nonlinear kernels are $O(\varepsilon)$, it can be seen that

$$u_{1,m}[n] = G^{(1)}\left\{u_{1,m-1}[n]\right\} + G^{\text{NL}}\left\{u_{0,m-1}[n]\right\}.$$

If we define

$$v_m[n] \triangleq u_{0,m}[n] + u_{1,m}[n],$$

then $v_m[n]$ is a reasonable approximation of $u_m[n]$ with the residue error satisfying $u_m[n] - v_m[n] = O(\varepsilon^2)$. Combining the results above, it can be shown that

$$v_m[n] = G^{(1)}\left\{u_{0,m-1}[n]\right\} + G^{(1)}\left\{u_{1,m-1}[n]\right\} + G^{\text{NL}}\left\{u_{0,m-1}[n]\right\} = G^{(1)}\left\{v_{m-1}[n]\right\} + G^{\text{NL}}\left\{u_{0,m-1}[n]\right\},$$

where the last step uses the linearity of the operator $G^{(1)}\left\{\cdot\right\}$. Finally, the above equation can be implemented by the system in Figure 5-4 (b), where the signals $v_m[n]$ $(0 \leq m \leq M)$ are the approximation of $u_m[n]$ in Figure 5-4 (a) up to the order of $O(\varepsilon)$.

Then, we consider the approximation of the modular Volterra system in Figure 5-1. Since the output of the modular Volterra system is a linear combination of the signals $u_m[n]$ $(0 \leq m \leq M)$ in Figure 5-4 (a), and each $u_m[n]$ can be approximated by $v_m[n]$, it can be shown from Figure 5-4 (b) that the modular Volterra system can be approximated by the structure[4] in Figure 5-5.

For parameter estimation, a possible choice of the performance criterion is the mean squared error $\mathbb{E}\left\{|y[n] - \tilde{y}[n]|^2\right\}$. Since the input signal of each module $G^{\text{NL}}\left\{\cdot\right\}$ in Figure 5-5 depends only on the signal $x[n]$ and the FIR filter $G_1(z)$ that is already available, the signals $\tilde{y}_i[n]$ $(1 \leq i \leq M)$ in Figure 5-5 depend linearly on the coefficients of the nonlinear kernels. The final output signal $\tilde{y}[n]$ is a linear combination of $\tilde{y}_i[n]$, so it also has a linear dependence on the coefficients of the

[4]Again, we use $M = 4$ as an example in Figure 5-5.

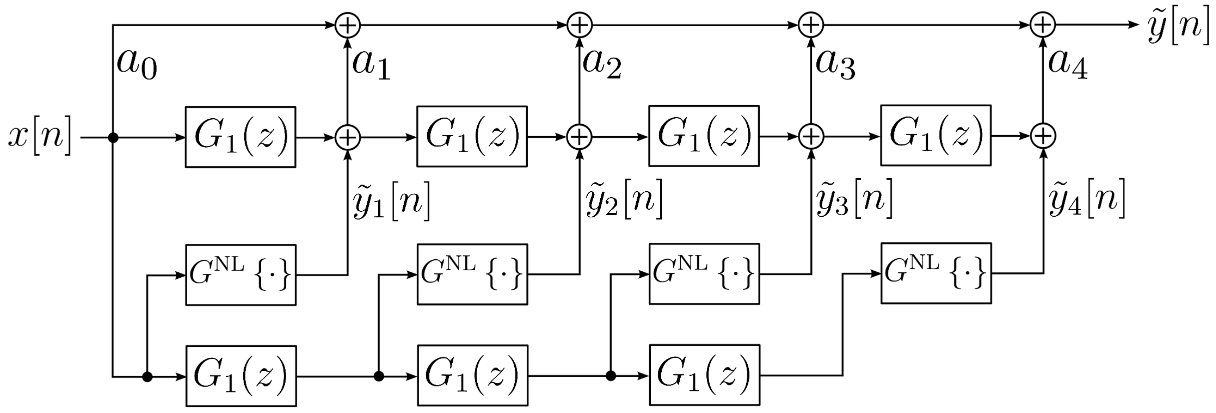


Figure 5-5: Approximation of a modular Volterra system under the weak nonlinearity assumption, with parameter $M = 4$.

nonlinear kernels in $G^{\mathrm{NL}}\{\cdot\}$, which enables least squares estimation for $G^{\mathrm{NL}}\{\cdot\}$ using the higher-order statistics between the input and output signals.

In summary, the parameter estimation algorithm for the modular Volterra system in Figure 5-1 is described as follows, which is under the assumption that the linear kernel dominates the nonlinear kernels in the Volterra module $G\{\cdot\}$.

Parameter Estimation Algorithm for Modular Volterra System

(1) Using the second-order correlation functions, obtain an order-(MN) FIR filter $\hat{H}_1(z)$ that minimizes the mean squared error of the output.

(2) Apply approximate polynomial decomposition algorithms to obtain FIR filters $F(z)$ and $G_1(z)$ such that $\hat{H}_1(z) \approx F(G_1(z))$.

(3) Using higher-order statistics between the input and output signals, obtain the least squares estimation of the nonlinear kernels in $G^{\mathrm{NL}}\{\cdot\}$ in the approximate system shown in Figure 5-5.

Since the filter $F(z)$ and the linear kernel of $G\{\cdot\}$ are estimated without considering the nonlinear kernels of $G\{\cdot\}$, there is generally no guarantee for global optimality of the algorithm above. In addition, if the nonlinear kernels of $G\{\cdot\}$ have coefficients that are comparable in magnitude to those of the linear kernel, or if the input signal $x[n]$ has sufficient large magnitude, then the above algorithm may have noticeable estimation error since the weak nonlinearity assumption does not hold.

5.4 Numerical Evaluation

5.4.1 Setup

This section evaluates the parameter estimation algorithm using synthetic modular Volterra systems, where the coefficients of both the Volterra module $G_{\text{syn}}\{\cdot\}$ and the FIR filter $F_{\text{syn}}(z)$ are randomly generated and follow zero-mean Gaussian distribution. To ensure weak nonlinearity, the average magnitude of the coefficients of the nonlinear kernels of $G_{\text{syn}}\{\cdot\}$ is 1/20 of that of the linear kernel. The input signals to the synthetic modular Volterra systems are white random processes with length 20000, where each element follows uniform distribution[5] between -1 and 1. Observation noise with SNR 40dB is added to the synthetic output signal of the modular Volterra systems. The highest order K of the Volterra module $G_{\text{syn}}\{\cdot\}$, the maximum delay (i.e. the number of state variables) of $G_{\text{syn}}\{\cdot\}$, and the order of the FIR filter $F_{\text{syn}}(z)$ are all swept between 2 to 4. At each parameter setting, 100 random examples of the FIR filter $F_{\text{syn}}(z)$, the Volterra module $G_{\text{syn}}\{\cdot\}$, and input-output signal pairs are generated.

We apply the two-step algorithm to obtain the coefficients of the estimated FIR filter $F_{\text{est}}(z)$ and the kernels of the estimated Volterra module $G_{\text{est}}\{\cdot\}$. Then, we obtain the estimated output signal by feeding the input signal into the estimated modular Volterra system. For comparison purposes, we consider the following two structures for the estimated system: (1) the modular Volterra system where the order and delay of the estimated systems $G_{\text{est}}\{\cdot\}$ and $F_{\text{est}}(z)$ are equal to those of the synthetic systems $G_{\text{syn}}\{\cdot\}$ and $F_{\text{syn}}(z)$, which is the setup for system identification; (2) the modular FIR system where $F_{\text{est}}(z)$ has the same order as $F_{\text{syn}}(z)$ while $G_{\text{est}}\{\cdot\}$ has only the linear kernel, and we ensure that $G_{\text{est}}\{\cdot\}$ has the same number of degrees of freedom as the synthetic Volterra module $G_{\text{syn}}\{\cdot\}$. In other words, this second setting has the estimated system with a mismatched structure. Since the estimated system can have a different structure from the true synthetic system, it is inconvenient

[5]In order to avoid input signals with big magnitude that can violate the weak nonlinearity assumption in this chapter, the uniform distribution is chosen due to its strict limit on the magnitude of the input signals.

to evaluate the estimation performance in terms of the differences in the kernels; instead, we consider the error between the estimated output signal and the original noisy synthetic output signal. If the SNR between these two output signals is higher than a threshold, then we consider the estimated system is sufficiently close to the true synthetic system in terms of input-output relationship and thus the parameter estimation is successful. Without loss of generality, we choose this threshold of SNR as 20dB in the simulation.

5.4.2 Results and Observations

We show the success percentage using the two structures for the estimated system, namely the modular Volterra system and the (mismatched) modular FIR filter, in Table 5.1.

We can have the following observations from Table 5.1. First, the success rates reduce as the increase of the delay and order of the modular system. In our current setting, the increase of the delay and order generally introduces stronger nonlinear effects of the modular system, which makes the weak nonlinearity assumption less precise and causes the parameter estimation to become more challenging. Second, the success rates using the mismatched modular FIR structure are noticeably lower than those using the correct structure. We can then conclude that the nonlinearity introduced by the Volterra modules is not efficiently represented by a modular FIR filter, even with the same number of degrees of freedom. This observation may potentially suggest the necessity of nonlinear kernels in order to precisely model or identify a modular Volterra system, despite the weak nonlinearity assumption. Third, for a fixed order of $F(z)$ and a fixed structure of the estimated system, the success rates generally seem to decrease with the number of degrees of freedom of $G\{\cdot\}$.

5.5 Chapter Conclusion and Future Work

This chapter considers the modular Volterra system that corresponds to modular composition in Section 1.1.4. We consider the modular Volterra system that is created

Table 5.1: Success rates of the parameter estimation algorithm for modular Volterra system.

Structural Parameters of the Synthetic System				Success Percentage Using Different System Structures in Estimation	
Order of FIR $F_{\text{syn}}(z)$	Order of Module $G_{\text{syn}}\{\cdot\}$	Delay of Module $G_{\text{syn}}\{\cdot\}$	Degrees of Freedom of $G_{\text{syn}}\{\cdot\}$	Modular Volterra System	Modular FIR
2	2	2	10	100	68
2	2	3	15	100	60
2	2	4	21	100	42
2	3	2	20	100	48
2	3	3	35	98	16
2	3	4	56	84	6
2	4	2	35	96	23
2	4	3	70	70	7
2	4	4	126	35	2
3	2	2	10	99	26
3	2	3	15	96	10
3	2	4	21	95	5
3	3	2	20	67	10
3	3	3	35	19	1
3	3	4	56	10	0
3	4	2	35	27	0
3	4	3	70	3	0
3	4	4	126	1	0
4	2	2	10	87	10
4	2	3	15	59	3
4	2	4	21	38	0
4	3	2	20	23	1
4	3	3	35	4	0
4	3	4	56	1	0
4	4	2	35	3	0
4	4	3	70	0	0
4	4	4	126	0	0

by replacing each delay element in a FIR filter $F(z)$ in the direct form implementation with an identical copy of a Volterra series module $G\{\cdot\}$. Since the modular Volterra system uses identical modules, it can benefit hardware implementation especially for VLSI technology. In addition, if the Volterra module $G\{\cdot\}$ has only the linear kernel, the modular Volterra system becomes the modular FIR filter that is systematically studied in [2].

Given statistical information between input and output signals, a two-step parameter estimation algorithm is proposed to obtain the coefficients of $F(z)$ and $G\{\cdot\}$, where we assume weak nonlinearity of $G\{\cdot\}$. The first step neglects the nonlinear kernels in $G\{\cdot\}$ and simplifies the modular Volterra system into a modular FIR filter with the transfer function as the polynomial composition of $F(z)$ and the linear kernel of $G\{\cdot\}$. Therefore, approximate polynomial decomposition algorithms can be utilized to determine these parameters. In the second step, the nonlinear kernels of $G\{\cdot\}$ are estimated with the available $F(z)$ and the linear kernel of $G\{\cdot\}$. Due to the weak nonlinearity assumption, a linear equation can be approximately established between the output signal and the unknown coefficients of the nonlinear kernels of $G\{\cdot\}$, to which the least squares solution is available. Simulation results have shown that this parameter estimation algorithm is effective when the order and delay (i.e. the number of state variables) of $F(z)$ and $G\{\cdot\}$ are not high; in addition, the estimated system with the correct nonlinear structure has significantly better performance than that with a mismatched modular FIR structure, although the number of degrees of freedom is maintained.

There are a number of directions for further development on the modular Volterra system. First, it would be very interesting to identify physical devices or systems which naturally create a modular nonlinear structure, which can serve as concrete applications of the proposed algorithm. Second, the current setup of numerical evaluation is still limited. A more general setup can be the comparison among a modular Volterra system, a modular FIR filter, and a Volterra system that all have the same number of degrees of freedom, where the goal is to approximate a nonlinear system that does not have a Volterra representation with a finite number of terms or to design a system with specifications in terms of input-output higher-order statistics. Third, the current parameter estimation algorithm could be interpreted as a linearized approximation for the modular Volterra system. Similar to the Taylor expansion of a function, the zeroth-order approximation in our problem is the modular FIR system without the nonlinear kernels, and the first-order approximation has the extra terms that involve a single coefficient of the nonlinear kernels. However, there could be higher-order approximations of the modular Volterra system that consider nonlinear interactions between the coefficients of the nonlinear kernels, which could potentially improve the precision of parameter estimation and thus is worth further study. Fourth, although challenging, the parameter estimation without the assumption of weak nonlinearity is worth considering, which might potentially be applied to a wider class of nonlinear systems. Fifth, we use the mean squared error in the output signal as the error criterion of the estimation, and it would be interesting to consider alternative criteria. Sixth, the modular Volterra system in this thesis replaces each delay element in a FIR filter with a Volterra module, and it would be interesting to further consider using IIR filters with feedback loops for modular systems instead of FIR filters.

Chapter 6

Learning Two-level Boolean Functions

In contrast to the systems in the previous chapters that have continuous-valued input and output signals, this chapter focuses on systems with multi-dimensional binary input variables and a single binary output, i.e. Boolean functions[1]. In particular, we focus on learning two-level Boolean functions from a dataset for classification purposes, with the joint criteria of accuracy and function simplicity. The Boolean functions we consider are in the conjunctive normal form (CNF, "AND-of-ORs") or disjunctive normal form (DNF, "OR-of-ANDs") [53]; as is discussed in Section 1.1.3, these functions belong to the multivariate composition of one-level Boolean functions. This composition significantly improves the model richness, and the two-level functions can represent any Boolean function [34] when the negation of each input variable is available.

The two-level Boolean function is an important subclass of rule-based classifiers, which have wide applications in various signal processing fields such as speech recognition [29], smart grid [30], and expert systems [31]. Typically, a Boolean function connects a subset of the binary input variables with the logical conjunction ("AND"), disjunction ("OR"), and negation ("NOT") to form the prediction of the binary output variable. To simplify description and avoid potential confusion, the

[1]The majority of the content in this chapter is included in [138].

terminologies *feature* and *label* will be used in this thesis to denote each dimension of the input variables and the output variable, respectively. For instance, a Boolean function in [33] for the prediction of 10-year coronary heart disease (CHD) risk for a male at the age of 45 is as follows:

IF 1. NOT smoke; OR

2. (total cholesterol $<$ 160) AND (systolic blood pressure $<$ 140);

THEN (10 year CHD risk $<$ 5%)=TRUE.

The example above is a two-level Boolean function in DNF, where in the lower level, conjunctions of binary features build *clauses* (i.e. one-level Boolean functions) and in the upper level, the disjunction of the clauses forms the predictor. As shown in the example above, continuous-valued features may be incorporated by converting them to binary with threshold comparisons. In this chapter, we assume that both the input and output variables are fully provided in the dataset. In other words, we do not consider the design for conversion from general variables to binary ones.

As is discussed in Section 1.1.3, a two-level Boolean function in CNF or DNF has the following multivariate function composition structure:

$$\hat{y} = F(G_1(x_1, \cdots, x_d), \cdots, G_R(x_1, \cdots, x_d)), \tag{6.1}$$

where $(x_1, \cdots, x_d)$ and $\hat{y}$ denote the input and output variables, respectively, and each $G_r(x_1, \cdots, x_d)$ $(1 \leq r \leq R)$ corresponds to a clause. For DNF, the functions $F(v_1, \cdots, v_R)$ and $G_r(x_1, \cdots, x_d)$ $(1 \leq r \leq R)$ in (6.1) are defined as

$$F(v_1, \cdots, v_R) = \bigvee_{r=1}^{R} v_r, \quad G_r(x_1, \cdots, x_d) = \bigwedge_{j \in \mathbf{X}_r} x_j, \tag{6.2}$$

where each $\mathbf{X}_r$ $(1 \leq r \leq R)$ is a subset of the index set $\{1, 2, \cdots, d\}$, and the symbols "$\vee$" and "$\wedge$" denote the logical disjunction ("OR") and conjunction ("AND"), respectively. By swapping the logical disjunction and conjunction, we obtain the functions F and G_r ($1 \leq r \leq R$) for CNF as follows.

$$F(v_1, \cdots, v_R) = \bigwedge_{r=1}^{R} v_r, \quad G_r(x_1, \cdots, x_d) = \bigvee_{j \in \mathbf{X}_r} x_j. \tag{6.3}$$

The goal of this chapter is to learn a two-level Boolean function for classification from a given training dataset by determining all the functions $G_r(x_1, \cdots, x_d)$ for $1 \leq r \leq R$ in (6.1) (or equivalently, the subsets $\mathbf{X}_r$ in (6.2) and (6.3)). In addition to the classification accuracy on the dataset, the simplicity of the Boolean function is also important for the two reasons below. First, if we measure the simplicity of a Boolean function by the total number of features used $\sum_r |\mathbf{X}_r|$, then a function with a small total number of features used yields the benefit of *interpretability* [32], which is attracting considerable attention in machine learning and has substantial importance in a wide range of applications such as law and medicine [33, 36–38]. In these fields, predictions from classification models are generally presented to a human decision maker/agent who makes the final decision. Such a decision maker often needs an understanding of the reasons for the prediction before accepting the result; thus, high prediction accuracy without providing the reasons is not sufficient for the model to be trusted. For example, convincing reasons are legally required in fraud detection [38] to establish a claim of fraud. In addition to directly learning interpretable models (e.g. decision lists [139], decision trees [140]) from data, researchers have also been interested in constructing interpretable approximations for black-box models (e.g. neural networks), which provide insights and interpretability while remaining faithful to the black-box models [141,142]. The second reason to favor simple functions is to avoid overfitting; since real world datasets inevitably have noise, excessively complicated functions may overfit and model the random noise instead of the fundamental relationship in the data, which leads to low predictive performance on general untrained data.

Despite the importance of Boolean functions, learning two-level Boolean functions with high accuracy and simplicity from a dataset is a considerable challenge since it is inherently combinatorial [143,144]. Even a simpler problem of learning a one-level

Boolean function (i.e. a single conjunctive or disjunctive clause) is already NP-hard in certain formulation [145]. In addition, two-level Boolean functions are substantially more expressive than one-level functions. As mentioned in Section 1.1.3, if the input features are binary and we include all negations, then two-level functions can represent any Boolean function of the input features [34], which does not hold for one-level functions. The expressiveness of two-level functions also suggests that they are more challenging to learn than one-level functions. Due to this complexity, as we overview in Section 6.1, existing methods are mostly heuristic and/or greedy and there has been limited work on principled yet tractable approaches.

This chapter introduces a unified optimization framework for learning two-level Boolean functions from a dataset that achieves good balance between the joint criteria of accuracy and function simplicity. We propose two formulations. The first formulation aims to minimize a weighted combination of the total number of classification errors and the total number of features used. Based on this formulation, a linear programming relaxation approach is developed, which explicitly utilizes the composition structure of the two-level Boolean functions. The second formulation replaces the 0-1 classification error with the Hamming distance from the current two-level function to the closest function that correctly classifies a training example. With this second formulation, block coordinate descent and alternating minimization algorithms are developed, which iteratively update a certain part in the composed two-level function. Experiments show that two-level functions can have considerably higher accuracy than one-level functions. The two algorithms based on the Hamming distance formulation obtain very good tradeoffs between accuracy and simplicity. In addition, we tackle the issue of fractional optimal solutions to linear programming relaxations and introduce a new binarization method to convert the solutions of the linear programs into binary values.

The remainder of this chapter is organized as follows. Section 6.1 reviews existing work on two-level Boolean function learning. After the problem formulations in Section 6.2, optimization approaches are introduced in Section 6.3 and evaluated in Section 6.4. Section 6.5 concludes this chapter and presents future work.

6.1 Review of Existing Work

The two-level Boolean functions in this chapter are examples of decision rule lists [139], which have been extensively studied in various fields such as pattern recognition and machine learning, where a number of strategies have been proposed [34]. The covering strategy [139, 146–150] sequentially constructs each clause in a two-level Boolean function; in each step, it builds a new clause generally in a greedy manner, and then removes the newly covered training examples or adjusts the weights on all training examples for future steps. The bottom-up strategy [151–153] successively combines more specific clauses into more general clauses according to local criteria like pairwise similarity. A more flexible multi-phase strategy [154–156] is to first discover a large set of candidate *local patterns* (i.e. clauses), then heuristically select a subset of informative clauses, and finally construct a two-level function by considering the selected clauses as new binary features. A fourth strategy is to convert trained decision trees into decision lists [140,157]. Unlike our proposed approach, the above strategies lack a single, principled objective to drive the learning process. Moreover, they employ heuristics that leave room for improvements on both accuracy and function simplicity.

In addition to the symbolic approaches above, Bayesian approaches in [33] and [158] apply approximate inference algorithms to produce posterior distribution over decision lists; however, the assignment of prior and likelihood may not always be clear, and certain approximate inference algorithms may have high computational cost. In contrast to two-level Boolean functions which combine clauses with logical conjunction or disjunction, more general classifiers can also take a weighted combination of learned clauses [159, 160]. Although the flexibility of weighted combination may improve accuracy, the weighted model may reduce interpretability compared with two-level Boolean functions [37].

There has been some prior work on optimization-based formulations for two-level Boolean function learning. To our best knowledge, the most relevant prior work is [145], where a linear programming framework is proposed to learn a clause, based on which two other algorithms are used for rule set learning, namely set covering

[150] and boosting. Although we apply this clause learning method as a component in some of our algorithms, there are significant differences between our work and [145]. As mentioned before, two-level functions that we consider are significantly more expressive and much more challenging to learn than a single clause. In addition, the greedy style of the set covering method in [145] leaves room for improvement, and the weighted combination of clauses in boosted classifiers reduces interpretability. Another work on DNF learning [161] provides a mixed integer program formulation named OOA (Optimized *Or's of And's*) and a different heuristic formulation OOAx (Optimized *Or's of And's* with Approximations). The mixed integer program in OOA is similar to our first formulation with the 0-1 error but without the relaxation to linear programming, which may thus result in quite high computational complexity. OOAx is similar to the multi-phase strategy above in using heuristic approaches to discover and select clauses before an optimization formulation is used to build a DNF with the selected clauses.

Boolean functions in CNF and DNF have also been studied in other fields that are either less related to this chapter or have a different setup. From the theoretical perspective, learnability of Boolean formulae is considered in [144] within the framework of probably approximately correct learning. Different from our problem, the setup of [144] and related work typically assumes positive or negative training examples can be generated on demand and without noise. In the field of digital circuit design, simplifying a function in DNF or CNF that exactly matches a given truth table can lead to implementation efficiency [35]. However, for our problem, it is neither needed nor desirable to have a two-level Boolean function exactly match a noisy dataset. In the work on functional composition [2], a composed representation of multivariate discrete-valued functions is proposed, where the individual functions to form the composition can be arbitrary. In contrast, each individual function in the composed form of the two-level Boolean functions in this chapter has to be either conjunction or disjunction of its input variables, which trades off flexibility with a more principled structure. An arbitrary discrete-valued function with a finite alphabet can be converted to a binary function by indicators; since two-level Boolean functions can represent any Boolean function, restricting our focus to CNF and DNF does not necessarily reduce the expressiveness of the function class.

6.2 Problem Formulation

We consider the binary supervised classification, where the training dataset has n labeled examples. The i^{th} example has a binary output label $y_i \in \{0, 1\}$ and in total d input binary features[2] $a_{i,j} \in \{0, 1\}$ $(1 \leq j \leq d)$. The goal is to learn a classifier $\hat{y}(\cdot)$ in the Conjunctive Normal Form ("AND-of-ORs") that can generalize well to unseen input feature combinations. Recall the two-level Boolean function in (6.1), where in the lower level each clause is formed by the disjunction of a selected subset of input features $\mathbf{X}_r$ $(1 \leq r \leq R)$, and in the upper level the final predictor is formed by the conjunction of all clauses. We introduce the binary decision variables $w_{j,r} \in \{0, 1\}$ to represent whether to include the j^{th} feature x_j in the r^{th} clause, i.e. $w_{j,r} = 1$ if and only if $j \in \mathbf{X}_r$. From (6.3), the output of the r^{th} clause for the i^{th} training example is

$$\hat{v}_{i,r} = G_r\left(a_{i,1}, \cdots, a_{i,d}\right) = \bigvee_{j \in \mathbf{X}_r} a_{i,j} = \bigvee_{j=1}^{d} \left(a_{i,j} w_{j,r}\right), \text{ for } 1 \leq i \leq n,\ 1 \leq r \leq R. \quad (6.4)$$

Then, the predictor $\hat{y}_i$ satisfies

$$\hat{y}_i = F\left(\hat{v}_{i,1}, \cdots, \hat{v}_{i,R}\right) = \bigwedge_{r=1}^{R} \hat{v}_{i,r}, \text{ for } 1 \leq i \leq n. \quad (6.5)$$

To mitigate the need for careful specification of the model parameter R, a mechanism to "disable" a clause can be introduced to reduce the total number of actual clauses if the assigned R is too large. For a CNF Boolean function, a clause can be regarded as disabled if its output is always 1. Thus, we can pad the input feature matrix with a trivial "always true" feature $a_{i,0} = 1$ for all training examples, and also include the corresponding decision variables $w_{0,r}$ for all clauses; if $w_{0,r} = 1$, then the r^{th} clause has output 1 and is thus disabled in the CNF Boolean function. Using this mechanism,

[2] We assume the negation of each feature is included as another input feature; if not, we can pad the input features with negations.

the parameter R becomes the *maximum* number of clauses.

From (6.4) and (6.5), the goal of learning two-level Boolean functions is converted to the determination of the binary parameters $w_{j,r}$ ($0 \leq j \leq d$, $1 \leq r \leq R$), which will be the focus of the parameter estimation algorithms below.

In certain cases, DNF functions could be more preferable than CNF. An algorithm for CNF can be used to learn DNF by De Morgan's laws:

$$y = \bigvee_{r=1}^{R} \left(\bigwedge_{j \in \mathbf{X}_r} x_j \right) \Leftrightarrow \overline{y} = \bigwedge_{r=1}^{R} \left(\bigvee_{j \in \mathbf{X}_r} \overline{x}_j \right)$$

where $\overline{y}$ and $\overline{x}_j$ mean the negation of binary variables y and x_j, respectively. To learn a DNF function with a CNF learning algorithm, we can first negate both features and labels of all training examples, then learn a CNF function with the negated features and labels, and finally use the decision variables $w_{j,r}$ with the original features to construct a DNF function. Since the formulations of CNF are slightly more concise, Sections 6.2 and 6.3 focus on CNF only.

Two formulations are introduced with different accuracy costs in Section 6.2.1 and 6.2.2, respectively.

6.2.1 Formulation with 0-1 Error

A natural choice for the accuracy term in the objective is the total number of misclassifications (i.e. 0-1 error for each training example). With the regularization term as the sum of the number of features used in each clause, a formulation is as below

$$\min_{w_{j,r}} \sum_{i=1}^{n} |\hat{y}_i - y_i| + \theta \cdot \sum_{r=1}^{R} \sum_{j=1}^{d} w_{j,r} \tag{6.6}$$

$$\text{s.t. } \hat{y}_i = \bigwedge_{r=1}^{R} \left(\bigvee_{j=1}^{d} (a_{i,j} w_{j,r}) \right), \text{ for } 1 \leq i \leq n, \tag{6.7}$$

$$w_{j,r} \in \{0,1\}, \text{ for } 1 \leq j \leq d, \ 1 \leq r \leq R,$$

where the regularization parameter θ controls the tradeoff between accuracy and function simplicity. For the decision variables $w_{0,r}$ from the mechanism to “disable” a clause that is mentioned before, the regularization cost for $w_{0,r}$ can be lower than other variables or even zero. In addition, this clause disabling mechanism can also be regarded as a part of the regularization, whereby the cost of activating a clause is weighed against the accuracy improvement that it brings.

6.2.2 Formulation with Minimal Hamming Distance

Instead of using the 0-1 error to measure accuracy, it may be desirable to have a more fine-grained measure such as the minimal Hamming distance that will be explained below. For instance, consider two Boolean functions in CNF, both with two clauses, predicting the same training example with ground truth label $y_i = 1$. Suppose both clauses in the first function predict 0, while only one clause in the second function predicts 0 and the other predicts 1. Although both CNF functions misclassify this training example after taking “AND” of their two clauses, the second CNF function is closer to correct than the first one. If we use an iterative algorithm to refine the learned function, it might be beneficial for the accuracy cost term to favor this second CNF function, which could push the solution towards being correct. An additional motivation for the Hamming loss is to avoid identical (and thus redundant) clauses, by training each clause with a different subset of training examples, as done in [149].

In this second formulation, the accuracy cost for a single training example is the minimal Hamming distance from a given CNF function to an *ideal* CNF function, where the latter means a function that correctly classifies this training example. The Hamming distance between two CNF functions is the total number of $w_{j,r}$ that are different in the two functions. An intuitive explanation of this minimal Hamming distance is the smallest number of modifications (i.e. negations) of the decision variables $w_{j,r}$ in the current CNF function that are needed to correct a misclassification on a training example, i.e. how far is the function from being correct.

For mathematical formulation, we introduce *ideal clause outputs* $v_{i,r}$ with $1 \leq i \leq n$ and $1 \leq r \leq R$ to represent the clause outputs of a CNF function that correctly classifies the i^{th} training example. The values of $v_{i,r}$ are always consistent with the ground truth labels, i.e. $y_i = \bigwedge_{r=1}^{R} v_{i,r}$ for all $1 \leq i \leq n$. We let $v_{i,r}$ have a ternary alphabet $\{0, 1, \text{DC}\}$, where $v_{i,r} = \text{DC}$ means that we "don't care" about the value of $v_{i,r}$; the conjunction with DC satisfies $0 \wedge \text{DC} = 0$ and $1 \wedge \text{DC} = \text{DC} \wedge \text{DC} = \text{DC}$. With this setup, if $y_i = 1$, then $v_{i,r} = 1$ for all $1 \leq r \leq R$; if $y_i = 0$, then $v_{i,r_0} = 0$ for at least one value of r_0, and we can have $v_{i,r} = \text{DC}$ for all $r \neq r_0$. In implementation, $v_{i,r} = \text{DC}$ implies the removal of the i^{th} training example in the training or updating for the r^{th} clause, which leads to a different training subset for each clause.

Denote η_i as the minimal Hamming distance from the current CNF function $w_{j,r}$ to an ideal CNF function for the i^{th} training example. We derive η_i for positive and negative training examples, respectively. Since $y_i = 1$ implies $v_{i,r} = 1$ for all r, for each clause with output 0 in the current function, at least one positive feature needs to be included to match $v_{i,r} = 1$. Thus, the minimal Hamming distance for a positive training example is the number of clauses with output 0:

$$\eta_i = \sum_{r=1}^{R} \max \left\{ 0, \left(1 - \sum_{j=1}^{d} a_{i,j} w_{j,r} \right) \right\}, \text{ for } y_i = 1.$$

For $y_i = 0$, we first consider for fixed r the minimal Hamming distance between the r^{th} clauses of the current function and an ideal function where $v_{i,r} = 0$. We need to negate $w_{j,r}$ in the current function for j with $w_{j,r} = a_{i,j} = 1$ to match $v_{i,r} = 0$, and thus the minimal Hamming distance of this clause is $\sum_{j=1}^{d} a_{i,j} w_{j,r}$. Then, since $v_{i,r} = 0$ needs to hold for at least one value of r while all other $v_{i,r}$ can be DC, the minimal Hamming distance of the CNF function is given by the minimum over r, i.e. setting $v_{i,r_0} = 0$ with

$$r_0 = \arg\min_{1 \leq r \leq R} \left(\sum_{j=1}^{d} a_{i,j} w_{j,r} \right). \tag{6.8}$$

Combining all analysis above, the new formulation with the minimal Hamming distance cost is as below

$$\min_{w_{j,r}} \quad \sum_{i=1}^{n} \eta_i + \theta \cdot \sum_{r=1}^{R} \sum_{j=1}^{d} w_{j,r} \tag{6.9}$$

$$\text{s.t.}\quad \eta_i = \sum_{r=1}^{R} \max\left\{0, \left(1 - \sum_{j=1}^{d} a_{i,j} w_{j,r}\right)\right\}, \text{ for } y_i = 1, \tag{6.10}$$

$$\eta_i = \min_{1 \le r \le R} \left(\sum_{j=1}^{d} a_{i,j} w_{j,r}\right), \text{ for } y_i = 0, \tag{6.11}$$

$$w_{j,r} \in \{0,1\}, \text{ for } 1 \le j \le d,\ 1 \le r \le R.$$

To simplify description of algorithms later, we show a formulation (6.12) below, which is equivalent to (6.9) but involves both $v_{i,r}$ and $w_{j,r}$. Taking the minimization over $v_{i,r}$ in (6.12) with fixed $w_{j,r}$ eliminates the variables $v_{i,r}$, and (6.12) becomes identical to (6.9).

$$\begin{aligned} \min_{w_{j,r},\ v_{i,r}} \quad & \sum_{i=1}^{n} \sum_{r=1}^{R} \left[\mathbb{1}_{v_{i,r}=1} \cdot \max\left\{0, \left(1 - \sum_{j=1}^{d} a_{i,j} w_{j,r}\right)\right\}\right. \\ & \left. + \mathbb{1}_{v_{i,r}=0} \cdot \sum_{j=1}^{d} a_{i,j} w_{j,r}\right] + \theta \cdot \sum_{r=1}^{R} \sum_{j=1}^{d} w_{j,r} \end{aligned} \tag{6.12}$$

$$\text{s.t.}\quad \bigwedge_{r=1}^{R} v_{i,r} = y_i, \text{ for } 1 \le i \le n, \tag{6.13}$$

$$v_{i,r} \in \{0, 1, \text{DC}\}, \text{ for } 1 \le i \le n,\ 1 \le r \le R,$$

$$w_{j,r} \in \{0,1\}, \text{ for } 1 \le j \le d,\ 1 \le r \le R.$$

The binary variables $w_{j,r}$ can be relaxed to $0 \le w_{j,r} \le 1$. The minimum over r in (6.11) implies the non-convexity of such continuous relaxation with $R > 1$, making the exact solution challenging. Letting $R = 1$ in formulation (6.9) after the relaxation, it can be seen that we recover the formulation for one-level Boolean function learning in [145].

Although the Hamming distance formulation is a more fine-grained measure that can potentially be more effective for improving the learned function in an iterative algorithm, the accuracy cost in (6.9) is higher than or equal to the cost of the 0-1 error formulation in (6.6) when the decision variables $w_{j,r}$ are binary, i.e. the Hamming distance formulation may over-penalize the accuracy cost. In certain parameter settings, the Hamming cost may potentially not always be balanced between the positive and negative training examples. For instance, when $R = 1$, the accuracy cost η_i for positive training examples in (6.10) happens to be equivalent to the 0-1 cost, while the cost for negative training examples in (6.11) can be higher than 1. We can observe that the Hamming cost might potentially over-penalize misclassifications for negative training examples when $R = 1$.

Conditions for Tight Relaxation of $w_{j,r}$ in the Noiseless Formulation

In this section, we show conditions under which the optimal solution to the relaxation of a variant Hamming distance formulation is guaranteed to be binary, i.e. the relaxation is tight. Two relaxations are considered, namely $0 \le w_{j,r} \le 1$ and $w_{j,r} \ge 0$. To simplify, we assume that the input dataset is noiseless and modify our goal as determining the function in CNF that has perfect accuracy and uses the smallest total number of features. With this modified goal, a variant of the Hamming distance formulation is as follows.

$$\min_{w_{j,r}} \quad \sum_{r=1}^{R}\sum_{j=1}^{d} w_{j,r} \tag{6.14}$$

$$\text{s.t.} \quad \sum_{r=1}^{R} \max\left\{0, \left(1 - \sum_{j=1}^{d} a_{i,j} w_{j,r}\right)\right\} = 0, \text{ for } y_i = 1, \tag{6.15}$$

$$\min_{1 \le r \le R} \left(\sum_{j=1}^{d} a_{i,j} w_{j,r}\right) = 0, \text{ for } y_i = 0, \tag{6.16}$$

$$w_{j,r} \in \{0, 1\}, \text{ for } 1 \le j \le d, \; 1 \le r \le R. \tag{6.17}$$

This variant formulation is obtained by setting $\eta_i = 0$ $(1 \le i \le n)$ in (6.9). All the feasible solutions to (6.14) have perfect accuracy, and the optimal solution has the minimal total number of features. To avoid binary optimization, the constraint (6.17) can be relaxed to $0 \le w_{j,r} \le 1$ or even $w_{j,r} \ge 0$. The focus here is conditions under which the original and the relaxed formulations for (6.14) have identical optimal solutions (up to a permutation of the clauses).

First, we organize all the input features for all the training examples as a feature matrix $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_d]$, where each row represents a training example and each column corresponds to a dimension of the features. The K-disjunct and K-conjunct properties[3] that characterize feature similarity are reviewed as follows [145].

Definition 1. [145]. *A binary matrix $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_d]$ satisfies the K-disjunct property if the union of the 1's in any K columns[4] does not cover all the 1's in a different $(K+1)$-th column, i.e. $\bigvee_{i=1}^{K} \mathbf{a}_{k_i}$ does not cover all the 1's in $\mathbf{a}_{k_0}$ where $k_0 \neq k_i$ for $1 \leq i \leq K$. The matrix $\mathbf{A}$ satisfies K-conjunct property if the union of the 0's in any K columns does not cover all the 0's in a different $(K+1)$-th column, i.e. $\bigwedge_{i=1}^{K} \mathbf{a}_{k_i}$ does not cover all the 0's in $\mathbf{a}_{k_0}$ where $k_0 \neq k_i$ for $1 \leq i \leq K$.*

If a feature matrix $\mathbf{A}$ is K-conjunct (or K-disjunct), then for a given feature and other K features, there is at least one training example to distinguish the former from the conjunction (or disjunction) of the latter. Moreover, if the 0's of a feature column are covered by another, then the matrix cannot satisfy the K-conjunct property for any K, which can limit the application of the K-conjunct property. As an extension of the K-conjunct property to matrices with a column covering another, we define the *generalized K-conjunct property* as follows.

Definition 2. *A binary matrix $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_d]$ satisfies the generalized K-conjunct property, if the union of the 0's in any K columns does not cover all the 0's in a $(K+1)$-th column unless the 0's in this $(K+1)$-th column are covered by a single column among the K columns. More explicitly, if the 0's in $\mathbf{a}_{k_0}$ are not covered by any of $\mathbf{a}_{k_i}$, then $\bigwedge_{i=1}^{K} \mathbf{a}_{k_i}$ does not cover all the 0's in $\mathbf{a}_{k_0}$. A similar definition is available for the generalized K-disjunct property.*

Second, we define the *augmented feature matrix* $\mathbf{E}_M(\mathbf{A})$ for a feature matrix $\mathbf{A}$.

Definition 3. *For a binary feature matrix $\mathbf{A}$, its augmented feature matrix $\mathbf{E}_M(\mathbf{A})$ has each column as the disjunction of m features from $\mathbf{A}$ where $1 \leq m \leq M$. All the combinations of m features with $1 \leq m \leq M$ are incorporated in $\mathbf{E}_M(\mathbf{A})$.*

[3]In [145], tight relaxation conditions are proposed for their formulation to learn one-level Boolean functions, which utilize these properties.

[4]These K columns are not necessarily mutually distinct, i.e. we choose the K columns with replacement.

In other words, the augmented feature matrix lists the output values of each disjunctive clause containing at most M features as a new feature column. For $M \geq 2$, since there always exists a column in $\mathbf{E}_M(\mathbf{A})$ that covers all the 0's in another column, the matrix $\mathbf{E}_M(\mathbf{A})$ cannot satisfy the K-conjunct property; however, $\mathbf{E}_M(\mathbf{A})$ can still satisfy the generalized K-conjunct property as we will discuss later.

The following theorem shows a sufficient condition for the relaxed formulation for (6.14) to have the same optimal solution as the original binary program.

Theorem 6.1. *Denote $\mathbf{W}^* = [\mathbf{w}_1^*, \ldots, \mathbf{w}_R^*]$ as the true CNF function that generates the training dataset, i.e. $y_i = \bigwedge_{r=1}^{R} \left(\bigvee_{j=1}^{d} (a_{i,j} w_{j,r}^*) \right)$ for all $1 \leq i \leq n$. In addition, each clause $\mathbf{w}_r^*$ has at most M features where $M < d$, and none of the clauses is redundant (i.e. the 1's in the output of the r_1^{th} clause on the dataset are not covered by the r_2^{th} clause for all $r_1 \neq r_2$). If all the three following assumptions hold,*

(a) *the feature matrix $\mathbf{A}$ is M-disjunct,*

(b) *the augmented feature matrix $\mathbf{E}_M(\mathbf{A})$ is generalized R-conjunct,*

(c) *each clause $\mathbf{w}_r^*$ is exactly "exemplified" by the negation of (at least) a training example; i.e. for each $1 \leq r \leq R$, there is some i such that $w_{j,r}^* = \overline{a_{i,j}}$ for all $1 \leq j \leq d$,*

then we have the two following guarantees:

(i) *the feasible solutions to the relaxed formulation for (6.14) with $0 \leq w_{j,r} \leq 1$ only consist of the true CNF function $\mathbf{W}^*$ up to a permutation of the clauses,*

(ii) *the optimal solutions to the relaxed formulation for (6.14) with $w_{j,r} \geq 0$ exactly recover $\mathbf{W}^*$ up to a permutation of the clauses.*

The proof for Theorem 6.1 is in Appendix E. We have the following remarks on Theorem 6.1:

(i) There are a number of possibilities to improve Theorem 6.1 in the current form. First, the assumption (c) seems to limit the application of the theorem.

In fact, the formulation (6.14) becomes the disjunctive clause (i.e. one-level Boolean function) recovery problem in [145] if we set $R = 1$; however, the assumption (c) is not needed for the tight relaxation condition for one-level Boolean function recovery in [145] and thus Theorem 6.1 is not a generalization of the results in [145], which may leave room for improvement. Second, the theorem assumes no noise in the dataset, and it is interesting to explore tight relaxation conditions with noisy dataset. Third, the theorem considers only the effects of the relaxation of the binary variables $w_{j,r}$, however, the relaxed formulation remains non-convex for $R \geq 2$ and thus challenging to solve precisely. As a result, performance bound of the algorithms that are used for solving the non-convex relaxed formulation should be useful.

(ii) Roughly speaking, the K-disjunct or conjunct properties indicate the differences among features; if the features are sufficiently independent in the dataset, then these property would be satisfied [145]. An extreme case is that the dataset includes every 2^d binary patterns of the features, which satisfies all assumptions (a)-(c).

(iii) If we consider another variant Hamming distance formulation to directly construct DNF instead of CNF, then the assumption (c) may become slightly more natural: each clause $\mathbf{w}_r^*$ is exactly "exemplified" by a training example (instead of the negation of the example).

(iv) The assumption (b) in Theorem 6.1 uses the generalized R-conjunct property of the augmented matrix $\mathbf{E}_M(\mathbf{A})$. However, this property can be directly tested on the input data matrix $\mathbf{A}$ without explicitly constructing the entire augmented matrix $\mathbf{E}_M(\mathbf{A})$, as shown by Theorem 6.2 below.

Theorem 6.2. *A sufficient condition for the generalized R-conjunct property of the augmented matrix $\mathbf{E}_M(\mathbf{A})$ is: for any two disjoint subsets of features $\mathcal{P}$ and $\mathcal{Z} \subseteq \{1, \ldots, d\}$ with cardinality constraints $1 \leq |\mathcal{P}| \leq \min\{R, d\}$ and $1 \leq |\mathcal{Z}| \leq M$, there exists a training example $a_{i,j}$ that satisfies $a_{i,j} = 1$ for all $j \in \mathcal{P}$ and $a_{i,j} = 0$ for* *all $j \in \mathcal{Z}$. Furthermore, if the matrix $\mathbf{A}$ is M-disjunct, then this condition is also necessary.*

The proof for Theorem 6.2 is in Appendix E. From Theorem 6.2, we can have the following sufficient but not necessary condition for the assumptions (a) and (b) in Theorem 6.1. The proof for Theorem 6.3 is omitted.

Theorem 6.3. *If for any subsets $\mathcal{S} \subseteq \{1, \ldots, d\}$ with cardinality $|\mathcal{S}| = \min\{R + M, d\}$, the data matrix $\mathbf{A}$ contains at least a training example for each of the $2^{|\mathcal{S}|}$ binary patterns of the features within the subset $\mathcal{S}$, then the assumptions* (a) *and* (b) *in Theorem 6.1 both hold.*

6.3 Optimization Approaches

This section discusses various optimization approaches to the two-level Boolean function learning problem, i.e. the estimation of the binary parameters $w_{j,r}$ in (6.4). Based on the formulation (6.6) in Section 6.2.1, we develop a linear programming relaxation approach in Section 6.3.1. Based on the formulation (6.12) in Section 6.2.2, we propose the block coordinate descent algorithm in Section 6.3.2 and the alternating minimization algorithm in Section 6.3.3. All algorithms use linear programming relaxations and the dimensions of the resulting linear programs are analyzed in Section 6.3.4. Section 6.3.5 considers the binarization problem for the case of non-binary solutions to the linear programs.

6.3.1 Two-level Linear Programming Relaxation

This approach considers the 0-1 error formulation (6.6) and explicitly utilizes the composition structure of the two-level functions as shown in (6.1). By applying the idea of replacing binary operations "AND" and "OR" with linear-algebraic operations, a linear programming relaxation will be finally formulated.

If we consider "AND" and "OR" as functions, then they are defined only on binary inputs. Consequently, there are various interpolations of these functions to the continuous domain, and both convex and concave interpolations exist for both functions. The "OR" function has the following interpolations [162]

$$\bigvee_{j=1}^{d} x_j = \max_{1\le j\le d}\{x_j\} = \min\left\{1,\ \sum_{j=1}^{d} x_j\right\},$$

where the first is convex and the second is concave, both of which are the respective tightest interpolations. The logical "AND" function also has the tightest convex and concave interpolations as [162]

$$\bigwedge_{j=1}^{d} x_j = \max\left\{0,\ \left(\sum_{j=1}^{d} x_j\right) - (d-1)\right\} = \min_{1\le j\le d}\{x_j\}.$$

As shown by (6.1), the output $\hat{y}_i$ of the two-level Boolean function is a composition of one-level Boolean functions. By a proper composition of the interpolations of each one-level Boolean function, we can obtain convex and concave interpolations of $\hat{y}_i$. Specifically, from (6.4) and (6.5), a convex interpolation of $\hat{y}_i$ is

$$\hat{y}_i = \max\left\{0,\ \left(\sum_{r=1}^{R} \max_{1\le j\le d}\{a_{i,j}w_{j,r}\}\right) - (R-1)\right\},$$

and a concave interpolation can also be obtained similarly.

Denote the 0-1 error cost for the i^{th} training example as $\psi_i \triangleq |\hat{y}_i - y_i|$. Since the errors ψ_i in (6.6) should be minimized, if $y_i = 1$, then $\psi_i = 1-\hat{y}_i$ and thus we need the concave interpolation for $\hat{y}_i$; if $y_i = 0$, then $\psi_i = \hat{y}_i$ and thus the convex interpolation is needed. Finally, the formulation in (6.6) can be exactly converted into a mixed integer program:

$$\min_{w_{j,r},\psi_i,\beta_{i,r}} \quad \sum_{i=1}^{n}\psi_i + \theta\cdot\sum_{r=1}^{R}\sum_{j=1}^{d} w_{j,r} \tag{6.18}$$

$$\text{s.t.}\quad \psi_i \ge 0,\ 1\le i\le n,$$

$$\psi_i \ge 1 - \sum_{j=1}^{d} a_{i,j}w_{j,r},\ \text{for } y_i = 1,\ 1\le r\le R,$$

$$\psi_i \geq \left(\sum_{r=1}^{R} \beta_{i,r}\right) - (R-1), \text{ for } i \text{ s.t. } y_i = 0,$$

$$\beta_{i,r} \geq a_{i,j} w_{j,r}, \text{ for } i \text{ s.t. } y_i = 0,\ 1 \leq j \leq d,\ 1 \leq r \leq R,$$

$$w_{j,r} \in \{0,1\},\ 1 \leq j \leq d,\ 1 \leq r \leq R.$$

Relaxing the decision variables to $0 \leq w_{j,r} \leq 1$ leads to a linear program.

Unfortunately, numerical results suggest that this linear programming relaxation is likely to have fractional values in the optimal solution $w_{j,r}$, and the optimal ψ_i may possibly be all close to 0, which may be undesirable since ψ_i aims to represent the 0-1 error cost term. A possible reason is that the gap between the convex and concave interpolations may loosen the linear program and enable fractional results with lower costs than binary solutions.

6.3.2 Block Coordinate Descent Algorithm

We now propose an algorithm that iteratively updates a single subset of features $\mathbf{X}_r$ with a fixed r, where $\mathbf{X}_r$ is defined in the two-level Boolean function (6.3). In other words, this algorithm considers the decision variables in a single clause ($w_{j,r}$ with a fixed r) as a block of coordinates, and performs block coordinate descent to minimize the Hamming distance cost in (6.12). Each iteration updates a single clause with all the other clauses fixed, using the one-level Boolean function learning algorithm in [145]. We denote r_0 as the clause to be updated.

The optimization of (6.12) even with $(R-1)$ clauses fixed still involves a joint minimization over w_{j,r_0} and the ideal clause outputs $v_{i,r}$ for $y_i = 0$ ($v_{i,r} = 1$ for $y_i = 1$ and thus fixed), so the exact solution could still be challenging. To simplify, we fix the values of $v_{i,r}$ for $y_i = 0$ and $r \neq r_0$ to the actual clause outputs $\hat{v}_{i,r}$ in (6.4) with the current $w_{j,r}$ ($r \neq r_0$). Now we assign v_{i,r_0} for $y_i = 0$: if there exists $v_{i,r} = \hat{v}_{i,r} = 0$ with $r \neq r_0$, then this training example is guaranteed to be correctly classified and we can assign $v_{i,r_0} = \text{DC}$ to minimize the objective in (6.12); in contrast, if $\hat{v}_{i,r} = 1$ holds for all $r \neq r_0$, then the constraint (6.13) requires $v_{i,r_0} = 0$.

This derivation leads to the updating process as follows. To update the r_0^{th} clause, we remove all training examples that have label $y_i = 0$ and are already predicted as 0 by at least one of the other $(R-1)$ clauses, and then update the r_0^{th} clause with the remaining training examples using the one-level function learning algorithm.

There are different choices of which clause to update in an iteration, such as cyclical or random updating. We can also try the update for each clause and then greedily choose the one with the minimum cost, as is used in our experiments.

The initialization of $w_{j,r}$ also has different choices. For example, one option is the set covering method, as is used in our experiments. Random or all-zero initialization can also be used.

6.3.3 Alternating Minimization Algorithm

This section proposes the alternating minimization algorithm that uses the Hamming distance formulation (6.12). This algorithm alternately minimizes between the decision variables $w_{j,r}$ and the ideal clause outputs $v_{i,r}$. Each iteration has two steps: update $v_{i,r}$ with the current $w_{j,r}$, and update $w_{j,r}$ with the new $v_{i,r}$. The latter step is simpler and will be first discussed.

With fixed values of $v_{i,r}$, the minimization over $w_{j,r}$ is relatively straight-forward: the objective in (6.12) is separated into R terms, each of which depends only on a single clause $w_{j,r}$ with a fixed r. Thus, all clauses are decoupled in the minimization over $w_{j,r}$, and the problem becomes parallel learning of R one-level clauses. Explicitly, the update of the r^{th} clause removes training examples with $v_{i,r} = \text{DC}$ and then uses the one-level Boolean function learning algorithm [145].

The update over $v_{i,r}$ with fixed $w_{j,r}$ follows the discussion in Section 6.2.2: for positive training examples $y_i = 1$, $v_{i,r} = 1$, and for the negative training examples $y_i = 0$, $v_{i,r_0} = 0$ for r_0 defined in (6.8) and $v_{i,r} = \text{DC}$ for $r \neq r_0$. For negative training examples with a "tie", i.e. non-unique r_0 in (6.8), tie breaking is achieved by a "clustering" approach. First, for each clause $1 \leq r_0 \leq R$, we compute its cluster center in the feature space by taking the average of $a_{i,j}$ (for each j) over training examples i for which r_0 is minimal in (6.8) (including ties). Then, each training example with a tie is assigned to the clause with the closest cluster center in ℓ_1-norm among all minimal r_0 in (6.8).

Similar to the block coordinate descent algorithm, various options exist for initializing $w_{j,r}$ in this algorithm. The set covering approach is used in our experiments.

6.3.4 Complexity of the Linear Programming Formulations

We now consider computational complexity of the proposed algorithms by characterizing the dimensions of the linear programming formulations. If we denote n_1 and n_0 as the numbers of training examples with $y_i = 1$ and $y_i = 0$, respectively, then the two-level linear program in (6.18) has $O(Rd + Rn_0 + n)$ variables and $O(Rn_1 + Rdn_0)$ constraints. The block coordinate descent and the alternating minimization algorithms are both iterative, and require solving R linear programs per iteration. However, each single linear program does not use all the training data; if n_r denotes the number of training examples used for updating a clause, then the linear program to update that clause has $O(d + n_r)$ variables and $O(d + n_r)$ constraints. Thus, despite having to solve multiple linear programs, the reduction in the dimensions of each single linear program can still result in greater overall efficiency compared with the non-iterative two-level linear programming formulation.

6.3.5 Redundancy Aware Binarization

This section discusses a solution to a potential issue with the linear programming relaxation that is widely used in this chapter. If the optimal solution to a linear program turns out to have fractional values, then we need to convert them into binary. If a linear program already yields a binary optimal solution, then the binarization methods here will not change it.

A straight-forward binarization method is to compare each $w_{j,r}$ from the linear programs with a specified threshold. However, empirical results suggest that the resulting binarized function may have *redundancy*, making the function unnecessarily complex.

The following improved binarization method considers three types of redundancies, each associated with a set of binary features that we call a *redundancy set*. Among the features in each redundancy set, no more than one feature will appear in any single clause of an optimal CNF function[5].

The first type of redundancy set corresponds to *nested* features. If binary features a_{i,j_1} and a_{i,j_2} satisfy $a_{i,j_1} \leq a_{i,j_2}$ for all training examples, then these two features cannot both appear in a single clause in the optimal CNF function; otherwise, since $a_{i,j_1} \bigvee a_{i,j_2} = a_{i,j_2}$, removing a_{i,j_1} from the clause keeps the same output and reduces the total number of features, leading to a better function. If there is a nested set $a_{i,j_1} \leq a_{i,j_2} \leq \ldots \leq a_{i,j_P}$ $(\forall 1 \leq i \leq n)$, then at most one feature from this set can be selected in a single clause in the optimal CNF function.

The second type consists of complementary binary features and applies when we use the mechanism to "disable" a clause as explained in Section 6.2. Since complementary features a_{i,j_1} and a_{i,j_2} satisfy $a_{i,j_1} \bigvee a_{i,j_2} = 1$ $(\forall 1 \leq i \leq n)$, the optimal CNF function cannot have both of them in a single clause, otherwise disabling this clause by $w_{0,r} = 1$ and $w_{j,r} = 0$ $(j > 0)$ keeps the output and improves function simplicity.

The third type also applies only when we use the mechanism to disable a clause. This type can happen with two nested sets that are pairwise complementary, and an example of such sets is binary features obtained by thresholding continuous-valued features. To illustrate, suppose we have six binary features from thresholding the same continuous feature c_i with thresholds $\tau_1 < \tau_2 < \tau_3$:

$$a_{i,1} = (c_i \leq \tau_1)\,,\ a_{i,2} = (c_i \leq \tau_2)\,,\ a_{i,3} = (c_i \leq \tau_3)\,,$$
$$a_{i,4} = (c_i > \tau_1)\,,\ a_{i,5} = (c_i > \tau_2)\,,\ a_{i,6} = (c_i > \tau_3)\,.$$

The "zigzag" path $(a_{i,4}, a_{i,5}, a_{i,2}, a_{i,3})$ forms a redundancy set, since at most one out of the four features can be selected in a fixed clause of the optimal CNF function, otherwise either the first or the second redundancies above will happen

[5]This statement holds for both formulations (6.6) and (6.12); for simplicity, we will focus on the formulation (6.6) for illustration.

and thus the function is not optimal. There are typically multiple "zigzag" paths, e.g. $(a_{i,4}, a_{i,1}, a_{i,2}, a_{i,3})$ and $(a_{i,4}, a_{i,5}, a_{i,6}, a_{i,3})$.

The new binarization approach takes the above types of redundancies into account. For illustration, suppose all binary features are obtained by thresholding continuous-valued features. For a given clause and a single continuous-valued feature, we may sweep over all non-redundant combinations of the binary features induced by this continuous feature and obtain the one with minimal cost. Since the total number of non-redundant combinations for nested and zigzag features is linear and quadratic with the number of thresholds, respectively, the sweeping is efficient with a single continuous feature. However, for multiple continuous features, joint minimization is combinatorial and challenging. Thus, to simplify matters, we first sort continuous features in decreasing order as determined by the sum of corresponding decision variables in the optimal solution to the linear programming relaxation. Then the decision variables corresponding to each continuous feature are sequentially binarized as described above.

6.4 Numerical Evaluation

6.4.1 Setup

This section evaluates the algorithms with UCI repository datasets [163]. To facilitate comparison with the most relevant prior work [145], we use the same 8 datasets as in that work: Indian liver patient dataset (ILPD), Ionosphere (Ionos), BUPA liver disorders (Liver), Parkinsons (Parkin), Pima Indian diabetes (Pima), connectionist bench sonar (Sonar), blood transfusion service center (Trans), and breast cancer Wisconsin diagnostic (WDBC). Each continuous-valued feature is converted to binary using 10 quantile-based thresholds.

The goal is to learn a DNF function ("OR-of-ANDs") from each dataset. We use stratified 10-fold cross validation and then average the test and training error rates. All linear programs are solved by IBM CPLEX version 12 [120]. The regularization

Table 6.1: Ten-fold average test error rates (unit: %). Standard error of the mean is shown in parentheses.

Dataset	TLP	BCD	AM	OCFL	SC	DList	C5.0	CART
ILPD	28.6(0.3)	28.6(0.2)	28.6(0.2)	28.6(0.2)	28.6(0.2)	36.5(1.4)	30.5(2.0)	32.8(1.3)
Ionos	8.3(1.2)	9.4(1.1)	11.4(1.1)	9.7(1.5)	10.5(1.3)	19.9(2.3)	7.4(2.1)	10.8(1.2)
Liver	44.9(0.9)	37.1(3.2)	39.1(2.5)	45.8(2.2)	41.7(2.8)	45.2(2.6)	36.5(2.4)	37.1(2.5)
Parkin	14.4(1.4)	12.8(2.2)	15.9(2.9)	16.4(2.1)	14.9(1.9)	25.1(3.3)	16.4(2.7)	13.9(2.9)
Pima	26.8(1.8)	26.8(1.7)	23.8(2.0)	27.2(1.5)	27.9(1.5)	31.4(1.6)	24.9(1.7)	27.3(1.5)
Sonar	31.3(3.2)	29.8(3.0)	25.5(2.4)	34.6(2.7)	28.8(2.9)	38.5(3.6)	25.0(4.2)	31.7(3.5)
Trans	23.8(0.8)	23.8(0.1)	23.8(0.1)	23.8(0.1)	23.8(0.1)	35.4(2.4)	21.7(1.2)	25.4(1.7)
WDBC	7.6(1.1)	6.2(1.2)	6.5(0.9)	9.3(2.0)	8.8(2.0)	9.7(0.8)	6.5(1.1)	8.4(1.0)

parameter is $\theta = A \times 10^B$ where we sweep $A = 1, 2, 5$ and $B = -4, -3, -2, -1, 0, 1$, for a total of 18 values. We use the redundancy aware binarization and the mechanism to "disable" a clause.

Algorithms in comparison and their abbreviations are: two-level linear programming relaxation (TLP), block coordinate descent (BCD), alternating minimization (AM), one-level conjunctive function learning (OCFL, equivalent to setting $R = 1$ for BCD or AM) and set covering (SC), the last two from [145], decision list in IBM SPSS (DList), and decision trees (C5.0: C5.0 with rule set option in IBM SPSS, CART: classification and regression trees algorithm in Matlab's classregtree function). The maximum number of iterations in BCD and AM is set as 100. Without loss of generality, we set the maximum number of clauses $R = 5$ for TLP, BCD, AM, and SC.

We show the test error rates, the total number of features used in the functions, and Pareto fronts indicating the tradeoff between accuracy and simplicity.

6.4.2 Accuracy and Function Simplicity

In this section, we apply a second cross validation within the training partition to choose the optimal parameter θ among the 18 values that has the highest accuracy, and then evaluate its performance on the test partition. The mean test error rates and the standard error of the mean are listed in Table 6.1. We refer the reader to [145] for results from other classifiers that are not interpretable; the accuracy of

Table 6.2: Ten-fold average numbers of features

Dataset	TLP	BCD	AM	SC	DList	C5.0
ILPD	4.8	0.0	0.0	0.0	5.4	45.5
Ionos	30.9	12.4	12.9	11.1	7.7	13.6
Liver	6.5	9.3	7.7	5.2	2.2	46.6
Parkin	9.0	8.2	12.6	3.2	2.1	16.6
Pima	15.3	2.2	2.0	2.4	8.6	38.2
Sonar	27.8	14.2	23.6	9.0	1.9	27.3
Trans	3.5	0.0	0.0	0.0	3.8	6.7
WDBC	21.8	13.6	11.9	8.7	4.0	15.8

our algorithms is generally competitive with them.

Table 6.2 provides the 10-fold average of the total number of features in the learned functions as a measure for function simplicity. No features are counted if a clause is disabled.

Table 6.1 shows that two-level functions obtained by our algorithms (TLP, BCD, and AM) are more accurate than the one-level functions from OCFL for most datasets, which demonstrates the expressiveness of two-level functions.

Among optimization-based two-level Boolean function learning approaches, BCD and AM generally have superior accuracy to TLP and SC. All these four approaches substantially outperform DList in terms of accuracy on all datasets. Compared with C5.0, BCD and AM obtain simpler functions (i.e. smaller total numbers of features) with quite competitive accuracy. Compared with CART, BCD has higher or equal accuracy on all datasets, and AM is also superior overall. In addition, AM achieves the highest accuracy on the Pima dataset among the interpretable models in Table 6.1, and BCD obtains the highest accuracy on the Parkin and WDBC datasets.

To evaluate the new redundancy aware binarization, we compare the numbers of features in the learned functions of the BCD, AM, and SC algorithms with $R = 5$ using the simple threshold comparison (threshold at 0.2) and the new binarization methods. Table 6.3 shows the average results over 10 folds and across all the datasets. We can see that the redundancy aware binarization substantially reduces the number of features and thus improves function simplicity.

Table 6.3: Average numbers of features with different binarization

Binarization Method	BCD	AM	SC
Threshold Comparison	11.9	11.2	12.3
Redundancy Aware	7.5	8.8	5.0

As a brief evaluation of the clause disabling mechanism, we consider the number of "active" clauses (i.e. not disabled by setting $w_{0,r} = 1$) in the learned functions, again averaged over 10 folds and 8 datasets. BCD, AM, and SC use on average 2.1, 2.4, and 1.7 active clauses, respectively, showing that clause disabling is effective when the maximum number of clauses $R = 5$ is larger than needed. In addition, the average number of clauses for DList is 2.7; thus, BCD and AM outperform DList in function simplicity if we consider the total number of clauses as an alternative measure.

6.4.3 Pareto Fronts

A DNF function can be considered *dominated* by another function if the former has higher error rate and uses more features than the latter. For a fixed algorithm, if we learn DNF functions with each of the 18 values of θ, then the pairs of accuracy and total number of features of the DNF functions that are not dominated by any other DNF function constitute the *Pareto front* for this algorithm [164], which shows the optimal tradeoff boundary in the space of accuracy v.s. function simplicity achieved by varying the regularization parameter. The Pareto fronts of the BCD, AM, and SC algorithms are shown in Figure 6-1, where we include the results for both using and not using the clause disabling mechanism for each algorithm. For ease of visualization, the dominated points are not shown and non-dominated points are connected by line segments. We use the Liver and Pima datasets as illustrating examples.

Comparing the Pareto fronts in Figure 6-1, we can have the following observations. For the more difficult Liver dataset where $R = 5$ clauses are not too many, using or not using clause disabling (i.e. more or less regularization) accesses different parts of the space of accuracy v.s. function simplicity, as shown in Figure 6-1 (b). In contrast, $R = 5$ is already unnecessarily large for the simpler Pima dataset, so clause disabling allows a much improved tradeoff (Pareto fronts to the lower left in Figure 6-1 (d)). In addition, the Pareto fronts clearly show the superiority of BCD and AM to SC.

6.5 Chapter Conclusion and Future Work

This chapter focuses on the two-level Boolean functions as an example of the multivariate function composition that is discussed in Section 1.1.3. By composing the simpler model of one-level Boolean functions, the two-level functions significantly improve the model richness and can represent any Boolean function if the negation of each input feature is available.

We provide two optimization-based formulations for learning two-level Boolean functions from a dataset. The first formulation is based on 0-1 classification error and the second on Hamming distance. Based on these formulations, we propose algorithms based on linear programming relaxation, block coordinate descent, and alternating minimization. For a variant of the Hamming distance formulation in the noiseless case, sufficient conditions are provided for the relaxed formulation to have binary optimal solutions, although the relaxed formulation remains non-convex and possibly challenging to solve precisely.

Numerical results show that two-level Boolean functions typically have considerably lower error rates than one-level functions. In addition, the block coordinate descent and alternating minimization algorithms provide very good tradeoffs between accuracy and function simplicity.

The clause disabling mechanism is effective for improving the accuracy-simplicity tradeoff on simple datasets when the specified number of clauses is too large. The new redundancy-aware binarization has been shown to reduce the total number of features compared with simple thresholding binarization.

There are a few directions for future work. First, alternative accuracy cost can be used in the problem formulation. For instance, the Hamming distance cost empirically improves the performance of the iterative algorithms; however, the 0-1 error cost is naturally used for performance evaluation in many situations. Since the

Hamming cost may over-penalize the classification error, formulations that combine the Hamming cost and the 0-1 cost are possibly useful to further improve the performance. Second, alternative initializations can be used for the block coordinate descent and alternating minimization algorithms, and it is also interesting to test the sensitivity of these algorithms on the initialization. Third, other binarization methods can be explored that yield binary variables with potentially less modification to the fractional solutions of the linear programming relaxation. Another interesting and more sophisticated approach to obtain binary solutions is to utilize integer programming techniques such as branch-and-bound. Fourth, there are opportunities from the theoretical perspective. It would be interesting to consider conditions more general than the theorems in Section 6.2.2, which guarantee that the optimal solutions to the relaxed formulation remain binary. Moreover, although more challenging, it is useful to characterize the gap between the solutions obtained by our algorithms and the optimal solution to the non-convex formulation with the Hamming distance cost. Fifth, we can design methods for the conversion from general feature variables to binary variables other than the quantile-based thresholding that is used in simulation, which may improve the overall performance.

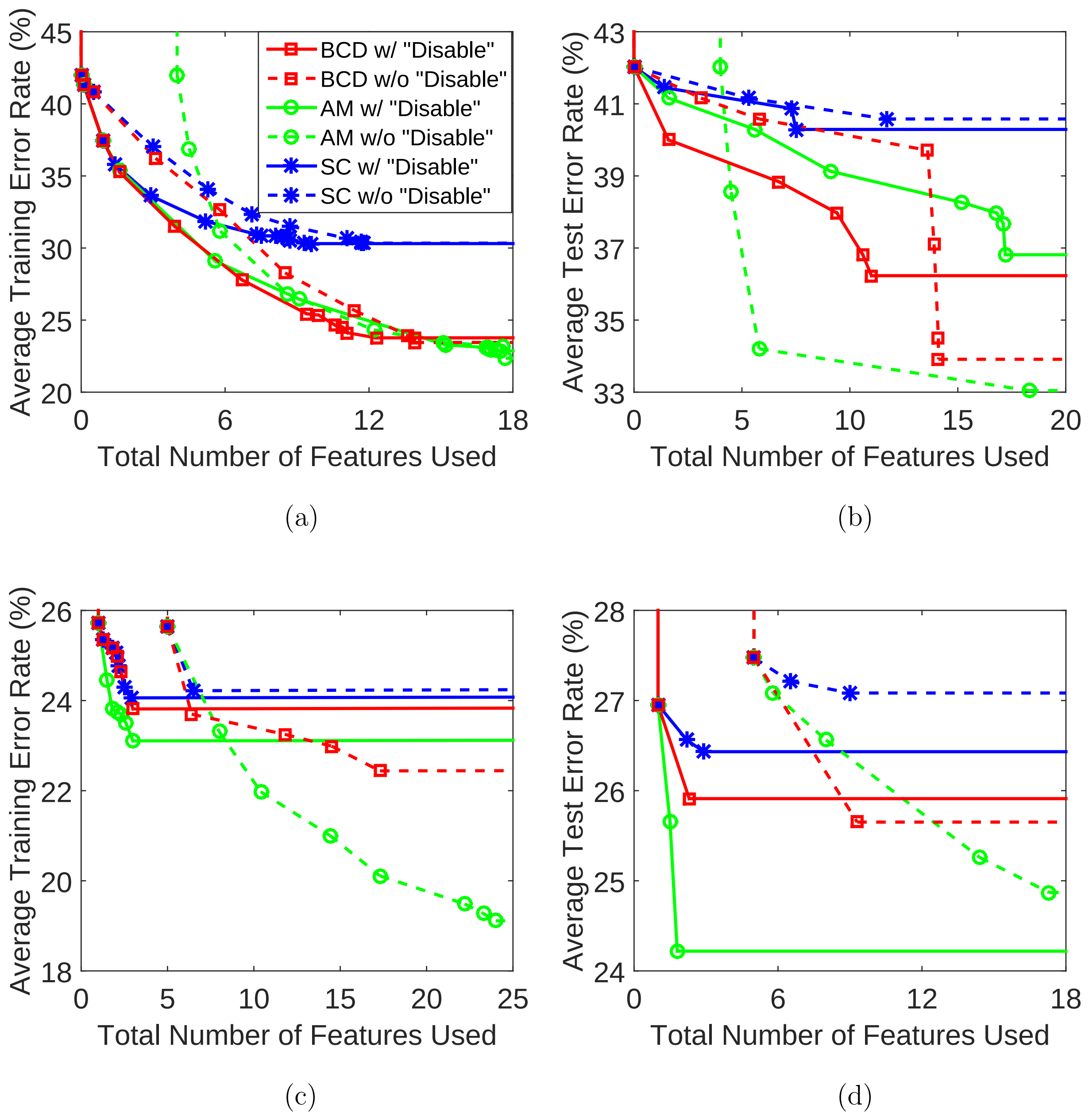


Figure 6-1: Pareto fronts: (a) Liver training error rate, (b) Liver test error rate, (c) Pima training error rate, (d) Pima test error rate. All figures use the same legends as in (a).

Chapter 7

Conclusion and Future Work

In this thesis, parameter estimation algorithms are provided for structures for system representation that can be viewed as a type of composition, which includes operator composition, modular composition, and multivariate function composition. These composition structures all belong to constrained parametric representations, which have wide applications in signal processing. We focus on a few important classes of systems with a composition structure, which include both linear and nonlinear systems with continuous-valued signals, as well as Boolean functions that have binary input and output variables. Utilizing proper composition can potentially lead to a number of advantages in signal processing, including the reduction of the total number of independent parameters that achieves computational and representational efficiency, structural modularity that can benefit hardware implementation, and improvement of model richness with the ability to form more sophisticated models of systems or functions by the composition of simple ones.

The first part of this thesis considers operator composition that naturally corresponds to a cascade of systems, where each system corresponds to an operator that works on the set of signals of interest. The cascade representations of both linear and nonlinear systems are discussed in this thesis, where the former focuses on the example of 2D FIR filters, and the latter considers the nonlinear block-oriented models with a cascade of memoryless nonlinear functions and LTI blocks. For the 2D FIR filters, the aim is to approximate a 2D filter by the cascade of a pair of 2D filters with lower orders, which can reduce the total number of independent parameters and achieve computational efficiency for spatial domain implementation. Equivalently, a 2D signal can be approximated by the convolution of a pair of 2D signals with shorter horizontal and vertical lengths, which can result in representational compression. In the transform domain, these two problems both become the approximate bivariate polynomial factorization. We formulate the factorization problem into rank-one matrix approximation in a dimensionally lifted parameter space, and propose a new algorithm that is referred to as lifted alternating minimization. This new algorithm is shown to outperform the other methods in comparison in our simulation with synthetic polynomials, followed by the existing singular value projection algorithm that also works with the low rank matrix formulation. An opportunity for future improvement on the algorithms based on low rank matrix approximation is in the situation where the two polynomial factors have exactly the same degrees, since this situation leads to empirically reduced performance possibly due to higher geometrical complication in the solution space. In addition, alternative criteria for approximation other than the ℓ_2-norm or Frobenius-norm of error could be a possible direction for future work.

For the block-oriented cascade representations of nonlinear systems, we focus on the following two structures. The first structure is a discrete-time Hammerstein model in which the linear subsystem is general, and we generalize an existing estimation procedure from a finite-dimensional to an infinite-dimensional parameter space, using a generalized SVD technique for infinite-dimensional quasimatrices. For the second structure, we aim to model a nonlinear system by its inverse, where the model for the inverse system is a cascade of memoryless nonlinear functions and FIR filters. This may be considered as a generalization of the all-pole modeling technique with additional flexibility due to the introduction of nonlinear functions. In numerical simulation, the gradient descent method overall outperforms the other approaches we consider. However, the parameter estimation problem becomes challenging with the increase of the total number of blocks in the cascade, which leaves room for future improvement.

The second part of this thesis discusses modular composition that refers to the replacement of each delay element in a system block diagram with an identical copy of another system module. We focus on the example of modular Volterra system, which has structural modularity that can potentially simplify the design and verification for hardware implementation. Under the assumption that the coefficients of the nonlinear kernels of the Volterra module have sufficiently smaller magnitude compared with those of the linear kernel, we propose a two-step parameter estimation algorithm using the statistical information between the input and output signals. The first step of this algorithm neglects the nonlinear kernels and estimates the remaining parameters, and then the second step fills in the estimation of the nonlinear kernels. Numerical evaluation shows that the algorithm is effective when the order and delay (i.e. the number of state variables) of the modular system are not high. In addition, two structures for the estimated system are used in numerical evaluation: one is modular Volterra system with the correct structural parameters, and the other is modular FIR filter without the nonlinear kernels but with the same number of degrees of freedom. A comparison between these two structures shows that the former structure has much better performance and thus modeling the nonlinear kernels is of significant importance. Possible directions for future work include parameter estimation in the more challenging situation without the assumption of weak nonlinearity, as well as determining the physical devices and systems that naturally have a modular Volterra structure.

The third part of this thesis focuses on learning two-level Boolean functions from a training dataset with the joint criteria of accuracy and function simplicity, where the two-level Boolean functions can be interpreted as the composition of certain one-level multivariate Boolean functions. These two-level functions can represent any Boolean function when the negation of each input variable is available, which thus have significantly higher model richness compared with the one-level functions. Two optimization-based formulations are provided with the first based on the 0-1 classification error and the second on Hamming distance. With these formulations, we propose algorithms that utilize linear programming relaxation, block coordinate descent, and alternating minimization. For a variant of the Hamming distance formulation in the noiseless case, sufficient conditions are provided for the relaxed formulation to have binary optimal solutions. Numerical results show that two-level Boolean functions typically have considerably lower error rates than one-level functions. The block coordinate descent and alternating minimization algorithms provide very good tradeoffs between accuracy and function simplicity. These two algorithms overall outperform the other algorithms except for C5.0 in terms of accuracy; compared with C5.0, the block coordinate descent and alternating minimization algorithms have similar accuracy and much better function simplicity. Moreover, the clause disabling mechanism is effective for improving the accuracy-simplicity tradeoff on simple datasets when the specified number of clauses is too large, and the new redundancy-aware binarization has been shown to reduce the total number of features compared with simple thresholding binarization. Directions for future work include alternative accuracy cost in the formulation, alternative initialization approaches for the iterative algorithms, and alternative methods for the conversion from continuous-valued feature variables to binary variables.

Appendix A

Further Discussion on Operator Mapping

This appendix shows further discussion on operator mapping in Section 1.1.4 that is used to mathematically describe the modular composition. In particular, we focus on the modular composition by replacing each delay element in a FIR filter $F(z^{-1})$ with an identical copy of another system module $G\{\cdot\}$, as is shown in Figure 1-2. In Section 1.1.4, we have introduced that the modular composition can be interpreted from the following perspective of operator mapping: the system within which we embed another module creates a higher-level operator, which has both its input and output as lower-level operators that work on the set of signals.

For a FIR filter $F(z^{-1})$ in the direct form structure, the higher-level operator $\mathcal{F}$ has a close relationship with the transfer function $F(z^{-1})$. If we denote $\mathcal{F}\langle G\rangle\{\cdot\}$ as the system obtained by embedding $G\{\cdot\}$ into $F(z^{-1})$, then the operator $\mathcal{F}$ associated with Figure 1-2 (b) satisfies

$$\begin{aligned}
\mathcal{F}\langle G\rangle\{x[n]\} &= a_0 \cdot x[n] + a_1 \cdot G\{x[n]\} + a_2 \cdot (G \circ G)\{x[n]\} + \cdots + a_M \cdot G^{[M]}\{x[n]\} \\
&= \sum_{m=0}^{M} a_m \cdot G^{[m]}\{x[n]\} \\
&= \left(\sum_{m=0}^{M} a_m \cdot G^{[m]}\right)\{x[n]\}, \qquad \text{(A.1)}
\end{aligned}$$

where $G^{[m]}\{\cdot\}$ denotes the cascade system with m modules of $G\{\cdot\}$. Consequently, if we symbolically substitute z^{-m} in $F(z^{-1})$ with $G^{[m]}$ (i.e. interpreting the m^{th} power of the independent variable as the m^{th} iterate of the operator), then $F(z^{-1})$ becomes $\mathcal{F}\langle G\rangle$.

For other block diagram structures of the FIR system $F(z^{-1})$ that are different from Figure 1-2 (a), the relationship (A.1) does not necessarily hold. Generally, the modular system $\mathcal{F}\langle G\rangle\{\cdot\}$ with an arbitrary module $G\{\cdot\}$ depends not only on the transfer function $F(z^{-1})$ but also on the block diagram structure[1]. As an example, Figure A-1 (a) shows the transposed structure of the direct form [26], which has the same transfer function as Figure 1-2 (a); however, the modular systems in Figure A-1 (b) and Figure 1-2 (b) are typically different for a general module $G\{\cdot\}$. While this structure-dependent property may cause potential challenges in obtaining the closed-form operator expression for a general modular system, it also introduces potential flexibility to represent different systems with the same key blocks and different interconnections.

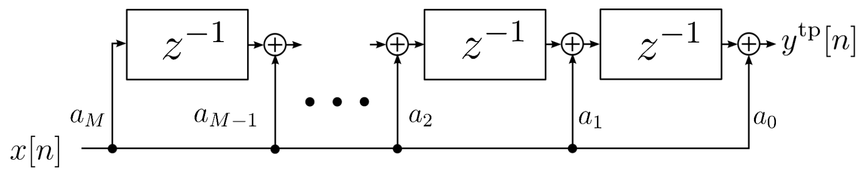
Since the modular filter in Figure 1-1 (b) can be recovered from the modular system in Figure 1-2 (b) with the additional constraint that $G\{\cdot\}$ is a LTI filter, we revisit the modular filter from the perspective of operator mapping. Since $G\{\cdot\}$ is a linear filter, its multiple cascade $G^{[m]}\{\cdot\}$ has the transfer function as $(G(z^{-1}))^m$. Thus, applying the relationship (A.1), the transfer function of the system $\mathcal{F}\langle G\rangle\{\cdot\}$ becomes exactly the function composition $(F\circ G)(z^{-1})$. In addition, when $G\{\cdot\}$ is a LTI filter, the modular systems generated from different block diagram structures[2] of $F(z^{-1})$ have the same transfer function and are thus structure-independent.

Finally, we have a few remarks on the operator mapping that is discussed above.

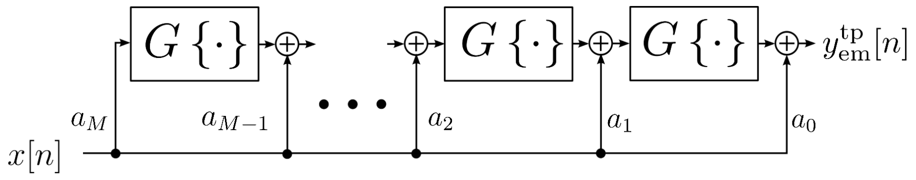
(a) The operator mapping could be considered as a generalization of the *matrix polynomial* [165] in linear algebra, which refers to a polynomial with square

[1]For simplicity, the influence of initial states of delay elements is not considered in our discussion, i.e. we assume that all delay elements are initial-rest.

[2]We assume that the block diagrams have only delay elements, scaling coefficients, and addition blocks. The conclusion may not hold for more sophisticated structures with expanders or decimators, e.g. polyphase decomposition [26].



(a): FIR filter $F(z^{-1})$ in the transposed structure of the direct form



(b): Modular system with the transposed structure of $F(z^{-1})$

Figure A-1: Embedding $G\{\cdot\}$ into the FIR filter $F(z^{-1})$ in the transposed structure of the direct form.

matrices as variables and using matrix multiplication rules. Each matrix naturally corresponds to a linear operator; however, the operators in our discussion can be nonlinear and do not need to correspond to a matrix.

(b) The operator mapping could be considered as a generalization of the second interpretation of univariate function composition at the beginning of Section 1.1.1, if we substitute the operator $\mathcal{F}_{\mathrm{lc}}$ that works on set of functions by the higher-level operator $\mathcal{F}$ that works on the set of lower-level operators $\mathscr{U}$ in (1.4).

(c) For clarity, we compare cascade system and modular system in Table A.1 on both the description from the perspective of operators and the property of the transfer functions in the special situation where the systems involved are constrained to be LTI systems.

Table A.1: Comparison between cascade and embedded systems

System Structure	Operator Description	Transfer Function if the Systems are LTI
Cascade System	Operator Composition	Multiplication
Modular System	Operator Mapping	Function Composition

Appendix B

Alternative Low Rank Matrix Approximation Algorithms

This appendix reviews three existing algorithms for low rank matrix approximation, namely nuclear norm minimization [114], singular value projection (SVP) [115], and atomic decomposition for minimum rank approximation (ADMiRA) [116]. These algorithms are applied to our problem of bivariate polynomial factorization and compared in the numerical simulation in Section 3.4.3. The formulation (3.22) with the squared ℓ_2-norm of approximation error is used for the algorithms in this appendix, namely

$$\begin{aligned} \min_{\mathbf{M}} \quad & V_{\text{poly}}(\mathbf{M}) \triangleq \sum_{i=0}^{P} \sum_{j=0}^{Q} \left(h_{i,j} - \sum_{(a,b) \in \mathcal{C}(i,j)} M_{a,b} \right)^2, & \text{(B.1)} \\ \text{s.t.} \quad & \mathbf{M} \text{ is rank one}, & \text{(B.2)} \end{aligned}$$

where the index sets $\mathcal{C}(i,j)$ are from (3.18).

B.1 Nuclear Norm Minimization

The nuclear norm minimization algorithm considers a convex formulation that is related to the low rank matrix approximation problem [114]. In particular, the rank constraint (B.2) is converted into the objective function as follows:

$$\min_{\mathbf{M}} \sqrt{\sum_{i=0}^{P}\sum_{j=0}^{Q}\left(h_{i,j} - \sum_{(a,b)\in\mathcal{C}(i,j)} M_{a,b}\right)^2} + \lambda \cdot \operatorname{rank}(\mathbf{M}), \tag{B.3}$$

where λ is the weight of the rank term.

Since the rank of a matrix is a non-convex function, it can be replaced by the convex *nuclear norm*, which refers to the summation of the singular values of a matrix, i.e.

$$\min_{\mathbf{M}} \sqrt{\sum_{i=0}^{P}\sum_{j=0}^{Q}\left(h_{i,j} - \sum_{(a,b)\in\mathcal{C}(i,j)} M_{a,b}\right)^2} + \lambda \cdot \|\mathbf{M}\|_*, \tag{B.4}$$

where $\|\mathbf{M}\|_*$ denotes the nuclear norm. The convex formulation (B.4) is already solvable by existing optimization solvers such as CVX [133]. From the perspective of convex approximation, the relationship between the nuclear norm and the rank of a matrix [114] is similar to that between the ℓ_1-norm and the ℓ_0-norm of a vector, where the latter relationship has been widely used in compressive sensing [4]. More generally, the approximation techniques of both the nuclear norm and ℓ_1-norm belong to the class of atomic norms [166].

B.2 Singular Value Projection

The singular value projection (SVP) algorithm has an iterative process [115]. In the k^{th} iteration, gradient descent is performed at the current solution $\mathbf{M}^{(k-1)}$ with respect to the objective function (B.1), and then the result is projected to the rank-one matrix set by taking SVD and keeping only the largest singular value. A detailed description of this algorithm is as follows.

```
SINGULAR VALUE PROJECTION [115]
(1)  Initialize $\mathbf{M}^{(0)} = \mathbf{0}$. Let $k = 1$.
(2)  In the $k^{\text{th}}$ iteration, perform the following steps:
(3)      Compute the gradient: $\mathbf{G}_{\mathbf{M}}^{(k)} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{M}} V_{\text{poly}}(\mathbf{M}^{(k-1)})$.
```

(4) Perform gradient descent: $\mathbf{M}_{\text{temp}}^{(k)} = \mathbf{M}^{(k-1)} - \mu \cdot \mathbf{G}_{\mathbf{M}}^{(k)}$.

(5) Project the result onto the set of rank-one matrices:

Compute the singular value decomposition of $\mathbf{M}_{\text{temp}}^{(k)}$, and construct the projection as $\mathbf{M}^{(k)} = \mathbf{u}_{\text{temp}}^{(k)} \cdot \sigma_{\text{temp}}^{(k)} \cdot \left(\mathbf{v}_{\text{temp}}^{(k)}\right)^{\text{T}}$, in which $\sigma_{\text{temp}}^{(k)}$ is the largest singular value of $\mathbf{M}_{\text{temp}}^{(k)}$, and $\mathbf{u}_{\text{temp}}^{(k)}$ and $\mathbf{v}_{\text{temp}}^{(k)}$ are the corresponding left and right singular vectors, respectively.

(6) $k \leftarrow k + 1$.

(7) Iterate until the results between consecutive iterations have difference smaller than a threshold, or a threshold on the iteration steps is reached.

(8) Let $\mathbf{f} = \mathbf{u}_{\text{temp}}^{(k)}$ and $\mathbf{g} = \sigma_{\text{temp}}^{(k)} \cdot \mathbf{v}_{\text{temp}}^{(k)}$, and obtain $F(x, y)$ and $G(x, y)$.

Certain conditions are proposed in [115] for the SVP algorithm to converge to the optimal solution with high probability; however, these conditions generally are not guaranteed in our problem.

Practically, in Step (4) that performs gradient descent, the choice of the step size μ has a tradeoff between possibly reducing the likelihood of divergence and potentially improving the convergence rate. A small step size μ makes it less likely to diverge while requires more steps to converge. However, due to the projection in Step (5) onto the set of rank-one matrices, the precise analysis of the convergence criterion with respect to μ in a general setting seems unclear.

B.3 Atomic Decomposition for Minimum Rank Approximation

The atomic decomposition for minimum rank approximation algorithm (ADMiRA) [116] takes iterations to obtain the solution. It can be high-levelly regarded as a generalization of the Compressive Sampling Matching Pursuit (CoSaMP) algorithm [167] from the sparse signal recovery problem to low rank matrix approximation. The ADMiRA algorithm keeps track of a set of candidates for special rank-one matrices (referred to as "atoms" in [116]), and aims to determine a rank-r matrix

$\mathbf{M}$ with low error from the available linear observations $\|\mathbf{h} - \mathcal{H}(\mathbf{M})\|^2$, where $\mathbf{h}$ is the vector of observations and $\mathcal{H}(\cdot)$ is a linear operator from the space of matrices to the observations. In the problem of bivariate polynomial factorization, $\mathbf{h}$ is the vector of reorganized coefficients of the input polynomial $H(x, y)$, and $\mathcal{H}(\cdot)$ corresponds to the summation of elements of the matrix $\mathbf{M}$ in (3.17) that correspond to similar terms.

Each iteration in ADMiRA consists of three main steps: first, new atoms are added to the set of candidates by the reduction of the current approximation error, so the set is enlarged and has more candidates than the target rank of the solution matrix; second, a least squares solution is obtained within the space spanned by the atoms in the enlarged candidate set; third, the solution in the second step is projected to the rank-r matrix set using SVD, and the candidate set is also updated to have only r atoms. A detailed description of this algorithm with the special setting $r = 1$ is as follows.

```
Atomic Decomposition for Minimum Rank Approximation [116]
(1)    Initialize $\mathbf{M}^{(0)} = \mathbf{0}$. Let $k = 1$.
       The candidate set of rank-one matrices $\mathbb{O} = \emptyset$.
(2)    In the $k^{\text{th}}$ iteration, perform the following steps:
(3)      Enlarge the candidate set $\mathbb{O}$:
(3.1)      Compute the matrix $\mathbf{\Delta M} = \mathcal{H}^* \left(\mathbf{h} - \mathcal{H}(\mathbf{M}^{(k-1)})\right)$, where $\mathcal{H}^*$ is the
           adjoint operator of $\mathcal{H}$.
(3.2)      Obtain the SVD of $\mathbf{\Delta M}$, and use the singular vectors associated
           with each of the two largest singular values to construct two
           rank-one matrices referred to as $\mathbf{E}_1$ and $\mathbf{E}_2$.
(3.3)      Let $\mathbb{O} = \mathbb{O} \cup \{\mathbf{E}_1, \mathbf{E}_2\}$.
(4)      Least squares solution in the span of $\mathbb{O}$:
         $\mathbf{M}^{(k)}_{\text{temp}} = \arg\min_{\mathbf{M} \in \text{span}(\mathbb{O})} \|\mathbf{h} - \mathcal{H}(\mathbf{M})\|_2$.
(5)      Project the result onto the set of rank-one matrices:
(5.1)      Compute the singular value decomposition of $\mathbf{M}^{(k)}_{\text{temp}}$.
(5.2)      Construct the projection as $\mathbf{M}^{(k)} = \mathbf{u}^{(k)}_{\text{temp}} \cdot \sigma^{(k)}_{\text{temp}} \cdot \left(\mathbf{v}^{(k)}_{\text{temp}}\right)^{\mathrm{T}}$, in which
           $\sigma^{(k)}_{\text{temp}}$ is the largest singular value of $\mathbf{M}^{(k)}_{\text{temp}}$, and $\mathbf{u}^{(k)}_{\text{temp}}$ and $\mathbf{v}^{(k)}_{\text{temp}}$ are
```

```
        the corresponding left and right singular vectors, respectively.
(5.3)   Update the candidate set $\mathbb{O} = \left\{ \mathbf{u}_{\mathrm{temp}}^{(k)} \cdot \left( \mathbf{v}_{\mathrm{temp}}^{(k)} \right)^{\mathrm{T}} \right\}$.
(6)   $k \leftarrow k + 1$.
(7) Iterate until the results between consecutive iterations have difference
    smaller than a threshold, or a threshold on the iteration steps is reached.
(8) Let $\mathbf{f} = \mathbf{u}_{\mathrm{temp}}^{(k)}$ and $\mathbf{g} = \sigma_{\mathrm{temp}}^{(k)} \cdot \mathbf{v}_{\mathrm{temp}}^{(k)}$, and obtain $F(x, y)$ and $G(x, y)$.
```

For performance guarantee, there are conditions in [116] for the ADMiRA algorithm to obtain a low rank solution that is very close to the optimal matrix; however, similar to the situation with the SVP algorithm, these conditions generally are not guaranteed in our problem.

Appendix C

Proofs for Theorems 3.1 and 3.2

This appendix demonstrates Theorems 3.1 and 3.2 in Section 3.4.2, which establish the relationship between the formulations (3.22) and (3.23).

First, we show the proof for Theorem 3.1.

Proof for Theorem 3.1: Define $\mathbf{D} = \widehat{\mathbf{M}} - \widetilde{\mathbf{M}}$ as the difference between a matrix $\widehat{\mathbf{M}}$ in the linear space $\mathcal{U}_{\text{coeff}}$ and the fixed matrix $\widetilde{\mathbf{M}} \in \mathcal{U}_{\text{rank}}$. Then, our goal is to minimize

$$\|\mathbf{D}\|_{\mathrm{F}}^2 = \sum_a \sum_b D_{a,b}^2 = \sum_{i=0}^{P} \sum_{j=0}^{Q} \left(\sum_{(a,b)\in\mathcal{C}(i,j)} D_{a,b}^2 \right), \tag{C.1}$$

where in the last step we utilize the fact that the index sets $\mathcal{C}(i,j)$ form a partition of all the elements in the matrix $\mathbf{D}$. Therefore, if we could minimize $\sum_{(a,b)\in\mathcal{C}(i,j)} D_{a,b}^2$ for each (i,j), then the Frobenius norm of $\mathbf{D}$ is minimized. From the definition of $\mathcal{U}_{\text{coeff}}$ in (3.21), we know that $\mathbf{D}$ satisfies

$$\sum_{(a,b)\in\mathcal{C}(i,j)} D_{a,b} = h_{i,j} - \sum_{(a,b)\in\mathcal{C}(i,j)} \widetilde{M}_{a,b}.$$

The Cauchy's inequality implies that

$$\sum_{(a,b)\in\mathcal{C}(i,j)} D_{a,b}^2 \geq \frac{1}{|\mathcal{C}(i,j)|} \cdot \left(\sum_{(a,b)\in\mathcal{C}(i,j)} D_{a,b} \right)^2 = \frac{1}{|\mathcal{C}(i,j)|} \cdot \left(h_{i,j} - \sum_{(a,b)\in\mathcal{C}(i,j)} \widetilde{M}_{a,b} \right)^2, \tag{C.2}$$

with the equality achieved when

$$D_{a,b} = \frac{h_{i,j} - \sum_{(a',b')\in\mathcal{C}(i,j)} \widetilde{M}_{a',b'}}{|\mathcal{C}(i,j)|} \tag{C.3}$$

holds for all $(a,b) \in \mathcal{C}(i,j)$.

In summary, the matrix $\widehat{\mathbf{M}}^{\text{proj}}$ that minimizes (3.23) has the expression below

$$\widehat{M}^{\text{proj}}_{a,b} = \widetilde{M}_{a,b} + \frac{R(i,j)}{|\mathcal{C}(i,j)|}, \text{ for } (a,b) \in \mathcal{C}(i,j), \text{ for all } (0,0) \le (i,j) \le (P,Q),$$

where

$$R(i,j) = h_{i,j} - \sum_{(a',b')\in\mathcal{C}(i,j)} \widetilde{M}_{a',b'}.$$

As a result, Theorem 3.1 is proved. ■

Then, we show the proof for Theorem 3.2.

Proof for Theorem 3.2: From the proof for Theorem 3.1 above, we see that for a rank-one matrix $\widetilde{\mathbf{M}} \in \mathcal{U}_{\text{rank}}$, its projection $\widehat{\mathbf{M}}^{\text{proj}}$ onto the linear space $\mathcal{U}_{\text{coeff}}$ achieves the equality in (C.2),

$$\left(h_{i,j} - \sum_{(a,b)\in\mathcal{C}(i,j)} \widetilde{M}_{a,b} \right)^2 = |\mathcal{C}(i,j)| \cdot \sum_{(a,b)\in\mathcal{C}(i,j)} \left(\widehat{M}^{\text{proj}}_{a,b} - \widetilde{M}_{a,b} \right)^2.$$

If we take the summation over all $(0,0) \le (i,j) \le (P,Q)$, then it results in

$$V_{\text{poly}}(\widetilde{\mathbf{M}}) = \sum_{i=0}^{P}\sum_{j=0}^{Q} \left(h_{i,j} - \sum_{(a,b)\in\mathcal{C}(i,j)} \widetilde{M}_{a,b} \right)^2 = \sum_{i=0}^{P}\sum_{j=0}^{Q} \left(|\mathcal{C}(i,j)| \cdot \sum_{(a,b)\in\mathcal{C}(i,j)} \left(\widehat{M}^{\text{proj}}_{a,b} - \widetilde{M}_{a,b} \right)^2 \right).$$

As a result, we obtain the following bounds

$$\left(\min_{(i,j)} |\mathcal{C}(i,j)| \right) \cdot \sum_{i=0}^{P}\sum_{j=0}^{Q} \left(\sum_{(a,b)\in\mathcal{C}(i,j)} \left(\widehat{M}^{\text{proj}}_{a,b} - \widetilde{M}_{a,b} \right)^2 \right) \le V_{\text{poly}}(\widetilde{\mathbf{M}})$$

$$\leq \left(\max_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot \sum_{i=0}^{P}\sum_{j=0}^{Q} \left(\sum_{(a,b)\in\mathcal{C}(i,j)} \left(\widehat{M}_{a,b}^{\text{proj}} - \widetilde{M}_{a,b}\right)^2 \right),$$

and further combining the fact that the sets $\mathcal{C}(i,j)$ form a partition of all the elements in the dimensionally lifted matrices results in

$$\left(\min_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot \left\|\widehat{\mathbf{M}}^{\text{proj}} - \widetilde{\mathbf{M}}\right\|_{\text{F}}^2 \leq V_{\text{poly}}(\widetilde{\mathbf{M}}) \leq \left(\max_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot \left\|\widehat{\mathbf{M}}^{\text{proj}} - \widetilde{\mathbf{M}}\right\|_{\text{F}}^2.$$

Applying the definition of $V_{\text{Fro}}(\widetilde{\mathbf{M}},\ \widehat{\mathbf{M}})$ from (3.23) into the equation above, we have arrived at the statement (3.27).

In order to show (3.28), the optimal solutions to the formulations (3.22) and (3.23) are denoted as

$$\begin{aligned} \mathbf{M}^{\text{poly}} &= \underset{\mathbf{M}\,\in\,\mathcal{U}_{\text{rank}}}{\arg\min} \quad V_{\text{poly}}(\mathbf{M}), && \text{(C.4)} \\ \left(\widetilde{\mathbf{M}}^{\text{Fro}},\ \widehat{\mathbf{M}}^{\text{Fro}}\right) &= \underset{(\widetilde{\mathbf{M}},\ \widehat{\mathbf{M}})\,\in\,\mathcal{U}_{\text{rank}}\times\mathcal{U}_{\text{coeff}}}{\arg\min} \quad V_{\text{Fro}}(\widetilde{\mathbf{M}},\ \widehat{\mathbf{M}}). && \text{(C.5)} \end{aligned}$$

Letting $\widetilde{\mathbf{M}}$ in (3.27) take the value of $\widetilde{\mathbf{M}}^{\text{Fro}}$, we know that $\widehat{\mathbf{M}}^{\text{Fro}}$ becomes the corresponding $\widehat{\mathbf{M}}^{\text{proj}}$, which says

$$V_{\text{poly}}(\widetilde{\mathbf{M}}^{\text{Fro}}) \leq \left(\max_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot V_{\text{Fro}}(\widetilde{\mathbf{M}}^{\text{Fro}},\ \widehat{\mathbf{M}}^{\text{Fro}}) = \left(\max_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot V_{\text{Fro}}^{\text{opt}}.$$

The optimality of $\mathbf{M}^{\text{poly}}$ implies

$$V_{\text{poly}}^{\text{opt}} = V_{\text{poly}}(\mathbf{M}^{\text{poly}}) \leq V_{\text{poly}}(\widetilde{\mathbf{M}}^{\text{Fro}}) \leq \left(\max_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot V_{\text{Fro}}^{\text{opt}}.$$

Letting $\widetilde{\mathbf{M}}$ in (3.27) take the value of $\mathbf{M}^{\text{poly}}$, we know that

$$\begin{aligned} V_{\text{poly}}^{\text{opt}} &= V_{\text{poly}}(\mathbf{M}^{\text{poly}}) \\ &\geq \left(\min_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot V_{\text{Fro}}(\mathbf{M}^{\text{poly}},\ \widehat{\mathbf{M}}^{\text{proj}}) \end{aligned}$$

$$
\begin{aligned}
&\geq \left(\min_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot V_{\text{Fro}}(\widetilde{\mathbf{M}}^{\text{Fro}},\ \widehat{\mathbf{M}}^{\text{Fro}}) \\
&= \left(\min_{(i,j)} |\mathcal{C}(i,j)|\right) \cdot V_{\text{Fro}}^{\text{opt}}.
\end{aligned}
$$

Consequently, the statement (3.28) is demonstrated. As a result, we have completed the proof for Theorem 3.2. ■

Appendix D

Derivation of the Optimal Solution (4.9)

This appendix presents the derivation of the optimal solution (4.9) to the formulation (4.8), under the assumption that the matrix $\mathbf{S}(e^{j\omega})$ in (4.6) has full rank for each ω. In fact, we present the derivation of the multi-input single-output (MISO) Wiener's filter [121, 122]. Figure 4-2 shows that $\tilde{y}_l[n]$ is the output of the LTI system $H_l(e^{j\omega})$ with the input $\tilde{u}_l[n] = f_l(x[n])$, and thus the following correlation functions are obtained from the basic properties of correlation functions [26],

$$\begin{aligned} R_{y,\tilde{y}_l}[m] &\triangleq \mathbb{E}\left\{y[n+m]\cdot \tilde{y}_l^*[n]\right\} = R_{y,f_l}[m] * h_l^*[-m], \\ R_{\tilde{y}_l,\tilde{y}_k}[m] &\triangleq \mathbb{E}\left\{\tilde{y}_l[n+m]\cdot \tilde{y}_k^*[n]\right\} = R_{f_l,f_k}[m] * h_l[m] * h_k^*[-m], \end{aligned}$$

where $R_{y,f_l}[m]$ and $R_{f_l,f_k}[m]$ are introduced in (4.4) and (4.5), respectively; the superscript "$*$" denotes the complex conjugate; $h_l[m]$ is the impulse response of the LTI system $H_l(e^{j\omega})$.

The corresponding power spectral density functions satisfy

$$\begin{aligned} S_{y,\tilde{y}_l}(e^{j\omega}) &= S_{y,f_l}(e^{j\omega})\cdot H_l^*(e^{j\omega}), \\ S_{\tilde{y}_l,\tilde{y}_k}(e^{j\omega}) &= S_{f_l,f_k}(e^{j\omega})\cdot H_l(e^{j\omega})\cdot H_k^*(e^{j\omega}). \end{aligned}$$

Since the output signal $\tilde{y}[n] = \sum_{l=1}^{L} \tilde{y}_l[n]$, the power spectral density functions with $\tilde{y}[n]$ satisfy

$$\begin{aligned} S_{y,\tilde{y}}(e^{j\omega}) &= \sum_{l=1}^{L} S_{y,\tilde{y}_l}(e^{j\omega}) = \sum_{l=1}^{L} S_{y,f_l}(e^{j\omega}) \cdot H_l^*(e^{j\omega}), && \text{(D.1)} \\ S_{\tilde{y},\tilde{y}}(e^{j\omega}) &= \sum_{l=1}^{L}\sum_{k=1}^{L} S_{\tilde{y}_l,\tilde{y}_k}(e^{j\omega}) = \sum_{l=1}^{L}\sum_{k=1}^{L} S_{f_l,f_k}(e^{j\omega}) \cdot H_l(e^{j\omega}) \cdot H_k^*(e^{j\omega}). && \text{(D.2)} \end{aligned}$$

For simplification of the notations, we introduce the vectors $\mathbf{h}(e^{j\omega})$ and $\mathbf{d}(e^{j\omega})$ as

$$\begin{aligned} \mathbf{h}(e^{j\omega}) &\triangleq \begin{bmatrix} H_1(e^{j\omega}) \\ \vdots \\ H_L(e^{j\omega}) \end{bmatrix}, && \text{(D.3)} \\ \mathbf{d}(e^{j\omega}) &\triangleq \begin{bmatrix} S_{y,f_1}(e^{j\omega}) \\ \vdots \\ S_{y,f_L}(e^{j\omega}) \end{bmatrix}. && \text{(D.4)} \end{aligned}$$

We can see that (D.1) and (D.2) become

$$\begin{aligned} S_{y,\tilde{y}}(e^{j\omega}) &= \mathbf{h}^\dagger(e^{j\omega}) \cdot \mathbf{d}(e^{j\omega}), && \text{(D.5)} \\ S_{\tilde{y},\tilde{y}}(e^{j\omega}) &= \mathbf{h}^\dagger(e^{j\omega}) \cdot \mathbf{S}^{\mathrm{T}}(e^{j\omega}) \cdot \mathbf{h}(e^{j\omega}) = \mathbf{h}^\dagger(e^{j\omega}) \cdot \mathbf{S}^*(e^{j\omega}) \cdot \mathbf{h}(e^{j\omega}), && \text{(D.6)} \end{aligned}$$

in which the superscripts "$^{\mathrm{T}}$" and "†" indicate the transpose and the conjugate transpose, respectively; the matrix $\mathbf{S}(e^{j\omega})$ is introduced in (4.6), and in the last step of (D.6) we apply the property $S_{f_l,f_k}(e^{j\omega}) = S_{f_k,f_l}^*(e^{j\omega})$ and thus $\mathbf{S}^{\mathrm{T}}(e^{j\omega}) = \mathbf{S}^*(e^{j\omega})$.

In the first step of the parameter estimation procedure in Section 4.1.3, the objective function is (4.8), which can be simplified as

$$\begin{aligned} &\mathbb{E}\left\{|y[n] - \tilde{y}[n]|^2\right\} \\ =\ &\mathbb{E}\left\{y[n]y^*[n] - \tilde{y}[n]y^*[n] - y[n]\tilde{y}^*[n] + \tilde{y}[n]\tilde{y}^*[n]\right\} \\ =\ &R_{y,y}[0] - R_{\tilde{y},y}[0] - R_{y,\tilde{y}}[0] + R_{\tilde{y},\tilde{y}}[0] \end{aligned}$$

$$\begin{aligned} &= R_{y,y}[0] + \frac{1}{2\pi}\int_{-\pi}^{\pi} \left(-S_{\tilde{y},y}(e^{j\omega}) - S_{y,\tilde{y}}(e^{j\omega}) + S_{\tilde{y},\tilde{y}}(e^{j\omega})\right) \mathrm{d}\omega \\ &= R_{y,y}[0] + \frac{1}{2\pi}\int_{-\pi}^{\pi} \big(- \mathbf{d}^\dagger(e^{j\omega}) \cdot \mathbf{h}(e^{j\omega}) - \mathbf{h}^\dagger(e^{j\omega}) \cdot \mathbf{d}(e^{j\omega}) + \mathbf{h}^\dagger(e^{j\omega}) \cdot \mathbf{S}^*(e^{j\omega}) \cdot \mathbf{h}(e^{j\omega})\big)\mathrm{d}\omega, \end{aligned} \tag{D.7}$$

where the step (D.7) utilizes (D.5) and (D.6), as well as the fact that $S_{\tilde{y},y}(e^{j\omega}) = S^*_{y,\tilde{y}}(e^{j\omega})$.

Since each frequency in the objective function (D.7) is independent of all the other frequencies, the objective function (D.7) can be minimized over $\mathbf{h}(e^{j\omega})$ pointwisely for each frequency, i.e.

$$\mathbf{h}^{\text{opt}}(e^{j\omega}) = \underset{\mathbf{h}(e^{j\omega})}{\arg\min} \ \left(-\mathbf{d}^\dagger(e^{j\omega}) \cdot \mathbf{h}(e^{j\omega}) - \mathbf{h}^\dagger(e^{j\omega}) \cdot \mathbf{d}(e^{j\omega}) + \mathbf{h}^\dagger(e^{j\omega}) \cdot \mathbf{S}^*(e^{j\omega}) \cdot \mathbf{h}(e^{j\omega})\right). \tag{D.8}$$

We first show that the matrix $\mathbf{S}^*(e^{j\omega})$ is positive semidefinite. By definition, we aim to show that for any $\mathbf{q} \in \mathbb{C}^L$, it holds that $\mathbf{q}^\dagger \cdot \mathbf{S}^*(e^{j\omega}) \cdot \mathbf{q} \geq 0$. If we consider the stochastic signal $\tilde{f}[n]$ below,

$$\tilde{f}[n] \triangleq \sum_{l=1}^{L} q_l \cdot f_l(x[n]),$$

then the auto-correlation function of $\tilde{f}[n]$ is

$$R_{\tilde{f},\tilde{f}}[m] = \sum_{l=1}^{L}\sum_{k=1}^{L} q_l \cdot q_k^* \cdot \mathbb{E}\left\{f_l(x[n+m]) \cdot f_k^*(x[n])\right\} = \sum_{l=1}^{L}\sum_{k=1}^{L} q_l \cdot q_k^* \cdot R_{f_l,f_k}[m].$$

We then see that the power spectral density of $\tilde{f}[n]$ is

$$\begin{aligned} S_{\tilde{f},\tilde{f}}(e^{j\omega}) &= \sum_{l=1}^{L}\sum_{k=1}^{L} q_l \cdot q_k^* \cdot S_{f_l,f_k}(e^{j\omega}) \\ &= \sum_{l=1}^{L}\sum_{k=1}^{L} q_l \cdot q_k^* \cdot S^*_{f_k,f_l}(e^{j\omega}) \\ &= \mathbf{q}^\dagger \cdot \mathbf{S}^*(e^{j\omega}) \cdot \mathbf{q}. \end{aligned}$$

Since the power spectral density of the stochastic signal $\tilde{f}[n]$ is always non-negative, the result above shows that

$$\mathbf{q}^\dagger \cdot \mathbf{S}^*(e^{j\omega}) \cdot \mathbf{q} \geq 0$$

and thus $\mathbf{S}^*(e^{j\omega})$ is positive semidefinite.

Moreover, with the assumption that $\mathbf{S}(e^{j\omega})$ always has full rank, the matrix $\mathbf{S}^*(e^{j\omega})$ at each frequency is actually positive definite. As a result, we can factorize the matrix $\mathbf{S}^*(e^{j\omega})$ as

$$\mathbf{S}^*(e^{j\omega}) = \mathbf{Q}^\dagger(e^{j\omega}) \cdot \mathbf{Q}(e^{j\omega}),$$

where $\mathbf{Q}(e^{j\omega}) \in \mathbb{C}^{L\times L}$ is an invertible matrix. We can then simplify (D.8) as

$$\mathbf{h}^{\text{opt}}(e^{j\omega}) = \underset{\mathbf{h}(e^{j\omega})}{\arg\min} \ \left\| \mathbf{Q}(e^{j\omega}) \cdot \mathbf{h}(e^{j\omega}) - \left(\mathbf{Q}^\dagger(e^{j\omega})\right)^{-1} \cdot \mathbf{d}(e^{j\omega}) \right\|_2^2 - \mathbf{d}^\dagger(e^{j\omega}) \cdot \left(\mathbf{S}^*(e^{j\omega})\right)^{-1} \cdot \mathbf{d}(e^{j\omega}),$$

where $\mathbf{d}(e^{j\omega})$ is introduced in (D.4). We can observe that the optimal solution to the formulation above is

$$\mathbf{h}^{\text{opt}}(e^{j\omega}) = \left(\mathbf{Q}(e^{j\omega})\right)^{-1} \cdot \left(\mathbf{Q}^\dagger(e^{j\omega})\right)^{-1} \cdot \mathbf{d}(e^{j\omega}) = \left(\mathbf{S}^*(e^{j\omega})\right)^{-1} \cdot \mathbf{d}(e^{j\omega}),$$

which completes the derivation of the optimal solution (4.9).

Finally, we can obtain the minimal value of the objective function in (4.8). Applying (4.9) in (D.7), it can be seen that

$$\min_{\mathbf{h}(e^{j\omega})} \ \mathbb{E}\left\{ |y[n] - \tilde{y}[n]|^2 \right\} = R_{y,y}[0] - \frac{1}{2\pi} \int_{-\pi}^{\pi} \left(\mathbf{d}^\dagger(e^{j\omega}) \cdot \left(\mathbf{S}^*(e^{j\omega})\right)^{-1} \cdot \mathbf{d}(e^{j\omega}) \right) \mathrm{d}\omega,$$

which is the minimal mean squared error in the first step of the parameter estimation algorithm in Section 4.1.3.

Appendix E

Proofs for Theorems 6.1 and 6.2

This appendix shows the proofs for Theorems 6.1 and 6.2 in Section 6.2.2, which propose sufficient conditions under which the relaxation with $0 \le w_{j,r} \le 1$ or $w_{j,r} \ge 0$ for the variant Hamming distance formulation (6.14) is guaranteed to have binary optimal solutions.

First, we show the proof for Theorem 6.1.

Proof for Theorem 6.1: The key part of this proof aims to show that for any *feasible* solution $\widetilde{\mathbf{W}}$ to the relaxed formulation for (6.14) with $w_{j,r} \ge 0$, each clause in $\mathbf{W}^*$ corresponds to a clause in $\widetilde{\mathbf{W}}$; more specifically, for the r^{th} clause $\mathbf{w}_r^*$ in the true CNF function, there is a clause $\widetilde{\mathbf{w}}_{r'}$ in any feasible solution such that (i) $\widetilde{w}_{j,r'} = 0$ if $w_{j,r}^* = 0$, and (ii) $\widetilde{w}_{j,r'} \ge 1$ if $w_{j,r}^* = 1$. Since there are no redundant clauses in the true CNF function, the above mapping between clauses in the true CNF and in any feasible solution is bijective.

Without loss of generality, this proof focuses on the r^{th} clause $\mathbf{w}_r^*$. We define $\mathcal{P}_r^* = \{j : w_{j,r}^* = 1\}$ and $\mathcal{Z}_r^* = \{j : w_{j,r}^* = 0\}$ as the index sets of positive and zero items in this clause.

First we show that in any feasible solution $\widetilde{\mathbf{W}}$ to the formulation (6.14) with relaxation $w_{j,r} \ge 0$, there is a clause satisfying $\widetilde{w}_{j,r'} = 0$ for all $j \in \mathcal{Z}_r^*$. We know from assumption (c) that there exists a training example i such that $w_{j,r}^* = \overline{a_{i,j}}$ for all $1 \le j \le d$. It is clear that the r^{th} clause of this training example has value 0, and therefore its true output label is $y_i = 0$. From the constraint (6.16), any feasible solution $\widetilde{\mathbf{W}}$ to the relaxed formulation has a clause $\widetilde{\mathbf{w}}_{r'}$ that satisfies

$$\sum_{j=1}^{d} a_{i,j}\widetilde{w}_{j,r'} = 0.$$

Since $a_{i,j} = \overline{w^*_{j,r}}$ and $\widetilde{w}_{j,r} \geq 0$, it is shown that $\widetilde{w}_{j,r'} = 0$ for all $j \in \mathcal{Z}^*_r$.

The next step is to show that $\widetilde{w}_{j,r'} \geq 1$ for all $j \in \mathcal{P}^*_r$. For each $j_0 \in \mathcal{P}^*_r$, we introduce an auxiliary clause $\widehat{\mathbf{w}}$ with elements

$$\widehat{w}_{j'} = \begin{cases} w^*_{j',r} & \text{for } j' \neq j_0, \\ 0 & \text{for } j' = j_0. \end{cases}$$

We aim to show that there exists a training example $a_{l,j}$ with $y_l = 1$ in the dataset, on which the clause $\widehat{\mathbf{w}}$ has output 0. Combining the M-disjunct property from assumption (a) and the fact $|\mathcal{P}^*_r| \leq M$, we know that there exists a training example with $a_{i',j_0} = 1$ and $a_{i',j'} = 0$ for all $j' \in \mathcal{P}^*_r \backslash \{j_0\}$, on which $\widehat{\mathbf{w}}$ has output 0 while $\mathbf{w}^*_r$ has output 1; in the corresponding columns in $\mathbf{E}_M(\mathbf{A})$, the 0's in the column of $\mathbf{w}^*_r$ do not cover the 0's for $\widehat{\mathbf{w}}$. Since we assume that the true CNF has no redundant clauses, the 0's for clause $\mathbf{w}^*_t$ with $t \neq r$ do not cover the 0's for $\mathbf{w}^*_r$ and thus do not cover the 0's for $\widehat{\mathbf{w}}$. Consequently, the 0's for $\widehat{\mathbf{w}}$ are not covered by any individual clause in $\mathbf{W}^*$; with the generalized R-conjunct property of the augmented feature matrix $\mathbf{E}_M(\mathbf{A})$, the 0's for $\widehat{\mathbf{w}}$ are not covered by the union of all the 0's for $\mathbf{w}^*_t$ $(1 \leq t \leq R)$, where the latter is the 0's in the true labels. Consequently, we see that there exists a training example $a_{l,j}$ with $y_l = 1$, on which the clause $\widehat{\mathbf{w}}$ has output 0.

Since $\widehat{\mathbf{w}}$ has output 0 for the l^{th} training example, we know $a_{l,j'} = 0$ for all $j' \in \mathcal{P}^*_r \backslash \{j_0\}$; combined with the fact that $\widetilde{w}_{j',r'} = 0$ for $j' \in \mathcal{Z}^*_r$, the constraint (6.15) indicates

$$\sum_{j'=1}^{d} a_{l,j'}\widetilde{w}_{j',r'} = \sum_{j' \in \mathcal{P}^*_r} a_{l,j'}\widetilde{w}_{j',r'} = a_{l,j_0}\widetilde{w}_{j_0,r'} \geq 1,$$

which leads to $\widetilde{w}_{j_0,r'} \geq 1$. Since j_0 is an arbitrary item in $\mathcal{P}^*_r$, it follows that $\widetilde{w}_{j,r'} \geq 1$ for all $j \in \mathcal{P}^*_r$.

Up to now, we have shown that any feasible solution $\widetilde{\mathbf{W}}$ to the formulation (6.14) with relaxation $w_{j,r} \geq 0$ is guaranteed to have the following properties: for the r^{th} clause $\mathbf{w}_r^*$, there exists a corresponding clause $\widetilde{\mathbf{w}}_{r'}$ such that $\widetilde{w}_{j,r'} = 0$ when $w_{j,r}^* = 0$, and $\widetilde{w}_{j,r'} \geq 1$ when $w_{j,r}^* = 1$. The non-redundancy property of the true CNF function shows that two clauses in $\mathbf{W}^*$ cannot correspond to the same clause in $\widetilde{\mathbf{W}}$, which leads to a bijective mapping between the clauses in $\mathbf{W}^*$ and $\widetilde{\mathbf{W}}$.

The last step to establish the two claims in Theorem 6.1 is as follows:

(i) If we use the relaxation $0 \leq w_{j,r} \leq 1$, then any feasible solution has corresponding clauses satisfying $\widetilde{w}_{j,r'} \geq 1$ and $\widetilde{w}_{j,r'} \leq 1$ for j such that $w_{j,r}^* = 1$. We then have $\widetilde{w}_{j,r'} = w_{j,r}^*$ for all $1 \leq j \leq d$, i.e. the corresponding clauses are identical. The bijective clause mapping argument shows that any feasible solution $\widetilde{\mathbf{W}}$ is identical to the true CNF function $\mathbf{W}^*$ up to a permutation of the clauses.

(ii) If we use the relaxation $w_{j,r} \geq 0$, then any feasible solution $\widetilde{\mathbf{W}}$ satisfies

$$\sum_{j=1}^{d} \widetilde{w}_{j,r'} = \sum_{j \in \mathcal{P}_r^*} \widetilde{w}_{j,r'} \geq |\mathcal{P}_r^*| \,.$$

Therefore, taking the summation over $1 \leq r' \leq R$, we know the objective in (6.14) satisfies

$$\sum_{r'=1}^{R} \sum_{j=1}^{d} \widetilde{w}_{j,r'} \geq \sum_{r=1}^{R} |\mathcal{P}_r^*| = \sum_{r=1}^{R} \sum_{j=1}^{d} w_{j,r}^*.$$

As a result, the minimal objective value is $\sum_{r=1}^{R} |\mathcal{P}_r^*|$; the optimal solution has to satisfy $\widetilde{w}_{j,r'} = 1$ for $j \in \mathcal{P}_r^*$. Combined with the results above, we have finally shown that any optimal solution to (6.14) with relaxation $w_{j,r} \geq 0$ is identical to the true CNF function up to a permutation of the clauses.

This has completed the proof for Theorem 6.1. ■

Next, we show the proof for Theorem 6.2.

Proof for Theorem 6.2: First we show the sufficiency. For any $(R+1)$ columns in the augmented matrix $\mathbf{E}_M(\mathbf{A})$, their corresponding disjunctive clauses are denoted as $\widehat{\mathbf{w}}_0, \ldots, \widehat{\mathbf{w}}_R$, and we consider the coverage of the 0's for $\widehat{\mathbf{w}}_0$ by the union of the 0's for the other R clauses. If the 0's for $\widehat{\mathbf{w}}_0$ are not covered by the 0's for any of $\widehat{\mathbf{w}}_r$ $(1 \leq r \leq R)$, then the selected features in $\widehat{\mathbf{w}}_r$ are not covered by $\widehat{\mathbf{w}}_0$. We denote x_{j_r} as a feature that is selected in $\widehat{\mathbf{w}}_r$ but not in $\widehat{\mathbf{w}}_0$, and construct the feature subsets $\mathcal{P} = \{j_r : \ 1 \leq r \leq R\}$. In addition, we define $\mathcal{Z} = \{j : \ x_j \text{ is selected in } \widehat{\mathbf{w}}_0\}$. These two subsets are disjoint by construction and satisfy $1 \leq |\mathcal{P}| \leq \min\{R, d\}$ and $1 \leq |\mathcal{Z}| \leq M$, so there exists a training example $a_{i,j}$ that satisfies $a_{i,j} = 1$ for all $j \in \mathcal{P}$ and $a_{i,j} = 0$ for all $j \in \mathcal{Z}$. For this training example, $\widehat{\mathbf{w}}_r$ has output 1 for all $1 \leq r \leq R$, while $\widehat{\mathbf{w}}_0$ has output 0; consequently, the 0's for $\widehat{\mathbf{w}}_0$ are not covered by the union of the 0's for $\widehat{\mathbf{w}}_r$ $(1 \leq r \leq R)$, implying the generalized R-conjunct property of $\mathbf{E}_M(\mathbf{A})$.

Then we show the necessity of this condition when $\mathbf{A}$ is M-disjunct. We denote $\mathcal{P} = \{j_1, j_2, \ldots, j_{|\mathcal{P}|}\}$ and define $\widehat{\mathbf{w}}_r$ as the clause that selects a single feature x_{j_r} $(1 \leq r \leq |\mathcal{P}|)$. We define another clause $\widehat{\mathbf{w}}_0$ that takes the disjunction of all features in the subset $\mathcal{Z}$. For each fixed $1 \leq r \leq |\mathcal{P}|$, since $j_r \notin \mathcal{Z}$, the M-disjunct property of the matrix $\mathbf{A}$ implies that there exists a training example on which $\widehat{\mathbf{w}}_0$ has output 0 while the j_r^{th} feature is 1, and therefore the 0's for $\widehat{\mathbf{w}}_0$ are not covered by those for $\widehat{\mathbf{w}}_r$. Since $|\mathcal{P}| \leq R$, the generalized R-conjunct property of $\mathbf{E}_M(\mathbf{A})$ implies that there exists a training example $a_{i,j}$ on which $\widehat{\mathbf{w}}_0$ has output 0 and all $\widehat{\mathbf{w}}_r$ $(1 \leq r \leq |\mathcal{P}|)$ have output 1. By definition, this training example has $a_{i,j} = 1$ for all $j \in \mathcal{P}$ and $a_{i,j} = 0$ for all $j \in \mathcal{Z}$. ■

Bibliography

[1] S. Demirtas and A. V. Oppenheim, "Modular filters based on filter sharpening and optimal minimax design," in *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on.* IEEE, 2013, pp. 197–202.

[2] S. Demirtas, "Functional composition and decomposition for signal processing," Ph.D. dissertation, Massachusetts Institute of Technology, 2014.

[3] S. Treitel and J. L. Shanks, "The design of multistage separable planar filters," *IEEE Transactions on Geoscience Electronics*, vol. 9, no. 1, pp. 10–27, 1971.

[4] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[5] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[6] N. Srebro, "Learning with matrix factorizations," Ph.D. dissertation, 2004.

[7] J. D. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22nd international conference on Machine learning.* ACM, 2005, pp. 713–719.

[8] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Advances in neural information processing systems*, 2009, pp. 2080–2088.

[9] T. Bouwmans, "Recent advanced statistical background modeling for foreground detection-a systematic survey," *Recent Patents on Computer Science*, vol. 4, no. 3, pp. 147–176, 2011.

[10] M. Hubert and S. Engelen, "Robust PCA and classification in biosciences," *Bioinformatics*, vol. 20, no. 11, pp. 1728–1736, 2004.

[11] J. F. Ritt, "Prime and composite polynomials," *Transactions of the American Mathematical Society*, vol. 23, no. 1, pp. 51–66, 1922.

[12] D. Kozen and S. Landau, "Polynomial decomposition algorithms," *Journal of Symbolic Computation*, vol. 7, no. 5, pp. 445–456, 1989.

[13] R. M. Corless, M. W. Giesbrecht, D. J. Jeffrey, and S. M. Watt, "Approximate polynomial decomposition," in *Proceedings of the 1999 international symposium on Symbolic and algebraic computation.* ACM, 1999, pp. 213–219.

[14] M. Giesbrecht and J. May, "New algorithms for exact and approximate polynomial decomposition," *Symbolic-Numeric Computation*, pp. 99–112, 2007.

[15] R. Zippel, "Rational function decomposition," in *Proceedings of the 1991 international symposium on Symbolic and algebraic computation.* ACM, 1991, pp. 1–6.

[16] C. Alonso, J. Gutierrez, and T. Recio, "A rational function decomposition algorithm by near-separated polynomials," *Journal of Symbolic Computation*, vol. 19, no. 6, pp. 527–544, 1995.

[17] T. Kimura, "On the iteration of analytic functions," *Funkcial. Ekvac*, vol. 14, pp. 197–238, 1971.

[18] M. Kuczma, B. Choczewski, and R. Ger, *Iterative functional equations.* Cambridge University Press, 1990.

[19] B. O. Koopman, "Hamiltonian systems and transformation in Hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[20] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," *Journal of Fluid Mechanics*, vol. 641, pp. 115–127, 2009.

[21] Y. Lan and I. Mezić, "Linearization in the large of nonlinear systems and Koopman operator spectrum," *Physica D: Nonlinear Phenomena*, vol. 242, no. 1, pp. 42–53, 2013.

[22] A. V. Oppenheim, D. Johnson, and K. Steiglitz, "Computation of spectra with unequal resolution using the fast Fourier transform," *Proceedings of the IEEE*, vol. 59, no. 2, pp. 299–301, 1971.

[23] C. Asavathiratham, P. E. Beckmann, and A. V. Oppenheim, "Frequency warping in the design and implementation of fixed-point audio equalizers," in *Applications of Signal Processing to Audio and Acoustics, 1999 IEEE Workshop on.* IEEE, 1999, pp. 55–58.

[24] D. Wei and A. V. Oppenheim, "Sampling based on local bandwidth," in *Signals, Systems and Computers (ASILOMAR), 2007 Conference Record of the Forty First Asilomar Conference on*, 2007, pp. 1103–1107.

[25] D. Wei, "Sampling based on local bandwidth," Master's thesis, Massachusetts Institute of Technology, 2007.

[26] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010.

[27] F. Giri and E.-W. Bai, *Block-oriented nonlinear system identification.* Springer, 2010, vol. 1.

[28] C. Bergman, *Universal algebra: Fundamentals and selected topics.* CRC Press, 2011.

[29] D. J. Litman, M. A. Walker, and M. S. Kearns, "Automatic detection of poor speech recognition at the dialogue level," in *Proceedings of Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, Jun. 1999, pp. 309–316.

[30] R. Mitchell and R. Chen, "Behavior-rule based intrusion detection systems for safety critical smart grid applications," *Smart Grid, IEEE Transactions on*, vol. 4, no. 3, pp. 1254–1263, Sep. 2013.

[31] A. Abraham, "Rule-based expert systems," *Handbook of Measuring System Design*, 2005.

[32] J. Feldman, "Minimization of Boolean complexity in human concept learning," *Nature*, vol. 407, no. 6804, pp. 630–633, Oct. 2000.

[33] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan, "Building interpretable classifiers with rules using Bayesian analysis," *Department of Statistics Technical Report tr609, University of Washington*, Dec. 2012.

[34] J. Fürnkranz, D. Gamberger, and N. Lavrač, *Foundations of rule learning.* Springer Science & Business Media, Nov. 2012.

[35] P. C. McGeer, J. V. Sanghavi, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "ESPRESSO-SIGNATURE: A new exact minimizer for logic functions," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 1, no. 4, pp. 432–440, Dec. 1993.

[36] A. Vellido, J. D. Martín-Guerrero, and P. J. Lisboa, "Making machine learning models interpretable," in *Proceeding of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Apr. 2012, pp. 163–172.

[37] A. A. Freitas, "Comprehensible classification models – a position paper," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 1, pp. 1–10, Mar. 2014.

[38] V. S. Iyengar, K. B. Hermiz, and R. Natarajan, "Computer-aided auditing of prescription drug claims," *Health Care Management Science*, vol. 17, no. 3, pp. 203–214, Sep. 2014.

[39] V. Volterra, "Theory of functionals and of integrals and integro-differential equations," Madrid 1927 (Spanish), translated version reprinted New York: Dover Publications, 1959.

[40] N. Wiener, "Response of a non-linear device to noise," Tech. Rep., 1942, Declassified Jul 1946, Published as rep. no. PB-1-58087.

[41] J. S. Lim, *Two-dimensional signal and image processing.* Englewood Cliffs, NJ, USA: Prentice Hall, 1990.

[42] S. Demirtas, G. Su, and A. V. Oppenheim, "Exact and approximate polynomial decomposition methods for signal processing applications," in *Proceeding of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 5373–5377.

[43] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[44] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," 2016, book in preparation for MIT Press. [Online]. Available: http://www.deeplearningbook.org

[45] J. Hastad, "Almost optimal lower bounds for small depth circuits," in *Proceedings of the eighteenth annual ACM symposium on Theory of computing.* ACM, 1986, pp. 6–20.

[46] I. Wegener and B. Teubner, *The complexity of Boolean functions.* BG Teubner Stuttgart, 1987, vol. 1.

[47] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[48] V. G. Mertzios and A. N. Venetsanopoulos, "Block decomposition structures for the fast modular implementation of two-dimensional digital filters," *Circuits, Systems and Signal Processing*, vol. 8, no. 2, pp. 163–185, 1989.

[49] M. M. Vai, *VLSI design.* CRC Press, 2000.

[50] A. Hammerstein, "Nichtlineare integralgleichungen nebst anwendungen," *Acta Mathematica*, vol. 54, no. 1, pp. 117–176, 1930.

[51] S. Rangan, G. Wolodkin, and K. Poolla, "Identification methods for Hammerstein systems," in *Proceedings of Control and Decision Conference*, 1995, pp. 697–702.

[52] E.-W. Bai, "An optimal two-stage identification algorithm for Hammerstein–Wiener nonlinear systems," *Automatica*, vol. 34, no. 3, pp. 333–338, 1998.

[53] S. Givant and P. Halmos, *Introduction to Boolean algebras.* Springer Science & Business Media, 2008.

[54] G. Su, "Polynomial decomposition algorithms in signal processing," Master's thesis, Massachusetts Institute of Technology, 2013.

[55] P. Aubry and A. Valibouze, "Algebraic computation of resolvents without extraneous powers," *European Journal of Combinatorics*, vol. 33, no. 7, pp. 1369 – 1385, 2012.

[56] S. Demirtas, G. Su, and A. V. Oppenheim, "Sensitivity of polynomial composition and decomposition for signal processing applications," in *Signals, Systems and Computers (ASILOMAR), Conference Record of the Forty Sixth Asilomar Conference on*, Nov. 2012, pp. 391–395.

[57] J. Kaiser and R. Hamming, "Sharpening the response of a symmetric nonrecursive filter by multiple use of the same filter," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 5, pp. 415–422, 1977.

[58] E. A. Guillemin, *Synthesis of Passive Networks.* Wiley, New York, NY, 1957.

[59] R. W. Daniels, *Approximation Methods for Electronic Filter Design.* McGraw-Hill, New York, NY, 1974.

[60] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[61] E. Schröder, "Ueber iterirte functionen," *Mathematische Annalen*, vol. 3, no. 2, pp. 296–322, 1870.

[62] M. Budišić, R. Mohr, and I. Mezić, "Applied Koopmanism," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 4, p. 047510, 2012.

[63] A. Mauroy, I. Mezić, and J. Moehlis, "Isostables, isochrons, and Koopman spectrum for the action–angle representation of stable fixed point dynamics," *Physica D: Nonlinear Phenomena*, vol. 261, pp. 19–30, 2013.

[64] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

[65] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *Journal of Computational Dynamics*, vol. 1, no. 2, pp. 391–421, 2014.

[66] N. Santitissadeekorn and E. M. Bollt, "The infinitesimal operator for the semigroup of the frobenius-perron operator from image sequence data: vector fields and transport barriers from movies," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 17, no. 2, p. 023126, 2007.

[67] N. Wiener, *Nonlinear problems in random theory.* Cambridge, Massachusetts: Technology Press of Massachusetts Institute of Technology, New York: John Wiley, 1958.

[68] S. Ikehara, "A method of Wiener in a nonlinear circuit," *Tech. Report No. 217, Research Laboratory of Electronics, Massachusetts Institute of Technology*, 1951.

[69] M. Schetzen, "The Volterra and Wiener theories of nonlinear systems," 1980.

[70] S. Boyd and L. Chua, "Fading memory and the problem of approximating nonlinear operators with Volterra series," *IEEE Transactions on circuits and systems*, vol. 32, no. 11, pp. 1150–1161, 1985.

[71] S. Billings, "Identification of nonlinear systems – a survey," in *IEE Proceedings D-Control Theory and Applications*, vol. 127, no. 6. IET, 1980, pp. 272–285.

[72] J. M. Mendel, "Tutorial on higher-order statistics (spectra) in signal processing and system theory: theoretical results and some applications," *Proceedings of the IEEE*, vol. 79, no. 3, pp. 278–305, 1991.

[73] D. Gabor, W. Wilby, and R. Woodcock, "A universal non-linear filter, predictor and simulator which optimizes itself by a learning process," *Proceedings of the IEE-Part B: Electronic and Communication Engineering*, vol. 108, no. 40, pp. 422–435, 1961.

[74] R. H. Flake, "Volterra series representation of time-varying nonlinear systems," in *Joint Automatic Control Conference*, no. 1, 1963, p. 375.

[75] R. Roy and J. Sherman, "A learning technique for Volterra series representation," *IEEE Transactions on Automatic Control*, vol. 12, no. 6, pp. 761–764, 1967.

[76] D. Brillinger, "The identification of polynomial systems by means of higher order spectra," *Journal of Sound and Vibration*, vol. 12, no. 3, pp. 301–313, 1970.

[77] H. Barker and R. Davy, "2nd-order Volterra kernel measurement using pseudorandom ternary signals and discrete Fourier transforms," in *Proceedings of the Institution of Electrical Engineers*, vol. 126, no. 5. IET, 1979, pp. 457–460.

[78] A. French and E. Butz, "Measuring the Wiener kernels of a non-linear system using the fast Fourier transform algorithm," *International Journal of Control*, vol. 17, no. 3, pp. 529–539, 1973.

[79] Y. W. Lee and M. Schetzen, "Measurement of the Wiener kernels of a non-linear system by cross-correlation," *International Journal of Control*, vol. 2, no. 3, pp. 237–254, 1965.

[80] G. Palm and T. Poggio, “The Volterra representation and the Wiener expansion: validity and pitfalls,” *SIAM Journal on Applied Mathematics*, vol. 33, no. 2, pp. 195–216, 1977.

[81] G. Palm and T. Poggio, “Stochastic identification methods for nonlinear systems: an extension of the Wiener theory,” *SIAM Journal on Applied Mathematics*, vol. 34, no. 3, pp. 524–535, 1978.

[82] S. Orcioni, M. Pirani, and C. Turchetti, “Advances in Lee–Schetzen method for Volterra filter identification,” *Multidimensional Systems and Signal Processing*, vol. 16, no. 3, pp. 265–284, 2005.

[83] J. Kim and K. Konstantinou, “Digital predistortion of wideband signals based on power amplifier model with memory,” *Electronics Letters*, vol. 37, no. 23, p. 1, 2001.

[84] L. Ding, R. Raich, and G. T. Zhou, “A Hammerstein predistortion linearization design based on the indirect learning architecture,” in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 3. IEEE, 2002, pp. III–2689.

[85] E. Eskinat, S. H. Johnson, and W. L. Luyben, “Use of Hammerstein models in identification of nonlinear systems,” *AIChE Journal*, vol. 37, no. 2, pp. 255–268, 1991.

[86] Y. Zhu, “Distillation column identification for control using Wiener model,” in *American Control Conference, 1999. Proceedings of the 1999*, vol. 5. IEEE, 1999, pp. 3462–3466.

[87] I. W. Hunter and M. J. Korenberg, “The identification of nonlinear biological systems: Wiener and Hammerstein cascade models,” *Biological cybernetics*, vol. 55, no. 2-3, pp. 135–144, 1986.

[88] M. Schoukens, R. Pintelon, and Y. Rolain, “Parametric identification of parallel Hammerstein systems,” *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 12, pp. 3931–3938, 2011.

[89] M. Schoukens and Y. Rolain, “Parametric identification of parallel Wiener systems,” *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 10, pp. 2825–2832, 2012.

[90] S. Billings and S. Fakhouri, “Theory of separable processes with applications to the identification of nonlinear systems,” *Electrical Engineers, Proceedings of the Institution of*, vol. 125, no. 10, pp. 1051–1058, 1978.

[91] M. Enqvist and L. Ljung, “Linear approximations of nonlinear FIR systems for separable input processes,” *Automatica*, vol. 41, no. 3, pp. 459–473, 2005.

[92] E.-W. Bai, "Frequency domain identification of Hammerstein models," *IEEE transactions on automatic control*, vol. 48, no. 4, pp. 530–542, 2003.

[93] A. Hagenblad, L. Ljung, and A. Wills, "Maximum likelihood identification of Wiener models," *Automatica*, vol. 44, no. 11, pp. 2697–2705, 2008.

[94] I. Goethals, K. Pelckmans, J. A. Suykens, and B. De Moor, "Identification of MIMO Hammerstein models using least squares support vector machines," *Automatica*, vol. 41, no. 7, pp. 1263–1272, 2005.

[95] I. Goethals, K. Pelckmans, J. A. Suykens, and B. De Moor, "Subspace identification of Hammerstein systems using least squares support vector machines," *IEEE Transactions on Automatic Control*, vol. 50, no. 10, pp. 1509–1519, 2005.

[96] E.-W. Bai, "A blind approach to the Hammerstein–Wiener model identification," *Automatica*, vol. 38, no. 6, pp. 967–979, 2002.

[97] L. Vanbeylen, R. Pintelon, and J. Schoukens, "Blind maximum likelihood identification of Hammerstein systems," *Automatica*, vol. 44, no. 12, pp. 3139–3146, 2008.

[98] M. Felsberg, "On the design of two-dimensional polar separable filters," in *Signal Processing Conference, 2004 12th European*. IEEE, 2004, pp. 417–420.

[99] J. McClellan and T. Parks, "Equiripple approximation of fan filters," *Geophysics*, vol. 37, no. 4, pp. 573–583, 1972.

[100] T. Sekiguchi and Y. Karasawa, "Wideband beamspace adaptive array utilizing FIR fan filters for multibeam forming," *IEEE Transactions on Signal Processing*, vol. 48, no. 1, pp. 277–284, 2000.

[101] T. Sasaki, "Approximate multivariate polynomial factorization based on zero-sum relations," in *Proceedings of International Symposium on Symbolic and Algebraic Computation*, Jul. 2001, pp. 284–291.

[102] Z. Mou-Yan and R. Unbehauen, "Approximate factorization of multivariable polynomials," *Signal processing*, vol. 14, no. 2, pp. 141–152, Mar. 1988.

[103] T. Sasaki, M. Suzuki, M. Kolár, and M. Sasaki, "Approximate factorization of multivariate polynomials and absolute irreducibility testing," *Japan journal of industrial and applied mathematics*, vol. 8, no. 3, pp. 357–375, Oct. 1991.

[104] W. M. Ruppert, "Reducibility of polynomials $f(x, y)$ modulo p," *Journal of Number Theory*, vol. 77, no. 1, pp. 62–70, Aug. 1999.

[105] R. M. Corless, M. W. Giesbrecht, M. Van Hoeij, I. S. Kotsireas, and S. M. Watt, "Towards factoring bivariate approximate polynomials," in *Proceedings of International Symposium on Symbolic and Algebraic Computation*. ACM, Jul. 2001, pp. 85–92.

[106] S. Gao, E. Kaltofen, J. May, Z. Yang, and L. Zhi, "Approximate factorization of multivariate polynomials via differential equations," in *Proceedings of International Symposium on Symbolic and Algebraic Computation.* ACM, Jul. 2004, pp. 167–174.

[107] E. Kaltofen, J. P. May, Z. Yang, and L. Zhi, "Approximate factorization of multivariate polynomials using singular value decomposition," *Journal of Symbolic Computation*, vol. 43, no. 5, pp. 359–376, May 2008.

[108] J. B. Rosen, H. Park, and J. Glick, "Total least norm formulation and solution for structured problems," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 1, pp. 110–126, Jan. 1996.

[109] P. Lemmerling, "Structured total least squares: analysis, algorithms and applications," Ph.D. dissertation, K. U. Leuven (Leuven, Belgium), 1999.

[110] B. De Moor, "Total least squares for affinely structured matrices and the noisy realization problem," *Signal Processing, IEEE Transactions on*, vol. 42, no. 11, pp. 3104–3113, Nov. 1994.

[111] V. A. Zorich, *Mathematical Analysis I.* New York, NY, USA: Springer Science & Business Media, 2004.

[112] I. M. Vinogradov and M. Hazewinkel, *Encyclopaedia of mathematics.* Kluwer Academic Publ, 2001.

[113] D. Mead, "Newton's identities," *The American mathematical monthly*, vol. 99, no. 8, pp. 749–751, Oct. 1992.

[114] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review*, vol. 52, no. 3, pp. 471–501, 2010.

[115] P. Jain, R. Meka, and I. S. Dhillon, "Guaranteed rank minimization via singular value projection," in *Advances in Neural Information Processing Systems*, 2010, pp. 937–945.

[116] K. Lee and Y. Bresler, "ADMiRA: Atomic decomposition for minimum rank approximation," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4402–4416, 2010.

[117] E. Moore, "On the reciprocal of the general algebraic matrix," *Bull. Amer. Math. Soc*, vol. 26, pp. 394–395, 1920.

[118] R. Penrose, "A generalized inverse for matrices," in *Mathematical proceedings of the Cambridge philosophical society*, vol. 51, no. 03. Cambridge Univ Press, 1955, pp. 406–413.

[119] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, Sep. 1936.

[120] "IBM ILOG CPLEX optimization studio," http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud.

[121] A. V. Oppenheim and G. C. Verghese, *Signals, systems and inference.* Pearson, 2015.

[122] D. T. Slock, "Spatio-temporal training-sequence based channel equalization and adaptive interference cancellation," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, vol. 5. IEEE, 1996, pp. 2714–2717.

[123] G. W. Stewart, *Afternotes goes to graduate school: lectures on advanced numerical analysis.* Siam, 1998.

[124] Z. Battles and L. N. Trefethen, "An extension of MATLAB to continuous functions and operators," *SIAM Journal on Scientific Computing*, vol. 25, no. 5, pp. 1743–1770, 2004.

[125] Z. Battles, "Numerical linear algebra for continuous functions," Ph.D. dissertation, University of Oxford, 2005.

[126] A. Townsend and L. N. Trefethen, "Continuous analogues of matrix factorizations," in *Proc. R. Soc. A*, vol. 471, no. 2173. The Royal Society, 2015, p. 20140585.

[127] G. Mzyk, *Combined parametric-nonparametric identification of block-oriented systems.* Springer, 2014.

[128] M. Schetzen, "Theory of p^{th}-order inverses of nonlinear systems," *IEEE Transactions on Circuits and Systems*, vol. 23, no. 5, pp. 285–291, 1976.

[129] C. Eun and E. J. Powers, "A new Volterra predistorter based on the indirect learning architecture," *IEEE Transactions on Signal Processing*, vol. 45, no. 1, pp. 223–227, 1997.

[130] G. U. Yule, "On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 226, pp. 267–298, 1927.

[131] G. Walker, "On periodicity in series of related terms," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 131, no. 818, pp. 518–532, 1931.

[132] D. P. Bertsekas, *Nonlinear programming.* Athena scientific Belmont, 1999.

[133] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[134] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds. Springer-Verlag Limited, 2008, pp. 95–110, http://stanford.edu/~boyd/graph_dcp.html.

[135] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[136] C. L. Nikias and J. M. Mendel, "Signal processing with higher-order spectra," *IEEE signal processing magazine*, vol. 10, no. 3, pp. 10–37, 1993.

[137] E. Cheney, "Introduction to approximation theory," 1966.

[138] G. Su, D. Wei, K. R. Varshney, and D. M. Malioutov, "Learning sparse two-level Boolean rules," in *Proceeding of IEEE International Workshop on Machine Learning for Signal Processing*, Sep. 2016.

[139] R. L. Rivest, "Learning decision lists," *Machine learning*, vol. 2, no. 3, pp. 229–246, Nov. 1987.

[140] J. R. Quinlan, "Simplifying decision trees," *International Journal of Man-machine Studies*, vol. 27, no. 3, pp. 221–234, Sep. 1987.

[141] M. W. Craven and J. W. Shavlik, "Extracting tree-structured representations of trained networks," *Advances in Neural Information Processing Systems*, pp. 24–30, 1996.

[142] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen, "Using neural network rule extraction and decision tables for credit-risk evaluation," *Management Science*, vol. 49, no. 3, pp. 312–329, Mar. 2003.

[143] L. G. Valiant, "Learning disjunction of conjunctions," in *Proceeding of International Joint Conference on Artificial Intelligence*, Aug. 1985, pp. 560–566.

[144] M. Kearns, M. Li, L. Pitt, and L. Valiant, "On the learnability of Boolean formulae," in *Proceedings of the 19th annual ACM symposium on Theory of computing*, Jan. 1987, pp. 285–295.

[145] D. M. Malioutov and K. R. Varshney, "Exact rule learning via Boolean compressed sensing," in *Proceedings of International Conference on Machine Learning*, Jun. 2013, pp. 765–773.

[146] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, no. 4, pp. 261–283, Mar. 1989.

[147] W. W. Cohen, "Fast effective rule induction," in *Proceedings of International Conference on Machine Learning*, Jul. 1995, pp. 115–123.

[148] W. W. Cohen and Y. Singer, "A simple, fast, and effective rule learner," in *Proc. Nat. Conf. Artif. Intell.*, 1999, pp. 335–342.

[149] J. Fürnkranz, "Separate-and-conquer rule learning," *Artificial Intelligence Review*, vol. 13, no. 1, pp. 3–54, Feb. 1999.

[150] M. Marchand and J. Shawe-Taylor, "The set covering machine," *Journal of Machine Learning Research*, vol. 3, pp. 723–746, 2002.

[151] S. Salzberg, "A nearest hyperrectangle learning method," *Machine Learning*, vol. 6, no. 3, pp. 251–276, May 1991.

[152] P. Domingos, "Unifying instance-based and rule-based induction," *Machine Learning*, vol. 24, no. 2, pp. 141–168, Aug. 1996.

[153] M. Muselli and D. Liberati, "Binary rule generation via Hamming Clustering," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 14, no. 6, pp. 1258–1268, Nov. 2002.

[154] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," in *Proceeding of International Conference on Knowledge Discovery and Data Mining*, Aug. 1998, pp. 80–86.

[155] B. Liu, Y. Ma, and C. K. Wong, "Improving an association rule based classifier," in *European Conference on Principles of Data Mining and Knowledge Discovery*, Sep. 2000, pp. 504–509.

[156] A. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz, "From local patterns to global models: The LeGo approach to data mining," in *Proceeding of ECML PKDD Workshop (LeGo-08)*, Sep. 2008, pp. 1–16.

[157] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," in *Proceedings of International Conference on Machine Learning*, Jul. 1998, pp. 144–151.

[158] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille, "Bayesian Or's of And's for interpretable classification with application to context aware recommender systems," MIT, Tech. Rep., 2015, submitted.

[159] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik, "An implementation of logical analysis of data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 12, no. 2, pp. 292–306, Mar. 2000.

[160] P. L. Hammer and T. O. Bonates, "Logical analysis of data – an overview: from combinatorial optimization to medical applications," *Annals of Operations Research*, vol. 148, no. 1, pp. 203–225, Nov. 2006.

[161] T. Wang and C. Rudin, “Learning optimized Or’s of And’s,” *arXiv preprint arXiv:1511.02210*, Nov. 2015.

[162] R. Hähnle, “Proof theory of many-valued logic–linear optimization–logic design: Connections and interactions,” *Soft Computing*, vol. 1, no. 3, pp. 107–119, Sep. 1997.

[163] M. Lichman, “UCI machine learning repository,” http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences, 2013.

[164] D. Fudenberg and J. Tirole, *Game Theory.* The MIT Press, Cambridge, MA, 1991.

[165] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix polynomials.* SIAM, 2009.

[166] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky, “The convex geometry of linear inverse problems,” *Foundations of Computational mathematics*, vol. 12, no. 6, pp. 805–849, 2012.

[167] D. Needell and J. A. Tropp, “CoSaMP: Iterative signal recovery from incomplete and inaccurate samples,” *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, 2009.

Epilogue

A Ph.D. journey typically has much richer and broader experience than the pure development of technical contents in the thesis. Even if constrained to the technical perspective, the path that leads to the final shape of the thesis can be rather "nonlinear", with fun stories and significant deviation from the order in which the contents are presented. Looking back the six years at MIT working with Al, this epilogue is concerned with some memories of my pleasant experiences and especially the development of this thesis.

After coming to MIT in 2011, I completed my master's program in the first two years. My master's thesis was on polynomial decomposition algorithms in signal processing, for which I was in a close collaboration with Dr. Sefa Demirtas, who was a Ph.D. candidate in the same group as me and then became one of my Ph.D. thesis committee members a few years later. We learned and proposed algorithms for univariate polynomial decomposition, among which a method noticed by Sefa was based on the rank-deficient Ruppert matrix approximation. Although this method seemed to have certain numerical challenges for polynomial decomposition, it inspired me to think about other topics later in my Ph.D. program. I performed two internships during my master's program, one with Dr. Davis Y. Pan at Bose Corporation and the other with Dr. Arthur J. Redfern at Texas Instruments. Both managers are great and very nice people, with whom I had a lot of awesome experiences not only for the projects but also for after-work activities such as sailing and playing golf.

In Fall 2013, my Ph.D. research started in the direction of nonlinear signal processing, with Al's question *"What are interesting nonlinear phenomena that can* *benefit signal processing?"* As a result, I was exposed to Volterra models, inversion of a nonlinear system, and various creative methods developed by Digital Signal Processing Group alumni (e.g. Dr. Kevin M. Cuomo, Dr. Steven H. Isabelle, Dr. Andrew C. Singer, and Dr. Wade P. Torres) that made use of chaotic systems or nonlinear differential equations for signal processing purposes, which I fully appreciated. Aside from research, I was one of the teaching assistants for Al's course *Discrete-time Signal Processing*. Al's lectures in that semester were videotaped for a MOOC (massive open online course) on edX, and it was a pleasant experience to participate in the development of new techniques and materials that were used in the MOOC.

In Spring 2014, I took the course *Information Acquisition and Processing* offered by Dr. Petros T. Boufounos. Inspired by the topic of compressive sensing that was an important part of the course, I investigated the problem of sparse signal recovery with nonlinear observations, where in particular each observed variable had a quadratic form with respect to the sparse signal. In fact, this problem had been explored in the compressive sensing community; related to the traditional compressive sensing problem, two possible methods to this problem were convex relaxation and greedy approaches. Although this problem was not directly related to my thesis, the low rank matrix approximation formulation for bivariate polynomial factorization in this thesis may be considered as high-levelly related to the compressive sensing problem. In the following summer, I had a pleasant internship at A9.com, a subsidy of Amazon.com, working with Dr. Simant Dube on fun computer vision problems.

Inspired by the fact that Fourier series can be a better approximation than Taylor series when the criterion is the global approximation error on an interval rather than locally at a single point, I considered alternative criteria for the approximation of nonlinear systems other than the commonly used ℓ_2-norm of error. As an example, I considered the minimax approximation of a higher-order Volterra system with a lower-order system. As Al suggested, we started with the most tractable situation where the lower-order Volterra system is a linear system. Similar to the alternation theorem that characterizes the minimax approximation of a one-dimensional function, I discovered a sufficient and necessary condition for a given linear system to be the optimal minimax approximation for a nonlinear Volterra system with finite memory states. However, the generalization of this result seemed challenging if a more sophisticated lower-order Volterra system was used in approximation.

As the exploration evolved, the direction of my Ph.D. project gradually moved towards a combination of nonlinear systems and composition structures. As a motivating problem, I was interested in modeling a nonlinear or time-variant system with the cascade of nonlinear blocks, time-warping blocks, and linear filters, where the cascade can be interpreted as the composition of operators. Another contributing motivation for nonlinear systems in composition structures was the work of Dr. Pablo Martínez Nuevo (my office mate) and Hsin-Yu (Jane) Lai on amplitude sampling, which represents a signal with a series of timestamps at which the value of the signal crosses a set of pre-specified thresholds. There turned out to be a pair of signals that are related in a nonlinear pattern and can be conveniently described by function composition. Then I noticed that there were existing works on iterative function equations, some of which had been applied in signal processing and system representation. Aside from research, I served as a teaching assistant for a second time in Spring 2015, for Prof. Gregory W. Wornell and Prof. Stefanie S. Jegelka who taught the course *Inference and Information*. In addition to the enjoyable TA experience, I personally gained much deeper understanding on the related topics including decision and estimation theory, inference algorithms, information geometry, and asymptotics, which I found really attractive.

In Summer 2015, I had a terrific internship at IBM Research, working with Dr. Dennis Wei, Dr. Kush R. Varshney, and Dr. Dmitry M. Malioutov. The project for the internship was to learn two-level Boolean functions from a dataset as a human-interpretable classifier, where the two-level Boolean function has the form of multivariate function composition; Chapter 6 of my thesis originated from this internship project. Feeling attracted, I further developed this project after coming back to MIT from the internship, especially on the theoretical analysis for the noiseless formulation. With exploration of the disjunct and conjunct properties, I came up with the sufficient conditions that guarantee the optimal solution to the relaxed formulation be binary, which was shown in Theorem 6.1 in Section 6.2.2.

The thesis topic became more inclined to composition structures as time went by, for which I personally considered that cascade systems would be unified as operator composition. The exploration of operator composition led me to an interesting observation that a cascade representation of a two-dimensional FIR filter achieves parameter reduction if the order in each dimension is maintained, which other researchers had already discovered. I then became more interested in bivariate polynomial factorization, which was gradually developed into Chapter 3 of this thesis. Initially, I considered the bivariate polynomial factorization algorithm that utilizes the rank-deficient approximation of the Ruppert matrix, which was explored in my master's thesis on univariate polynomial decomposition. As mentioned, when applied to univariate polynomial decomposition, the empirical issues of numerical precision associated with this algorithm limited the degrees of the input polynomials. However, I was curious about the performance of this algorithm on bivariate polynomial factorization, which could have a different empirical performance from that of polynomial decomposition. Unfortunately, the direct application of this algorithm was actually not smooth; moreover, this algorithm does not control the degrees of the two factors, which is different from our problem formulation. Therefore, we decided not to stick with this algorithm.

Seeking for other methods for bivariate polynomial factorization, I noticed the approaches that use the zero-sum properties of the root functions. Interestingly, my master's thesis also proposed an algorithm for univariate polynomial decomposition, which is based on related zero-sum properties of roots and formulates a mixed integer program. In the spirit of combining integer program with zero-sum properties for bivariate polynomial factorization, I developed the zero-sum mixed integer program algorithm in Section 3.4.1 of this thesis, which turned out not robust for noise on the polynomial coefficients.

A new method for bivariate polynomial factorization was inspired from a weekly meeting with Al in a manner characterized by *"free association"*, which has occurred commonly and is valued by Al's style. There was a seminar on optimization that could be interpreted from the concept of dimensional lifting, which Al mentioned in an individual meeting and connected to a few past techniques developed by DSPG alumni. Though it was not clear at the first glance whether the concept of dimensional lifting could be technically related to bivariate polynomial factorization, later I came up with the idea of using the outer product of two coefficient vectors, which was developed into the low rank matrix formulation and the lifted alternating minimization algorithm in Section 3.4.2. For simplicity, the initial parameters in the simulation were chosen as $\deg_x(F) = \deg_y(F) = \deg_x(G) = \deg_y(G)$, and this new algorithm unfortunately yielded low successful factorization percentage. As I summarized up the work on bivariate polynomial factorization, Al suggested to include the existing works on separable filters in references and to articulate the relationship between polynomial factorization and separable filters, i.e. the separable filters can be considered as a special situation of factorization with each polynomial factor dependent on a single variable. Although this special situation did not seem to be core from a technical perspective, to my surprise, after a few weeks I suddenly realized that the separable filters should be guaranteed the optimal solution for my formulation with a single SVD step, which is the first step in the lifted alternating minimization algorithm. The guaranteed optimality for this special case triggered me to conjecture that the lifted alternating minimization algorithm might have a more satisfactory performance than its low success rate in my previous simulation. By a deeper thinking of the low rank matrix formulation, I finally came up with a possible reason why the situation with $\deg(F) = \deg(G)$ could be especially challenging – in this situation, the two factors can be swapped with different matrices (i.e. the outer product of associated vectors) in the dimensionally lifted parameter space but with the same product polynomial, and therefore the solution space could have a complicated geometrical structure. As shown in Section 3.4.3, simulation results with $\deg(F) \neq \deg(G)$ typically yielded considerably improved performance compared to those with $\deg(F) = \deg(G)$, which was used in my first round of simulation. This improved performance showed the usefulness of the low rank formulation for bivariate polynomial factorization. After further discussion with Petros, I searched and compared with some existing works on low rank matrix approximation.

Thinking of a combination between nonlinear signal processing and operator composition, I explored more deeply in the thread of the blockwise cascade representations of nonlinear systems. Inspired by the over-parametrization approach for Hammerstein systems where the LTI block has a rational transfer function, I was curious about whether this thought can be applied to a Hammerstein system with a general LTI filter, where the latter problem should be more challenging since it has an infinite-dimensional parameter space. Surprisingly, there was already an existing mathematical method for quasimatrix approximation that could be used for the problem that I considered, which led to the results in Section 4.1. Since the study of blockwise cascade models for nonlinear systems typically considered no more than three blocks, the more general situation with more than three blocks in cascade was studied in Section 4.2, with the focus of nonlinear system inversion.

Since Sefa's thesis was on function composition and decomposition, the articulation of the concept of composition under consideration has been brought up multiple times in individual and group meetings over the past years, including whether or how to distinguish function and operator composition. As Al and Sefa suggested, an interpretation of composition could be the replacement of variables of a function or modules of a system, and the modular Volterra system in Chapter 5 can be considered as a combination of this interpretation with nonlinear systems. After developing the parameter estimation algorithm for a modular Volterra system under the weak nonlinearity assumption, I later observed that such modular composition with a nonlinear Volterra module can be described by neither function nor operator composition in the traditional sense. Instead, it may correspond to operator mapping that is mentioned in Section 1.1.4.

After the development of the technical materials, the last stage of my Ph.D. journey focused on the theme and story of the thesis, which Al referred to as the design of *"patch quilt"*. Although the thesis was developed with the general concept of composition, multiple iterations were taken among Al, George, Sefa, and me to articulate the focus of the thesis and to have a unified perspective summarizing the different parts. The key words of this thesis could be *composition* and *parameter estimation*, and Al suggested in a group meeting that the composition structures can be viewed as a special and important class of constrained parametric models, which I feel is a really nice perspective.

As a fun exercise, Al encourages his students to summarize the development of thesis in a *"six-word novel"*. In retrospect, I feel when looking for a research topic or a solution to a problem, opportunities can occasionally be hidden in an area too familiar to be noticed; after becoming familiar with numerous existing works in a specific area, it may be natural to have the misleading impression that all interesting problems or all useful solutions in this area have already been sufficiently discovered, which can usually turn out not the case. However, the discovery of these nearby opportunities might not result from logical thinking in a focused pattern, but rather from sudden inspiration that results from making free association and considering seemingly unrelated problems. Another personal feeling of mine is that when a method does not get a satisfactory performance, the cause may possibly not be the uselessness of the algorithm but the settings for evaluation, such as what happened to the lifted alternating minimization algorithm for bivariate polynomial factorization with the parameter setting $\deg(F) = \deg(G)$. In this case, a simple adjustment of peripheral settings could possibly result in a related problem for which the same algorithm has satisfactory performance, i.e. finding the right nail for a hammer. Notwithstanding, the motivation to make such a simple but important adjustment may be observations obtained from freely considering a totally different problem that seems far from the core target. As a summary of these two feelings, my six words would be: free association discovers hidden opportunities nearby.