Functional Composition and Decomposition for Signal Processing

by

Sefa Demirtas

B.S. Electrical and Electronics Eng., Bogazici University (2007)
S.M. EECS, Massachusetts Institute of Technology (2009)

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 2014

Author . . . .
Department of Electrical Engineering and Computer Science
May 21, 2014

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

Accepted by . . . .
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Functional Composition and Decomposition for Signal Processing

by

Sefa Demirtas

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2014, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Functional composition, the application of one function to the results of another function, has a long history in the mathematics community, particularly in the context of polynomials and rational functions. This thesis articulates and explores a general framework for the use of functional composition in the context of signal processing. Its many potential applications to signal processing include utilization of the composition of simpler or lower order subfunctions to exactly or approximately represent a given function or data sequence. Although functional composition currently appears implicitly in a number of established signal processing algorithms, it is shown how the more general context developed and exploited in this thesis leads to significantly improved results for several important classes of functions that are ubiquitous in signal processing such as polynomials, frequency responses and discrete multivariate functions. Specifically, the functional composition framework is exploited in analyzing, designing and extending modular filters, separating marginalization computations into more manageable subcomputations and representing discrete sequences with fewer degrees of freedom than their length and region of support with implications for sparsity and efficiency.

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

Acknowledgments

The last few years that I spent at MIT have been hard but absolutely rewarding and fun at the same time. Although this is an incomplete list, I would like to thank many people who had their own share in turning this difficult marathon into an enjoyable journey for me. I would like to start with my supervisor Al Oppenheim, whose passionate and excellent guidance made it possible for me to comfortably navigate the rocky paths to a thesis. His experience and wisdom provided me with a great perspective and insight into doing research, teaching and thinking. Al, thank you for sharing my enthusiasm and happiness at times of exciting outcomes and supporting me at times of desperation and hardship. I am grateful for the amazing research home that you provided for me, a place that well exceeded my expectations when I knocked on your door several years ago. It was a pleasure, an invaluable experience and a privilege to work and teach with you.

I am greatly indebted to my thesis committee, Pablo Parrilo and George Verghese, who provided me with very useful comments during the development of this thesis. The great question from Pablo in the first meeting regarding the communities we were trying to impact helped me clarify the language of the thesis. George kindly followed up with the conversations that started in committee meetings, which helped me shape interesting applications and future directions. Their feedback significantly improved the way this thesis was written. I also had many useful discussions regarding polynomials with Bjorn Poonen from Mathematics Department at early stages of the thesis. I would like to thank him for his time and the useful directions he pointed at.

I had the good fortune of having great friends with great minds during my time at the Digital Signal Processing Group (DSPG) between 2010 and 2014, including the alumni who never could leave this fun place: Ballard Blair, Charlie Rohrs, Dan Dudgeon, Dennis Wei, Guolong Su, Jeremy Leow, Joe McMichael, Jon Paul Kitchens, Martin McCormick, Milutin Pajovic, Pablo Nuevo, Petros Boufounos, Qing Zhuo, Sally Wolfe, Shay Maymon, Tarek Lahlou, Tom Baran, Xue Feng, Yuantao Gu, Zahi Karam. The intellectual level of any discussion with them related to research or

anything else quickly climbed to a level that is hard to attain anywhere else. We had very productive brainstorming sessions in group meetings, or intellectual food fights as Al would like to refer to them, which helped me at almost every stage of my thesis. The fun group outings we had are already among my unforgettable memories. I would like to thank each and every one of them for making this place a friendly environment to do research and to be at. I also would like to thank Eric Strattman, Kathryn Fischer and Laura von Bosau for their help and support in the smooth running of DSPG, and providing more smiles than we already had. My colleagues and the residents of the fifth and sixth floors of RLE contributed significantly to the great friendly work environment, and I would like to thank them all for their warmth.

I would like to send my special thanks to Guolong, my extremely kind, hardworking and helpful colleague and officemate for all his support and useful discussions during my research, all the polynomial decomposition coding and simulations he took off my shoulders at my busiest times, the long sleepless nights he worked with me to meet deadlines and last but not least for his great friendship. Guolong, I really appreciate what you did over the last few years, thank you very much. I also would like thank Dennis, Martin, Milutin, Shay and Tarek for always being available for long, exciting and helpful discussions when I wanted to borrow their brains spontaneously.

My thesis have greatly benefitted from very interesting questions that arose in my discussions with several creative and bright people outside of DSPG. I would like to thank Arthur Redfern, Ben Vigoda, Charles Sestok, Fernando Mujica, Jeff Bernstein, Joyce Kwong and Theo Weber for the directions that they suggested which eventually became useful applications that fit within the composition framework explored in my thesis. Ben, Jeff and Theo, thank you for your suggestion regarding decomposing factor tables and the many useful follow up discussions that eventually shaped the chapter and the applications regarding the decomposition of multivariate functions. Arthur, Charles, Fernando and Joyce, thank you for the great brainstorming sessions that made it clearer to us the importance and convenience of modular filters.

I feel very fortunate to have my amazing Bostonian or once-Bostonian friends, the presence of whom was a great support during my time at MIT and provided the much needed distractions from research in forms of live music performances, potlucks, weekend getaways or just a cup of coffee in an afternoon: Alessio Spantini, Aylin Kentkur, Banu Erdim, Baris Nakiboglu, Donghyun Jin, Ebru Bekaslan, Eray Sabancilar, Guner Dincer Celik, Hakan Sonmez, Halil Tekin, Huseyin Erdim, Ilke Kalcioglu, Jelena Pajovic, Jungwoo Joh, Keren Miller, Murad Abu-Khalaf, Nalan Senol Cabi, Ozan Candogan, Ozgur Amac, Renin Hazan, Saban Bilek, Serkan Cabi, Ted Golfinopoulos, Yalcin Cayir, and my little baby friends Yarden Maymon and Petar Pajovic. Thank you all for being my friends and always being there for me.

I would like to express my deepest gratitude to my friend Baris Nakiboglu. Baris provided me with the greatest moral as well as technical support during my difficult times when switching from my previous research discipline into a new one for which the background building process would be difficult and time consuming. Baris, thank you for encouraging me to move forward even when many others approached the field switching idea suspiciously. You should know that you were the one who tipped the scale to the correct side, and thank you for being that person.

I would like to send my heart-felt thanks to my parents, Seher Demirtas and Selah Demirtas, my brother Mustafa and my sister Yeliz, who always made me feel their encouragement and pride, which fueled all my endeavors in my life including my PhD adventure. Mom, Dad, thank you for raising me to be who I am today and for your unconditional love and support. I also would like to sincerely thank the Tokuccu family: Munteha, Sevilay, Senay, Esra, Ibrahim, Emra and Eda, none of my achievements would be possible without your support, love and acceptance of me as a member in your family for as long as I have known myself.

Finally, I would like to thank my lovely wife Selda Celen Demirtas for her endless support during the development of this thesis. Every time I attempt to compare my life before and after meeting her, I get tongue-tied and fail in finding the right words to thank her enough for the difference she has made for me. Selda, your presence has made the world a beautiful place for me in which to live, thrive and achieve myself. Who you have been to me and what you have done for me made me happier, stronger, more successful, courageous and optimistic. Thank you for being in my life.

Contents

Chapter 1

Introduction

Signal processing is a rich discipline in which functional composition and decomposition can potentially be utilized in a variety of creative ways. In a broad sense, the aim of this thesis is to create a systematic framework in which these two operations can be exploited more fully in signal processing applications. From an analysis point of view, one can often gain further insight into existing techniques by reinterpreting them in terms of functional composition and decomposition. From a synthesis point of view, one can develop new algorithms and techniques which inherit desirable properties of these two operations. Moreover, computations can be performed more efficiently and data can be represented more compactly in information systems in the presence of a compositional structure.

In Section 1.1, functional composition and decomposition operations will be defined. Their different interpretations will be shown to correspond to parallelization, cascading and recursion, which are among methods that are often used to tackle computationally difficult tasks. In Section 1.2, certain desirable aspects and implications of composition and decomposition will be introduced as the focus of the framework to be exploited in signal processing, namely an alternative way for compact representations of signals, modularity in designs and separability of computations. Section 1.3 will conclude the chapter with an outline of the other chapters.

1.1 Functional Composition and Decomposition

Functional composition is the application of one function to the results of another function. Conversely, functional decomposition is directed toward expressing a given decomposable function as a composition of other functions, usually of lower order or complexity. If the function is not decomposable, functional decomposition may be applied to obtain a decomposable approximation. In this thesis, two notations will be used interchangeably to denote the composition of two functions A and B, namely $A(B(\cdot))$ and $A \circ B(\cdot)$. Compositions of two operators will be distinguished by using curly brackets, i.e. $A\{B\{\cdot\}\}$.

Functional composition can be interpreted conceptually as a sequence of operators applied to an input function or variable. This corresponds to cascading subfunctions in order to obtain a more complex function, or cascading subsystems in order to achieve a more sophisticated system to process an input. One simple example is the application of filtering to an input signal $x[n]$ using a cascade of two lower order subfilters, which can be associated with operators $\mathcal{G}\{\cdot\}$ and $\mathcal{F}\{\cdot\}$, respectively. The result of the filtering operation can be expressed as the composition of these operators acting on the input, namely $\mathcal{F}\{\mathcal{G}\{x[n]\}\}$. This composition takes a simple form if the filters are linear and time invariant (LTI), in which case the z-transform of the composition can be expressed as the product of individual z-transforms of the filters with that of the input signal,

$$\mathcal{F}\{\mathcal{G}\{x[n]\}\} \overset{z}{\longleftrightarrow} F(z)G(z)X(z).$$

Therefore, composition of cascaded subsystems in the case of LTI systems is commutative, a property that is usually lacking in the composition of other more general functions including nonlinear filters. This observation raises an interesting question as to what classes of operators accept a rather simple representation in other domains when they are composed. This question is not the main focus of this thesis but is a promising future direction.

Another case which can be associated with functional composition is that of dividing a complex function into subfunctions the results of which are not required by each other in advance and therefore could be obtained separately and combined appropriately at a later stage. This corresponds to parallelization of a task in which each subtask can be performed by an independent subsystem such as multiple processors or even different computers in a network. In signal processing, a common scheme where parallelization is used is the implementation of high order infinite impulse response LTI filters as a combination of several low order subfilters, in which usually the low order filters are obtained by a partial fraction expansion. For two such subfilters $\mathcal{G}_1\{\cdot\}$ and $\mathcal{G}_2\{\cdot\}$, the output becomes the composition

$$\mathcal{F}\{\mathcal{G}_1\{x[n]\}, \mathcal{G}_2\{x[n]\}\} = \mathcal{G}_1\{x[n]\} + \mathcal{G}_2\{x[n]\} \overset{z}{\longleftrightarrow} G_1(z)X(z) + G_2(z)X(z)$$

where $x[n]$ is the input and $\mathcal{F}\{\cdot,\cdot\}$ simply corresponds to the summation of its arguments in this example. This is a simple example of composing the multivariate function $\mathcal{F}$ with univariate functions $G_1(z)$ and $G_2(z)$. In this thesis, composition of multivariate functions will emerge in a discussion of decreasing the computational complexity in certain classes of problems requiring marginalization.

A recursive approach to solving computationally difficult problems can also be associated with a composition of subfunctions, where these subfunctions are similar to the original function applied to easier subproblems. A very successful application of this approach in the context of signal processing is the Fast Fourier Transform (FFT) algorithms to compute the Discrete Fourier Transform (DFT) of a long sequence $x[n]$. In the decimation-in-time FFT algorithm, denoting the N-point DFT of a length-N sequence with the operator $\mathcal{G}_N\{\cdot\}$ leads to a recursion

$$\mathcal{G}_N\{x[n]\} = \mathcal{F}\{\mathcal{G}_{\frac{N}{2}}\{x_e[n]\}, \mathcal{G}_{\frac{N}{2}}\{x_o[n]\}\},$$

where $x_e[n]$ and $x_o[n]$ are subsequences of $x[n]$ consisting of its even and odd indexed terms. The operator $\mathcal{F}\{\cdot,\cdot\}$ corresponds to combining its two arguments through simple additions as well as multiplications with different roots of -1, therefore DFT of a long sequence can be computed more efficiently by combining the DFT of its subsequences recursively.

So far, functional composition and decomposition were conceptually associated with cascading, parallelization and recursion by representing subcomputations as operators. In these approaches, the composition of operators does not necessarily lead to the composition of the actual functions representing these operators. For example, the cascade of two LTI filters was represented as $\mathcal{F}\{\mathcal{G}\{\cdot\}\}$ with an operator representation of each filter. However the actual mathematical representation of this operation is two convolutions involving impulse responses and does not involve composition of these impulse responses. In this thesis, composition and decomposition of actual functions will be explored rather than their operator representations. In other words, in the context of this thesis, composition will refer to the mathematical operation of embedding functions into others through a direct replacement of variables with functions.

1.2 A Framework for Signal Processing

1.2.1 Goals

It is the main goal in this thesis to develop a systematic framework in which functional composition and decomposition can be exploited more fully in signal processing. Towards this goal, existing functional composition and decomposition algorithms in the mathematical literature will be identified, implemented, extended or new algorithms will be proposed that also accommodate the common optimality and efficiency criteria of signal processing. A complete discussion regarding composition and decomposition of all types of functions is neither possible nor meaningful. Therefore, the focus in this thesis is on certain classes of functions that are ubiquitous in signal processing, namely univariate polynomials, frequency responses and discrete multivariate functions. Once the tools are developed, some of the existing signal processing applications in the literature will be revisited and re-interpreted within this framework illustrating its additional benefits, and also new applications will be formulated.

Three common themes will appear as algorithms are developed and applications are formulated in the following chapters; namely sparsity, modularity and separability. Each of these themes will have a close relationship with a corresponding interpretation of composing two functions. The relationship between the themes and the functions that constitute the focus in this thesis are depicted in Figure 1-1.

Compact Representations and Sparsity

In a parametric representation of two functions F and G, their composition $F \circ G$ can be interpreted as expanding some or all parameters of F with the parameters of G. Parameters can be coefficients of a polynomial, variables in a multivariate function or sample values of bandlimited functions. In most compositions, the number of parameters in the composition $F \circ G$ well exceeds the total number of parameters in F and G. This suggests an opportunity for a more compact representation of a decomposable function in terms of the parameters of its components rather than its direct parametric representation. This can be viewed as an alternative way to reduce the number of required parameters to represent such functions with implications for sparsity.

Modular Structures

Another interpretation of functional composition is to embed one function G into another function F to obtain $F \circ G$. If a function is used to represent a signal processing operation or task, the composition $F \circ G$ may correspond to repeating the subtask G at different processing levels encapsulated by the main task F. The implementation of the subtask G may be accomplished by a standardized and optimized off-line design, which can then be repeatedly used at each processing level it is needed. This naturally leads to a modular pattern with the main module being G.

Separation of Computations

A usual approach to simplify difficult computational problems is to divide them into more manageable parts, for example in the case of factorable functions. The in-

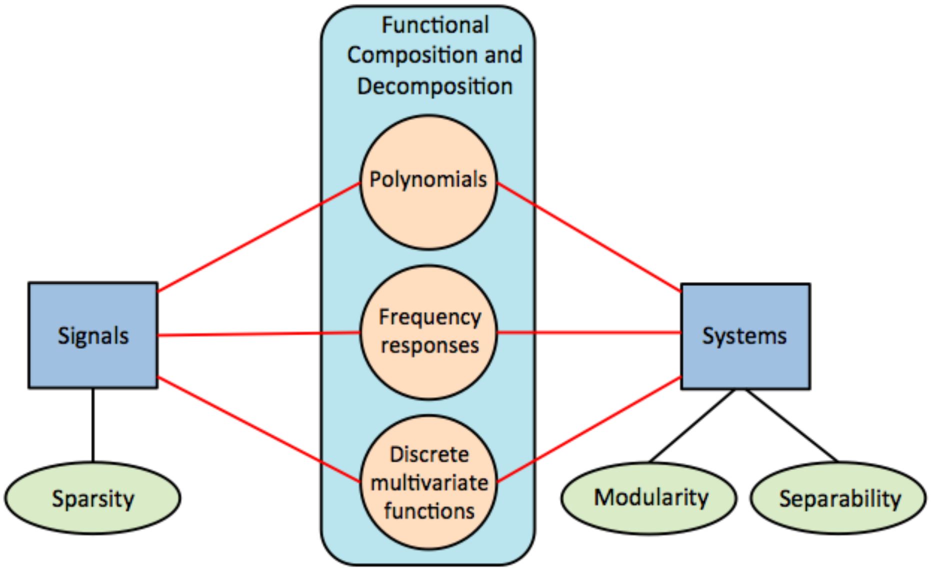


Figure 1-1: Functions for which composition and decomposition operations will be discussed and the implications of these operations for signals and systems that will arise in the context of examples discussed.

terpretation of functional composition as embedding subtasks in other tasks also provides an alternative method to separate computations into smaller subcomputations. Computational efficiency usually follows by carefully framing and scheduling the subcomputations.

1.2.2 Main Contributions

Functional composition and decomposition have manifested themselves in a variety of contexts in signal processing. However, they were often not identified explicitly and not manipulated utilizing the mathematical formalism. In this thesis, formal mathematical representations of composition and decomposition will be adapted whenever possible, which will make it more convenient to exploit them by borrowing techniques from the mathematics literature. This also constitutes the basis for a framework to systematically approach certain signal processing applications.

A chapter of this thesis is devoted to an overview of existing decomposition methods, their evaluation and development of a new decomposition method for an important class of functions for signal processing, namely polynomials. The identification of polynomial decomposition techniques as a potentially useful signal processing tool presents a new viewpoint to manipulate finite length discrete time signals and LTI systems. As computational challenges ever evolve and signal processing keeps offering new and creative solutions for them, composition and decomposition of polynomials may emerge as a promising set of operations to exploit in applications involving finite length sequences.

Designing modular filters constitutes an important subset of applications that are advocated in this thesis, which arises as an application of frequency response composition and decomposition. Development of a technique to decompose frequency responses leads to a convenient framework to design and analyze modular filters, revisit and re-interpret filter sharpening applications as a special case of modularity, which in turn allows improving and generalizing sharpening methods.

A further accomplishment is that decomposable multivariate functions are shown to be potentially as useful as their factorable counterparts for an important class of signal processing and machine learning applications, namely those that require marginalizations. This is accomplished by identifying a close relationship between decomposability and factorability of multivariate functions by introducing latent variables and temporarily increasing the dimensionality of these functions. This allows exploitation of some well-known and computationally efficient methods in the case of decomposable multivariate functions which were originally developed for marginalizing factorable functions. The relationship between decomposability of a lower dimensional function and the factorability of an associated higher dimensional function also appears in the literature for polynomials, a property which deserves further consideration as to whether it exists more generally than in the case of these two classes of functions.

1.3 Outline

In Chapter 2, applications from the existing signal processing literature that can be interpreted as a form of composition and decomposition are discussed. This chapter also reviews the basic concepts of time and frequency warping since these are commonly exploited in this literature.

Chapter 3 explores polynomial composition and decomposition and compares the implementation of several polynomial decomposition algorithms, including both exact and approximate decompositions. A new method for approximate polynomial decomposition is introduced. The chapter concludes with the discussion of sensitivity of polynomial composition and decomposition operations and methods for obtaining equivalent decompositions with lower sensitivity.

In Chapter 4, the composition and decomposition of frequency responses are defined and methods are developed for their decomposition into a rational function and a polynomial. The decomposition quality is specified in terms of the Chebyshev norm of the difference between the given frequency response and its approximation as a composition. The method is also extended to the cases where the decomposition quality is specified based on approximating the magnitude of a given frequency response with the magnitude of a composition, which, for example, becomes useful in designing analog modular filters.

Composition and decomposition of discrete multivariate functions are discussed in Chapter 5. For multivariate functions, their decomposability and factorability are shown to be related by artificially increasing the dimensionality of the function through the introduction of latent variables. This relationship allows using well-established matrix factorization algorithms to decompose discrete multivariate functions.

Several applications of functional composition and decomposition are discussed in Chapter 6. These applications show that the functional composition viewpoint leads to efficiency in representations, implementations and computations. These applications shown here are only a few examples of a possibly much larger set of applications that can be formulated in the richness of signal processing.

Chapter 2

Composition and Decomposition in Signal Processing

Functional composition and decomposition, although not identified as such explicitly, have appeared in a variety of signal processing contexts. Phase modulation is one such example where a carrier sinusoid is composed with the signal to be transmitted and as a result, the carrier signal experiences a time warping in the form of local changes in its frequency and phase. A similar effect on signals that can be interpreted as functional composition is the wow and flutter in musical recordings which stem from imperfect and variable-speed recording and playback, where the varying speed can be associated with a warping function. In other examples, functional composition has been intentionally introduced into signal processing systems and algorithms in order to exploit the benefits of time and frequency transformations as well as reusing the same signal processing blocks repeatedly to avoid the expense of designing larger systems in one step.

In this chapter, several signal processing contexts that can be interpreted from a composition viewpoint are described to illustrate that composition is not a totally unconventional concept in signal processing and its benefits have been recognized, yet it is still far from being fully exploited in a systematical manner. Since time and frequency transformations are commonly exploited in many of these contexts, as a first step, these two operations are reviewed and re-interpreted as a form of functional composition.

2.1 Time and Frequency Transformations

2.1.1 Time Transformations

Time transformation of a signal $x(t)$ by another function of time $\gamma(t)$ can be defined as the substitution of its time variable t with $\gamma(t)$, i.e. $x(t) \to x(\gamma(t))$, and is a generic example of composition of functions. The resulting signal can be represented as $y(t) = x \circ \gamma(t)$ with the composition notation described in Chapter 1. An example of time transformation is shown Figure 2-1 with $x(t) = 2\sin(5\pi t + 0.34\pi)$ and $\gamma(t) = \tanh(3t)$. The independent variable axis, time axis in this case, gets warped in a way consistent with $\gamma(t)$, the time warping function. This can be visually justified by examining the time plots of $x(t)$ and $y(t) = x(\gamma(t))$. The time axis gets locally compressed when the slope of $\gamma(t)$ is greater than the slope of the identity warping function $\gamma_{id}(t) = t$, i.e. unity, and locally expanded when the slope is less than unity. For an unambiguous recovery of $x(t)$ from $y(t)$, $\gamma(t)$ is required to be known and invertible. In that case

$$x(t) = y(\gamma^{-1}(t)), \tag{2.1}$$

i.e., $x(t)$ can be obtained by the application of the inverse time warping function to $y(t)$.

2.1.2 Frequency Transformations

Frequency transformations of discrete time signals can be viewed as a special case of transforming, or composing, the corresponding z-transform since the Fourier transform of a signal is the evaluation of its z-transform on the unit circle. Only the discrete time case will be explored as the discussion of continuous time signals is similar. Since the z-transform $F(z)$ of a causal discrete time signal $f[n]$ of length $M + 1$

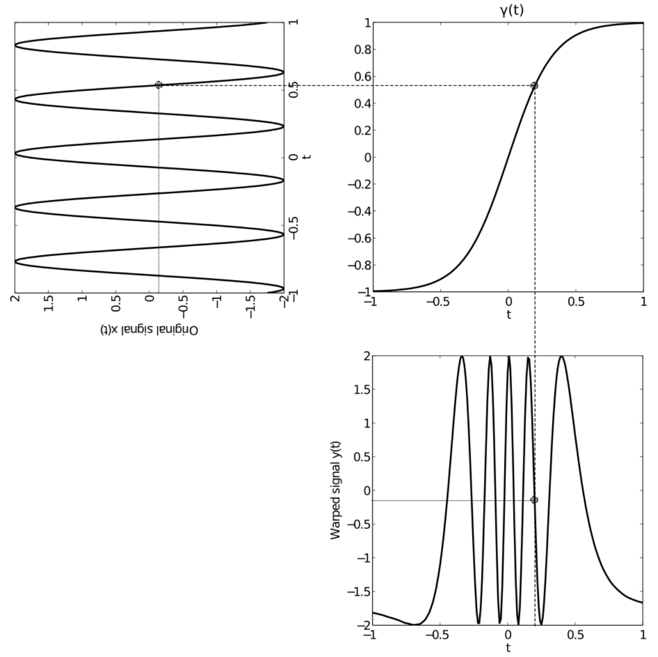


Figure 2-1: Time transformation $\gamma(t) = \tanh(3t)$ applied to the signal $x(t) = 2\sin(5\pi t + 0.34\pi)$.

is given by

$$F(z) = \sum_{k=0}^{M} f_k z^{-k}, \tag{2.2}$$

its composition with the z-transform $G(z)$ of another causal discrete time signal $g[n]$ can be defined as

$$H(z) = F(G(z)) \triangleq \sum_{k=0}^{M} f_k G^k(z), \tag{2.3}$$

where $H(z)$ is the resulting z-transform. This corresponds to replacing z^{-1} in the definition of $F(z)$ with $G(z)$, or equivalently, to composing the polynomial $F(\cdot)$ with another function of z^{-1}, namely $G(z)$.

The frequency transformation implied by the composition in equation (2.3) can be interpreted as follows. Before the composition, the DTFT $F(e^{j\omega})$ is specified by computing the z-transform $F(z)$ on the unit circle, which is parametrized as $z^{-1} = e^{-j\omega}$ and is transversed by sweeping ω from 0 to 2π. The composition in equation (2.3) transforms the sequence $f[n]$ to $h[n]$ by substituting $G(z)$ for z^{-1}. Therefore the DTFT $H(e^{j\omega})$ corresponds to computing $F(z)$ on the contour parametrized by $z^{-1} = G(e^{j\omega})$ in the complex plane. This can be viewed as warping the unit circle as specified by $G(e^{j\omega})$. Figure 2-2 illustrates the warping of the unit circle by

$$z^{-1} = G(e^{j\omega}) = -1 + 0.5e^{-j\omega} + 0.4z^{-2j\omega} \tag{2.4}$$

which are the set of new points on which $F(z)$ is computed to yield $H(e^{j\omega})$. The computation of the z-transform on contours other than the unit circle has proved useful in different signal processing contexts, for example the chirp z-transform [43].

An important class of mappings $G(z)$ are all pass functions which satisfy $|G(e^{j\omega})| = 1$. The importance of such mappings stems from the fact that they map the unit circle onto itself, therefore the Fourier transform before and after a composition are frequency warped versions of one another. This also implies that compositions of all-pass mappings with this property results in an all-pass mapping with the same property as the unit circle is mapped onto itself by each map in the composition chain. Mappings using all-pass functions have proven to be very important in signal processing applications and several examples of their use are shown in Section 2.2.

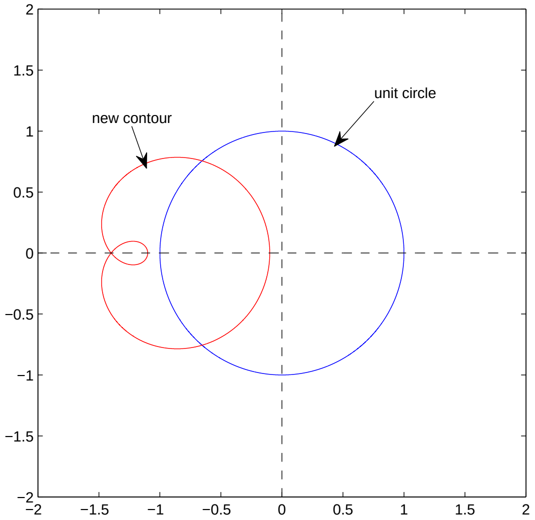


Figure 2-2: The unit circle in the complex plane, parametrized by $z^{-1} = e^{-j\omega}$, and its image under the transformation $z^{-1} = G(e^{j\omega}) = -1 + 0.5e^{-j\omega} + 0.4z^{-2j\omega}$.

2.2 Previous Work

2.2.1 Nonuniform sampling and local bandwidth

It is a well-known fact that a bandlimited signal can be represented by and recovered from its uniform samples taken at a rate at least twice its highest frequency. In cases where a bandlimited signal experiences a time transformation, this property may be lost as the transformation renders the signal non-bandlimited in general. However, the convenience of representing a signal using its finite-rate samples has tempted several authors [14,55] to investigate other means to sample non-bandlimited signals obtained by time warping bandlimited signals and reconstruct them from these samples, where the signal can be sampled at a finite rate consistent with a notion of local bandwidth corresponding to the time warping function in this context.

Given that a non-bandlimited signal $f(t)$ is in fact obtained by time warping a signal $g(t)$ bandlimited to ω_c, i.e.

$$f(t) = g(\gamma(t)), \tag{2.5}$$

where $\gamma(t)$ is an invertible warping function with $\gamma^{-1}(t) = \alpha(t)$, Wei [55] proposes using the system depicted in Figure 2-3 to sample and reconstruct $f(t)$. Reversal of the time warping in the first stage can be viewed as a preconditioning of the signal to avoid aliasing in the subsequent uniform sampling process. The samples taken in the third step correspond to a non-uniform grid in the original time domain, supporting the intuitive notion of the local bandwidth as the samples are denser when the $\gamma'(t)$ is larger, corresponding to a higher local bandwidth. This method utilizes compositions with functions of time in order to transform a non-bandlimited signal in an invertible way to a bandlimited one to exploit the efficient sampling and reconstruction schemes for the latter.

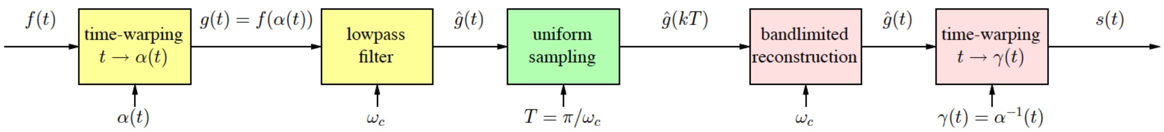


Figure 2-3: A sampling and reconstruction scheme for a non-bandlimited signal $f(t)$ obtained by time warping a signal $g(t)$ bandlimited to ω_c, where $f(t) = g(\gamma(t))$. Figure is adapted from [55].

2.2.2 FFT for Unequal Resolution Spectra

Oppenheim et al. [39] showed that it is possible to use the FFT in order to compute the DFT efficiently on a nonuniform frequency grid after appropriately warping the Fourier transform in the frequency variable, which can be recognized as a form of composition as discussed in Section 2.1.2. In the proposed method, the Fourier transform is effectively composed by a nonlinear function such that a uniform sampling grid, on which a DTFT can be efficiently sampled using the FFT, corresponds to a desired nonuniform sampling grid for the original Fourier transform. This is accomplished by transforming the original sequence $f[n]$ to a new sequence $h[n]$ satisfying the desired relationship between their corresponding Fourier transforms.

Figure 2-4 illustrates how a sequence $h[n]$ is obtained from a causal discrete time sequence $f[n]$. First, $f[n]$ is time reversed and is provided as the input to a system consisting of all-pass filters after the first two subsystems and which is tapped after each block in the chain. Each block in this network and hence the resulting $h[n]$ is parametrized by the real number a. The discrete time sequence $h[n]$ is specified as the values recorded at these taps at $n = 0$, i.e.,

$$h[n] = \tilde{h}_n[0]. \tag{2.6}$$

The relationship between $H(e^{j\omega})$ and $F(e^{j\omega})$ is given by

$$H(e^{j\omega}) = F(e^{j\theta_\alpha(\omega)}) \tag{2.7}$$

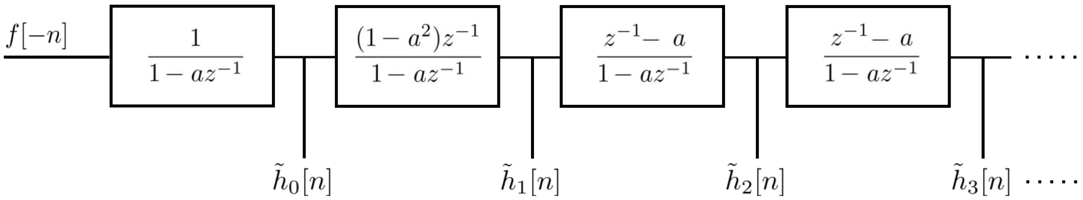


Figure 2-4: The system to obtain $h[n] = \tilde{h}_n[0]$ from $f[n]$ with the frequency response relationship given in equation (2.7) [39].

where

$$\theta_\alpha(\omega) = \arctan \frac{(1-\alpha^2)\sin\omega}{-2\alpha + (1+\alpha^2)\cos\omega}. \tag{2.8}$$

This is a parametric warping of the frequency axis with the parameter α. The DFT of $h[n]$ can be efficiently computed using the FFT, which computes the equally spaced samples of $H(e^{j\omega})$, and hence the non-uniformly spaced samples of $F(e^{j\theta_a})$ as desired. This method can be interpreted as an indirect utilization of composition in the frequency domain to extend the efficiency of the FFT to computations of the DFT on non-uniform grids.

2.2.3 Frequency Transformations of Prototype Filters

Frequency selective filters can be designed by applying an algebraic transformation to a prototype filter, which is usually selected as a low-pass filter. This can be interpreted as another form of functional composition in the context of signal processing. The idea is applicable to both continuous and discrete time filters, and is only illustrated for discrete time in this section.

In Section 2.1.2, frequency transformations were expressed in terms of composing the z-transform $F(z)$ of a discrete time sequence $f[n]$ by a mapping $G(z)$, and computing the Fourier transform on the resulting system function by setting $z^{-1} = e^{-j\omega}$. If $F(z)$ is a rational system function of a causal and stable filter, the system function after the composition is usually required to remain rational and correspond to a causal and stable system. These requirements place certain constraints on the mapping $G(z)$, namely, $G(z)$ must be a rational function of z^{-1} and the inside of the unit circle must be mapped to the inside of the unit circle so that the poles are not mapped across the unit circle [40]. Moreover, for the resulting Fourier transform to take values from the range set of $F(e^{j\omega})$, the unit circle must be mapped onto itself, which requires $|G(e^{j\omega})| = 1$ as discussed in Section 2.1.2.

It was shown in [15] that the most general form of the mappings $G(z)$ satisfying these conditions is of the form

$$G(z) = \pm \prod_{k=1}^{N} \frac{z^{-1} - \alpha_k}{1 - \alpha_k z^{-1}}, \tag{2.9}$$

i.e., the product of a finite number of all-pass system functions each with a parameter a_k. The simplest mapping that maps a low-pass filter $F(z)$ to another lowpass filter is

$$G(z) = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}. \tag{2.10}$$

A prototype low-pass filter $F(z)$ with a cut-off frequency θ_p can be mapped to $F(G(z))$ using such a mapping to obtain a low-pass filter with cutoff frequency

$$\omega_p = \arctan \frac{(1 - \alpha^2) \sin \theta_p}{2\alpha + (1 + \alpha^2) \cos \theta_p}. \tag{2.11}$$

This frequency transformation resembles the nonlinear transformation applied to a discrete time sequence for the efficient computation of its DTFT on a nonuniform grid as described in Section 2.2.2. However, in this setting, the purpose of the transformation is to relocate the cut-off frequency. Equations (2.8) and (2.11) describe an equivalent relationship between the frequency variables ω and θ, and this relationship is illustrated in Figure 2-5.

Compositions with mappings of the form (2.10), i.e., frequency transformations can be used to obtain frequency selective filters more general than another low-pass filter. For example, in order to obtain a band-pass filter with a desired lower cut-off frequency ω_{p1} and a desired higher cut-off frequency ω_{p2}, the mapping

$$G(z) = \frac{z^{-2} - \frac{2\alpha k}{k+1} z^{-1} + \frac{k-1}{k+1}}{\frac{k-1}{k+1} z^{-2} - \frac{2\alpha k}{k+1} z^{-1} + 1} \tag{2.12}$$

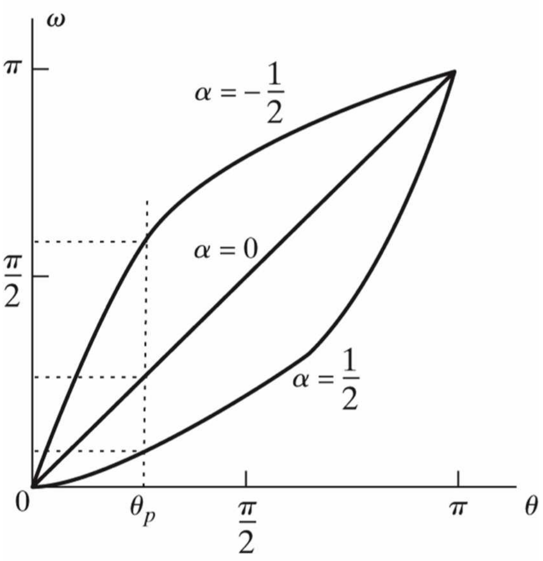


Figure 2-5: The parametric relationship between the frequency variable of a prototype low-pass filter θ and the frequency variable ω of a low-pass filter obtained after the transformation in equation (2.11). Figure is adapted from [40].

is used for composing the prototype low-pass filter with the cut-off frequency θ_p, where

$$k = \cot\left(\frac{\omega_{p2} - \omega_{p1}}{2}\right)\tan\left(\frac{\theta_p}{2}\right) \tag{2.13}$$

and

$$\alpha = \frac{\cos(\frac{\omega_{p2}+\omega_{p1}}{2})}{\cos(\frac{\omega_{p2}-\omega_{p1}}{2})}. \tag{2.14}$$

For a complete list of transformations from a low-pass filter to low-pass, high-pass, band-pass and band-stop filters, the reader is referred to [15] or [40].

2.2.4 Design of Audio Filters

A desirable property of frequency selective filters obtained using frequency transformations as in Section 2.2.3 is the fact that the resulting filter exhibits the same extremal values in the pass-bands and the stop-bands as that of the prototype filter since the composition distorts only the frequency axis. This guarantees that, for example, the specifications for the maximum allowable ripple size are not violated after the frequency warping, a fact that is often exploited in designing minimax-optimal

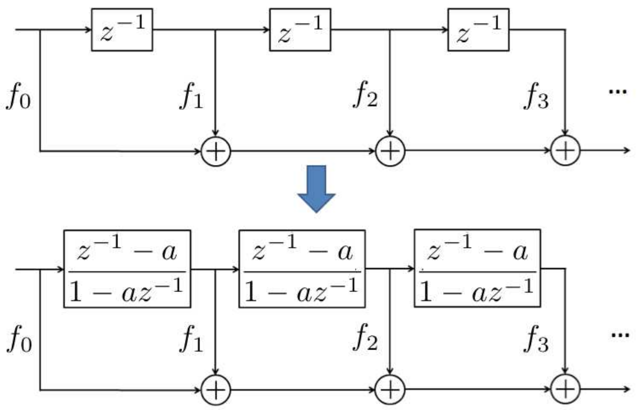

z^{-1}
z^{-1}
z^{-1}
f_0
f_1
f_2
f_3
...
$\frac{z^{-1}-a}{1-az^{-1}}$
$\frac{z^{-1}-a}{1-az^{-1}}$
$\frac{z^{-1}-a}{1-az^{-1}}$
f_0
f_1
f_2
f_3
...

Figure 2-6: Composing a filter $F(z)$ to accommodate audio filter specifications with logarithmic frequency ranges by substituting its delay elements for all pass systems [4].

filters. One example is an audio filter design algorithm introduced in [4].

Due to the natural frequency sensitivity of the human auditory system, frequency specifications in audio and speech applications are usually given on a logarithmic scale. Frequency responses of the filters that are used in these applications have most of the detail at low frequencies. This unbalance in the specifications between low and high frequency ranges causes difficulties in designing filters such as yielding very high filter orders or not converging at all. The design procedure proposed in [4] to alleviate this problem utilizes the idea of warping the frequency scale nonlinearly similar to the methods in Section 2.2.3. This is accomplished by composing the z-transform of the filter by the all pass system function as illustrated in Figure 2-6, which leads to composing the frequency response $F(e^{j\omega})$ of the filter with the function in equation (2.8). The choice of α in the range $[-1, 0]$ corresponds to expanding the frequency axis at low frequencies and compressing at high frequencies, making the design space suitable for logarithmic specifications. Low order and high quality audio equalizers can easily be obtained by composing an FIR filter designed with well-established techniques such as the Parks-McClellan algorithm [42] using this method.

2.2.5 Parks-McClellan Algorithm

Another example of a signal processing context in which functional composition has indirectly appeared is the design of linear phase FIR filters. One approach to designing linear phase causal FIR filters is to design a zero phase FIR filter and delay the sequence in time to the point it becomes causal as in the design of Parks-McClellan filters [42]. This approach utilizes the symmetry in the coefficients of the filter to optimize them indirectly by representing the Fourier transform as a composition of a polynomial and a trigonometric function. For example, a finite length sequence $h[n]$ that is symmetric around zero has a real-valued Fourier transform of the form

$$H(e^{j\omega}) = \sum_{n=-L}^{L} h[n]e^{-j\omega} = \sum_{n=0}^{L} h[n]\cos n\omega \tag{2.15}$$

which can be rewritten as

$$H(e^{j\omega}) = \sum_{n=0}^{L} b_n(\cos\omega)^n \tag{2.16}$$

where $b_n, n = 0, 1, \ldots, L$, the coefficients of the polynomial in $\cos\omega$, depend on the values of $h[n]$. Hence the Fourier transform can be expressed as the composition of a polynomial

$$P(w) = \sum_{n=0}^{L} b_n w^n \tag{2.17}$$

with the function $f(\omega) = \cos\omega$. Due to this specific form of the Fourier transform, Parks and McClellan [42] were able to express the linear phase FIR filter design problem as a polynomial fitting problem, which is well studied in mathematics, and devised the celebrated Parks-McClellan FIR filter design algorithm.

2.2.6 Extending Filter Dimensionality

Mersereau et al. [35] extended the idea by Parks and McClellan to 2-D linear phase FIR filter design. Similar to its one dimensional counterpart in equation (2.15), a 2-D linear phase FIR filter $h[m, n]$ has a frequency response

$$H(e^{j\omega_1}, e^{j\omega_2}) = \sum_{m=0}^{M_1}\sum_{n=0}^{M_2} h[m, n]\cos m\omega_1 \cos n\omega_2. \tag{2.18}$$

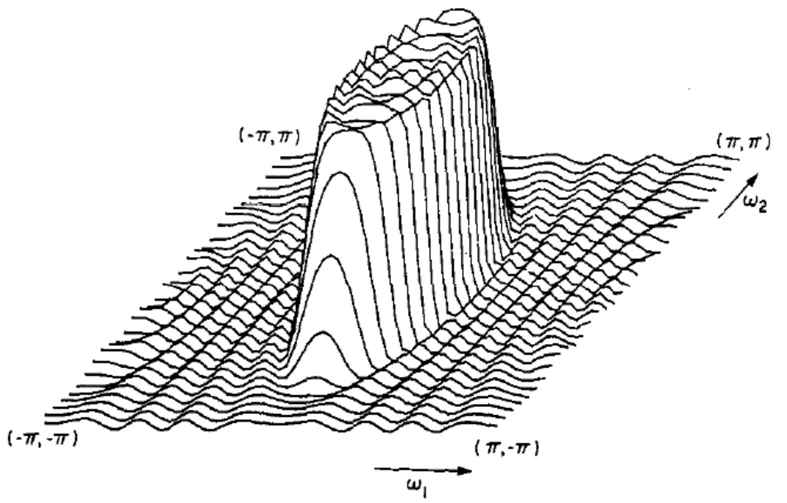


Figure 2-7: A 2-D low-pass filter obtained from a 1-D low-pass filter using McClellan transformation. Figure adapted from [35].

The procedure to obtain 2-D filters from 1-D filters by taking advantage of this similarity in the forms of Fourier transform is called McClellan transformation and involves the substitution

$$\cos\omega \leftrightarrow \sum_{p=0}^{P}\sum_{q=0}^{Q} t[p,q]\cos p\omega_1 \cos q\omega_2 \tag{2.19}$$

in the Fourier transform of the 1-D filter given in equation (2.16). In other words, the univariate polynomial in $\cos\omega$ given in equation (2.16) is composed by a bivariate function of ω_1 and ω_2 given in equation (2.19) to yield the Fourier transform of the 2-D filter

$$H(e^{j\omega_1}, e^{j\omega_2}) = \sum_{n=0}^{L} b_n \left[\sum_{p=0}^{P}\sum_{q=0}^{Q} t[p,q]\cos p\omega_1 \cos q\omega_2\right]^n . \tag{2.20}$$

The substitution given in equation (2.19) also describes the relationship between the 1-D filter response and the 2-D filter response implicitly. Each frequency in the one dimensional ω space corresponds to a contour in the two dimensional frequency plane indexed by ω_1 and ω_2. The shape of the contour is determined completely by the coefficients $t(p,q),\ 0 \le p \le P,\ 0 \le q \le Q$. The value of $H(e^{j\omega})$ at $\omega = \omega_0$ will be identical to the value of $H(e^{j\omega_1}, e^{j\omega_2})$ on the contour which corresponds to ω_0, but the variations from one contour to another will be determined by the coefficients of the 1-D filter $h[n]$. This approach which uses functional composition for 2-D filter design separates the problem into two tractable subproblems, namely the design of the 1-D filter $h[n]$ and the design of the contour parameters $t(p, q)$. This procedure is similar to the case of designing filters from a prototype using frequency transformations and the ripple sizes are not affected by the increase in dimensionality. Figure 2-7 illustrates an example of a two dimensional low-pass filter that is obtained by extending a 1-D Paks-McClellan filter using McClellan transformation.

2.2.7 Filter Sharpening

In cases where multiple identical linear phase filters with an inadequate frequency selectivity are available, it is possible to obtain improved overall frequency characteristics that exhibits smaller deviations from zero in the stop-bands and smaller deviations from unity in the pass-bands. This can be achieved by using replicas of the given filter through an interconnection of adders and gains. This procedure is usually referred to as filter sharpening [26]. In this section, several approaches by different authors to the filter sharpening problem will be reviewed. These approaches result in polynomial transformations applied on the filter, which is another interesting and useful example that can be interpreted as a composition in the context of signal processing, namely the composition of a polynomial and a filter frequency response. However, all of these methods are ad hoc, some of them require exhaustive searches and even in that case lead to suboptimal solutions. In Chapter 6, the compositional structure of sharpened filters will be exploited in order to obtain both superior and systematic methods for filter sharpening as compared to the current methods.

A straightforward approach to sharpening a filter with a frequency response $G(e^{j\omega})$ is to cascade the filter with itself to obtain a response $G^2(e^{j\omega})$, but this has an adverse effect in the passband since squaring will increase the deviation from unity. Tukey [52] proposed a method called *twicing* which involves filtering the input with $G(e^{j\omega})$ and adding back to the input the residual between the input and the output before a

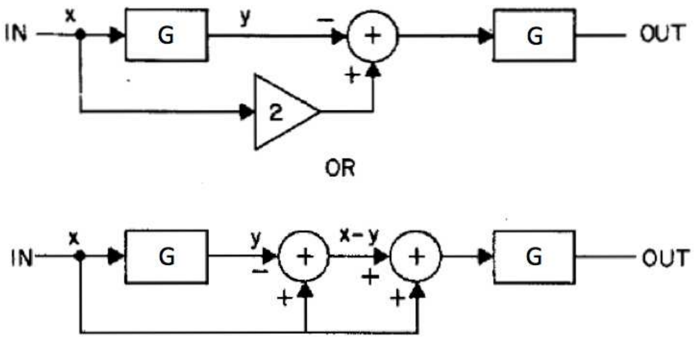


Figure 2-8: Implementation of twicing as proposed in [52]. Figure is adapted from [26].

second stage of filtering. The effective frequency response in this case becomes

$$\begin{aligned} H_{\text{tw}}(e^{j\omega}) &= (1 + (1 - G(e^{j\omega})))G(e^{j\omega}) \\ &= 2G(e^{j\omega}) - G^2(e^{j\omega}) \end{aligned} \tag{2.21}$$

The implementation of twicing is illustrated in Figure 2-8.

Kaiser and Hamming [26] observed that the effective transformation $2x - x^2$ that is being applied to $G(e^{j\omega})$ in twicing has a desirable attenuating effect on the passband deviations from unity but an undesirable magnification effect on stop-band deviations from zero, the exact opposite effects observed with cascading corresponding to the transformation x^2. They explained the effect of these transformations through the value of their slope at $x = 0$ for stop-band and $x = 1$ for passband; a zero slope will attenuate the magnitude of deviations and a slope that is greater than unity will increase the deviations. Therefore, they proposed using transformations $A(x)$, which they referred to as amplitude change functions, with vanishing derivatives at both $x = 0$ and $x = 1$ in addition to the constraint $A(0) = 0$ and $A(1) = 1$. This latter constraint guarantees mapping the magnitude in the stop-bands to zero and the magnitude in the pass-bands to unity. For example, the smallest order polynomial transformation satisfying all of these constraints is $3x^2 - 2x^3$. The comparison of these amplitude change functions is illustrated in Figure 2-9.

Kaiser and Hamming [26] provided a general formula to yield higher order polynomials with higher order tangencies, i.e. vanishing derivatives, at $x = 0$ and $x = 1$

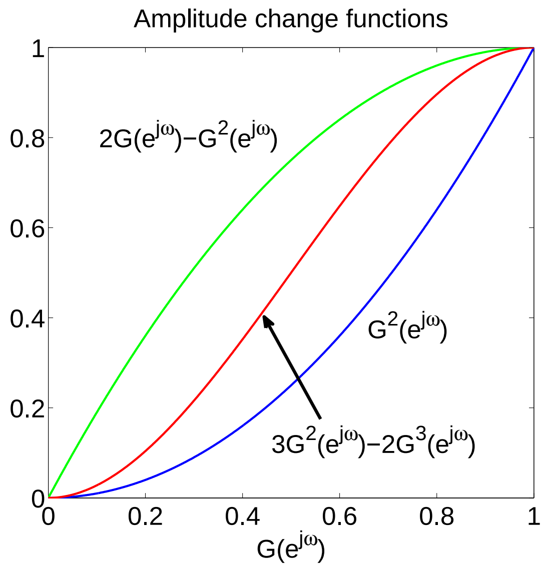


Figure 2-9: Comparison of three different amplitude change functions [26].

satisfying $A(0) = 0$ and $A(1) = 1$. Moreover, the order of tangencies at these two points are not required to be equal. A polynomial with n-th order tangency at zero and m-th order tangency at unity is given by

$$A(x) = x^{n+1} \sum_{k=0}^{m} \frac{(n+k)!}{n!k!}(1-x)^k. \tag{2.22}$$

Kaiser and Hamming's work in [26] on filter sharpening of linear phase filters inspired other authors [12, 38, 48] to approach this problem in a slightly more structured way than originally attempted. In this new approach, the overall design after sharpening was referred to as tapped cascaded interconnection of FIR subfilters.

Nakamura and Mitra [38] advanced the notion of filter sharpening in two directions. The first one is to find the amplitude change function coefficients that not only satisfy the desired values and the order of tangencies at $x = 0$ and $x = 1$, but also minimize the mean squared error between the sharpened filter response and the ideal response. The latter is achieved at the expense of requiring a higher order polynomial than otherwise needed. Their second contribution is to introduce modifier filters $T(e^{j\omega})$ with coefficients of the form $\left(\frac{1}{2}\right)^k$, where k is an integer, such that they can be implemented using only bit shift operators and no explicit multiplications. The purpose of using these modifier filters was to approximate the behavior of the subfilters and reduce their burden to meet the subfilter specifications, and then cascade them with the modifier filter to meet the specifications more accurately. Coefficients of a similar form were later explored not only for subfilter modifiers but also for the amplitude change functions to obtain multiplierless filters by Chen [12].

Saramaki [48] viewed the designs using tapped cascaded interconnection of FIR subfilters as a form of a frequency transformation where the coefficients of the amplitude change function $F(z)$ corresponded to the prototype filter coefficients, or the *tap coefficients*, and the subfilter $G(z)$ as the frequency transformation similar to the discussion in Section 2.2.3. Moreover, he considered the filter approximation in Chebyshev norm rather than to minimize mean square error unlike in [38], and discussed the following four approximation problems:

i. Given the number of subfilters, optimize simultaneously the subfilter and the tap coefficients such that the composite filter satisfies the given specifications with a minimum subfilter order.

ii. Given the subfilter order, optimize simultaneously the subfilter and the tap coefficients to meet the given overall specifications with a minimum number of subfilters.

iii. Given a prescribed subfilter, optimize the tap coefficients to meet the given overall specification with a minimum number of subfilters.

iv. Given the tap coefficients and the number of subfilters, optimize the subfilter to meet the given overall specifications with a minimum subfilter order.

In order to understand the unified approach Saramaki proposed for these four problems, consider the desired overall filter specifications

$$1 - \delta_p \leq H(e^{j\omega}) \leq 1 + \delta_p, \;\; \omega \in I_{pass} \tag{2.23a}$$

$$-\delta_s \leq H(e^{j\omega}) \leq \delta_s, \;\; \omega \in I_{stop}, \tag{2.23b}$$

where δ_p and δ_s are the allowed pass-band and stop-band ripple sizes, and I_{pass} and I_{stop} are the union of pass-band and stop-band frequency intervals, respectively. The design of such a filter is divided into two parts, namely the design of the prototype low-pass filter $F(z)$ such that

$$1 - \delta_p \leq F(e^{j\Omega}) \leq 1 + \delta_p, \;\; 0 \leq \Omega \leq \Omega_p \tag{2.24a}$$

$$-\delta_s \leq F(e^{j\Omega}) \leq \delta_s, \;\; \Omega_s \leq \Omega \leq \pi, \tag{2.24b}$$

and

$$0 \leq G(e^{j\omega}) \leq \Omega_p, \;\; \omega \in I_{pass} \tag{2.25a}$$

$$\Omega_s \leq G(e^{j\omega}) \leq \pi, \;\; \omega \in I_{stop}, \tag{2.25b}$$

where Ω_p and Ω_s are the pass-band and the stop-band edge frequencies for the prototype low-pass filter $F(z)$. The specifications for the prototype and the subfilter guarantee that every frequency in the pass-band, $\omega \in I_{pass}$, is mapped by $G(e^{j\omega})$ to the $[0, \Omega_p]$ interval in which the prototype filter $F(e^{j\Omega})$ is within the specified range $[1 - \delta_p, 1 + \delta_p]$ for the pass-band. Similarly, for every frequency in the stop-band, $\omega \in I_{stop}$, is guaranteed to be mapped to $[\Omega_s, \pi]$ interval in which the prototype filter $F(e^{j\Omega})$ is within the specified range $[-\delta_s, \delta_s]$ for the stop-band. Figure 2-10 depicts the design of a band-pass filter $H(e^{j\omega})$ using the porposed mapping from a low-pass filter $F(e^{j\Omega})$ and the corresponding $G(e^{j\omega})$.

Although this frequency transformation point of view presents an intuitive way to think about tapped cascaded interconnection of identical subfilters, the optimal choice of Ω_p and Ω_s for a given order of prototype or subfilter requires an exhaustive search for each pair (Ω_p, Ω_s), and a design of Parks-McClellan filter for each pair to see if the design criteria are met since this is a non-convex problem in general. Suboptimal designs can be obtained by fixing either one or both of these two parameters. However, the methodology still requires that both the prototype and the subfilter be designed as

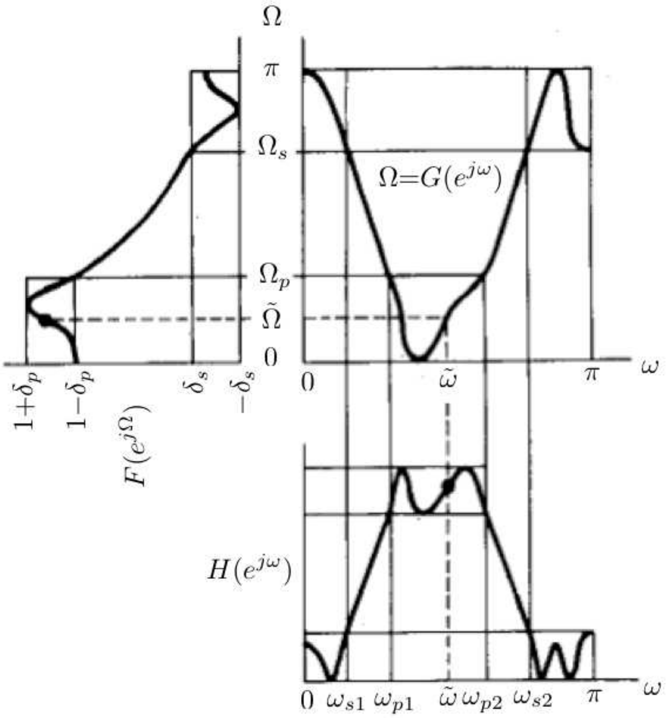
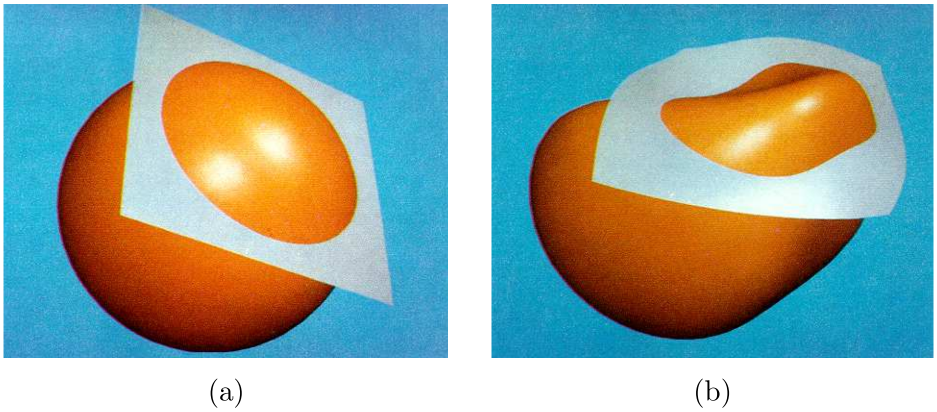


Figure 2-10: Design of a band-pass filter using the low-pass prototype and a mapping as in equations (2.24) and (2.25). Figure is adapted from [48].

a symmetric linear phase filter, which limits the classes of filters that can be sharpened or used in such a structure.

2.2.8 Other contexts

There are many other contexts in which functional composition appears in signal processing implicitly or explicitly. For example nesting all pass filters into systems with feedback, which corresponds to composing z-transforms, is proposed as an efficient method for artificial reverberation [49]. In robotics, the location of the effector at the tip of a robot arm with multiple joints can be expressed as a functional composition, where the location of each joint is a function of the previous joint location and its own parameters, and techniques for multivariate polynomial decompositions can be exploited to decrease on-line computations during robot operation [23, 37]. In computer-aided geometric design, composition can be used for the polynomial reparametrization of Bezier simplexes [18], and for modeling and manipulating ob-



(a) (b)

Figure 2-11: A computer graphics of a sphere and a plane experiencing deformation that is modeled by a functional composition. Figure is adapted from [50].

jects in deformable media [50]. Figures 2-11a and 2-11b illustrate objects in a deformable medium, where the deformation is modeled as the composition of a function for object representations with the deformation function.

2.3 Chapter Conclusions

The signal processing contexts from the current literature discussed in this chapter provide evidence that functional composition has appeared in many contexts directly and indirectly. In some of these, it was natural to interpret the operations on signals and systems as a form of composition such as phase modulation or the phenomena of wow and flutter in musical recordings although the benefit of a functional composition point of view is not immediately obvious. However in other applications which exploit time and frequency transformations, composition and decomposition were intentionally introduced as they were recognized to be useful for designing and generalizing filters, extending sampling and reconstruction schemes and computing spectra efficiently. A third class of applications can be considered to be those in which reusing a limited class of subsystems are promoted to obtain more sophisticated overall systems such as the filter sharpening methods. Although it is natural and straightforward to interpret this latter class from a functional composition perspective, their current analyses were not performed from this way which could have allowed gaining further insight into these applications and achieving further improvements. Moreover, the utilization of functional composition and decomposition have been rather limited in scope in all of these applications, focusing mostly on warping either time or frequency, and failing in exploiting other aspects such as their potential for more compact representations, modular structures and structured computations as discussed in Chapter 1. After developing composition and decomposition algorithms in the following chapters, examples representing these additional aspects will be discussed illustrating the breadth of applications that can possibly be formulated or revisited within this systematic framework.

Chapter 3

Polynomial Composition and Decomposition

A fundamental tool in signal processing to represent and manipulate discrete sequences, namely the z-transform, is a polynomial for the finite length case. Therefore, univariate polynomials constitute an important class of functions that are ubiquitous in signal processing. In this chapter, composition and decomposition of univariate polynomials will be discussed as an important component of a framework in which these two operations can be potentially exploited in signal processing applications.

In Section 3.1, the polynomial decomposition scenarios will be introduced. Several methods exist in the current mathematics literature to decompose polynomials if they are known to be decomposable. In this case, they will be referred to be exactly decomposable polynomials and the methods to obtain exact decompositions will be presented in Section 3.2. At the end of this section, their performances will be compared on a relatively large set of randomly generated polynomials.

If a polynomial is not exactly decomposable, it can be approximated as the composition of lower order polynomials. This will be referred to as approximate polynomial decomposition. Decomposing polynomials approximately is a more difficult problem than the exact decomposition case and is relatively less studied and understood in the mathematics literature. In addition to presenting the existing approximate decomposition algorithms in Section 3.3, a new approximate decomposition algorithm will be developed in Section 3.4 in an attempt to obtain better performance and it will be compared to the existing algorithms.

In Section 3.5, the sensitivity of polynomial composition and decomposition algorithms with respect to perturbations in polynomial coefficients will be investigated. It will be shown that this sensitivity can be lowered by departing to equivalent decompositions obtained with the methods discussed in that section. Section 3.6 will conclude the chapter.

3.1 Decomposition Scenarios

Consider $F(x)$, the polynomial that represents a length-$(M+1)$ sequence f_n,

$$F(x) = \sum_{n=0}^{M} f_n x^n, \tag{3.1}$$

which corresponds to the z-transform[1] of f_n for $x = z^{-1}$. Composing $F(x)$ with another polynomial that represents a length-$(N+1)$ sequence g_n yields

$$H(x) = F(G(x)) = \sum_{n=0}^{M} f_n G^n(x). \tag{3.2}$$

If a polynomial $H(x)$ can be represented as a composition of two polynomials with orders greater than unity as in equation (3.2), then $H(x)$ is referred to as a decomposable polynomial. The sequence represented by $H(x)|_{x=z^{-1}}$ becomes

$$h_n = f_0(g_n^{(0)}) + f_1(g_n^{(1)}) + f_2(g_n^{(2)}) + f_3(g_n^{(3)}) + \cdots + f_M(g_n^{(M)}) \tag{3.3}$$

[1] In different disciplines, the z-transform is defined with $x = z$. Although this thesis adopts the common notation in the signal processing contexts with $x = z^{-1}$, z-transforms will still be referred to as polynomials representing finite length sequences in the context of this thesis.

where $g_n^{(i)}$ corresponds to i self-convolutions of the sequence g_n. This relationship can be expressed through a matrix equation

$$\mathbf{Cf} = \mathbf{h} \tag{3.4}$$

or, written explicitly,

$$\begin{bmatrix} | & | & | & & | \\ | & | & | & & | \\ | & | & | & & | \\ | & | & | & & | \\ g_n^{(0)} & g_n^{(1)} & g_n^{(2)} & \cdots & g_n^{(M)} \\ | & | & | & & | \\ | & | & | & & | \\ | & | & | & & | \\ | & | & | & & | \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_M \end{bmatrix} = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ \vdots \\ h_{MN} \end{bmatrix}, \tag{3.5}$$

where the ith column of matrix $\mathbf{C}$ is the sequence $g_n^{(i-1)}$. The coefficient vectors of $F(x), G(x)$ and $H(x)$ will be denoted by $\mathbf{f}, \mathbf{g}$ and $\mathbf{h}$, respectively, in the sequel. The number of rows in $\mathbf{C}$ is $MN + 1$, i.e. the length of $g_n^{(M)}$ which is the highest order self convolution of g_n in equation (3.3), and the number of columns is $M + 1$.

The problem of decomposing polynomials can be divided into three categories. Given that a polynomial $H(x)$ is decomposable as in equation (3.2) and $G(x)$ is known, finding $F(x)$ has a straightforward solution due to the linear relationship between $\mathbf{f}$ and $\mathbf{h}$ as in equation (3.4). $\mathbf{C}$ is a matrix with fewer columns than rows, therefore the solution for $\mathbf{f}$ will be unique if it is full-rank. This is indeed the case unless the leading coefficient g_M is zero, in which case $G(x)$ can be regarded to have one less order.

The other two decomposition scenarios assume that even though $H(x)$ is known to be decomposable, $G(x)$ is unknown. In one case, $F(x)$ may be provided, and in the other case it may be also unspecified. Since the relationship between $\mathbf{g}$ and $\mathbf{h}$ is non-linear, both of these problems are much harder to solve than the case when $G(x)$ is known. The specification of $F(x)$ does not facilitate the solution of the nonlinear problem of determining a candidate for $G(x)$ within the existing decomposition algorithms as will be discussed in Section 3.2. However, its specification is useful to uniquely determine a decomposition for $H(x)$ since, otherwise, there are infinitely many equivalent decompositions that can be obtained using compositions with first order polynomials as will be described later in Section 3.5.4. As discussed in Section 3.2, all existing decomposition algorithms have assumed the most general case in which both $F(x)$ and $G(x)$ are not specified, focused on determining a candidate $G(x)$ first and then used the linear relationship in equation (3.5) to determine a candidate $F(x)$.

3.2 Exact Decomposition Algorithms

In this section, four polynomial decomposition algorithms from the mathematics and computer science literature are introduced, which obtain the components $F(x)$ and $G(x)$ when the polynomial $H(x)$ is indeed a composition, i.e., $H(x) = F(G(x))$. These algorithms focus on obtaining the decomposition factor $G(x)$ first since $F(x)$ can be obtained relatively easily from the linear relationship given in equation (3.5) once $G(x)$ is known.

3.2.1 Barton-Zippel algorithm

One of the earliest attempts to find a polynomial decomposition algorithm was proposed by Barton and Zippel [6, 7], motivated by the search for an algorithm that allows expressing polynomials in terms of compositions of low order polynomials for efficient root finding and symbolic computations. Following the work in Fried and MacRae [20], they showed that a polynomial $H(x)$ has another polynomial $G(x)$ as a decomposition factor if and only if the bivariate polynomial $\phi_G(y, z)$ divides $\phi_H(y, z)$ with no remainder resulting in another bivariate polynomial in y and z. Here, the bivariate polynomial $\phi_A(y,z)$ for a K-th order polynomial $A(x)$ is defined as

$$\phi_A(y,z) \triangleq \frac{A(y)-A(z)}{y-z} = \sum_{k=1}^{K} a_k \frac{y^k - z^k}{y-z} = \sum_{k=1}^{K} a_k \left(\sum_{k'=0}^{k-1} y^{k-k'-1} z^{k'} \right). \tag{3.6}$$

These bivariate polynomials have a specific symmetry in their coefficients, namely the terms that have the same total order p of the variables y and z also have the same coefficients a_{p+1}.

It is relatively straightforward to show that if $H(x) = F(G(x))$, then $\phi_G(y,z)$ has to divide $\phi_H(y,z)$ with no remainder. More specifically,

$$\phi_H(y,z) = \frac{H(y)-H(z)}{y-z} = \sum_{n=1}^{M} f_n \frac{G^n(y)-G^n(z)}{y-z}, \tag{3.7}$$

where $\phi_G(y,z)$ can be factored out from the summation, therefore divisibility is guaranteed.

The algorithm that was developed by Barton and Zippel [6] takes as input the polynomial to be decomposed, namely $H(x)$; obtains the bivariate polynomial $\phi_H(y,z)$ and examines all of its factors to find a factor that has the form of $\phi_G(y,z)$. The requirement to examine all combinations of the factors to obtain a factor of the form $\phi_G(y,z)$ makes this algorithm computationally inefficient since the number of combinations to be examined is exponential in the number of factors.

3.2.2 Alagar-Thanh algorithm

This algorithm proposed in [2] uses the fact that the derivative of a decomposable polynomial $H(x)$ of the form in equation (3.2) has $G'(x)$ as one of its factors. Specifically,

$$H'(x) = F'(G(x))G'(x).$$

The algorithm focuses only on the factors of the univariate polynomial $H'(x)$ with appropriate orders since the order of $G'(x)$ is restricted to be one less than a factor of the order of $H(x)$. Every factor of $H'(x)$ is integrated to obtain a candidate for

$G(x)$ where the choice for the integration constant does not affect the next step, namely the computation of $\phi_G(y,z)$. The polynomial $G(x)$ is a valid decomposition factor if $\phi_G(y,z)$ divides $\phi_H(y,z)$. Since the polynomial factorization is performed on a univariate polynomial rather than a bivariate one, this is a more efficient method than the Barton-Zippel algorithm. However,the requirement to examine each possible factor remains as an undesirable computational burden.

3.2.3 Kozen-Landau algorithm

A more systematic polynomial decomposition algorithm than the previous two algorithms is given by Kozen and Landau [30], which has the complexity $\mathcal{O}(P^2)$, where P is the order of $H(x)$. M and N, the orders of $F(x)$ and $G(x)$ respectively, are required as part of the input. The algorithm uses the fact that the coefficients of N highest order terms in $H(x)$ are determined only by f_M, namely the coefficient of the highest order term in $F(x)$, and all the coefficients of $G(x)$. This can be seen from the matrix equation (3.5) since the last N entries in every column of matrix $\mathbf{C}$ except its last column consist of zeros, and only f_M among the coefficients of $F(x)$ multiplies this last column to yield the N highest order terms in $H(x)$.

As the first step of the decomposition, $H(x)$ is scaled to be monic, i.e. to have unity as the coefficient of the highest order term, an operation that does not affect decomposability. Restricting $G(x)$ and $F(x)$ to be also monic, the coefficients of $G(x)$ is obtained in the order of decreasing powers through solving N equations iteratively involving the coefficients of $G(x)$, a straightforward and well-outlined procedure described in [30]. After the decomposition is obtained for the monic polynomial, the scaling is undone.

Although this algorithm requires the orders M and N of the decomposition components as the input, it is computationally much more efficient than the previous algorithms. If the information for orders is not available a priori, the algorithm is required to run more than once, however only for orders M and N the product of which equals the order of $H(x)$. This algorithm along with the Aubry-Valibouze algorithm that will be discussed next are mainly used for later simulations since they are more systematic compared to the other algorithms as well as having lower computational complexity.

3.2.4 Aubry-Valibouze Algorithm

A different decomposition algorithm proposed by Aubry and Valibouze [5] utilizes the relationship between the coefficients of a polynomial and the power sum of its roots known as the Newton identities. More specifically, the coefficients of an N^{th} order monic polynomial $G(x)$ can be uniquely determined from the k^{th} power sums $s_k,\ k = 1, \dots N$, of its roots defined as

$$s_k = \sum_{i=1}^{N} r_{g,i}^k, \tag{3.8}$$

where $r_{g,i}\ i = 1, \dots N$, are the roots of $G(x)$. The Newton identities relate the polynomial coefficients g_n and the power sums s_k as in [27]

$$s_k + g_{n-1}s_{k-1} + \cdots + g_{n-k+1}s_1 = -kg_{n-k} \quad \text{for } 1 \leq k \leq n. \tag{3.9}$$

In other words, there is a one-to-one linear relationship between the coefficients of the n highest order coefficients and power sums of roots $s_k,\ k = 1, \dots, n$ for a polynomial.

The Aubry-Valibouze algorithm first normalizes the polynomial $H(x)$ to make it monic as in Kozen and Landau's algorithm. Since a polynomial $F(x)$ of order M can be written in terms of its roots as

$$F(x) = \prod_{j=1}^{M}(x - r_{f,j}) \tag{3.10}$$

where $r_{f,j}\ j = 1 \dots M$, are the roots of $F(x)$, a decomposable polynomial $H(x) = F(G(x))$ can be written as

$$H(x) = \prod_{j=1}^{M}(G(x) - r_{f,j}) \triangleq \prod_{j=1}^{M}\tilde{G}_j(x). \tag{3.11}$$

Each polynomial $\tilde{G}_j(x),\ j = 1 \dots M$, has the same coefficients as $G(x)$ except the constant term. By the one-to-one relationship implied by the Newton identities, all $\tilde{G}_j(x)$ have identical power sums of roots for the powers $k = 1, \dots, N$. Moreover, the roots of $\tilde{G}_j(x)$ are also the roots of $H(x)$, therefore the power sums of roots for any $\tilde{G}_j(x)$ can be computed by dividing that of $H(x)$ by M. The Newton identities can be used again to find the coefficient of $G(x)$ except its constant term using these power sums, and the constant term g_0 can be computed from $F(x)$ and $H(x)$. Finally the normalization to make $H(x)$ monic is undone.

3.2.5 Comparison of Exact Decomposition Methods

Due to their computational complexity and the existence of more systematic methods, the Barton-Zippel algorithm and the Alagar-Thanh algorithm are not included in the comparison of exact decomposition algorithms. Both the Kozen-Landau algorithm [30] and the Aubry-Valibouze's algorithm [5] are based on using the coefficients of N highest order terms in $H(x)$. Moreover, both algorithms are accurate and similar in performance for low order decompositions. For example, consider the composition of the 4-th order polynomial

$$F(x) = 0.3603 + 0.5697x - 0.3764x^2 - 0.9914x^3 + x^4 \tag{3.12}$$

and the 3-rd order polynomial

$$G(x) = -0.2921 + 0.6488x + 0.02611x^2 + x^3, \tag{3.13}$$

which yields the 12-th order polynomial

$$\begin{aligned} H(x) = 0.1937 + 0.2830x + 0.4341x^2 - 0.1198x^3 + 1.4098x^4 - 2.6497x^5 + 1.8790x^6 \\ - 4.0765x^7 + 2.3622x^8 - 1.9564x^9 + 2.5995x^{10} + 0.1044x^{11} + x^{12}. \end{aligned} \tag{3.14}$$

The Kozen-Landau and Aubry-Valibouze algorithms both recover $F(x)$ and $G(x)$ in equations (3.12) and (3.12) successfully when $H(x)$ as well as the orders $M = 4$ and $N = 3$ are provided as the input to these algorithms.

Due to representation of polynomial coefficients and their manipulations with finite precision, the performance of both algorithms deteriorates with increasing polynomial orders. On the other hand, computation of k^{th} power sums in equation (3.8) can be performed using directly the roots of $H(x)$ when they are provided rather than computing them from its coefficients in the implementation of the Aubry-Valibouze algorithm, which leads to significantly enhanced precision for the decomposition factors $G(x)$ and $F(x)$.

Figure 3-1 shows a comparison of the performance of three algorithms, namely the Kozen-Landau algorithm and the Aubry-Valibouze algorithm implemented two different ways: one using coefficients and the other using roots of $H(x)$. The polynomials $H(x)$ were obtained by composing random polynomials $F(x)$ and $G(x)$ with the coefficients of the highest order fixed to be unity to avoid degenerate cases and where the respective orders M and N are chosen equal and varied from 5 to 75 with increments of five. The decomposition is considered successful if the SNR is more than 80dB, where the error is defined as the energy in the difference between the true and the obtained decomposition factors. Both algorithms show an almost identical success rate since they use the same coefficients of $H(x)$ to determine $G(x)$ whereas the implementation of the Aubry-Valibouze algorithm using the roots of $H(x)$ directly outperforms the others significantly. For all of the polynomials of order 1600, $G(x)$ was successfully determined by this algorithm while this number dropped to 79 out of 100 for polynomials of order 4900. Once $G(x)$ is computed, $F(x)$ is also determined using the linear relationship in equation (3.4). However, the matrix inversion using finite precision deteriorates the success rate for $F(x)$ as the order of the polynomials increase. The Aubry-Valibouze algorithm manages to successfully obtain $F(x)$ up to higher orders by utilizing the relationship in equation (3.11) to find the roots of $F(x)$ and construct its coefficients when the roots of $H(x)$ are provided. The improved performance using the Aubry-Valibouze algorithm with the roots of $H(x)$ suggests that this algorithm may be more useful to use in a signal processing context when representations of signals and systems with poles and zeros are provided.

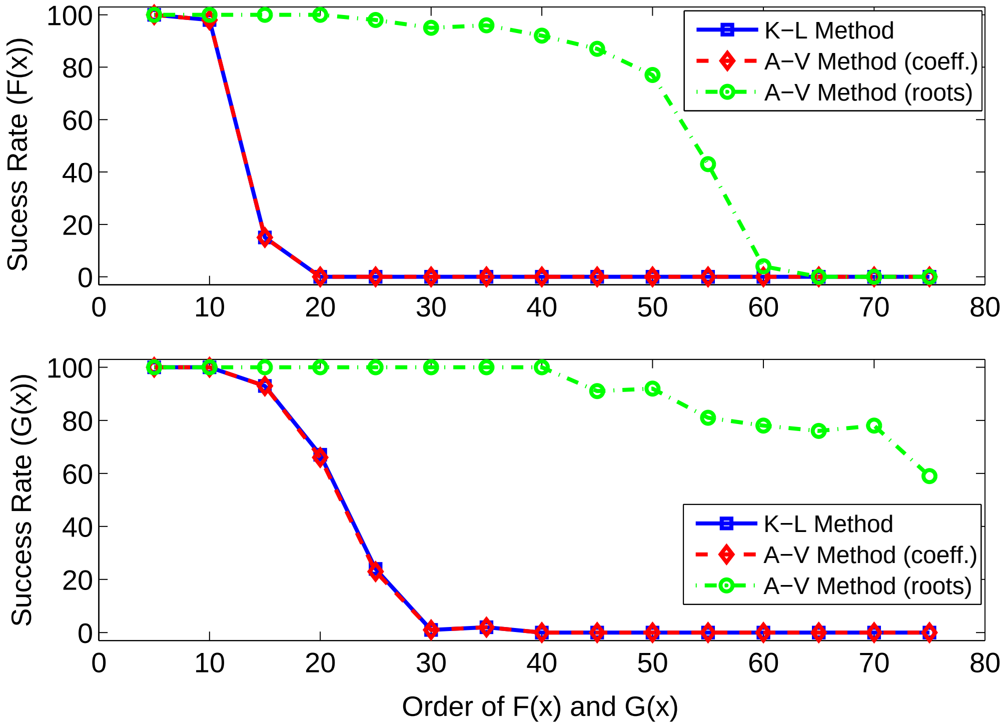


Figure 3-1: The comparison of number of successful decompositions of $H(x) = F \circ G(x)$ by the Kozen-Landau algorithm and the Aubry-Valibouze algorithm, where the latter is implemented using both coefficients and the roots of input polynomials $H(x)$.

3.3 Current Approximate Decomposition Methods

Section 3.2 focused on algorithms for obtaining the decomposition factors when it is known that a given polynomial is decomposable. Approximating non-decomposable polynomials by decomposable ones can be also of significant interest, particularly in applications such as signal representation and compression because of the inherent reduction in the number of free parameters. This section starts with the introduction of a mathematical tool, namely the Ruppert matrices, that has been exploited in some of the existing approximate polynomial decomposition algorithms. Following that, three approximate decomposition algorithms are introduced from the current literature, which can be viewed as an extension of the exact decomposition algorithms given in Section 3.2.

3.3.1 Ruppert Matrices

Barton and Zippel's exact polynomial decomposition algorithm was based on the fact that a polynomial $H(x)$ has another polynomial $G(x)$ as a decomposition factor if and only if the bivariate polynomial $\phi_G(y, z)$ divides $\phi_H(y, z)$. This required examining all factors of $\phi_H(y, z)$ to find one of the form $\phi_G(y, z)$ for potential decomposability. This examination was replaced by a stronger statement that required testing only the factorability of $\phi_H(y, z)$ ([21] Theorem 1, [53] Theorem 1). More specifically, a given polynomial $H(x)$ with a non-prime order is decomposable if and only if $\phi_H(y, z)$ is factorable with no restrictions on the form of the factors. This relationship between decomposability of a low dimensional function and the factorability of a higher dimensional function will also appear in the context of multivariate functions, suggesting a possibly deeper connection between these two properties of functions. This more general result for polynomials leads to a decomposability test which involves only checking the rank of a matrix constructed from its coefficients, namely the Ruppert matrix, which will be discussed in this section. The Ruppert matrix is a convenient tool that is exploited in the approximate decomposition algorithms described in Sections 3.3.3 and 3.3.4.

Factorability of $\phi_H(y, z)$ can be determined using a particular test for bivariate polynomial factorization that was introduced by Ruppert [47]. Specifically, $\phi_H(y, z)$ is factorable if and only if there exist two bivariate polynomials $r(y, z)$ and $s(y, z)$ that are not identically zero with $deg_{(y,z)}(r) \leq (P-2, P-1)$ and $deg_{(y,z)}(s) \leq (P-1, P-3)$ such that

$$\frac{\partial r}{\partial z}\phi_H - r\frac{\partial \phi_H}{\partial z} - \frac{\partial s}{\partial y}\phi_H + s\frac{\partial \phi_H}{\partial y} = 0, \tag{3.15}$$

where P is the order of $H(x)$. The existence of two such polynomials $r(y, z)$ and $s(y, z)$ only certifies factorability of $\phi_H(y, z)$; they are not necessarily its factors. Given $H(x)$ and therefore $\phi_H(y, z)$, the differential equation given in equation (3.15) is linear in the coefficients of $r(x, y)$ and $s(x, y)$. Therefore it can be rewritten as

$$\mathbf{Ru} = \mathbf{0}, \tag{3.16}$$

where $\mathbf{R}$ is a $(4P^2 - 10P + 6) \times (2P^2 - 3P)$ matrix the entries of which are linear functions of the coefficients of $H(x)$ and which is referred to as the Ruppert matrix of $H(x)$; and the vector $\mathbf{u}$ is obtained by concatenating the coefficient vectors of $r(y, z)$ and $s(y, z)$. Equation (3.16) implies that the existence of $r(y, z)$ and $s(y, z)$ that are not identically zero is equivalent to the Ruppert matrix $\mathbf{R}$ being rank deficient. More specifically, if $\mathbf{R}$ is rank deficient, then $\phi_H(y, z)$ is factorable and $H(x)$ is decomposable. The reverse statement is also true since all relationships are both necessary and sufficient.

The full-rank Ruppert matrix of a non-decomposable polynomial can provide additional information in addition to its non-decomposability. If $H(x)$ is a non-decomposable polynomial of order P, and if $\tilde{H}(x)$ is a decomposable polynomial with order at most P and $H(0) = \tilde{H}(0)$, then

$$||H - \tilde{H}||_2 \geq \frac{\sigma_{\min,\mathbf{R}}}{P^2\sqrt{2P^2 - P}}, \tag{3.17}$$

i.e., a decomposable polynomial $\hat{H}(x)$ has to be at least at a certain distance from a non-decomposable polynomial $H(x)$, where this distance is determined by the smallest singular value $\sigma_{\min,\mathbf{R}}$ of the Ruppert matrix [24, 28]. This associates a radius of non-decomposability with every non-decomposable polynomial $H(x)$, meaning that all polynomials within this distance to $H(x)$ are also non-decomposable.

The linearity of equation (3.15) in the coefficients of $\phi_H(y, z)$ allows rewriting the Ruppert matrix as the linear combination of a basis for Ruppert matrices [24]. More specifically, by defining a linear operator $\mathcal{R}_P$ that takes a univariate polynomial of order P as an argument and returns its Ruppert matrix, the computation of the

Ruppert matrix for $H(x)$ can be performed as

$$\mathbf{R} = \mathcal{R}_P\{H(x)\} = \mathcal{R}_P\{\sum_{i=0}^{P} h_i x^i\} = \sum_{i=1}^{P} h_i \mathcal{R}_P\{x^i\} \triangleq \sum_{i=1}^{P} h_i \mathbf{R_i}, \tag{3.18}$$

where the matrices $\mathbf{R_i},\ i = 1 \dots P$ are the Ruppert matrices for the monomials x^i computed by treating them as a P-th order polynomial $0x^P + x^i$ to yield the same size as $\mathbf{R}$. $\mathbf{R_i},\ i = 1 \dots P$ can be considered to be a basis for Ruppert matrices of all polynomials of order P, where the weight of each basis matrix is the corresponding polynomial coefficient h_i. The Ruppert matrix corresponding to the constant term in a polynomial consists of only zeros and thus it is not required in the basis. The formulation of the Ruppert matrix as in equation (3.18) will provide a basis for the approximate polynomial decomposition technique discussed in Section 3.3.4.

3.3.2 Iterative Approximate Decomposition

Corless et al [16] proposed an approximate decomposition method that starts from an initial guess for the decomposition factors $F(x)$ and $G(x)$, which are obtained using Kozen-Landau algorithm in Section 3.2, and iteratively obtain a nearby decomposable polynomial, where proximity is given in l_2 norm. The algorithm determines $\Delta G(x)$ at each iteration to minimize

$$\begin{aligned} &\|H(x) - F_k(G_k(x) + \Delta G(x))\| \\ &\approx \|H(x) - F_k(G_k(x)) - F_k'(G_k(x))\Delta G(x)\|, \end{aligned} \tag{3.19}$$

where the subscript k represents the current iteration step, $\|\cdot\|$ denotes the l_2 norm of polynomial coefficient vectors and only the first term in the Taylor series is taken into account since the coefficients of ΔG are assumed to be small at each iteration. Until the change $\Delta G(x)$ is below a certain threshold, $G_{k+1}(x)$ is obtained by updating $G_k(x)$ with $\Delta G(x)$ and $F_{k+1}(x)$ is evaluated by solving equation (3.5). This algorithm approximates a nonlinear optimization problem with a simpler one and if it converges, the convergence rate is linear. They also proposed a second algorithm that attempts to solve the nonlinear problem of minimizing $\|H(x)-F(G(x))\|$ directly using Newton iterations where $F(x)$ and $G(x)$ are perturbed together and convergence is quadratic at the expense of increased computational complexity. The quality of the decomposition obtained by these algorithms is highly dependent on the validity of the assumption that there is a decomposable polynomial close to $H(x)$ since the initial guess for $G(x)$ is obtained through an exact decomposition algorithm, namely the Kozen-Landau algorithm. Moreover, the quality of the initial guess is highly sensitive to the perturbation on the N highest order terms in $H(x)$ since Kozen-Landau algorithm uses these coefficients.

3.3.3 Decomposition by Approximate Factorization

Giesbrecht and May [24] exploited the relationship between the decomposability of a univariate polynomial $H(x)$ and factorability of the associated bivariate polynomial $\phi_H(y,z)$ as discussed in Section 3.3.1 in order to extend Barton and Zippel's exact decomposition algorithm to the case of approximate decomposition. In the case where $H(x)$ is not decomposable, $\phi_H(y,z)$ is not factorable and there is no guarantee of obtaining an approximate factor of the form $\phi_G(y,z)$ to find an approximate decomposition factor $G(x)$. Therefore an approximate factorization for $\phi_H(y,z)$ is obtained by the method described in [22] using the Ruppert matrix of $H(x)$. Each approximate factor of $\phi_H(y,z)$ is examined to determine the one closest to the form of a polynomial $\phi_G(y,z)$, namely a bivariate polynomial in which the terms with equal total order for y and z have the same coefficients. This is accomplished by computing the standard deviation of terms with the same total order, setting the maximum of these as the distance to a candidate $\phi_G(y,z)$ and choosing the factor with smallest distance. $G(x)$ and $F(x)$ can be obtained easily as before once $\phi_G(y,z)$ is known. A disadvantage of this algorithm is that it uses the result of the approximate factorization step and performs another approximation to find $\phi_G(y,z)$, which complicates the estimation of the quality of the obtained decomposition.

3.3.4 Decomposition by Riemannian SVD

As an alternative method for approximate decomposition of a polynomial $H(x)$, Botting [11] proposed a solution to the problem of finding a rank deficient Ruppert matrix the corresponding polynomial of which is close to $H(x)$, where distance is quantified as the l_2 norm of coefficient vector differences. The equivalence of decomposability of a polynomial and rank deficiency of its Ruppert matrix was established in Section 3.3.1. Specifically, the approximate decomposition problem was reduced to the optimization problem [17] specified as

$$\begin{aligned} &\underset{\tilde{h}_i, \mathbf{w}}{\text{minimize}} \quad \sum_{i=1}^{P} (\tilde{h}_i - h_i)^2 \\ &\text{subject to} \quad \tilde{\mathbf{R}}\mathbf{w} = 0 \quad \text{and} \quad \mathbf{w}^{\mathbf{T}}\mathbf{w} = 1 \end{aligned} \tag{3.20}$$

where $\tilde{h}_i,\ i = 1, \ldots, P$, are the coefficients of a decomposable polynomial $\tilde{H}(x)$ and $\tilde{\mathbf{R}}$ is its Ruppert matrix given by

$$\tilde{\mathbf{R}} = \sum_{i=1}^{P} \tilde{h}_i \mathbf{R_i} \tag{3.21}$$

as defined in equation (3.18). The first constraint corresponds to rank deficiency of $\tilde{\mathbf{R}}$ and the second constraint ensures that $\mathbf{w}$ is not identically zero preventing $\tilde{\mathbf{R}}$ from having a nontrivial null space. The optimization problem (3.20) is shown to be equivalent to a nonlinear generalized singular value decomposition referred to as Riemannian SVD in [17]. This corresponds to finding the triplet $(\mathbf{u}, \tau, \mathbf{v})$ corresponding to the smallest scalar τ that satisfies

$$\begin{aligned} \mathbf{R}\mathbf{v} &= \mathbf{D_v}\mathbf{u}\tau, & \mathbf{u^T}\mathbf{D_v}\mathbf{u} &= 1 \\ \mathbf{R^T}\mathbf{u} &= \mathbf{D_u}\mathbf{v}\tau, & \mathbf{v^T}\mathbf{D_u}\mathbf{v} &= 1 \end{aligned} \tag{3.22}$$

where $\mathbf{D_u}$ and $\mathbf{D_v}$ are matrices with entries quadratic in the vectors $\mathbf{u}$ and $\mathbf{v}$; and also a heuristic iterative solution is provided leading to a polynomial with a rank deficient Ruppert matrix and coefficients $\tilde{h}_i = h_i - \mathbf{u^T}\mathbf{R_i}\mathbf{v}\tau$. The iterations end when the smallest singular value of $\tilde{\mathbf{R}}$ becomes smaller than a given threshold, however no theoretical guarantee for convergence exists.

3.4 Approximate Decomposition based on STLN

In this section, a new approximate polynomial decomposition algorithm will be formulated. This algorithm will also exploit the Ruppert matrices in order to find a decomposable approximation to a non-decomposable polynomial by approximating its full rank Ruppert matrix with a rank-deficient one. The structure of a Ruppert matrix must be preserved while finding an approximation, hence the Structured Total Least Norm (STLN) method will be used [45].

In Section 3.4.1, the definition of STLN will be given as an extension to the total least squares (TLS) formulations. Section 3.4.2 develops the approximate polynomial decomposition based on STLN. This is followed by the comparison of certain approximate decomposition methods in Section 3.4.3.

3.4.1 Structured Total Least Norm

Given an overdetermined and nonconsistent set of linear equations

$$\mathbf{A}\mathbf{x} \approx \mathbf{b}, \tag{3.23}$$

the solution that minimizes $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ is given by the well known least squares solution. This solution leads to $\mathbf{A}\mathbf{x} = \mathbf{b} + \mathbf{\Delta b}$, i.e. only the entries of $\mathbf{b}$ are altered to satisfy the equation and $\mathbf{A}$ remains intact. Total least squares (TLS) is a generalization of this problem where the entries of $\mathbf{A}$ are also subject to possible change. More specifically the Frobenius norm of the matrix $[\mathbf{\Delta A}|\mathbf{\Delta b}]$ is minimized such that

$$(\mathbf{A} + \mathbf{\Delta A})\mathbf{x} = \mathbf{b} + \mathbf{\Delta b}. \tag{3.24}$$

This is equivalent to finding the closest rank deficient matrix $[\mathbf{A} + \mathbf{\Delta A}|\mathbf{b} + \mathbf{\Delta b}]$ to $[\mathbf{A}|\mathbf{b}]$ since equation (3.24) can be expressed as

$$[\mathbf{A} + \mathbf{\Delta A}|\mathbf{b} + \mathbf{\Delta b}]\mathbf{y} = 0, \tag{3.25}$$

where $\mathbf{y} = [\mathbf{x^T}\ , -1]^T$. The solution is obtained by suppressing the smallest singular value of the matrix $[\mathbf{A}|\mathbf{b}]$, however in general $\mathbf{A}$ and $[\mathbf{A}|\mathbf{b}]$ do not retain any of their previous structures such as sparsity, the structure of a Hankel or Toeplitz matrix or the special structure of a Ruppert matrix.

Imposing a structure preserving constraint to equation (3.25) to approximately solve equation (3.23) is the basis for the collective algorithms referred to as Structured Total Least Square Norm (STLN) algorithms for a general norm [45], and reduces to Structured Total Least Squares (STLS) for the choice of l_2 norm for which a solution was proposed in [17]. In fact, the approximate decomposition algorithm proposed by Botting [11] in Section 3.3.4 is an example of an STLS problem since it formulates the optimization problem in a way to preserve the Ruppert matrix structure while minimizing an l_2 norm, however it utilizes the specialized Riemann SVD solution developed in [17]. A more general solution for structure-preserving methods using the STLN framework was provided in [45], including the l_2 norm case, which will be explored in this section for a new approximate polynomial decomposition algorithm.

3.4.2 Algorithm Development

The exploitation of structure preserving low rank approximation formulations such as STLN as described in [45] for finding a rank deficient Ruppert matrix has been suggested as a potentially useful method ([29], Remark 6), however no implementations or results were reported. In this section, the approximate polynomial decomposition problem of $H(x)$ will be expressed in terms of approximate rank deficiency of its Ruppert matrix $\mathbf{R}$, which in turn will be expressed as the problem in equation (3.25) with $\mathbf{A}$ and $\mathbf{b}$ obtained from columns of $\mathbf{R}$ appropriately. $\mathbf{\Delta A}$ and $\mathbf{\Delta b}$ obtained this way will yield the required perturbation to $\mathbf{R}$ to render it rank deficient while preserving its Ruppert matrix structure. This will also yield the perturbations required for the coefficients of the polynomial $H(x)$ to make it decomposable. The developed algorithm will require solving a simple quadratic optimization problem at each iteration step. The matrices used in the iterative algorithm given in [45] are also modified here in order to minimize the perturbation on the polynomial coefficients as opposed to unnecessarily minimizing the total perturbation on the entries of its Ruppert matrix, which are not necessarily equivalent.

The rank deficiency of the Ruppert matrix can be imposed by setting one of its columns as a linear combination of other columns. In order to formulate this problem as in equation (3.25), the column vector $\mathbf{b}$ is chosen as one of the columns of the Ruppert matrix $\mathbf{R}$; and $\mathbf{A}$ is defined to be equal to $\mathbf{R}$ excluding the column vector $\mathbf{b}$. This specific column can be chosen as the one that minimizes the residual when expressed in terms of other columns, i.e.

$$\mathbf{b} = \arg\min_{\mathbf{b}} \left(\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2 \right), \tag{3.26}$$

where $\mathbf{x}$ represents a column of $\mathbf{R}$. The minimum value inside the parenthesis is ideally zero, which would correspond to an already rank deficient Ruppert matrix. Assuming $\mathbf{b}$ is chosen to be the k^{th} column in $\mathbf{R}$, one can define the vectors $\mathbf{b_i}$ as the k^{th} column vector of the basis Ruppert matrices $\mathbf{R_i}$ for $i = 1, \ldots, P$, where the basis Ruppert matrices $\mathbf{R_i}$ were defined in equation (3.18). Similarly the matrices $\mathbf{A_i}$ can be defined to be equal to $\mathbf{R_i}$ excluding the column $\mathbf{b_i}$. If $[\mathbf{\Delta A}|\mathbf{\Delta b}]$ is constrained to be of the form

$$\sum_{i=0}^{P} \alpha_i [\mathbf{A_i}|\mathbf{b_i}] \tag{3.27}$$

for real scalars $\alpha_i,\ i = 1 \ldots P$, the resulting matrix

$$[\mathbf{A} + \mathbf{\Delta A}|\mathbf{b} + \mathbf{\Delta b}]$$

in equation (3.25) will retain the same matrix structure as $[\mathbf{A}|\mathbf{b}]$ due to the linear relationship given in equation (3.18). The scalars α_i correspond to the perturbation in the coefficients h_i of the polynomial $H(x)$. The approximate decomposition algorithm can be formulated as an optimization problem in which the change in the coefficients of the polynomial to be decomposed are minimized subject to the rank deficiency constraint in equation (3.25), i.e.

$$\begin{aligned} &\underset{\alpha_i,\mathbf{y}}{\text{minimize}} \quad \sum_{i=1}^{P} \alpha_i^2 \\ &\text{subject to} \quad [\mathbf{A}+\mathbf{\Delta A}|\mathbf{b}+\mathbf{\Delta b}]\mathbf{y}=0. \end{aligned} \tag{3.28}$$

The similarity of the optimization problems given in equations (3.20) and (3.28) is obvious since we chose to minimize the l_2 norm of the coefficient perturbation vector $\boldsymbol{\alpha} = [\alpha_1 \ \alpha_2 \ \ldots \ \alpha_P]^T$. An explicit constraint for $\mathbf{y}$ to be nonzero is not required in equation (3.28) since it is already restricted to be of the form $\mathbf{y} = [\mathbf{x^T} \ , -1]^T$ with this formulation, which cannot be identically zero.

In order to solve the nonlinear and nonconvex optimization problem in equation (3.28), the following relaxation is considered,

$$\underset{\alpha_i,\mathbf{y}}{\text{minimize}} \quad \boldsymbol{\alpha^T}\boldsymbol{\alpha} + \lambda^2 \hat{\mathbf{r}}^{\mathbf{T}}\hat{\mathbf{r}} \tag{3.29}$$

where

$$\hat{\mathbf{r}} \overset{\Delta}{=} [\mathbf{A}+\mathbf{\Delta A}|\mathbf{b}+\mathbf{\Delta b}]\mathbf{y} \tag{3.30}$$

and λ is the penalty parameter for any nonzero residual $\hat{\mathbf{r}}$ and is required to be chosen appropriately large for a good approximation to the original problem. An iterative algorithm for the solution of a nonlinear optimization problem of the form (3.29) is given in [45]. Setting $\mathbf{y} = [\mathbf{x^T} \ , -1]^T$ in equation (3.30) yields

$$\begin{aligned} \hat{\mathbf{r}} &= \mathbf{Ax}+\mathbf{\Delta Ax}-\mathbf{b}-\mathbf{\Delta b} \\ &= \mathbf{Ax}+\mathbf{X}\boldsymbol{\alpha}-\mathbf{b}-\mathbf{Q}\boldsymbol{\alpha} = \mathbf{Ax}+\mathbf{K}\boldsymbol{\alpha}-\mathbf{b} \end{aligned} \tag{3.31}$$

where the matrix $\mathbf{K} = \mathbf{X} - \mathbf{Q}$ and the matrix $\mathbf{X}$ is defined by

$$\boldsymbol{\Delta}\mathbf{A}\mathbf{x} = \sum_{\mathbf{i=1}}^{\mathbf{P}} \alpha_{\mathbf{i}} \mathbf{A_i x} = \sum_{\mathbf{i=1}}^{\mathbf{P}} (\mathbf{A_i x})\alpha_{\mathbf{i}} \triangleq \mathbf{X}\boldsymbol{\alpha}. \tag{3.32}$$

More specifically the i^{th} column of $\mathbf{X}$ consists of $\mathbf{A_i x}$. Similarly, i^{th} column of $\mathbf{Q}$ consists of $\mathbf{b_i}$. A heuristic value for the penalty parameter λ that proved to yield reasonable results was the reciprocal of the minimum singular value of $\mathbf{K}$, namely $\frac{1}{\sigma_{\mathbf{K},min}}$. The steps of the iteration are summarized in Algorithm 1.

```
                    ALGORITHM 1
    Input: $H(x)$ with coefficients $h_i,\ i = 1, \dots, P$.
    Output: A rank deficient Ruppert matrix $\tilde{\mathbf{R}}$ corresponding
    to a polynomial $\tilde{H}(x)$ with coefficients $\tilde{h}_i$ close to that of $H(x)$.
Begin
    Set $\mathbf{k} = 1$. Specify $\mathbf{A}, \mathbf{b}$ from $\mathbf{R}$ as in (3.26).
    Set $\mathbf{x}^{[\mathbf{k}]} = \arg\min_{x} \|\mathbf{Ax} - \mathbf{b}\|_2$ and $\boldsymbol{\alpha}^{[\mathbf{k}]} = \mathbf{0}$.
    Obtain $\mathbf{A_i}, \mathbf{b_i}$ from $\mathbf{R_i},\ i = 1, \dots, P$.
    Set $\mathbf{Q} = [\mathbf{b_1} \mid \dots \mid \mathbf{b_P}]$.
  1. Set $\mathbf{X}^{[\mathbf{k}]} = [\mathbf{A_1 x}^{[\mathbf{k}]} \mid \dots \mid \mathbf{A_P x}^{[\mathbf{k}]}],\ \mathbf{K}^{[\mathbf{k}]} = \mathbf{X}^{[\mathbf{k}]} - \mathbf{Q},\ \hat{\mathbf{r}}^{[\mathbf{k}]} = \mathbf{Ax}^{[\mathbf{k}]} + \mathbf{K}^{[\mathbf{k}]}\boldsymbol{\alpha}^{[\mathbf{k}]} - \mathbf{b}$
    If $\mathbf{k} = 1$, set $\lambda = \dfrac{1}{\sigma_{\mathbf{K}^{[\mathbf{k}]},min}}$.
  2. Solve the following quadratic program:
    $\underset{\boldsymbol{\Delta\alpha}, \boldsymbol{\Delta}\mathbf{x}}{\text{minimize}} \left\| \begin{bmatrix} \lambda\mathbf{K}^{[\mathbf{k}]} & \lambda\mathbf{A} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \boldsymbol{\Delta\alpha} \\ \boldsymbol{\Delta}\mathbf{x} \end{pmatrix} + \begin{pmatrix} \lambda\hat{\mathbf{r}}^{[\mathbf{k}]} \\ \boldsymbol{\alpha}^{[\mathbf{k}]} \end{pmatrix} \right\|_2$
  3. Set $\mathbf{x}^{[\mathbf{k+1}]} \leftarrow \mathbf{x}^{[\mathbf{k}]} + \boldsymbol{\Delta}\mathbf{x},\ \boldsymbol{\alpha}^{[\mathbf{k+1}]} \leftarrow \boldsymbol{\alpha}^{[\mathbf{k}]} + \boldsymbol{\Delta\alpha},\ \mathbf{k} \leftarrow \mathbf{k} + 1$.
  4. If $\tilde{\mathbf{R}}^{[\mathbf{k}]} = \mathbf{R} + \sum_{i=1}^{P} \boldsymbol{\alpha}_i^{[\mathbf{k}]}\mathbf{R_i}$ is rank deficient, then $\tilde{h}_i = h_i + \boldsymbol{\alpha}_i^{[\mathbf{k}]}$ for $i = 1, \dots, P$, exit.
    If not, go to Step 1.
  End
```

3.4.3 Comparison of Approximate Decomposition Methods

In this section, the performance of the two most recent approximate decomposition methods based on the use of Ruppert matrices are compared first, namely the Riemann SVD (RiSVD) formulation summarized in Section 3.3.4 and the Structured Total Least Norm (STLN) formulation proposed in Section 3.4.2. Afterwards, they are both compared to the iterative approximate decomposition method discussed in Section 3.3.2, which will also be referred to as Corless' method.

One hundred decomposable polynomials $H(x)$ were obtained by composing an M^{th}-order random polynomial $F(x)$ with an N^{th}-order random polynomial $G(x)$, the coefficients of both of which were selected from a standard normal distribution except the highest order terms that are fixed to be unity to avoid degenerate compositions. The coefficient vector $\mathbf{h}$ of each polynomial $H(x)$ is then perturbed by an error vector $\mathbf{e}$ to obtain a nondecomposable polynomial $\hat{H}(x)$, where the coefficients of $\mathbf{e}$ are also obtained from a standard normal distribution and scaled so that $\|\mathbf{e}\|_2 = 10^{-2}\|\mathbf{h}\|_2$, i.e. the SNR is 40dB.

The iterations in both RiSVD and STLN methods were stopped when the Ruppert matrix is considered numerically rank deficient, where this is defined as the existence of a significantly large ratio between any two consecutive singular values among the smallest twenty singular values of the Ruppert matrix. More specifically, the Ruppert matrix is considered to be rank deficient when the maximum ratio between consecutive singular values are greater than one hundred times that of the original Ruppert matrix or 10^4, whichever is smaller.

Table 3.1 summarizes the results of the simulations for the STLN and RiSVD methods tested against non-decomposable polynomials of different orders. The decomposition success rates are calculated as the ratio of the number of cases where the ending criterion was met before one hundred iterations to the total number of polynomials that did not have numerically rank deficient Ruppert matrices at the initial stage. For the set of polynomials and the stopping criteria chosen, the STLN method proves to be more successful than the RiSVD method for all orders.

Table 3.1: Success Rates for RiSVD and STLN based Approximate Decomposition Methods (%)

$deg(F)$	$deg(G)$	$deg(F \circ G)$	STLN	RiSVD
2	2	4	100.0	73.0
2	3	6	97.0	2.0
3	2	6	96.0	9.0
2	4	8	92.0	5.0
4	2	8	94.8	7.3
3	3	9	86.0	5.0
2	5	10	81.0	1.0
5	2	10	90.0	10.0
2	6	12	79.0	2.0
3	4	12	83.7	12.2
4	3	12	82.2	10.0
6	2	12	95.0	11.3

Consider, for example, adding noise to the coefficients of the decomposable polynomial $H(x)$ in equation (3.14) to obtain a non-decomposable polynomial $\hat{H}(x)$ such that the SNR is 40dB. The application of STLN and RiSVD algorithms to $\hat{H}(x)$ yields the approximations given in the third and fourth columns in Table 3.2. Both of these polynomials are very close to each other as well as to $\hat{H}(x)$ in their coefficients. Figure 3-2 is the plot of the twenty smallest singular values of the three Ruppert matrices corresponding to the nondecomposable polynomial $\hat{H}(x)$ and its approximations obtained using RiSVD and STLN, where the singular values are depicted in the decreasing order for each matrix. Although the coefficient vectors of all three polynomials are very close with respect to the l_2 norm, only the polynomial that is the output of the STLN leads to a numerically rank deficient matrix in this example as the ratio between its two smallest singular values is large, more specifically 3.5×10^4.

The polynomials obtained by STLN and RiSVD methods that are numerically deemed as a decomposable approximation to $\hat{H}(x)$ may not always lead to a faithful decomposition when they are used as the input to an exact decomposition algorithm, such as the Kozen-Landau algorithm, to actually find a decomposition $F(G(x))$. This is indeed the case for H_{STLN} in this example. The fact that the radius of non-

Table 3.2: Coefficients of H in (3.14) in increasing order from top to bottom, $\hat{H}$ obtained by perturbing them with 40dB noise, and the resulting coefficients after the application of RiSVD, STLN and Corless' methods to $\hat{H}$.

H	$\hat{H}$	H_{RiSVD}	H_{STLN}	$H_{Corless}$
0.19372869	0.17480287	0.17480287	0.17480287	0.17567301
0.28302078	0.29436839	0.29246959	0.29477573	0.29724135
0.43414745	0.45491202	0.38919792	0.43389074	0.42877466
−0.11975589	−0.10554232	−0.17286798	−0.10524759	−0.09980779
1.40982439	1.39074151	1.39984861	1.40067728	1.38490538
−2.64965555	−2.6362043	−2.64909473	−2.65041201	−2.64146584
1.87902027	1.86431271	1.77910056	1.86102616	1.84951314
−4.07645854	−4.07629235	−4.06771098	−4.06253602	−4.04778661
2.36218459	2.34961771	2.3642983	2.33975812	2.34035946
−1.95636998	−1.9324207	−2.01021235	−1.94051827	−1.93297813
2.59948389	2.55459276	2.61287623	2.56956633	2.56956643
0.10442734	0.10923451	0.1423575	0.10494564	0.10494761
1.00000000	0.99627832	0.99627832	0.98489721	0.98489877

decomposability associated with every non-decomposable polynomial as specified in equation (3.17) is only a lower bound partially accounts for this observation. Other possible reasons are that the rank of the Ruppert matrix may itself be very sensitive to the polynomial coefficients, or the performance of the exact decomposition methods are extremely sensitive to the exactness of the coefficients of the polynomial to be decomposed.

The iterative approximate decomposition method introduced by Corless et al [16] and summarized in Section 3.3.2 suggests a decomposable approximation to a given polynomial at each step of the iteration. Therefore a decomposable approximation was obtained for all the examples shown here using Corless' algorithm. However a direct performance comparison cannot be made between this method and the STLN or RiSVD methods since Corless' method is not based on approximating Ruppert matrices. The initial guess in this algorithm is obtained using the Kozen-Landau exact decomposition algorithm that utilizes high order coefficients in $\hat{H}(x)$. For the example in Table 3.2, the last column corresponds to the decomposable approximation obtained by Corless' method, for which the approximate decomposition factors are

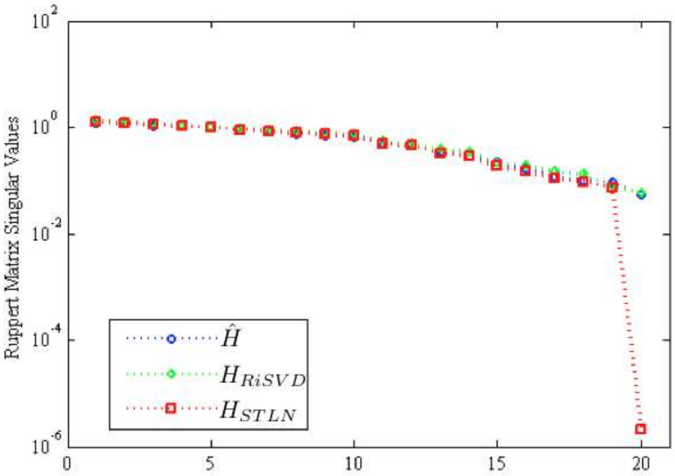


Figure 3-2: The smallest twenty singular values of the Ruppert matrices corresponding to the perturbed and nondecomposable polynomial $\hat{H}$, its approximation H_{RiSVD} as the output of the RiSVD algorithm and H_{STLN} as the output of the STLN algorithm, all shown in decreasing order. Only the Ruppert matrix of the polynomial obtained by the STLN method exhibits a large gap between its consecutive singular values, corresponding to a numerical rank deficiency.

obtained as

$$\hat{F}(x) = 0.3456 + 0.5829x - 0.3811x^2 - 0.9950x^3 + 0.9849x^4 \tag{3.33}$$

and

$$\hat{G}(x) = -0.2902 + 0.6512x + 0.0266x^2 + 1.0000x^3. \tag{3.34}$$

These polynomials constitute a very good approximation to those in equations (3.12) and (3.13). In all the examples shown in this section, the noise was evenly distributed on all coefficients of the polynomial and Corless' method yielded successful approximate decompositions with high SNR. However, the approximation is likely to be poor in cases where the highest order terms of the polynomial $\hat{H}(x)$ experience relatively more perturbation as these are used to find an initial guess as the first step of Corless' algorithm.

Approximate polynomial decomposition has in fact proved to be a very difficult nonlinear problem in both mathematics and computer science and it has been only considered for cases in which the coefficients of a given polynomial is known to be close to that of a decomposable one. Even in those cases, it remains to lack a solution which is consistently satisfactory for high polynomial orders and large or nonuniform perturbations on the coefficients. In cases where the polynomial is not known to be in the neighborhood of a decomposable polynomial, these algorithms summarized or developed in this chapter do not have any guarantees to yield an acceptable approximate decomposition. This stems from the non-convex nature of the approximate polynomial decomposition problem and the fact that the set of decomposable polynomials constitutes a relatively small subset of the space of polynomials.

3.5 Sensitivity Analysis

The focus of this section is the sensitivity of the polynomial composition and the decomposition operations. This is useful in understanding the types of signal processing applications in which these operations can be used and the extent to which they remain reliable. For example, such an analysis can suggest when a decomposable signal can be faithfully represented in terms of its components in the presence of quantization noise. Similarly, this analysis can quantify the performance of a filter designed as a generalized tapped delay line in the presence of error in the tap coefficients or the subfilters, where the generalized tapped delay lines will be described in Chapter 6.

3.5.1 Composition Sensitivity

The sensitivity of composition for a given decomposable polynomial $H(x)$ can be defined as the maximum magnification of an infitesmall perturbation Δu in its composing polynomials, i.e.

$$S_{U \to H} = \max_{\Delta \mathbf{u}} \frac{E_{\Delta \mathbf{h}}/E_{\mathbf{h}}}{E_{\Delta \mathbf{u}}/E_{\mathbf{u}}} \tag{3.35}$$

where U is either F or G depending on which is being perturbed, $E_{\mathbf{h}} = ||\mathbf{h}||_2^2$ is the energy of the coefficient vector $\mathbf{h}$, $E_{\mathbf{u}} = ||\mathbf{u}||_2^2$ is the energy of the coefficient vector $\mathbf{u}$ and $||\cdot||_2^2$ is the square of the two norm of a vector. The relative magnification in perturbation depends on the direction of the perturbation vector $\Delta\mathbf{u}$; and sensitivity is defined at the direction of maximum magnification.

Formulation of $S_{F\to H}$

Due to the linear relationship given in equation (3.5), a perturbation $\Delta\mathbf{f}$ in the coefficient vector of $F(x)$ will result in a change in the coefficients of $H(x)$ given by

$$\Delta\mathbf{h} = \mathbf{C}\Delta\mathbf{f}. \tag{3.36}$$

The sensitivity of composition with respect to $F(x)$ becomes, by equation (3.5), (3.35) and (3.36)

$$S_{F\to H} = \max_{\Delta\mathbf{f}} \frac{||\mathbf{C}\Delta\mathbf{f}||_2^2}{||\Delta\mathbf{f}||_2^2}\frac{||\mathbf{f}||_2^2}{||\mathbf{C}\mathbf{f}||_2^2}. \tag{3.37}$$

For a given decomposition of a polynomial $H(x)$ as $F \circ G(x)$, the factor $\frac{||\mathbf{f}||_2^2}{||\mathbf{C}\mathbf{f}||_2^2}$ is constant. The maximum value of $\frac{||\mathbf{C}\Delta\mathbf{f}||_2^2}{||\Delta\mathbf{f}||_2^2}$ is equal to $\sigma_{\mathbf{C},max}^2$, where $\sigma_{\mathbf{C},max}$ is the maximum singular value of $\mathbf{C}$. Therefore equation (3.37) becomes

$$S_{F\to H} = \sigma_{\mathbf{C},max}^2 \frac{||\mathbf{f}||_2^2}{||\mathbf{C}\mathbf{f}||_2^2}. \tag{3.38}$$

Furthermore, $\frac{||\mathbf{f}||_2^2}{||\mathbf{C}\mathbf{f}||_2^2}$ is bounded above by $\sigma_{\mathbf{C},min}^{-2}$ and bounded below by $\sigma_{\mathbf{C},max}^{-2}$ for any $\mathbf{f}$. Hence, regardless of $F(x)$, the sensitivity $S_{F\to H}$ satisfies

$$1 \le S_{F\to H} \le \frac{\sigma_{\mathbf{C},max}^2}{\sigma_{\mathbf{C},min}^2} \tag{3.39}$$

where $\frac{\sigma_{\mathbf{C},max}^2}{\sigma_{\mathbf{C},min}^2}$ is the square of the condition number of $\mathbf{C}$.

Formulation of $S_{G\to H}$

A perturbation Δg_k to g_k, namely the coefficient of x^k in $G(x)$, does not affect the coefficient of x^n in $H(x)$ for $k > n$. Such a perturbation results in the composition

$$\tilde{H}(x) = F(G(x) + \Delta g_k x^k) = H(x) + \Delta H(x) \tag{3.40}$$

where

$$\Delta H(x) \approx F'(G(x))\Delta g_k x^k \tag{3.41}$$

assuming Δg_k is small and only the first term in the Taylor series for equation (3.40) is considered. For $k \leq n$, equation (3.41) implies that the corresponding perturbation in h_n becomes

$$\Delta h_n = \Delta g_k d_{n-k} \tag{3.42}$$

where d_{n-k} is the coefficient of x^{n-k} in the polynomial $D(x)$ defined as

$$D(x) = F'(G(x)). \tag{3.43}$$

Perturbation of all the coefficients $g_k,\ k = 0, 1, \ldots, N$ results in the addition of error terms in equation (3.42), i.e.

$$\Delta h_n = \sum_{0 \leq k \leq \min(N,n)} \Delta g_k d_{n-k}. \tag{3.44}$$

Equivalently,

$$\Delta \mathbf{h} = \mathbf{D}\Delta \mathbf{g} \tag{3.45}$$

where $\mathbf{D}$ is an $(MN+1) \times (N+1)$ Toeplitz matrix the first column of which consists of the coefficients of the polynomial $D(x)$, namely $[d_0\ d_1\ d_2 \ldots d_{MN+1-N}]^T$, with zero padding of length N. The sensitivity of composition with respect to $G(x)$ becomes, by equations (3.35) and (3.45),

$$S_{G \rightarrow H} = \max_{\Delta \mathbf{g}} \frac{||\mathbf{D}\Delta \mathbf{g}||_2^2}{||\Delta \mathbf{g}||_2^2} \frac{||\mathbf{g}||_2^2}{||\mathbf{h}||_2^2}. \tag{3.46}$$

As in the previous section, for a given decomposition of $H(x)$ as $F \circ G(x)$, $\frac{||\mathbf{g}||_2^2}{||\mathbf{h}||_2^2}$ is constant. The maximum value of $\frac{||\mathbf{D}\Delta \mathbf{g}||_2^2}{||\Delta \mathbf{g}||_2^2}$ is $\sigma_{\mathbf{D},max}^2$, where $\sigma_{\mathbf{D},max}$ is the maximum singular value of $\mathbf{D}$. Therefore equation (3.46) becomes

$$S_{G \rightarrow H} = \sigma_{\mathbf{D},max}^2 \frac{||\mathbf{g}||_2^2}{||\mathbf{h}||_2^2}. \tag{3.47}$$

An upper bound for $S_{G\to H}$ can be obtained by an alternative representation of the coefficient vector of $D(x)$ given in equation (3.43) in the form of equation (3.5), i.e.

$$\mathbf{d} = \mathbf{C}\tilde{\mathbf{f}} = \mathbf{C}\mathbf{V}\mathbf{f} \tag{3.48}$$

where $\tilde{\mathbf{f}}$ is the coefficient vector of $F'(x)$ and $\mathbf{V}$ is the $(M+1)\times(M+1)$ matrix with superdiagonal elements $1, 2, \ldots M$ and zeros elsewhere, corresponding to the derivative operator. Due to the derivative operation, the order of the polynomial decreases by one, therefore the last element of $\tilde{\mathbf{f}}$ as well the last N elements of $\mathbf{d}$ are zero, but these are not discarded for the consistency of the sizes among matrices and multiplying vectors. For vectors $\mathbf{d}$, $\Delta\mathbf{h}$ and $\Delta\mathbf{g}$, which are related through equation (3.44), it can be shown that

$$\frac{E_{\Delta\mathbf{h}}}{E_{\Delta\mathbf{g}}} \leq (N+1)E_{\mathbf{d}}, \tag{3.49}$$

where the proof is given in Appendix A for the convolution of general sequences. Therefore, from the definition in equation (3.35), $S_{G\to H}$ can be bounded as

$$S_{G\to H} \leq (N+1)E_{\mathbf{d}}\frac{E_{\mathbf{g}}}{E_{\mathbf{h}}} = (N+1)||\mathbf{g}||_2^2\frac{||\mathbf{C}\mathbf{V}\mathbf{f}||_2^2}{||\mathbf{C}\mathbf{f}||_2^2}. \tag{3.50}$$

Defining

$$\mathbf{w} = \mathbf{C}\mathbf{f}, \tag{3.51}$$

it can be shown $\mathbf{f} = \left(\mathbf{C}^{\mathbf{T}}\mathbf{C}\right)^{\mathbf{-1}}\mathbf{C}^{\mathbf{T}}\mathbf{w}$ since $\mathbf{C}$ is full rank. Therefore equation (3.50) becomes

$$\begin{aligned} S_{G\to H} &\leq (N+1)||\mathbf{g}||_2^2\frac{||\mathbf{C}\mathbf{V}\left(\mathbf{C}^{\mathbf{T}}\mathbf{C}\right)^{\mathbf{-1}}\mathbf{C}^{\mathbf{T}}\mathbf{w}||_2^2}{||\mathbf{w}||_2^2} \\ &\leq (N+1)||\mathbf{g}||_2^2\,\sigma_{\mathbf{T},max}^2 \end{aligned} \tag{3.52}$$

where the matrix $\mathbf{T} = \mathbf{C}\mathbf{V}\left(\mathbf{C}^{\mathbf{T}}\mathbf{C}\right)^{\mathbf{-1}}\mathbf{C}^{\mathbf{T}}$ and $\sigma_{\mathbf{T},max}$ is the maximum singular value of $\mathbf{T}$.

3.5.2 Decomposition Sensitivity

Defining the sensitivity of decomposition directly as the relative magnification of perturbation in the components $F(x)$ and $G(x)$ when $H(x)$ is perturbed is not meaningful since a small perturbation $\mathbf{\Delta h}$ will render it nondecomposable in general. In other cases, $H(x)$ may remain decomposable but the new components $\hat{F}(x)$ and $\hat{G}(x)$ may have different orders than $F(x)$ and $G(x)$, respectively. These cases are excluded from a discussion regarding their sensitivity here as well since the decomposition process may be regarded as having failed by not predicting the orders of the components correctly. Consequently, the definition for sensitivity of the decomposition will be restricted to cases in which the perturbation preserves decomposability with components of the same order.

Perturbations in polynomials $F(x)$ and $G(x)$ may lead to much smaller perturbations in the coefficients of $H(x)$. For the inverse operation, this implies that decomposition under this specific perturbation in $H(x)$ will yield larger relative perturbations in $F(x)$ and $G(x)$. The sensitivity of decomposition hence can reasonably be defined as

$$S_{H\to U} = \max_{\mathbf{\Delta u}} \left(\frac{E_{\mathbf{\Delta h}}/E_{\mathbf{h}}}{E_{\mathbf{\Delta u}}/E_{\mathbf{u}}} \right)^{-1} \tag{3.53}$$

where again U is either F or G. $S_{H\to U}$ corresponds to the case where the perturbation on the components results in the direction of maximum attenuation.

Formulation of $S_{H\to F}$

The sensitivity associated with obtaining $F(x)$ from a decomposable polynomial $H(x)$ becomes, by equations (3.5), (3.53) and (3.36)

$$S_{H\to F} = \left(\min_{\mathbf{\Delta f}} \frac{||\mathbf{C\Delta f}||_2^2}{||\mathbf{\Delta f}||_2^2} \frac{||\mathbf{f}||_2^2}{||\mathbf{Cf}||_2^2} \right)^{-1} = \left(\sigma^2_{\mathbf{C},min} \frac{||\mathbf{f}||_2^2}{||\mathbf{Cf}||_2^2} \right)^{-1}. \tag{3.54}$$

Furthermore, $\left(\frac{||\mathbf{f}||_2^2}{||\mathbf{Cf}||_2^2} \right)^{-1}$ is bounded above by $\sigma^2_{\mathbf{C},max}$ and bounded below by $\sigma^2_{\mathbf{C},min}$. Hence similar to equation (3.39), for any $F(x)$, the sensitivity $S_{H\to F}$ is bounded by the square of the condition number of $\mathbf{C}$, which only depends on $G(x)$, i.e.

$$1 \leq S_{H\to F} \leq \frac{\sigma^2_{\mathbf{C},max}}{\sigma^2_{\mathbf{C},min}}. \tag{3.55}$$

Formulation of $S_{H\to G}$

The sensitivity associated with obtaining $G(x)$ from a decomposable polynomial $H(x)$ becomes, by equations (3.53) and (3.45),

$$S_{H\to G} = \left(\min_{\Delta\mathbf{g}} \frac{||\mathbf{D}\Delta\mathbf{g}||_2^2}{||\Delta\mathbf{g}||_2^2} \frac{||\mathbf{g}||_2^2}{||\mathbf{h}||_2^2} \right)^{-1} = \left(\sigma^2_{\mathbf{D},min} \frac{||\mathbf{g}||_2^2}{||\mathbf{h}||_2^2} \right)^{-1}. \tag{3.56}$$

3.5.3 Simulations

In the following subsections, several simulation results are provided to illustrate the sensitivity of the polynomial composition and decomposition operations. The coefficient vectors of the polynomials $F(x)$ and $G(x)$ were selected from the standard normal distribution by the `randn` function of MATLAB and were normalized to have unit energy. The effect of normalization and scaling will be discussed in Section 3.5.4.

Simulations for composition sensitivity

Evaluation of $S_{F\to H}$

In Section 3.5.1, $S_{F\to H}$ was shown to be bounded by the square of the condition number of $\mathbf{C}$ as given in equation (3.39) regardless of the specific value of $F(x)$. This bound is in fact attained if $\mathbf{f}$ is aligned with the right singular vector of $\mathbf{C}$ that corresponds to its smallest singular value, however for an average case the sensitivity is orders of magnitude lower than the square of the condition number as the simulations in this section suggests.

The sensitivity $S_{F\to H}$, as defined in equation (3.38), is shown in Fig. 3-3 as a function of the degree of $F(x)$. In Fig. 3-3, each point shows the median value of $S_{F\to H}$ obtained from composing one hundred instances of $F(x)$ of the corresponding order with each one of one hundred instances of $G(x)$ of order seven. The vertical

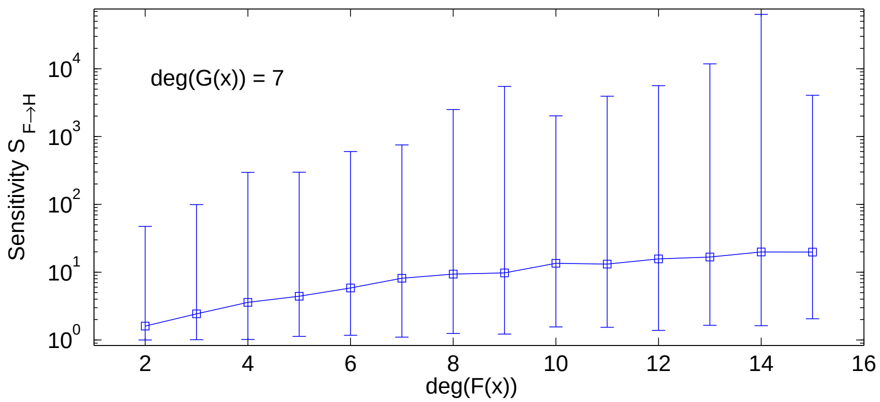


Figure 3-3: Sensitivity of the coefficients of $H(x)$ with respect to the coefficients of $F(x)$. The order of $G(x)$ is seven in all compositions. Each point is the median of $S_{F \to H}$ obtained from ten thousand compositions, where the vertical bars indicate the range from the maximum to the minimum values attained.

bars show the maximum and minimum sensitivities attained in these ten thousand compositions. For consistency, the same set of $G(x)$ were used for each degree of $F(x)$. The simulation results are consistent with the lower and upper bounds given in equation (3.39), namely 1 and the square of the condition number of $\mathbf{C}$, respectively. However the upper bound has been omitted from this figure due to very large values that exceed the display scale by multiple orders.

Evaluation of $S_{G \to H}$

The sensitivity $S_{G \to H}$, as defined in equation (3.47), is shown in Fig. 3-4 as a function of the degree of $G(x)$. In Fig. 3-4, each point indicates the median value of $S_{G \to H}$ obtained from composing one hundred instances of $G(x)$ of the corresponding order with each one of one hundred instances of $F(x)$ of order seven. The dashed line indicates the upper bound given in equation (3.52) where $||\mathbf{g}||_2^2 = 1$ and $\sigma_{\mathbf{T},max}$ is evaluated for the $G(x)$ that attains the maximum value of $S_{G \to H}$ in the simulations for each degree.

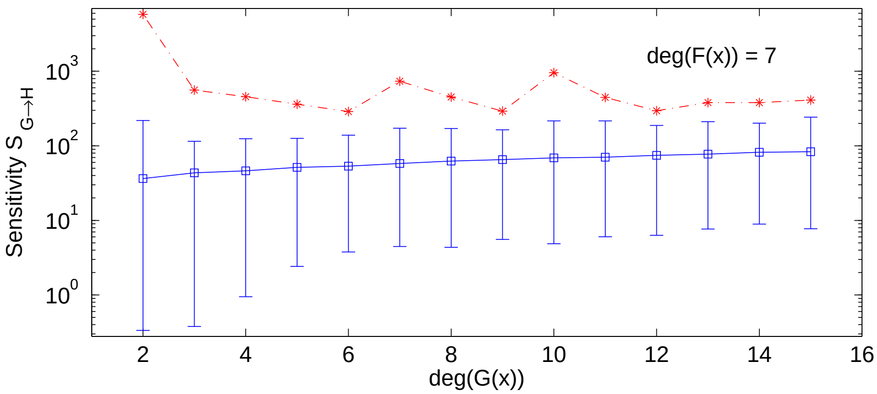


Figure 3-4: Sensitivity of the coefficients of $H(x)$ with respect to the coefficients of $G(x)$. The order of $F(x)$ is seven in all compositions. Each point is the median of $S_{G\to H}$ obtained from ten thousand compositions, where the vertical bars indicate the range from the maximum to the minimum values attained. The dashed line indicates the upper bound given in equation (3.52).

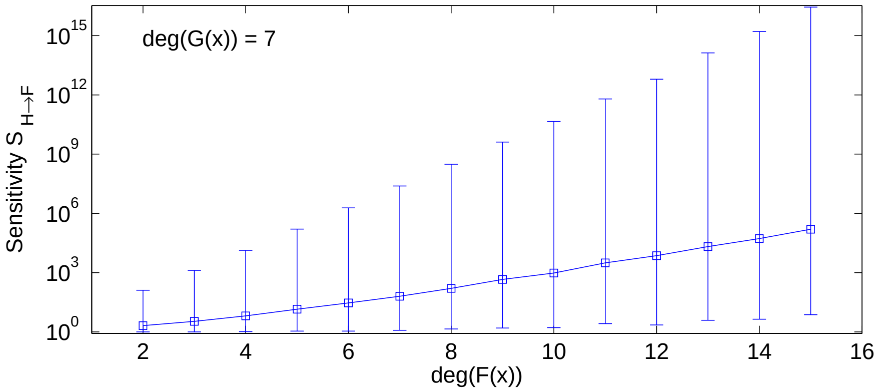


Figure 3-5: Sensitivity of the coefficients of $F(x)$ with respect to the coefficients of $H(x)$. The order of $G(x)$ is seven in all compositions. Each point is the median of $S_{H\to F}$ obtained from ten thousand compositions, where the vertical bars indicate the range from the maximum to the minimum values attained.

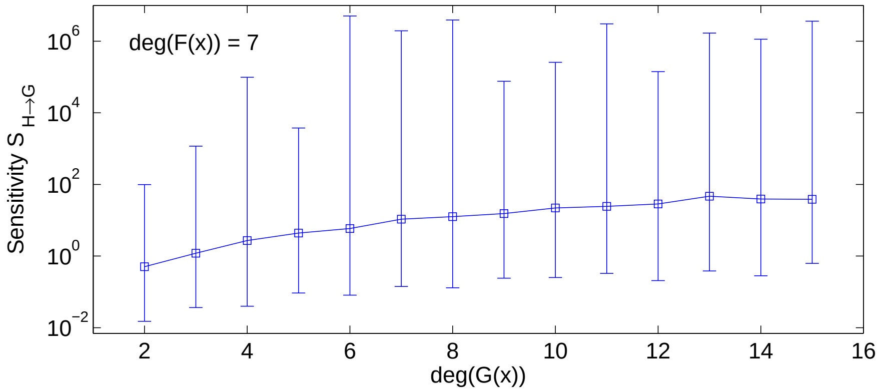


Figure 3-6: Sensitivity of the coefficients of $G(x)$ with respect to the coefficients of $H(x)$. The order of $F(x)$ is seven in all compositions. Each point is the median of $S_{H\to G}$ obtained from ten thousand compositions, where the vertical bars indicate the range from the maximum to the minimum values attained.

Simulations for decomposition sensitivity

Evaluation of $S_{H\to F}$

Fig. 3-5 illustrates the sensitivity of the coefficients of $F(x)$ with respect to the perturbations in $H(x)$, namely $S_{H\to F}$ as described in equation (3.54). The values are extracted from the experiments illustrated in Figure 3-3.

Evaluation of $S_{H\to G}$

$S_{H\to G}$, as described in equation (3.56) the sensitivity of the coefficients of $G(x)$ with respect to the perturbations in $H(x)$ is illustrated in Fig. 3-6. The values are extracted from the experiments performed in Figure 3-4.

3.5.4 Equivalent Decompositions

For a decomposable polynomial $H(x) = F(G(x))$, there exists infinitely many other pairs of polynomials that yields the same composition. This provides an opportunity to choose a decomposition that has a lower sensitivity than a given decomposition. This section discusses different ways of obtaining equivalent decompositions to be exploited in order to lower sensitivity in the sequel.

One way to obtain equivalent decompositions of a polynomial $H(x)$ is through compositions with first order polynomials. More specifically, given any first order polynomial $\lambda(x) = ax + b$ with $a \neq 0$ and with its inverse with respect to composition

$$\lambda^{-1}(x) = \frac{1}{a}x - \frac{b}{a}, \tag{3.57}$$

it is clear that

$$H(x) = F(G(x)) = (F \circ \lambda^{-1}) \circ (\lambda \circ G)(x) = \bar{F}(\bar{G}(x)). \tag{3.58}$$

Coefficients of $\bar{F}$ and $\bar{G}$ will be different in general yielding different sensitivities with respect to these coefficients.

Another way to obtain equivalent compositions is to exploit commutative polynomials [9], namely polynomial pairs that satisfy $F(G(x)) = G(F(x))$. One such class is the monomials, namely the polynomials of the form x^n where n is a positive integer. Another class of commutative polynomials is the Chebyshev polynomials which can be conveniently defined through trigonometric functions as

$$T_n(x) = \cos(n \cos^{-1}(x)). \tag{3.59}$$

Commutativity of the Chebyshev polynomials follows easily since

$$\begin{aligned} T_m \circ T_n(x) &= \cos(m \cos^{-1}(\cos(n \cos^{-1}(x)))) \\ &= \cos(mn \cos^{-1}(x)) \\ &= \cos(n \cos^{-1}(\cos(m \cos^{-1}(x)))) \\ &= T_n \circ T_m(x). \end{aligned} \tag{3.60}$$

Formalizing the work in [44], an *entire set of commutative polynomials* is defined in [9] as a set of polynomials which contains at least one of each positive degree, and in which any two members commute with each other. Furthermore, it is shown that only two such sets exist, which are the polynomials of the form $\lambda^{-1} \circ P_n \circ \lambda(x)$ where $P_n(x)$ is a monomial or a Chebyshev polynomial and $\lambda(x)$ is any first order polynomial. Both of these classes correspond to a rather restricted form of decomposability, and therefore they are relatively less useful in exploiting the existence of equivalent decompositions for lower sensitivity.

Similar to equation (3.5), equation (3.58) corresponds to the matrix equation

$$\mathbf{h} = \mathbf{Cf} = (\mathbf{CA})(\mathbf{A^{-1}f}) \tag{3.61}$$

where $\mathbf{A}$ is a square, upper triangular and invertible matrix kth column of which consists of $k-1$ self convolutions of the sequence $\{b, a\}$ or equivalently the coefficients of $(ax+b)^{k-1}$ in the ascending order. From equation (3.37), the sensitivity $S_{\bar{F}\to H}$ becomes

$$S_{\bar{F}\to H} = \max_{\Delta\mathbf{f}} \frac{||\mathbf{CA}\Delta\mathbf{f}||_2^2}{||\Delta\mathbf{f}||_2^2} \frac{||\mathbf{A^{-1}f}||_2^2}{||\mathbf{Cf}||_2^2}. \tag{3.62}$$

Although matrix $\mathbf{A}$ can be further factored as a product of two simpler matrices that depend only on a and b, respectively, it is not obvious how $S_{\bar{F}\to H}$ will behave as a joint function of a and b in general. The effect of pure scaling, which corresponds to the case $a > 0$, $b = 0$ and A is diagonal, can be inferred by examining the extremal values of a. More specifically, as a tends to infinity, the term $\max_{\Delta\mathbf{f}} \frac{||\mathbf{CA}\Delta\mathbf{f}||_2^2}{||\Delta\mathbf{f}||_2^2}$ also tends to infinity whereas the term $\frac{||\mathbf{A^{-1}f}||_2^2}{||\mathbf{Cf}||_2^2}$ tends to a finite constant number if the constant term of $F(x)$ is nonzero. The roles of these two terms are reversed as a tends to zero, which suggests the existence of a minimum at a finite value of $a > 0$.

Fig. 3-7 illustrates the effectiveness of choosing different values for a and b in order to reduce $S_{F\to H}$. Here, $F(x)$ and $G(x)$ are chosen to be the pair of polynomials that attained the largest sensitivity of 6.3×10^4 in Fig. 3-3 with $F(x)$ of order fourteen and $G(x)$ of order seven. The simulation results in Fig. 3-7 were obtained through an exhaustive search and indicate that $S_{\bar{F}\to H}$ gets larger as b tends to infinity in either direction for this pair of $F(x)$ and $G(x)$. $S_{\bar{F}\to H}$ attains its minimum at $a^* = 0.73$ and $b^* = 0.57$. Table 3.3 displays the values of all four sensitivities associated with this composition before and after composition with $\lambda(x) = 0.73x + 0.57$ and its inverse.

The effect of compositions with first order polynomials on $S_{G\to H}$ is relatively

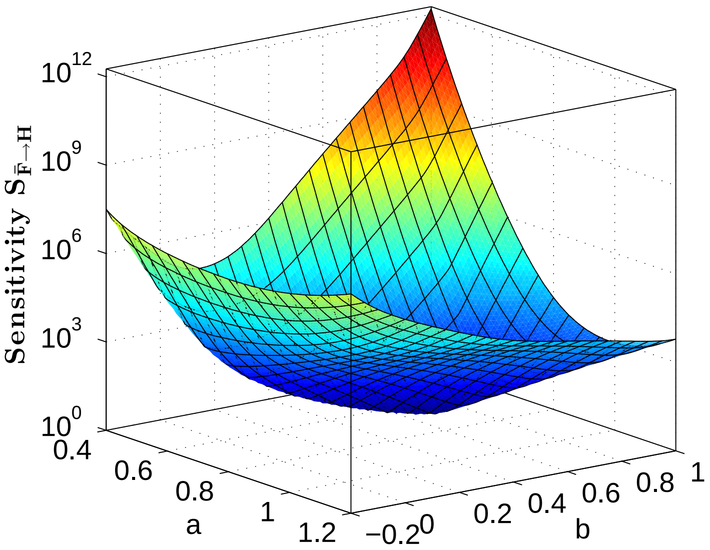


Figure 3-7: $S_{\bar{F}\to H}$ as a function of a and b. $F(x)$and $G(x)$ are chosen to be the pair of polynomials that attained the largest sensitivity in Fig. 3-3 with $F(x)$ of order fourteen and $G(x)$ of order seven.

more straightforward. The matrix $\mathbf{D}$ in the definition of $S_{G\to H}$ will be modified for the definition of $S_{\bar{G}\to H}$. More specifically,

$$H(x) = \bar{F} \circ \bar{G}(x), \tag{3.63}$$

therefore the columns of the modified matrix $\bar{\mathbf{D}}$ will consist of the coefficients of the polynomial $\bar{F}' \circ \bar{G}(x)$ instead of $F' \circ G(x)$. Since

$$\begin{aligned}\bar{F}' \circ \bar{G}(x) &= (F \circ \lambda^{-1})' \circ (\lambda \circ G(x)) \\ &= ((\lambda^{-1})' F' \circ \lambda^{-1}) \circ \lambda \circ G(x) = \frac{1}{a} D(x),\end{aligned} \tag{3.64}$$

Table 3.3: Sensitivity before and after composition with $a^*x + b^*$

Sensitivity	Original	at (a^*, b^*)
$S_{F\to H}$	6.3×10^4	2.2×10^0
$S_{G\to H}$	1.7×10^2	1.5×10^2
$S_{H\to F}$	1.1×10^8	3.5×10^1
$S_{H\to G}$	1.5×10^4	1.7×10^4

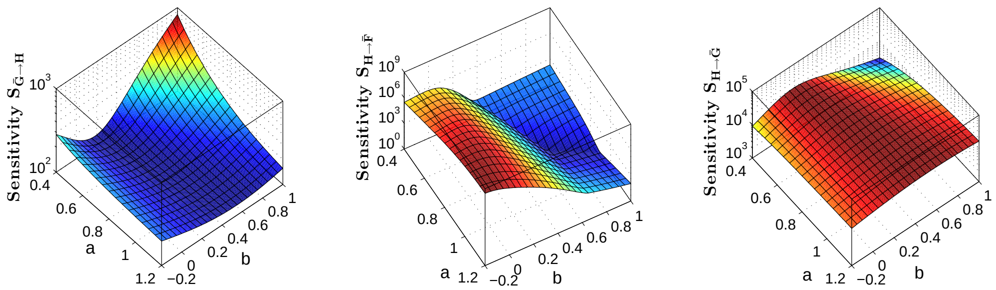


Figure 3-8: The behavior of $S_{\bar{G}\to H}$, $S_{H\to\bar{F}}$ and $S_{H\to\bar{G}}$ as a function of a and b for same pair of polynomials $F(x)$ and $G(x)$ as in Fig. 3-7.

the matrix $\bar{\mathbf{D}}$ is simply equal to $\mathbf{D}$ scaled with $\frac{1}{a}$. From equation (3.46), the sensitivity $S_{\bar{G}\to H}$ becomes,

$$S_{\bar{G}\to H} = \max_{\Delta\mathbf{g}} \frac{||\frac{1}{a}\mathbf{D}\Delta\mathbf{g}||_2^2}{||\Delta\mathbf{g}||_2^2} \frac{||a\mathbf{g} + b\mathbf{e}||_2^2}{||\mathbf{h}||_2^2} = \frac{||\mathbf{g} + \frac{b}{a}\mathbf{e}||_2^2}{||\mathbf{g}||_2^2} S_{G\to H} \tag{3.65}$$

where $\mathbf{e} = [1, 0, \ldots, 0]^T$ and it is the same size as $\mathbf{g}$. This implies that if $\left|g_0 + \frac{b}{a}\right| < |g_0|$ where g_0 is the constant term in $G(x)$, $S_{G\to H}$ will also be improved. This is indeed the case for the optimal point in Fig. 3-7 and introducing a linear composition to improve $S_{F\to H}$ has decreased $S_{G\to H}$. Due to its relationship with $S_{G\to H}$, the effect is reversed on $S_{H\to G}$ in such a way that their product remains the same. On the other hand, the effect on $S_{H\to\bar{F}}$ can be described at extreme values of a and b similarly to the case of $S_{\bar{F}\to H}$. Fig. 3-8 illustrates the behavior of all of these sensitivities as a function of a and b for same pair of polynomials $F(x)$ and $G(x)$. Since the optimal points are not the same for all sensitivities, a^* and b^* can be chosen depending on the application.

3.6 Chapter Conclusions

Polynomial composition and decomposition methods are an important part of a signal processing framework in which functional composition and decomposition are to be exploited. Although composing polynomials is a straightforward operation in terms of algebraic operations, their decomposition is not an easy task. In mathematics and computer science, decomposition methods have been developed to decompose polynomials that are known to have a decomposition, yielding one of the infinitely many choices for such a decomposition. A much more difficult problem involves approximating nondecomposable polynomials with those that are decomposable. Existing approximate decomposition methods as well as the new method introduced in this chapter yield satisfactory results only for low order polynomials that are fairly close to a decomposable polynomial, therefore approximate polynomial decomposition persists as an interesting and challenging problem.

The sensitivities of composition and decomposition were also discussed in this chapter, where natrix representation of composition is utilized to obtain closed form expressions for sensitivity measures. The existence of equivalent decompositions using first order polynomials and their inverses were also exploited to reduce sensitivity with respect to perturbations in the polynomial coefficients.

Chapter 4

Frequency Response Composition and Decomposition

Frequency response representations of signals and systems are central to signal processing. Therefore, identification and development of techniques for composing and decomposing frequency responses as well as formulating new and interesting applications using them are important parts of a framework that will be exploited in the context of signal processing.

The composition and decomposition of transfer functions that represent discrete time FIR sequences, i.e. z-transforms, were related to polynomial composition and decomposition as discussed in Chapter 3. The relationship between the z-transform and the Fourier transform suggests a straightforward extension of the techniques in Chapter 3 to the frequency response composition and decomposition. However, this would restrict its applications to finite sequences and FIR systems. Moreover, approximating the transfer function with a decomposable one with respect to a particular norm involving coefficients does not always guarantee a satisfactory approximation to the corresponding frequency response. An independent treatment of frequency response composition and decomposition from that of the transfer functions will provide flexibility for specifying more relevant approximation constraints as well as the ability to extend benefits of the framework to a wider class of signals including IIR systems and continuous time signals that cannot be represented by polynomials.

This chapter develops the mathematical methods that yield a decomposition of any desired frequency response $H(e^{j\omega})$ continuous in ω into two frequency responses $G(e^{j\omega})$ and $F(e^{j\omega})$, where F, but not necessarily G, is a polynomial in $e^{-j\omega}$, i.e.,

$$H(e^{j\omega}) \approx F(G(e^{j\omega})) = \sum_{k=0}^{M} f_k G^k(e^{j\omega}). \tag{4.1}$$

$G(e^{j\omega})$ will be assumed to be continuous and pre-specified, which is the case in certain signal processing applications that can potentially exploit this procedure. In cases for which $G(e^{j\omega})$ is not specified, it can, for example, be designed to approximate $H(e^{j\omega})$ with a low order rational frequency response before the decomposition so that the composition itself has a rational form. In the current approach, the coefficients f_k will be chosen to minimize the Chebyshev or l_∞ norm of the approximation error, the commonly preferred norm in several signal processing applications,

$$\begin{aligned} \underset{\mathbf{f}}{\text{minimize}} \quad & \Delta \\ \text{subject to} \quad & \|H(e^{j\omega}) - \sum_{k=0}^{K} f_k G^k(e^{j\omega})\|_\infty \leq \Delta. \end{aligned} \tag{4.2}$$

This problem can be solved easily for any finite set of points. However solving it on a continuum of frequency points requires specialized techniques. It is currently unclear how to extend the mathematical tools developed in this chapter to decompositions where $F(e^{j\omega})$ is a more general function than a polynomial. Therefore these cases are excluded from the current discussion.

In Section 4.1, the composition of a polynomial $F(\cdot)$ and a frequency response $G(e^{j\omega})$ will be treated as a special case of generalized polynomials, which is simply a linear combination of continuous functions. This point of view will allow borrowing two techniques from the existing mathematics literature with which optimal weights for a generalized polynomial can be computed, and therefore can be recast as a frequency response decomposition algorithm. The frequency response decomposition algorithm that is developed for the frequency responses of discrete time signals and systems will be extended to continuous time signals in a straightforward manner implying its generality.

Section 4.2 will present extensions to the decomposition algorithm where the decomposition quality is improved when the phase of the target response $H(e^{j\omega})$ is not required to be matched and there is an emphasis on its magnitude instead, another case that often arises in the context of signal processing applications.

4.1 Frequency Response Decomposition

The problem stated in (4.2) is a special case of the semi-infinite optimization problem

$$\begin{aligned} &\underset{\mathbf{f}}{\text{minimize}} \quad \Delta \\ &\text{subject to} \quad \|D(\omega) - \sum_{k=0}^{K} f_k U_k(\omega)\|_\infty \leq \Delta \end{aligned} \tag{4.3}$$

where $U_k(\omega)$, $k = 0, 1, \ldots, K$, and $D(\omega)$ are general continuous functions of ω. Following Cheney [13], a linear combination of continuous functions $U_k(\omega)$ will be referred to as a generalized polynomial in this thesis. The interpretation of the decomposition of a desired response $D(\omega) = H(e^{j\omega})$ as an approximation using a generalized polynomial with $U_k(\omega) = G^k(e^{j\omega})$ will be the central theme when developing and extending the frequency response decomposition algorithm. More specifically, the techniques from the mathematical literature that will yield the optimal weights f_k in (4.3) will constitute the basis for the frequency response decomposition algorithm.

Two such mathematical techniques will be discussed in this section. The first one is an efficient algorithm called the Remez Exchange Algorithm that will be discussed in Section 4.1.2. However this works only under very restrictive conditions one of which is called the Haar condition. The other is a less efficient algorithm called the First Algorithm of Remez, which is not limited by the constraints of the previous algorithm. This algorithm will be described in Section 4.1.3. Although the Haar condition is not required for the First Algorithm of Remez, its existence implies uniqueness of the optimal choice of weights. Since the Haar condition is central to the discussion of both algorithms, this section starts with its description.

4.1.1 Haar Condition and Best Approximations

A set of real or complex valued basis functions, $U_k(\omega), k = 0, 1, 2, \ldots, K$, is said to satisfy the Haar condition on a compact set $\mathcal{S}$ if each $U_k(\omega)$ is continuous and any function in their span,

$$\sum_{k=0}^{K} f_k U_k(\omega), \tag{4.4}$$

has at most K roots in $\mathcal{S}$ [34]. This is equivalent to the constraint on the $(K+1) \times (K+1)$ matrix

$$\mathbf{V} = \begin{bmatrix} U_0(\omega_0) & U_1(\omega_0) & U_2(\omega_0) & \ldots & U_K(\omega_0) \\ U_0(\omega_1) & U_1(\omega_1) & U_2(\omega_1) & \ldots & U_K(\omega_1) \\ U_0(\omega_2) & U_1(\omega_2) & U_2(\omega_2) & \ldots & U_K(\omega_2) \\ & & \vdots & & \\ U_0(\omega_K) & U_1(\omega_K) & U_2(\omega_K) & \ldots & U_K(\omega_K) \end{bmatrix} \tag{4.5}$$

to be full rank for every set of distinct frequencies $\{\omega_k \in \mathcal{S}, k = 0, 1, 2, \ldots, K\}$ [13].

An optimal approximation with respect to the Chebyshev norm to any continuous function $D(\omega)$ as a linear combination of $U_k(\omega)$ as in (4.3) exists, i.e. a set of weights $\{f_k\}$ can be found to achieve the optimal approximation [13]. The existence of the Haar condition within the basis functions is a necessary and sufficient condition for the uniqueness of this optimal approximation and the set of coefficients $\{f_k\}$ leading to it (Theorem 19 in [34], [13]). For example, the optimal approximation to any continuous function on a compact set $\mathcal{S}$ with an ordinary polynomial is unique since the set of monomials $\omega^k, k = 0, 1, \ldots, K$, satisfies the Haar condition on any compact set. This can be verified by the fact that the matrix $\mathbf{V}$ is a Vandermonde matrix for $U_k(\omega) = \omega^k$, which is guaranteed to be full rank if all ω_k are distinct.

4.1.2 Alternation Theorem and the Remez Exchange Algorithm

In cases where the basis functions $U_k(\omega)$ satisfy the Haar condition and both these and $D(\omega)$ are real valued, the problem stated in (4.3) accepts an efficient solution using the algorithm called the Remez Exchange Algorithm [13]. This algorithm exploits a characterization of the unique optimal approximation given by the alternation theorem [13,34], where uniqueness of the optimal solution follows from the Haar condition:

Theorem 4.1. *Alternation theorem: The function*

$$\tilde{D}(\omega) = \sum_{k=0}^{K} f_k U_k(\omega) \tag{4.6}$$

is the unique optimal approximation to $D(\omega)$ that is in the span of $\{U_k(\omega), k = 0, 1, \ldots, K\}$ if and only if the error function

$$E(\omega) = D(\omega) - \sum_{k=0}^{K} f_k U_k(\omega) \tag{4.7}$$

exhibits at least $K+2$ alternations, i.e., there are at least $K+2$ alternation points $\omega_i \in \mathcal{S}$ such that

$$\omega_1 < \omega_2 < \cdots < \omega_{K+1} < \omega_{K+2}$$

and

$$E(\omega_i) = -E(\omega_{i+1}) = \pm \max_{\omega \in \mathcal{S}} |E(\omega)|$$

for $i = 1, 2, \ldots, K+1$. In other words, for the unique optimal approximation $\tilde{D}(\omega)$, the absolute value of the error function attains its maximum value at the points ω_i where the sign of the error alternates from one alternation point to the next.

The Remez Exchange Algorithm [13,42] is an iterative algorithm which exploits the existence of at least $K+2$ alternation points and the behavior of the error function $E(\omega)$ at these points to find the unique optimal set of weights f_k in (4.3). It starts with an initial guess for the alternation points, and at each iteration the coefficients f_k are updated so that the maximum error on the discrete set of candidate alternation points is minimized. For the next iteration, the candidate points are exchanged with another set of $K+2$ points including the point of maximum error on $\mathcal{S}$. The iterations are continued until the change in the maximum error does not improve beyond a prespecified threshold. This procedure is guaranteed to converge where the convergence is quadratic [13]. A given frequency response $D(\omega)$ can be decomposed by using the Remez Exchange Algorithm by setting $U_k(\omega) = G^k(e^{j\omega})$.

The set of basis functions $(\cos\omega)^k, k = 0, 1, \ldots, L$, satisfy the Haar condition and are real valued. The well-known Parks-McClellan filter design algorithm [42] exploits the alternation theorem and the Remez Exchange Algorithm to design, for example, an even symmetric FIR filter for which the frequency response can be represented as

$$H(e^{j\omega}) = \sum_{k=0}^{L} f_k (\cos\omega)^k. \tag{4.8}$$

This implies that Parks-McClellan filter design algorithm can be re-interpreted as the decomposition of an ideal filter response $D(\omega)$ as $F(G(e^{j\omega}))$ with $G(e^{j\omega}) = \cos\omega$. This also suggests that the frequency response decomposition of the ideal filter response with more general real-valued functions $G^k(e^{j\omega})$ satisfying the Haar condition can be viewed as a generalization of the Parks-McClellan filter design algorithm.

A positive and continuous weight $W(\omega)$ can be imposed on the error function $E(\omega)$ in equation (4.7) in order to regulate the relative approximation quality on subsets of $\mathcal{S}$. In that case, the alternation theorem and Remez exchange algorithm can be re-expressed with respect to the weighted error

$$E_W(\omega) = W(\omega)\left[D(\omega) - \sum_{k=0}^{K} f_k U_k(\omega)\right]. \tag{4.9}$$

The weighted approximation problem becomes equivalent to approximating the target function $W(\omega)D(\omega)$ with a generalized polynomial where the basis functions are $\{W(\omega)U_k(\omega), k = 0, 1, \ldots, K\}$. This adjusted set of basis functions also satisfies the

Haar condition since the matrix

$$\begin{bmatrix} W(\omega_0)U_0(\omega_0) & W(\omega_0)U_1(\omega_0) & W(\omega_0)U_2(\omega_0) & \ldots & W(\omega_0)U_K(\omega_0) \\ W(\omega_1)U_0(\omega_1) & W(\omega_1)U_1(\omega_1) & W(\omega_1)U_2(\omega_1) & \ldots & W(\omega_1)U_K(\omega_1) \\ W(\omega_2)U_0(\omega_2) & W(\omega_2)U_1(\omega_2) & W(\omega_2)U_2(\omega_2) & \ldots & W(\omega_2)U_K(\omega_2) \\ & & & \vdots & \\ W(\omega_K)U_0(\omega_K) & W(\omega_K)U_1(\omega_K) & W(\omega_K)U_2(\omega_K) & \ldots & W(\omega_K)U_K(\omega_K) \end{bmatrix} \tag{4.10}$$

is full rank for every set of distinct frequencies $\{\omega_k \in \mathcal{S}, k = 0, 1, 2, \ldots, K\}$. That the matrix in (4.10) is full rank can be shown by expressing it as the product of two full rank matrices, namely the diagonal matrix $\mathbf{W}$ with diagonal entries $W(\omega_k)$ and $\mathbf{V}$ in equation (4.5).

Although the Remez exchange algorithm is an efficient algorithm to solve the frequency response decomposition as described, the requirements on $H(e^{j\omega})$ to be real and on $G^k(e^{j\omega}), k = 0, 1, \ldots, K$, to be both real and to satisfy the Haar condition are quite restricting on its applicability. This can be seen from the fact that the matrix $\mathbf{V}$ given in (4.5) is a Vandermonde matrix for this basis, and it is full rank if and only if $G(e^{j\omega})$ attains distinct values for each frequency $\omega \in \mathcal{S}$, i.e. if $G(e^{j\omega})$ is monotonic. It is desirable to depart to possibly less efficient algorithms that work for complex valued frequency responses and for basis functions that do not satisfy the Haar condition such as the First Algorithm of Remez described next.

4.1.3 The First Algorithm of Remez

An algorithm that yields a solution to problem (4.3) and hence the frequency response decomposition problem is the First Algorithm of Remez [13], which requires only the function $D(\omega)$ and the basis functions $U_k(\omega)$ to be continuous on the compact set $\mathcal{S}$. Therefore, the algorithm is applicable to cases where these functions are complex valued and where the Haar condition is not satisfied, two scenarios that very often appear in a signal processing setting. The steps of the First Algorithm of Remez are given in Algorithm 2.

ALGORITHM 2

```
Input: $U_k(\omega), k = 0, 1, \ldots, K$ and $D(\omega)$,
Output: $\mathbf{f}^* = \arg\min_{\mathbf{f}} \|D(\omega) - \sum_{k=0}^{K} f_k U_k(\omega)\|_\infty$.
Begin ($i = 1$)
0. Choose $\mathcal{S}^{[i]} = \{\omega_0, \omega_1, \ldots, \omega_m\} \subset \mathcal{S}$ such that
   $m \geq K$ and the matrix $[U_k(\omega_n)]_{k,n},\ k = 0, 1, \ldots, K;$
   $n = 0, 1, \ldots, m$ has column rank $K + 1$.
1. Set $\mathbf{f}^{[\mathbf{i}]} = \arg\min_{\mathbf{f}} \left\{ \max_{\omega \in \mathcal{S}^{(i)}} |D(\omega) - \sum_{k=0}^{K} f_k U_k(\omega)| \right\}$.
2. Find $\omega^{[i]} = \arg\max_{\omega \in \mathcal{S}} |D(\omega) - \sum_{k=0}^{K} f_k^{(i)} U_k(\omega)|$.
3. Set $\mathcal{S}^{[i+1]} \leftarrow \mathcal{S}^{[i]} \cup \{\omega^{[i]}\}$ and $i \leftarrow i + 1$, go to Step 1.
```

The First Algorithm of Remez starts with a discrete set of frequencies $\{\omega_i, i = 0, 1, \ldots, m\}$ in $\mathcal{S}$, where m is at least K. Although the Haar condition is not required to be satisfied by the basis functions, this initial set of frequency points must be chosen such that they yield a full-column-rank matrix

$$\mathbf{V}_{init} = \begin{bmatrix} U_0(\omega_0) & U_1(\omega_0) & U_2(\omega_0) & \ldots & U_K(\omega_0) \\ U_0(\omega_1) & U_1(\omega_1) & U_2(\omega_1) & \ldots & U_K(\omega_1) \\ U_0(\omega_2) & U_1(\omega_2) & U_2(\omega_2) & \ldots & U_K(\omega_2) \\ & & \vdots & & \\ U_0(\omega_K) & U_1(\omega_K) & U_2(\omega_K) & \ldots & U_K(\omega_K) \\ & & \vdots & & \\ U_0(\omega_m) & U_1(\omega_m) & U_2(\omega_m) & \ldots & U_K(\omega_m) \end{bmatrix}. \tag{4.11}$$

The lack of such a discrete set of points indicates that the basis is degenerate, i.e., some of the basis functions $U_k(\omega)$ are linearly dependent on others, in which case they are excluded from the basis. The optimization problem (4.3) is solved only for this discrete

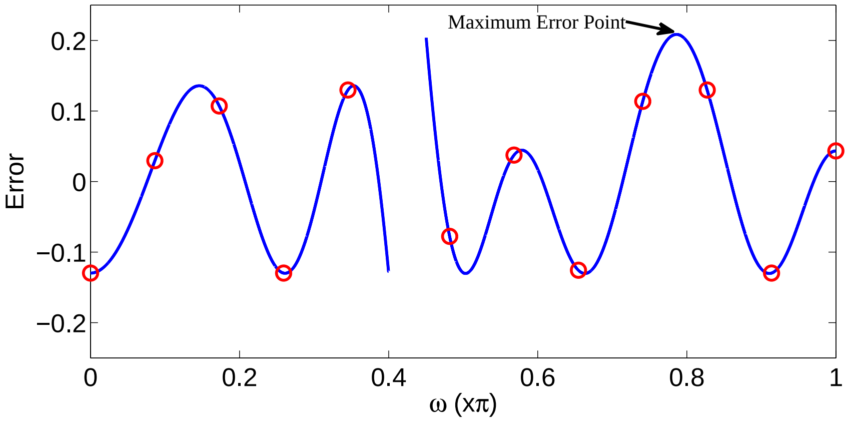


Figure 4-1: The First Algorithm of Remez computes the optimal approximation for a discrete set of points, and locates the maximum error point to include in the next iteration. In this example, the compact set $\mathcal{S}$ on which the approximation is performed is $[0, \pi]$.

set of points. The coefficients f_k obtained are used to evaluate the approximation on $\mathcal{S}$ and to locate the frequency at which the maximum approximation error occurs as illustrated in Figure 4-1. This frequency is added to the set of points for which the optimization problem will be solved in the next iteration, and the procedure is repeated until the minimax error at each iteration does not deviate by more than a pre-specified threshold.

Unlike the Remez exchange algorithm, the number of points for which an optimization problem is solved increases by one at each iteration in the First Algorithm of Remez. However, this does not present a serious computational concern as the optimization problem (4.3) is linear when the functions involved are real-valued, and is convex when they are complex-valued for a discrete set of frequencies, both cases of which can be solved efficiently, and the required number of iterations are quite few as will be shown in applications in Chapter 6. Moreover, it is shown in [19] that the exact computation of the location of the maximum error on $\mathcal{S}$ at each iteration is not required, and the algorithm has the same convergence guarantees when approximation error is computed on a "dense enough" and possibly irregular grid in $\mathcal{S}$.

At each iteration i, the minimax error on $\mathcal{S}$ increases as new points are added to the discrete set of points $\mathcal{S}^{(i)}$. The vector of optimal coefficients $\mathbf{f^{(i)}}$ obtained at each iteration is guaranteed to be in a bounded subset of $\mathcal{R}^{K+1}$, therefore the sequence of vectors $\mathbf{f^{(i)}}$ is guaranteed to have at least one clustering point [46]. Moreover, any of these clustering points will be an optimal choice for the coefficients f_k with the same maximum approximation error as the other cluster points. This implies that the coefficient vector sequence $\mathbf{f^{(i)}}$ does not necessarily converge, but the sequence of minimax error on $\mathcal{S}$ converges. The Haar condition for $\{U_k(\omega)\}$ is not required for the convergence of minimax error sequence, however its existence guarantees the uniqueness of the optimal choice for coefficients f_k [13]. These facts are proved in [13] and the proof is included in Appendix B with the notation of this chapter for convenience.

For the frequency response decomposition problem stated in (4.2), the matrix in equation (4.5) will take the form of a complex-valued Vandermonde matrix which is full rank if and only if $G(e^{j\omega})$ has distinct values for each $\omega \in \mathcal{S}$. When the Haar condition is not satisfied, there are multiple optimal choices including any clustering point of the coefficient vector sequence $\mathbf{f^{(i)}}$ obtained in Algorithm 2. In fact, any coefficient vector in the convex hull of the identified optimal coefficient vectors can be shown to be an optimal choice itself. In order to see this, consider two optimal coefficent vectors $\mathbf{a}$ and $\mathbf{b}$ that satisfy

$$\|D(\omega) - \sum_{k=0}^{K} a_k U_k(\omega)\|_\infty = \Delta_{opt} \tag{4.12}$$

and

$$\|D(\omega) - \sum_{k=0}^{K} b_k U_k(\omega)\|_\infty = \Delta_{opt} \tag{4.13}$$

where Δ_{opt} is the minimum attainable error among all choices of coefficients. It suffices to show that the coefficient vector in the midpoint $\frac{\mathbf{a}+\mathbf{b}}{2}$ is also an optimal choice as in

$$\begin{aligned}
&\left| D(\omega) - \sum_{k=0}^{K+1} \frac{a_k + b_k}{2} U_k(\omega) \right| \\
&= \frac{1}{2} \left| \left(D(\omega) - \sum_{k=0}^{K+1} a_k U_k(\omega) \right) + \left(D(\omega) - \sum_{k=0}^{K+1} b_k U_k(\omega) \right) \right| \\
&\leq \frac{1}{2} \left(\left| (D(\omega) - \sum_{k=0}^{K+1} a_k U_k(\omega) \right| + \left| (D(\omega) - \sum_{k=0}^{K+1} b_k U_k(\omega) \right| \right) \\
&\leq \Delta_{opt}
\end{aligned} \tag{4.14}$$

where the triangular inequality was used in the first inequality. However, since no coefficient set can achieve a smaller error than Δ_{opt}, the inequality becomes an equality proving the optimality of $\frac{\mathbf{a}+\mathbf{b}}{2}$.

4.1.4 The Frequency Response Decomposition Algorithm

Since the frequency response decomposition of a desired function $D(\omega)$ is a special case of the optimization problem (4.3) with $U_k(\omega) = G^k(e^{j\omega})$, its solution consists of one of the two algorithms developed in this section.

Frequency Response Decomposition Algorithm. Given a desired function $D(\omega)$, the basis functions $G^k(e^{j\omega}), k = 0, 1, \ldots, K$, and the compact set $\mathcal{S}$ on which the decomposition is to be performed, invoke the Remez Exchange Algorithm with $U_k(\omega) = G^k(e^{j\omega})$) if $D(\omega)$ and $G(e^{j\omega})$ are real valued and the functions $G^k(e^{j\omega}), k = 0, 1, \ldots, K$, satisfy the Haar condition. Otherwise, invoke the First Algorithm of Remez given in Algorithm 2.

Although the First Algorithm of Remez provides a slightly less efficient algorithm to solve the optimization problem (4.3) than the Remez Exchange Algorithm due to an increasing number of frequency points at each iteration for which the error is optimized, it will be adapted as the main choice to solve this problem in the rest of this chapter and for the applications in Chapter 6 since it applies to a much wider class of applications where the functions are complex valued and the Haar condition is not necessarily satisfied.

4.1.5 Decomposition on infinite intervals: Continuous time

The frequency response decomposition algorithm applies to problems defined only on compact sets. Continuous time signals and filters have frequency response representations that are defined on the entire real line as opposed to discrete time signals and systems which need to be specified only on the compact interval $[-\pi, \pi]$. Therefore, one way to perform the decomposition of such frequency responses defined on $(-\infty, \infty)$ is to transform the problem to a compact interval, for example through a frequency warping, before applying the decomposition algorithm developed in this section. In this case, such a transformation must also preserve the minimax approximation error profile in both the original domain and the warped domain. The bilinear transformation satisfies both of these properties and applications exploiting this transformation will be shown in Chapter 6.

4.2 Frequency Response Decomposition by Magnitude

Although the First Algorithm of Remez is able to locate the optimal solution by solving problem (4.3) efficiently even when $D(\omega)$ and $U_k(\omega)$ are complex valued functions, a general disadvantage of approximating complex valued functions when compared to those that are real valued is the additional requirement to match the phase of the approximating generalized polynomial $\sum_k f_k U_k(\omega)$ to that of $D(\omega)$, which are always matched when both are real regardless of the choice of coefficients f_k. The attained minimax error Δ_{opt} may not be satisfactory in cases where the phases of $D(\omega)$ and the basis functions $U_k(\omega)$ are arbitrary due to this additional burden on the approximation algorithm. The approximation quality may improve significantly if only the magnitude of a complex-valued function is desired to be approximated with that of a composition in an application. For example, the specification of continuous time LTI filters are provided as constraints on the magnitude of the frequency response. These applications can potentially benefit from an extension of the frequency response decomposition algorithm to the case where the approximation quality is specified with respect to the difference between $|H(e^{j\omega})|$ and $|F(G(e^{j\omega}))|$.

This section extends the frequency response decomposition algorithm given in Section 4.1.4 to the case of decomposition by magnitude. Since the functions of interest are in general complex valued, only the first Algorithm of Remez will be considered. In this case, the error to be minimized is given in terms of the difference of magnitudes of the target function $H(e^{j\omega})$ and its decomposition $\sum_k f_k G^k(e^{j\omega})$, and the problem of decomposition by magnitude,

$$\begin{aligned} &\underset{\mathbf{f}}{\text{minimize}} \quad \Delta \\ &\text{subject to} \quad \left\| \left|H(e^{j\omega})\right| - \left|\sum_{k=0}^{K} f_k G^k(e^{j\omega})\right| \right\|_\infty \leq \Delta, \end{aligned} \tag{4.15}$$

becomes a special case of the semi-infinite optimization problem

$$\begin{aligned} &\underset{\mathbf{f}}{\text{minimize}} \quad \Delta \\ &\text{subject to} \quad \left| M(\omega) - \left|\sum_{k=0}^{K} f_k U_k(\omega)\right| \right\|_\infty \leq \Delta \end{aligned} \tag{4.16}$$

with $M(\omega) = |H(e^{j\omega})|$ and $U_k(\omega) = G^k(e^{j\omega})$.

The frequency response decomposition algorithm described in Section 4.1.4 exploited the similarity of problems (4.2) and (4.3), where the latter was efficiently solved by the First Algorithm of Remez for complex valued functions. However, the decomposition problem by magnitude in (4.15) cannot directly exploit its similarity to (4.16) as the First Algorithm of Remez does not apply to problems of the latter form. Moreover, these problems are not convex with respect to the coefficients f_k unlike in the frequency response decomposition problem. In this section, an equivalent problem to that of (4.16) will be formulated that allows the utilization of the First Algorithm of Remez in an iterative algorithm for its solution.

4.2.1 An Alternating Projections Algorithm

The optimal solution for the optimization problem (4.16) does not change if its constraint is replaced by

$$\left\| \left|M(\omega)e^{j\Theta(\omega)}\right| - \left|\sum_{k=0}^{K} f_k U_k(\omega)\right| \right\|_\infty \leq \Delta \tag{4.17}$$

where $\Theta(\omega)$ is an arbitrary phase function since $e^{j\Theta(\omega)}$ has unit magnitude. In (4.17), specifying $\Theta(\omega) = \Theta_{\mathbf{f}}(\omega)$, the phase of the generalized polynomial $\sum_k f_k U_k(\omega)$ such that

$$\sum_{k=0}^{K} f_k U_k(\omega) = \left|\sum_{k=0}^{K} f_k U_k(\omega)\right| e^{j\Theta_{\mathbf{f}}(\omega)}, \tag{4.18}$$

renders the absolute value operations redundant since both complex terms have the same phase. Removing the absolute values yields an equivalent problem to (4.16),

$$\begin{aligned} &\underset{\mathbf{f}}{\text{minimize}} && \Delta \\ &\text{subject to} && \left\| M(\omega)e^{j\Theta_{\mathbf{f}}(\omega)} - \sum_{k=0}^{K} f_k U_k(\omega) \right\|_\infty \leq \Delta. \end{aligned} \tag{4.19}$$

The subscript $\mathbf{f}$ in $\Theta_{\mathbf{f}}(\omega)$ represents the dependency of the specified phase on the coefficients of the generalized polynomial.

The optimization problem (4.19) is not convex and cannot be solved exactly. Although (4.19) is equivalent to (4.16), its explicit dependence on $\mathbf{f}$ in two different terms in its constraint suggests an iterative algorithm each step of which is much easier to perform than solving (4.16) directly. In the i-th iteration of this iterative algorithm, the optimal coefficients $\mathbf{f}^{[i]}$ are obtained given the phase $\Theta^{[i-1]}(\omega)$ from the $(i-1)$-st iteration. This phase is updated for the next iteration with the optimal choice of $\Theta^{[i]}(\omega)$ given $\mathbf{f}^{[\mathbf{i}]}$. This procedure is outlined in Algorithm 3.

```
ALGORITHM 3
    **Input:** $U_k(\omega), k = 0, 1, \ldots, K$; $M(\omega)$; an arbitrary $\Theta^{[0]}(\omega)$
    **Output:** A local optimum for $\mathbf{f}^* = \arg\min_{\mathbf{f}} \|M(\omega)e^{j\Theta_{\mathbf{f}}(\omega)} - \sum_{k=0}^{K} f_k U_k(\omega)\|_\infty$.
    Set $i = 1$.
1. Set $\mathbf{f}^{\mathbf{[i]}} = \arg\min_{\mathbf{f}} \|M(\omega)e^{j\Theta^{[i-1]}(\omega)} - \sum_{k=0}^{K} f_k U_k(\omega)\|_\infty$.
2. Set $\Theta^{[i]}(\omega) = \arg\min_{\Theta(\cdot)} \|M(\omega)e^{j\Theta(\omega)} - \sum_{k=0}^{K} f_k^{[i]} U_k(\omega)\|_\infty$
3. Set $i \leftarrow i + 1$, go to Step 1.
```

Given the phase $\Theta^{[i-1]}(\omega)$ from the previous iteration, finding the optimal coefficient vector $\mathbf{f}^{[i]}$ to minimize the error in Step 1 of Algorithm 3 is an instance of an approximation problem by a generalized polynomial which can be solved easily using the First Algorithm of Remez. This step can be viewed as the projection of the function $M(\omega)e^{j\Theta^{[i-1]}(\omega)}$ onto the space of functions spanned by the basis functions $U_k(\omega), k = 0, 1, \ldots, K$, i.e.,

$$\mathcal{U} = \{P(\omega) \text{ s.t. } P(\omega) = \sum_{k=0}^{K} a_k U_k(\omega) \text{ for } a_k \in \mathcal{R}\}. \tag{4.20}$$

Choosing the same set of frequency points in the initial step each time the first algorithm of Remez is invoked guarantees the sequence $\mathbf{f}^{[i]}$ to be bounded in magnitude by the same number not only within one call of the algorithm but also throughout the iterations in Algorithm 3.

Once $\mathbf{f}^{[i]}$ is determined in Step 1, the optimal phase $\Theta^{[i]}(\omega)$ to attach to $M(\omega)$ can be determined by interpreting Step 2 as the projection of the generalized polynomial $\sum_k f_k^{[i]} U_k(\omega)$ onto the set of complex valued functions with magnitude $M(\omega)$, i.e.,

$$\mathcal{M} = \{R(\omega) \text{ s.t. } R(\omega) = M(\omega)e^{j\Theta(\omega)} \text{ for arbitrary } \Theta(\omega)\}. \tag{4.21}$$

This set does not correspond to a vector space and it is not even convex, however projections onto this set can be performed in a very straightforward manner. More specifically, for every ω,

$$\begin{aligned} &\left|M(\omega)e^{j\Theta(\omega)} - \sum_{k=0}^{K} f_k^{[i]} U_k(\omega)\right| = \left|M(\omega)e^{j\Theta(\omega)} - \left|\sum_{k=0}^{K} f_k^{[i]} U_k(\omega)\right| e^{j\Theta_{\mathbf{f}}^{[i]}(\omega)}\right| \\ &= |M(\omega)|^2 + \left|\sum_{k=0}^{K} f_k^{[i]} U_k(\omega)\right|^2 - 2\,|M(\omega)| \left|\sum_{k=0}^{K} f_k^{[i]} U_k(\omega)\right| \cos(\Theta(\omega) - \Theta_{\mathbf{f}}^{[i]}(\omega)) \end{aligned} \tag{4.22}$$

where the first equality follows from equation (4.18) for the definition of $\Theta_{\mathbf{f}}^{[i]}(\omega)$ and the second equality follows from the cosine theorem. The same optimal choice of $\Theta(\omega) = \Theta_{\mathbf{f}}^{[i]}(\omega)$ minimizes equation (4.22) for every frequency ω, hence it is the solution of Step 2 of Algorithm 3.

The two steps of Algorithm 3 correspond to alternating projections of a desired magnitude function $M(\omega)$ between the sets $\mathcal{U}$ defined in equation (4.20) and $\mathcal{M}$ defined in equation (4.21) as depicted in Figure 4-2. It is well known that if the two sets were both convex witn non-empty intersection, the sequence of functions obtained during this iterative procedure would converge to a function in $\mathcal{U} \cap \mathcal{M}$ yielding $\Delta = 0$, or, if the intersection is empty, converge to the closest point of $\mathcal{U}$ to $\mathcal{M}$ attaining the global minimum of Δ. Although the lack of convexity in $\mathcal{M}$ prevents from establishing such guarantees in the magnitude response decomposition problem, the maximum error $\Delta^{[i]}$ at each iteration can be shown to be a non-increasing sequence. More specifically, for any $i > 0$,

$$\begin{aligned} \Delta^{[i]} &= \|M(\omega)e^{j\Theta^{[i]}(\omega)} - \sum_{k=0}^{K} f_k^{[i]} U_k(\omega)\|_\infty \\ &\geq \|M(\omega)e^{j\Theta^{[i]}(\omega)} - \sum_{k=0}^{K} f_k^{[i+1]} U_k(\omega)\|_\infty \\ &\geq \|M(\omega)e^{j\Theta^{[i+1]}(\omega)} - \sum_{k=0}^{K} f_k^{[i+1]} U_k(\omega)\|_\infty \\ &= \Delta^{[i+1]}, \end{aligned} \tag{4.23}$$

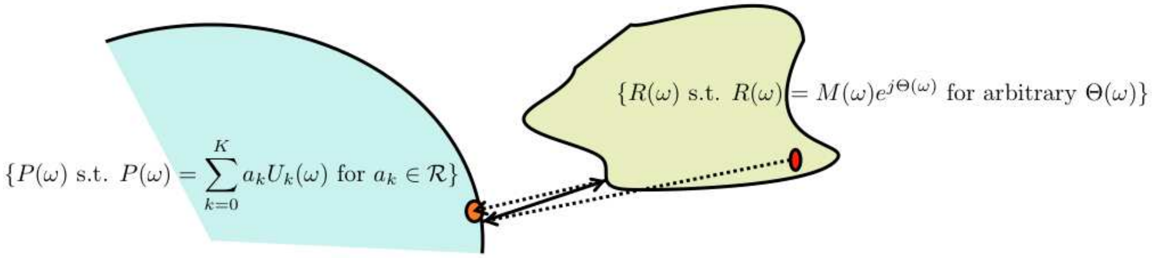

$\{R(\omega)$ s.t. $R(\omega) = M(\omega)e^{j\Theta(\omega)}$ for arbitrary $\Theta(\omega)\}$
$\{P(\omega)$ s.t. $P(\omega) = \sum_{k=0}^{K} a_k U_k(\omega)$ for $a_k \in \mathcal{R}\}$

Figure 4-2: The illustration of alternating projections of a function between two sets $\mathcal{U}$ defined in equation (4.20) and $\mathcal{M}$ defined in equation (4.21).

where the first inequality follows from the minimization at Step 1 and the second inequality follows from the minimization from Step 2 of Algorithm 3. Since $\Delta^{[i]}$ is bounded below by zero, it is going to converge and possibly to a positive value Δ_{opt}.

4.3 Chapter Conclusions

In this chapter, frequency response decomposition was treated as a special case of approximations with generalized polynomials in order to develop the decomposition algorithms in a more general setting and to utilize the mathematical tools available for generalized polynomials from the literature. The Remez Exchange Algorithm was shown to be an important tool that can be exploited to decompose real valued frequency responses when the basis functions $G^k(e^{j\omega}), k = 0, 1, \ldots, K$ are both real-valued and satisfy the Haar condition on the frequency intervals of interest. Although computationally efficient, such restrictions of the Remez Exchange algorithm motivated an alternative algorithm again due to Remez, the First Algorithm of Remez, to be extended for decomposing frequency responses which does not have these restrictions. This latter algorithm was extended to cases where only the magnitude of the composition approximates the magnitude of a desired function. An iterative algorithm was proposed for the solution of the decomposition problem by magnitude, where, although the non-convex problem is not solved exactly, the approximation error was shown to be monotonically non-increasing at each iteration step.

Chapter 5

Discrete Multivariate Function Composition and Decomposition

In the previous chapters, the focus has been on the composition and decomposition of one dimensional functions. In a signal processing framework in which functional composition and decomposition are to be exploited, it may be potentially useful to extend the discussion to multivariate functions as well.

Applications that are characterized by their requirement of taking summations, i.e. marginalizations, over multivariate functions constitute a large class of problems that can potentially benefit from the composition and the decomposition of multivariate functions. Some well-known examples that require marginalizations are nonlinear filtering, projections, computing expectations and messages in message passing algorithms in statistical inference problems. This chapter will develop a multivariate function decomposition algorithm in order to benefit from its potential computational efficiencies for this large set of problems. The computational benefits for the case of factorable functions in these applications are well known and several efficient algorithms have already been developed to exploit factorability. The decomposition algorithm which will be developed in this chapter will be formulated in a way to extend the benefits of factorability to decomposability and to exploit the algorithms previously developed in this new context. This will be accomplished by introducing latent variables to a given decomposable function and representing it as a product of functions each of which have fewer variables, in which case the efficient algorithms for the factorable functions can be invoked. This is done at the expense of artificially increasing the dimensionality of the given function. However a careful choice of an alphabet size for the latent variable will be shown to ensure this approach remains efficient. Moreover, if a given function is not exactly decomposable, it will be approximated by a decomposable representation as an alternative to approximate factorization in order to benefit from the same efficiencies.

In this thesis, attention will be restricted to discrete functions as the methods developed will exploit mathematical methods that assume discrete and finite alphabet sizes. However this will be sufficient to illustrate the feasibility and benefits of introducing decomposition in certain signal processing applications and provide a motivation for exploring decomposition algorithms for multivariate continuous functions in a future study.

In Section 5.1, the computational difficulties in the context of marginalization problems will be discussed with an emphasis on why factorability into functions with fewer variables have been widely considered as desirable. The class of applications known in the literature as marginalize-a-product-function (MPF) problems will be introduced [1], and a new class of applications will be defined as marginalize-a-decomposable-function (MDF) problems for which the computational benefits are to be extended.

In Section 5.2, multidimensional matrices are introduced as a natural and straightforward tool to represent discrete multivariate functions. Moreover, a basic form for a decomposable multivariate function, adapted from a stream of work in machine learning for its convenience, will be introduced. It will be shown that once a decomposition algorithm is developed for this basic form, other forms of decompositions can also be obtained by repeatedly invoking this algorithm on the subfunctions.

In Section 5.3, methods will be developed to represent decomposable functions as factorable functions at a higher dimension by introducing latent variables. The conditions under which such an operation will remain efficient will be discussed, and the decomposable representations will be manipulated to show that a decomposable function can be represented as a product of matrices. Moreover, although factor graphs can successfully capture the structure of factorable functions and not decomposable functions, the techniques developed in this section will lead to a factor graph representation that can capture the decomposability structure and lend itself to efficient computations.

The matrix representations of multivariate functions as well as the matrix product view of decomposition developed in Section 5.3 will constitute the basis of decomposition algorithms that will be developed in Section 5.4. More specifically, having reduced the decomposition problem to a matrix factorization problem, techniques such as singular value decomposition (SVD) and nonnegative matrix factorization (NMF) algorithms will be borrowed from the literature for developing the machinery for decomposition.

5.1 Efficient Marginalization

The problems that require marginalizing multivariate functions over some or all of their variables can be computationally intractable since the dimensionality over which the summation is performed increases exponentially with the number of variables. These problems become instances of marginalize-a-product-function (MPF) problems formally defined in [1, 41], for which well-known efficient algorithms exist, when the multivariate functions to be marginalized are factorable into local functions that depend only on subsets of variables. The efficiency of factorability stems from the possibility of distributing the summation over local functions hence reducing the dimensionality over which these operations are performed. For example, consider a 5-th order Volterra kernel of a nonlinear time invariant system which has a factorization of the form

$$h[n_1, n_2, n_3, n_4, n_5] = \psi_A[n_1, n_4, n_5]\psi_B[n_2, n_3, n_4], \tag{5.1}$$

where ψ_A and ψ_B are both multivariate functions themselves. In order to compute the corresponding 5-th Volterra term $y_5[n]$ in the series expansion for the output when the input is $x[n]$, the marginalization can be distributed over the two local functions

ψ_A and ψ_B instead of performing the marginalization over all possible five-tuples $\{n_1, \ldots, n_5\}$ at once,

$$
\begin{aligned}
y_5[n] &= \sum_{n_1}\sum_{n_2}\sum_{n_3}\sum_{n_4}\sum_{n_5} \psi_A[n_1, n_4, n_5]\psi_B[n_2, n_3, n_4] \prod_{k=1}^{5} x[n - n_k] \\
&= \sum_{n_4} x[n-n_4] \left(\sum_{n_1}\sum_{n_5} \psi_A[n_1, n_4, n_5] x[n-n_1] x[n-n_5] \right) \left(\sum_{n_2}\sum_{n_3} \psi_B[n_2, n_3, n_4] x[n-n_2] x[n-n_3] \right)
\end{aligned} \tag{5.2}
$$

where each local sum will be performed on a subset of variables, in which case much fewer computations will be required. Although the computational savings depend highly on the form of the actual factorization, the number of variables in the factors and their alphabet sizes, performing computations by exploiting the factorization will increase efficiency in general.

Following a naming convention similar to MPF, marginalizations over decomposable functions will be referred to as marginalize-a-decomposable-function (MDF) in this thesis. In order to benefit from the existing algorithms for MPF problems exploiting factorability, decompositions will be formulated as a generalization of factorization when the function is artificially and temporarily carried to higher dimensions through the introduction of latent variables. This will allow re-expressing MDF problems as MPF problems and apply the same efficient algorithms with little overhead due to the introduction of the latent variables.

5.2 Decomposable Representations of Discrete Multivariate Functions

A bivariate signal $F(\cdot, \cdot)$ of two variables $\{x, y\}$ can be represented either analytically in terms of x and y or specified for each and every value of the pair (x, y) if the alphabets of these variables are finite and discrete, i.e., they can only take finitely many distinct values. In many circumstances, an analytic expression for $F(x, y)$ is undefined, unknown or not meaningful such as with a natural image consisting of pixel values where the variations between pixel values cannot be modeled by an analytic function of the pixel coordinates. In such an image, the variables x and y correspond to coordinates of the pixels and they both take a finite number of discrete values, consecutive integers in this case. In other cases, there may be an analytic representation of $F(x, y)$ or the alphabets of the variables may be continuous. In an application where computations are to be performed or data to be stored, a common practice is to take the range of interest for the variables and discretize them, in which case the function values can be recorded as a matrix $\mathbf{M_F}$.

The matrix representation of bivariate discrete functions can be extended similarly to a multidimensional matrix, i.e. tensor, representation for discrete multivariate functions $H(\mathbf{X})$, where

$$\mathbf{X} = \{x_1, x_2, \ldots, x_K\} \tag{5.3}$$

is the set of K discrete variables with finite alphabets. The multidimensional matrix $\mathbf{T_H}$ has K dimensions. The size of $\mathbf{T_H}$ along its k-th dimension is D_{x_k}, namely the alphabet size of x_k, since it is indexed by x_k. The total number of entries in $\mathbf{T_H}$ is thus

$$D_{\mathbf{X}} = \prod_{k=1}^{K} D_{x_k}, \tag{5.4}$$

i.e., it is equal to the joint alphabet size of the K-tuple $(x_1, \ldots, x_K)$. Given an instantiation of $\mathbf{X}$, namely a set of values for $(x_1, x_2, \ldots)$, the value of a multivariate function $H(\mathbf{X})$ can be obtained by a look-up procedure from the multidimensional matrix $\mathbf{T_H}$ that represents it.

5.2.1 A Basic Decomposable Representation

A basic form that represents a function of a function in the context of multivariate functions is

$$F(x_{(f,1)}, \ldots, x_{(f,M)}, G(x_{(g,1)}, \ldots, x_{(g,N)})) \tag{5.5}$$

where the variable subindices f and g imply correspondence to F and G, respectively. In other words, in this basic decomposable form, G is a multivariate function evaluating to a single value and F is a function of this value along with other variables. The variables are all distinct, i.e., the set of variables of F and that of G are disjoint. This basic form was exploited in the context of machine learning due to its simplicity, in which intermediate meanings for a subset of variables were developed and referred to as *concepts* [57]. In that stream of work, this decomposition was not considered for its potential computational efficiency but rather to obtain groups of variables related to each other through a concept. In this thesis, this basic form will be adapted as the simplest form of decomposability which can be repeated for subfunctions as desired to obtain more sophisticated decompositions for the purposes of computational and representational efficiencies using multivariate functions in MDF problems. This requires obtaining decompositions of this form as the first step.

For any disjoint partitioning $\mathbf{X_A}$ and $\mathbf{X_B}$ of $\mathbf{X}$ satisfying $\mathbf{X_A} \cup \mathbf{X_B} = \mathbf{X}$ and $\mathbf{X_A} \cap \mathbf{X_B} = \emptyset$, $H(\mathbf{X})$ can always be obtained by a two-step tensor look-up procedure by introducing a latent variable c. The first tensor $\mathbf{T_G}$ of dimensionality N corresponds to a sub-function $c = G(\mathbf{X_B})$ and the second tensor $\mathbf{T_F}$ of dimensionality $M+1$ corresponds to a function $F(\mathbf{X_A}, c)$. Here, M and N are the number of elements in $\mathbf{X_A}$ and $\mathbf{X_B}$, respectively. The result will be a decomposable representation in the form

$$H(\mathbf{X_A}, \mathbf{X_B}) = F(\mathbf{X_A}, G(\mathbf{X_B})). \tag{5.6}$$

$\mathbf{T_G}$ can be obtained by defining any one-to-one function $G : \prod_{x_i \in \mathbf{X_B}} X_i \longrightarrow \mathcal{R}_G$, where X_i denotes the alphabet of the variable x_i and $\mathcal{R}_G$ denotes the alphabet of c, i.e. the set of all possible values that G takes over the alphabet of $\mathbf{X_B}$. Once G is chosen freely, F must be specified, for consistency, by

$$F(\mathbf{X_A}, G(\mathbf{X_B})) \triangleq H(\mathbf{X}), \tag{5.7}$$

which constitute the entries of $\mathbf{T_F}$. In this representation, G evaluates to a unique value c for each different instantiation of $\mathbf{X_B}$ since it is a one-to-one mapping, and that unique value along with the rest of the variables in $\mathbf{X_A}$ unambiguously determines the value of F, which is by definition set to $H(\mathbf{X})$.

5.2.2 Constraints on the size of $\mathcal{R}_G$

The decomposable representation of H, or equivalently of $\mathbf{T_H}$, as in equation (5.6) using a one-to-one function for G does not directly lead to a more compact or efficient representation. This can be seen from the fact that $\mathbf{T_H}$ has been replaced by a comparably large tensor $\mathbf{T_F}$ in addition to $\mathbf{T_G}$. The size of $\mathbf{T_F}$ is the same as that of $\mathbf{T_H}$ despite having smaller dimensionality of $M + 1$ since the dimension of $\mathbf{T_F}$ that is indexed by the latent variable c is the same size as the number of different configurations that $\mathbf{X_B}$ can have.

For a more efficient and compact representation, $G(\mathbf{X_B})$ has to reveal degeneracies in $H(\mathbf{X})$ with respect to $\mathbf{X_B}$, i.e., when $H(\mathbf{X}) = H(\mathbf{X_A}, \mathbf{X_B})$ takes the same values for different instantiations of $\mathbf{X_B}$, $G(\mathbf{X_B})$ should be specified in a way to reflect this by taking on the same value for those instantiations rather than being specified as a one-to-one mapping.

To illustrate this observation, consider the discrete multivariate function example

$$H(x_1, x_2, x_3) = \max(x_1, \min(x_2, x_3)) \tag{5.8}$$

given in [57]. The variables $x_1, x_2 \in \{lo, med, hi\}$, and $x_3 \in \{lo, hi\}$ index a three dimensional tensor $\mathbf{T_H}$ of dimensions $3 \times 3 \times 2$. For clarity and the sake of numerical manipulations, the values $\{lo, med, hi\}$ will be replaced by $\{-5, 0, 5\}$, which corresponds to the voltages in a bipolar signaling scheme with $hi = 5V$. For illustration, $\mathbf{T_H}$ has been unfolded into a partition matrix in Table 5.1 where the partition sets are $\mathbf{X_A} = \{x_1\}$ and $\mathbf{X_B} = \{x_2, x_3\}$, i.e., it is shown as a matrix where the rows are indexed by different values of x_1 and the columns are indexed by different values of the pair $\{x_2, x_3\}$. This representation conveniently reveals the degeneracies in H with respect to the variables $\{x_2, x_3\}$ in that for six different values of the pair, there are only three distinct columns in this specific tabular representation. This observation would have suggested defining a sub-function $c = G(x_2, x_3)$ that only takes three distinct values for all values of $\{x_2, x_3\}$ as shown at the bottom of Table 5.1, for example to choose $\mathcal{R}_G = \{1, 2, 3\}$. The new representation of $H(x_1, x_2, x_3)$ consists of two

Table 5.1: Partition matrix for $\mathbf{X_A} = \{x_1\}$ and $\mathbf{X_B} = \{x_2, x_3\}$

	x_2	-5	-5	0	0	5	5
x_1	x_3	-5	5	-5	5	-5	5
-5		-5	-5	-5	0	-5	5
0		0	0	0	0	0	5
5		5	5	5	5	5	5
$G(x_2, x_3)$		1	1	1	2	1	3

Table 5.2: Decomposed representation for $H(x_1, x_2, x_3)$ in Table 5.1. (a) Values for $G(x_2, x_3)$ (b) Values for $F(x_1, G) = H(x_1, x_2, x_3)$.

(a)

(x_2, x_3)	-5	5
-5	1	1
0	1	2
5	1	3

(b)

(x_1, G)	1	2	3
-5	-5	0	5
0	0	0	5
5	5	5	5

smaller tensors $\mathbf{T_G}$ and $\mathbf{T_F}$, in fact matrices since they are two dimensional as they are shown in Table 5.2.

In the specific example of equation (5.8), the total number of parameters or tensor entries to represent H has changed from 18 to 15. For larger alphabets and numbers of variables, the savings can be substantial. The limit on the alphabet size of c may be different for efficient representations of the multivariate function than the limit for efficient computations using the function in MDF problems as discussed in later sections.

5.2.3 Other Decomposable Representations

The basic decomposition of a multivariate function $H(\mathbf{X})$ into functions of a disjoint partition $\{\mathbf{X_A}, \mathbf{X_B}\}$ as in equation (5.6) is only one of the possible forms for a decomposition. Nevertheless, in this section, developing a basic one-step decomposition algorithm to yield this form will be shown to suffice to obtain many other more structured decomposition forms when it is applied iteratively. For example, a decomposition of the form

$$H(\mathbf{X}) = F(\mathbf{X_{A'}}, G_1(\mathbf{X_{B'}}, G_2(\mathbf{X_{C'}}))), \tag{5.9}$$

where $\{\mathbf{X_{A'}}, \mathbf{X_{B'}}, \mathbf{X_{C'}}\}$ is a disjoint partition of $\mathbf{X}$, can be obtained by applying the one-step decomposition algorithm to $H(\mathbf{X})$ with $\mathbf{X_A} = \mathbf{X_{A'}}$ and $\mathbf{X_B} = \mathbf{X_{B'}} \cup \mathbf{X_{C'}}$ to obtain the intermediate decomposition

$$H(\mathbf{X}) = F(\mathbf{X_{A'}}, G_{int}(\mathbf{X_{B'}} \cup \mathbf{X_{C'}})), \tag{5.10}$$

and re-applying the algorithm to $G_{int}(\mathbf{X_{B'}} \cup \mathbf{X_{C'}})$ to obtain the final decomposition in equation (5.9). Similarly, a decomposition of the form

$$H(\mathbf{X}) = F(G_A(\mathbf{X_{A'}}), G_B(\mathbf{X_{B'}}), \mathbf{X_{C'}}), \tag{5.11}$$

can be obtained by first applying the one-step decomposition algorithm to $H(\mathbf{X})$ with $\mathbf{X_A} = \mathbf{X_{A'}} \cup \mathbf{X_{C'}}$ and $\mathbf{X_B} = \mathbf{X_{B'}}$ to get an intermediate decomposition

$$H(\mathbf{X}) = F_{int}(\mathbf{X_{A'}} \cup \mathbf{X_{C'}}, G_B(\mathbf{X_{B'}})). \tag{5.12}$$

In the second step, defining a new variable "g_B" with an alphabet that is the same as possible values for $G_B(\mathbf{X_{B'}})$ allows the application of the same decomposition algorithm to F_{int} with $\mathbf{X_A} = \mathbf{X_{C'}} \cup \{g_B\}$ and $\mathbf{X_B} = \mathbf{X_{A'}}$ to obtain the decomposition in equation (5.11).

Decompositions into functions of non-disjoint partitions $\{\mathbf{X_A}, \mathbf{X_B}\}$ are not going to be considered in this thesis as they deteriorate the efficiency of operations that will be performed in an MDF problem. This stems from the fact that non-empty intersections between variable sets implies dependency of sub-functions in the decomposition and not a complete separation of the computation into easier local computations as will be discussed in the next section where decomposability will be formulated as a generalization of factorability.

5.3 Decomposition as a Generalization of Factorization

In this section, decomposable multivariate functions will be artificially carried to a higher dimension by introducing latent variables, where they can be represented as the marginalization of a factorable function over the latent variable. This procedure, which will be referred to as δ-decomposition since it involves the δ-function, will be shown to lead to representational and computational efficiencies if this function is to be used in a marginalize-a-decomposable-function (MDF) problem. Moreover, this decomposition will be extended to involve more general functions than the δ-function and make it possible to reduce the problem of functional decomposition to matrix factorization. This will establish a relationship between decomposability of a multivariate function and the factorability of a higher dimensional related function, a theme which also appeared in the case of polynomial decomposition, suggesting a more general relationship between these two operations for possibly larger classes of functions. Finally, the equivalence between decomposability and factorability at a higher dimensionality will be exploited to facilitate a factor graph representation of decomposable functions on which algorithms such as the sum-product algorithm can be run efficiently.

5.3.1 δ-Decomposition

A multivariate function with a basic decomposable representation of the form in equation (5.6) over a disjoint partition $\{\mathbf{X_A}, \mathbf{X_B}\}$ can be equivalently expressed as

$$\begin{aligned} H(\mathbf{X_A}, \mathbf{X_B}) &= F(\mathbf{X_A}, G(\mathbf{X_B})) \\ &= \sum_{c \in \mathcal{R}_G} F(\mathbf{X_A}, c) \cdot \delta(c, G(\mathbf{X_B})) \end{aligned} \tag{5.13}$$

where c is a latent variable with the alphabet $\mathcal{R}_G$, i.e., it represents all different possible values that $G(\mathbf{X_B})$ can take, and

$$\delta(x,y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}.$$

Equation (5.13) is a direct application of the sifting property of the $\delta(\cdot,\cdot)$ function –the summand will be nonzero only when $c = G(\mathbf{X_B})$, in which case it will yield $F(\mathbf{X_A}, c)$ establishing the desired equality. This representation of $H(\mathbf{X_A}, \mathbf{X_B})$ is in fact equivalent to the marginalization of a multivariate function $\hat{H}(\mathbf{X_A}, \mathbf{X_B}, c)$ over the latent variable c where

$$\hat{H}(\mathbf{X_A}, \mathbf{X_B}, c) = F(\mathbf{X_A}, c) \cdot \delta(c, G(\mathbf{X_B})), \tag{5.14}$$

i.e.,

$$H(\mathbf{X_A}, \mathbf{X_B}) = \sum_{c \in \mathcal{R}_G} \hat{H}(\mathbf{X_A}, \mathbf{X_B}, c). \tag{5.15}$$

The representation of $H(\mathbf{X})$ using $\delta(\cdot,\cdot)$ function as in equation (5.13) will be referred to as δ-decomposition in the sequel.

5.3.2 Representational Efficiency Using δ-Decomposition

In order to compare the total tensor sizes required to represent a multivariate function $H(\mathbf{X_A}, \mathbf{X_B})$ in its original form and using its δ-decomposition as in equation (5.13), consider the joint alphabet size $D_{\mathbf{X}}$ of the variables in $\mathbf{X} = \{x_k\}_{k=1}^{K}$ as defined in equation (5.4). $D_{\mathbf{X}}$ simply denotes the number of different configurations that a K-tuple $(x_1, \ldots, x_K)$ can take. A direct representation of $H(\mathbf{X_A}, \mathbf{X_B})$ would require a tensor of size $(D_{\mathbf{X_A}} \cdot D_{\mathbf{X_B}})$, which is equal to the joint alphabet size of all the variables in $\mathbf{X}$. On the other hand, a tensor of the size $D_{\mathbf{X_B}}$ to represent $G(\mathbf{X_B})$ and a tensor of size $D_{\mathbf{X_A}} \cdot |\mathcal{R}_G|$ to represent $F(\mathbf{X_A}, c)$ will suffice to represent its δ-decomposition. Therefore, a δ-decomposition with a disjoint partition $\{\mathbf{X_A}, \mathbf{X_B}\}$ is considered efficient if the total size of these two tensors is smaller than that required for the direct representation, i.e.,

$$|\mathcal{R}_G| < \frac{D_{\mathbf{X_A}} \cdot D_{\mathbf{X_B}} - D_{\mathbf{X_B}}}{D_{\mathbf{X_A}}}. \tag{5.16}$$

This comparison emphasizes the importance of having a small alphabet size for c for an efficient representation, which is consistent with the discussion in Section 5.2.2. The constraints for efficient representation versus efficient MDF computations are closely related by not identical as will be shown in Section 5.3.3, however $|\mathcal{R}_G|$ plays an important role for both kinds of efficiencies introduced by decomposition.

5.3.3 Computational Efficiency Using δ-Decomposition

$\hat{H}$ in equation (5.14) is a multivariate function that is factorable into local multivariate functions F and δ each depending only on a subset of its variables at the expense of increasing the dimensionality of H by one through the introduction of the latent variable c. In MDF problems, using its representation through $\hat{H}$ as in equation (5.15) instead of H directly will lead to more efficient computations due to the factorability of $\hat{H}$ if the alphabet of c is below a certain size, effectively transforming the problem into an equivalent MPF problem. More specifically, the marginalization of H over all of its variables cannot be distributed over local summations since it is not directly factorable

$$\sum_{\mathbf{X_A}} \sum_{\mathbf{X_B}} H(\mathbf{X_A}, \mathbf{X_B}), \tag{5.17}$$

whereas the summation over $\hat{H}$ can be obtained using the combination of local summations over smaller sets of variables

$$\begin{aligned} &\sum_{\mathbf{X_A}} \sum_{\mathbf{X_B}} \left(\sum_{c} \hat{H}(\mathbf{X_A}, \mathbf{X_B}, c) \right) \\ &= \sum_{c} \left(\sum_{\mathbf{X_A}} F(\mathbf{X_A}, c) \right) \left(\sum_{\mathbf{X_B}} \delta(c, G(\mathbf{X_B})) \right). \end{aligned} \tag{5.18}$$

The marginalization directly performed over H as in equation (5.17) spans an alphabet size of $(D_{\mathbf{X_A}} \cdot D_{\mathbf{X_B}})$. On the other hand, factorability of $\hat{H}$ allows the summation to be performed in two smaller sub-summations as in equation (5.18) repeated as many times as $|\mathcal{R}_G|$ for each different value of c, resulting in $|\mathcal{R}_G| \cdot (D_{\mathbf{X_A}} + D_{\mathbf{X_B}})$ additions. More specifically, computations using $\hat{H}$ will be more preferable if

$$|\mathcal{R}_G| < \frac{D_{\mathbf{X_A}} \cdot D_{\mathbf{X_B}}}{D_{\mathbf{X_A}} + D_{\mathbf{X_B}}}, \tag{5.19}$$

which is usually a very easy constraint to satisfy as the right hand side of inequality (5.19) increases exponentially with the number of variables.

The separation of the marginalization into local summations in equation (5.18) would not be possible if the partition sets $\mathbf{X_A}$ and $\mathbf{X_B}$ were non-disjoint since neither F nor δ could be factored out from the local summations over $\mathbf{X_A}$ or $\mathbf{X_B}$. In such a case, the summation over the variables in $\mathbf{X_A} \cap \mathbf{X_B}$ needs to be performed separately, similar to the summation over c, which therefore causes the number of required computations to multiply by their joint alphabet sizes. In order to avoid this reduction in efficiency, decompositions over non-disjoint partitions will be avoided.

5.3.4 General Decomposition

The efficiency of the computations using $\hat{H}$ in the equivalent MPF problem stems from its factorability as in equation (5.14). In fact, the computational efficiency is preserved if $\delta(c, G(\mathbf{X_B}))$ is replaced by a more general function $\tilde{G}(c, \mathbf{X_B})$ which can take values other than $\{0, 1\}$,

$$\hat{H}(\mathbf{X_A}, \mathbf{X_B}, c) = F(\mathbf{X_A}, c) \cdot \tilde{G}(c, \mathbf{X_B}), \tag{5.20}$$

since computations similar to equation (5.18) can still be performed over local summations. This leads to a more general representation of H than in equation (5.13),

$$H(\mathbf{X_A}, \mathbf{X_B}) = \sum_{c \in \mathcal{R}_G} F(\mathbf{X_A}, c) \cdot \tilde{G}(c, \mathbf{X_B}). \tag{5.21}$$

Therefore, δ-decomposition becomes a special case of the general decomposition with

$$\tilde{G}(c, \mathbf{X_B}) = \delta(c, G(\mathbf{X_B})). \tag{5.22}$$

Although computational efficiency is preserved when δ-decomposition is generalized as in equation (5.21), representational efficiency is not preserved as representing a general function $\tilde{G}(c, \mathbf{X_B})$ requires a larger tensor than the one required to represent $G(\mathbf{X_B})$ by a factor of $|\mathcal{R}_G|$. Therefore, the constraint on the alphabet size of c takes the same form of equation (5.19) for both efficient representation and computation using general decomposition.

5.3.5 Decomposition as a Matrix Factorization

The relationship in equations (5.13) and (5.21) can be viewed as matrix equations involving three matrices $\mathbf{M_F}$, $\mathbf{M_{\tilde{G}}}$ and $\mathbf{M_H}$, the entries of which consist of F, $\tilde{G}$ and H;

$$\mathbf{M_H} \triangleq \mathbf{M_F} \cdot \mathbf{M_{\tilde{G}}} \tag{5.23}$$

where the rows and columns of $\mathbf{M_F}$ are indexed by different values of $\mathbf{X_A}$ and c, respectively; and the rows and columns of $\mathbf{M_{\tilde{G}}}$ are indexed by different values of c and $\mathbf{X_B}$, respectively. Similarly, rows and columns of $\mathbf{M_H}$ are indexed by $\mathbf{X_A}$ and $\mathbf{X_B}$. In δ-decomposition where $\tilde{G}(c, \mathbf{X_B}) = \delta(c, G(\mathbf{X_B}))$, the entries of $\mathbf{M_{\tilde{G}}}$ equal 1 wherever $c = G(\mathbf{X_B})$ and 0 elsewhere, while they can take any real value for the general decomposition. The constraint that c has a small alphabet size corresponds to $\mathbf{M_F}$ having few columns and $\mathbf{M_{\tilde{G}}}$ few rows. Since the columns of $\mathbf{M_H}$ are a linear combination of the columns of $\mathbf{M_F}$, the rank of H is at most c and the representation of equation (5.23) reveals the degeneracy in $\mathbf{M_H}$, the original representation of $H(\mathbf{X_A}, \mathbf{X_B})$. This observation will be used to obtain general decomposition algorithms in the following sections.

5.3.6 Factor Graph Representation of Decomposable Multivariate Functions

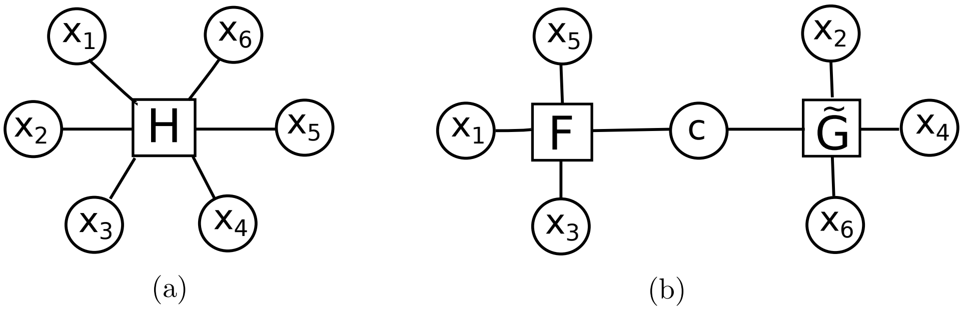
Factorable multivariate functions are conveniently expressed using a factor graph, a bipartite graph with variable nodes and factor nodes in which each factor node is connected to the variable nodes that it takes as a variable [31]. This representation of the function leads to a visualization of dependencies of factors on variables and more importantly suggests an order for computations that is the basis for several efficient message passing algorithms for MPF problems including the sum-product algorithm. The messages that are passed from factor nodes to variable nodes are usually the computational bottleneck as they require marginalization over the joint alphabets of all but one of their variables. The overall complexity increases exponentially with the maximum number of variables that are connected to the same factor, therefore it is important to be able to break factors into those which depend on only few variables.

Although factorability of a multivariate function can be directly represented by factor graphs, decomposability of a function $H(\mathbf{X})$ cannot be summarized and visualized using a factor graph in its original form, and the efficient message passing algorithms cannot be used directly. However, the factorable function $\hat{H}(\mathbf{X}, c)$ can be viewed as a factorable representative of $H(\mathbf{X})$ with a higher dimensionality since the result of any marginalization on $\hat{H}$ that includes the marginalization over c will be indistinguishable from those obtained using H directly. Therefore, the factor graph representation of $\hat{H}$ can be treated as that of H allowing the use of efficient message passing algorithms. As an example, Figure 5-1a illustrates the factor graph representation of a decomposable function

$$H(x_1, \ldots, x_6) = F(x_1, x_3, x_5, G(x_2, x_4, x_6)) \tag{5.24}$$

in its original form which fails in reflecting the structure it has, whereas Figure 5-1b illustrates the representation of this function in terms of its factorable representative

$$\hat{H}(x_1, \ldots, x_6, c) = F(x_1, x_3, x_5, c) \cdot \tilde{G}(c, x_2, x_4, x_6). \tag{5.25}$$



(a)

(b)

Figure 5-1: An example of a factor graph (a) before decomposition (b) after decomposition.

Both δ-decomposition and general decomposition allow increasing the dimensionality of a decomposable function by introducing a latent variable c to obtain a factorable representative on which efficient algorithms can be employed for MDF computations under reasonable constraints on the alphabet size of c. From this perspective, decomposability can be interpreted as the generalization of factorability and can exploit the computational efficiencies that factorability offers.

5.4 Decomposition Methods

A concept central to the multivariate function decomposition is a *partition matrix*, a convenient tool that was originally described in [57]. For a multivariate function $H(\mathbf{X})$ and a disjoint partition of its variables $\{\mathbf{X_A}, \mathbf{X_B}\}$, the partition matrix is a matrix representation of the function where the rows are indexed by $\mathbf{X_A}$ and the columns are indexed by $\mathbf{X_B}$. The partition matrix can be viewed as unfolding the high dimensional tensor $\mathbf{T_H}$ that represents $H(\mathbf{X})$ into a matrix form. An example of a partition matrix was given in Table 5.1 for the function in equation (5.8) with the partition sets $\mathbf{X_A} = \{x_1\}$ and $\mathbf{X_B} = \{x_2, x_3\}$.

In [57], partition matrices were used to decompose multi-valued functions in the context of machine learning. However, decomposition using partition matrices was not exploited for representational or computational efficiency. Rather, it was used as a tool for inferring missing observations and for obtaining meaningful intermediate *concepts*, i.e., functions that can be interpreted as the summary of the state of a subset of variables. In the sequel, multivariate functions are assumed to be known or observed for all possible configurations of the variables in $\mathbf{X}$ and decomposition methods will be developed to serve a different and more general purpose, namely to allow efficient representation and computation using decomposable multivariate functions in MDF problems. Once a decomposition is obtained, it will be checked whether the constraint in equation (5.16) or (5.19) is satisfied to declare a decomposition efficient for representational or computational purposes whereas approximate decomposition algorithms can impose these constraints if they are not readily met at the expense of a less accurate representation.

5.4.1 Exact δ-Decomposition Algorithm

Obtaining the Matrices for Sub-Functions F and G

In Section 5.3, an efficient decomposition for a multivariate function $H(\mathbf{X})$ was argued to be one of the form $F(\mathbf{X_A}, G(\mathbf{X_B}))$ with a disjoint partition $\{\mathbf{X_A}, \mathbf{X_B}\}$ and a small alphabet size for $\mathcal{R}_G$, i.e., the set of possible values for $G(\mathbf{X_B})$. Given a disjoint partition $\{\mathbf{X_A}, \mathbf{X_B}\}$ and the corresponding partition matrix $\mathbf{M_H}$ of $H(\mathbf{X})$, the identical columns in the partition matrix imply that for any of their column indices $\mathbf{X_B}$, the value of $F(\mathbf{X_A}, G(\mathbf{X_B}))$ will be the same for a given $\mathbf{X_A}$, therefore G can be specified in way to yield the same value of $c = G(\mathbf{X_B})$ for all these column indices. In this approach, the number of distinct columns in the partition matrix corresponds to the alphabet size for $\mathcal{R}_G$. In the example given in Table 5.1, all columns except the ones indexed by $\mathbf{X_B} = \{0, 5\}$ and $\mathbf{X_B} = \{5, 5\}$ are identical, therefore all of these column indices $\mathbf{X_B}$ were assigned the same value under G,

$$G(-5, -5) = G(-5, 5) = G(0, -5) = G(5, -5) \triangleq 1. \tag{5.26}$$

The column indices of the other two columns were assigned values of 2 and 3 under the function G. The exact values that column indices $\mathbf{X_B}$ are assigned under G, i.e., the alphabet values of the latent variable c, are arbitrary as along as they are distinct for distinct columns since c will be marginalized in all computations and these values will be merely used as the index for the columns of matrix $\mathbf{M_F}$ and the rows of matrix $\mathbf{M_{\tilde{G}}}$ as discussed in Section 5.3.5.

The first step of the exact δ-Decomposition algorithm given a disjoint partition $\{\mathbf{X_A}, \mathbf{X_B}\}$ is the construction of the partition matrix and assigning distinct values of c to $G(\mathbf{X_B})$ for the values of $\mathbf{X_B}$ that index distinct columns. These distinct columns construct the matrix $\mathbf{M_F}$, or equivalently, specify the function $F(\mathbf{X_A}, c)$. The row of the matrix $\mathbf{M_{\tilde{G}}}$ that corresponds to a particular value of c consists of all zeros except when $c = G(\mathbf{X_B})$, which represents the function $\delta(c, G(\mathbf{X_B}))$. For the specific example of the partition matrix given in Table 5.1,

$$\mathbf{M_F} = \begin{array}{|l|ccc|} \hline x_1, c & 1 & 2 & 3 \\ \hline -5 & -5 & 0 & 5 \\ 0 & 0 & 0 & 5 \\ 5 & 5 & 5 & 5 \\ \hline \end{array} \tag{5.27}$$

and

$$\mathbf{M_{\tilde{G}}} = \begin{array}{|lr|cccccc|} \hline & x_2 & -5 & -5 & 0 & 0 & 5 & 5 \\ c, & x_3 & -5 & 5 & -5 & 5 & -5 & 5 \\ \hline 1 & & 1 & 1 & 1 & 0 & 1 & 0 \\ 2 & & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array} \tag{5.28}$$

where the indices for the rows and columns are included and highlighted for clarity. The product of these two matrices as in equation (5.23) yields the partition matrix, $\mathbf{M_H}$, that is given in Table 5.1.

Once a δ-decomposition is obtained, the constraints in equations (5.16) and (5.19) can be checked to declare whether it is efficient for representing H or performing MDF computations using it. For the specific example in equation (5.8), the associated alphabet sizes are $D_{\mathbf{X_A}} = 3$, $D_{\mathbf{X_B}} = 6$ and $\mathcal{R}_G = 3$, therefore the constraint for efficient representation in equation (5.16) holds whereas the one for computations in (5.19) does not hold. In cases where the obtained decomposition is declared inefficient for a specific purpose, either another decomposition is explored using different partition sets or approximate decomposition algorithms are explored with the same partition sets that can impose a pre-specified alphabet size for the latent variable c.

Determining a Disjoint Partition $\{\mathbf{X_A}, \mathbf{X_B}\}$

In the construction of the decomposition in Table 5.2, a suitable partition was easily chosen to illustrate the inherent degeneracy in H as its canonical form in equation (5.8) is known. However, H may be specified by its corresponding tensor $\mathbf{T_H}$ and a canonical form is not always given a priori. In such cases, it may be necessary to exhaust all possible variable partitions to obtain an efficient decomposition. The number of such partitions grow exponentially with the number of variables and it is usually formidable to exhaust all possible partitions for many variables. In [10], considering partitions where $\mathbf{X_B}$ consists of only two or three closely related variables was proposed as it may be possible to associate an intermediate *concept*, i.e., meaning, to $G(\mathbf{X_B})$, in which case the number of possible partitions increase as a polynomial in the number of variables rather than exponentially. In an example given in [10], housing loan applicants are prioritized into five categories depending on twelve variables. A supervised decomposition with at most three variable partition sets yielded sub-functions $G(\mathbf{X_B})$ where, for instance, $\mathbf{X_B}$ includes the number of children, employment status and earnings, and the sub-function $G(\mathbf{X_B})$ can be interpreted as financial status. In other applications, a natural partitioning is implied from the context and this helps eliminate a great portion of irrelevant partitions or even directly suggests the most efficient partitioning. For example, in Bayesian inference problems, the joint distribution of unobserved parameters and observed variables can be decomposed as a function of observed variables and a sub-function of unobserved parameters.

5.4.2 Approximate δ-Decomposition Algorithms

δ-decomposition is efficient when the alphabet size of the latent variable c is small, or equivalently when there are few distinct columns in the partition matrix, as discussed in the previous section. However expecting few distinct columns in a partition matrix is not realistic for most multivariate functions and most variable partitions, especially when the function $H(\mathbf{X})$ takes values from a large range set such as the set of real numbers. In other cases, the function H may have been contaminated by noise causing identical columns to seem distinct. For example, the function in equation (5.8) may be observed through a non-ideal system yielding noisy values for $H(\mathbf{X})$,

$$\begin{array}{|cc|cccccc|} \hline & x_2 & -5 & -5 & 0 & 0 & 5 & 5 \\ x_1 & x_3 & -5 & 5 & -5 & 5 & -5 & 5 \\ \hline -5 & & -4.73 & -5.00 & -5.12 & 0.11 & -5.01 & 4.86 \\ 0 & & -0.04 & 0.07 & 0.13 & 0.01 & 0.19 & 5.02 \\ 5 & & 5.20 & 4.92 & 5.01 & 5.01 & 4.79 & 5.14 \\ \hline \end{array}. \tag{5.29}$$

In such cases, one approach for efficient decomposition is to cluster columns into a few classes and then replace each member in a class with a single representative column for that class. This can be viewed as a form of vector quantization. For example, a simple and straightforward scheme is to group similar columns and replace them with their vector average within each group. This leads to an approximate functional decomposition as the function values are modified to expose near-degeneracies.

In approximate δ-decomposition, the number of clusters for column vectors of the partition matrix and the clustering method can be chosen in the application context. The number of clusters corresponds to the number of distinct columns after the decomposition is completed, hence it also corresponds to the alphabet size of the latent variable c. The constraints in equations (5.16) and (5.19) can be used to determine an upper bound for the number of clusters for an efficient decomposition. Using k-means algorithm to cluster the column vectors in the partition table given in equation (5.29) into three groups minimizing the l_2 norm of the clustering error results in the approximate function values

$$\begin{array}{|ll|cccccc|}\hline & x_2 & -5 & -5 & 0 & 0 & 5 & 5 \\ x_1 & x_3 & -5 & 5 & -5 & 5 & -5 & 5 \\ \hline -5 & & -4.97 & -4.97 & -4.97 & 0.11 & -4.97 & 4.86 \\ 0 & & 0.09 & 0.09 & 0.09 & 0.01 & 0.09 & 5.02 \\ 5 & & 4.98 & 4.98 & 4.98 & 5.01 & 4.98 & 5.14 \\ \hline \end{array}. \tag{5.30}$$

The matrix $\mathbf{M_F}$, which represents $F(\mathbf{X_A}, c)$ and includes distinct columns in the partition matrix becomes

$$\mathbf{M_F} = \begin{array}{|l|ccc|}\hline x_1, c & 1 & 2 & 3 \\ \hline -5 & -4.97 & 0.11 & 4.86 \\ 0 & 0.09 & 0.01 & 5.02 \\ 5 & 4.98 & 5.01 & 5.14 \\ \hline \end{array} \tag{5.31}$$

and $\mathbf{M_{\tilde{G}}}$ that represents $\delta(c, G(\mathbf{X_B}))$ remains the same as that given in equation (5.28).

For different applications, other clustering algorithms such as k-medoids or affinity propagation can be used, which also perform the clustering by minimizing other error norms or *dissimilarity* measures. This may be beneficial when l_2 norm is not a suitable choice such as approximating probability density function values, where KL-divergence may be more appropriate.

5.4.3 Exact General Decomposition Algorithm

In δ-decomposition, the matrix $\mathbf{M_{\tilde{G}}}$ consists of only the values $\{0, 1\}$, where the row and column indices of nonzero entries can be interpreted as pointers to which column in the partition matrix $\mathbf{M_H}$ is identical to which column in $\mathbf{M_F}$ since $\mathbf{M_H} = \mathbf{M_F} \cdot \mathbf{M_{\tilde{G}}}$. On the other hand, in general decomposition, the columns of $\mathbf{M_F}$ can be viewed as basis vectors the linear combinations of which constitutes the columns of the partition matrix $\mathbf{M_H}$, where the linear combination coefficients are given as the columns of $\mathbf{M_{\tilde{G}}}$.

The number of basis vectors that are required is the same as the column rank of the partition matrix, which also corresponds to the alphabet size of the latent variable c. A partition matrix where all the columns are distinct will not yield an efficient δ-decomposition since the alphabet size of c equals the actual number of columns in the partition matrix $\mathbf{M_H}$, whereas general decomposition can exploit the low rank of the partition matrix and requires only an alphabet size equal to the rank of $\mathbf{M_H}$. The resulting matrix $\mathbf{M_F}$ for general decomposition will have fewer columns than that for a δ-decomposition and the matrix $\mathbf{M_{\tilde{G}}}$ will consist of more general values other than $\{0, 1\}$.

Given a disjoint partition $\{\mathbf{X_A}, \mathbf{X_B}\}$ for a function $H(\mathbf{X_A}, \mathbf{X_B})$, the exact general decomposition consists of obtaining the corresponding partition matrix $\mathbf{M_H}$, determining its rank r and expressing it as the product $\mathbf{M_F} \cdot \mathbf{M_{\tilde{G}}}$, where $\mathbf{M_F}$ has r columns and $\mathbf{M_{\tilde{G}}}$ has r rows. These matrices represent $F(\mathbf{X_A}, c)$ and $\tilde{G}(c, \mathbf{X_B})$, respectively, and their product corresponds to the representation of $H(\mathbf{X})$ as given in equation (5.21).

There are many matrix factorization algorithms available in the literature that will yield the so called *rank factorization* of the partition matrix $\mathbf{M_H}$. Although it is not the most computationally efficient method, singular value decomposition suggests a factorization in terms of its singular values and left and right singular vectors. The singular value decomposition of $\mathbf{M_H}$ is given by

$$\mathbf{M_H} = U \cdot \Sigma \cdot V^T \tag{5.32}$$

where the columns of U are the left singular vectors, the columns of V are the right singular vectors and diagonal elements of Σ are the singular values of $\mathbf{M_H}$. An equivalent representation is

$$\mathbf{M_H} = \sum_{i=1}^{r} u_i \sigma_i v_i^T \tag{5.33}$$

where u_i and v_i are left and right singular vectors corresponding to the singular value σ_i. With this representation of $\mathbf{M_H}$, the matrix $\mathbf{M_F}$ can be chosen to consist of columns $\{u_1\sigma_1, \ldots, u_r\sigma_r\}$ and $\mathbf{M_{\tilde{G}}}$ to consist of rows $\{v_1^T; \ldots; v_r^T\}$. This choice of the matrix pair $\mathbf{M_F}$ and $\mathbf{M_{\tilde{G}}}$ suggested by singular value decomposition is by no means unique as any invertible matrix $\mathbf{A}$ of the correct size will yield another valid pair,

$$\mathbf{M_H} = (\mathbf{M_F A}) \cdot (\mathbf{A}^{-1}\mathbf{M_{\tilde{G}}}) \triangleq \mathbf{M'_F} \cdot \mathbf{M'_{\tilde{G}}}. \tag{5.34}$$

A perturbation of $\mathbf{\Delta M_F}$ on $\mathbf{M_F}$ leads to an error $\mathbf{\Delta M_H} = \mathbf{\Delta M_F} \cdot \mathbf{M_{\tilde{G}}}$. Similarly, a perturbation of $\mathbf{\Delta M_{\tilde{G}}}$ on $\mathbf{M_{\tilde{G}}}$ leads to an error $\mathbf{\Delta M_H} = \mathbf{M_F} \cdot \mathbf{\Delta M_{\tilde{G}}}$. For multivariate functions, the sensitivity analysis for polynomials in Chapter 3 where compositions were also expressed in terms of matrix equations can be extended from the product of a matrix and a vector to a product of matrices. Different choices of an invertible matrix $\mathbf{A}$ in equation (5.34) parallels the method of obtaining equivalent polynomial compositions, which possibly leads to lower sensitivities.

5.4.4 Approximate General Decomposition Algorithms

General decomposition of a multivariate function $H(\mathbf{X})$ relaxes the constraint on the partition matrix to have few distinct columns and rather requires it to be a low rank matrix for an efficient decomposition. In certain cases, the partition matrix may not be rank deficient or its rank r may not be low enough to yield a desirable alphabet size for c that will meet the efficiency requirement given in equation (5.19). In an approximate general decomposition scheme, an alphabet size for c, or equivalently a rank for the partition matrix, $\hat{r} < r$ can be imposed by modifying the function values in the partition matrix. Similar to the development of the exact general decomposition algorithm, singular value decomposition in equation (5.33) suggests an approximate partition matrix $\mathbf{\hat{M}_H}$ that is of lower rank and a corresponding approximate general decomposition where $\mathbf{\hat{M}_H}$ is an optimal approximation to $\mathbf{M_H}$ with respect to the Frobenius norm. More specifically, for a pre-specified rank $\hat{r} < r$,

$$\mathbf{\hat{M}_H} = \sum_{i=1}^{\hat{r}} u_i \sigma_i v_i^T \tag{5.35}$$

where $\sigma_1, \ldots, \sigma_{\hat{r}}$ are the largest $\hat{r}$ singular values. One particular choice for the matrix $\mathbf{M}_\mathbf{F}$ consists of columns $\{u_1\sigma_1, \ldots, u_{\hat{r}}\sigma_{\hat{r}}\}$ and for $\mathbf{M}_{\tilde{\mathbf{G}}}$ consists of rows $\{v_1^T; \ldots; v_{\hat{r}}^T\}$ while other choices can be obtained as in equation (5.34). This is closely related to the principal component analysis method that treats the columns of $\mathbf{M}_\mathbf{H}$ as data [8].

5.4.5 Approximate Decomposition of Probability Density Functions

A multivariate function $H(\mathbf{X})$ to be decomposed may correspond to a probability density function involving the variables in $\mathbf{X}$. For an approximate decomposition to be also used as a probability density, the resulting component function values may be required to be nonnegative as well since certain inference methods rely explicitly on their nonnegativity. Therefore, a natural approach to decompose probability densities is to find an element-wise nonnegative approximation $\hat{\mathbf{M}}_\mathbf{H}$ to $\mathbf{M}_\mathbf{H}$ that has the desired rank $\hat{r}$ and use an exact general decomposition algorithm to obtain element-wise nonnegative $\mathbf{M}_\mathbf{F}$ and $\mathbf{M}_{\tilde{\mathbf{G}}}$ as in Section 5.4.3. Using singular value decomposition or principal component analysis approaches as in Section 5.4.4 will in general fail to meet these nonnegativity constraints.

The procedure to find a nonnegative approximation $\hat{\mathbf{M}}_\mathbf{H}$ to $\mathbf{M}_\mathbf{H}$ with low rank is not trivial in general as it is not a convex problem. A set of algorithms for the so called *nonnegative matrix factorization* (NMF) have been developed to find locally optimal lower rank approximations [32, 33]. The nonnegativity of $\hat{\mathbf{M}}_\mathbf{H}$ allows to find an approximation to minimize the KL-divergence from the true distribution, which is a preferred dissimilarity measure for probability distributions and is not defined for negative valued functions.

5.5 Chapter Conclusions

In this chapter, exact and approximate discrete multivariate function decomposition algorithms were developed for efficient representations and computations in the context of marginalize-a-decomposable-function" (MDF) problems. Several signal processing and machine learning applications require marginalization of multivariate functions and factorable multivariate functions have previously proved useful to make these computations efficient by the algorithms developed for marginalize-a-product-function (MPF) problems. Re-formulating decomposability as a generalization of factorability at a higher dimension allowed using well-known mathematical tools to decompose multivariate functions, and reducing MDF problems to MPF problems which can benefit from the algorithms developed for this latter class as an application will be shown in Chapter 6.

Chapter 6

Applications of Composition and Decomposition

In the previous chapters, functional composition and decomposition methods were developed for certain classes of functions as parts of a framework in which these two operations can be exploited more fully and in a more systematic way than it has been traditionally done. As discussed in Chapter 1, some of the benefits of incorporating functional composition and decomposition into signal processing are the possibility to design and analyze modular systems, separate computations into more manageable subcomputations and represent signals more compactly. This chapter illustrates several applications that constitute examples for each of these cases, and how the machinery developed in the previous chapters can be exploited in their development and analysis.

In Section 6.1, modularity in designing filters is explored. As an example for modular filter design, the section will start with revisiting the filter sharpening and re-interpreting it as a form of functional composition. This interpretation will be shown to lead to a more systematic way of sharpening filters as well as improved results compared to the traditional methods. Moreover, the developed framework will allow extending the modularity approach from sharpening a limited class of filters, namely those with a real valued frequency response, to designing modular filters as generalized tapped delay lines using filters with complex-valued frequency responses or even continuous time filters.

Another application that will be introduced within the decomposition framework involves marginalizations over multivariate functions which appear in many contexts in signal processing and machine learning as discussed in Chapter 5. Section 6.2 will illustrate the computational benefits of decomposing a multivariate function in the context of a marginalization problem, namely within the sum-product algorithm in the context of an image denoising example. First, the sum-product algorithm will be reviewed in the context of a factorable probability density function to illustrate how the separation of variables lead to a reduced computational burden. Then, a high dimensional multivariate function that represents a joint probability density of a group of pixels will be decomposed approximately to separate the variables similar to a factorable density in order to carry out the sum-product algorithm more efficiently in a binary text image denoising problem. Approximate decomposition of multivariate functions obtained this way will also suggest a more compact representation.

In Section 6.3, examples of using polynomial composition and decomposition will be given in which compact representations for discrete time sequences are obtained, and modular FIR filters are designed without requiring the pre-specification of a subfilter unlike in the case of filter sharpening. However, the low performance of approximate polynomial decomposition algorithms for general polynomials reduces the practicality of these applications.

6.1 Modular Filter Design

An interesting and useful application for which functional composition and decomposition provide a convenient framework is the design of modular filter structures. VLSI designers are increasingly advocating modularity in their designs, for example by dividing the overall system into either identical or few distinct sub-systems, and they have reduced the emphasis on the number of multipliers and the number of delay elements [36]. Modularity has the advantage of requiring a reduced number of different designs as well as the possibility of independent and efficient verification of sub-systems [36, 54].

A particular example of modular filters are those that are obtained by sharpening FIR filters with real valued frequency responses [26, 38, 48]. Filter sharpening emphasizes a useful aspect of modular filters as it allows obtaining sophisticated filter responses that can be reconfigured to adapt to changes in the specifications in the context of an application using a number of relatively simple and coarse subfilters. These subfilters may be designed offline with great precision and desired complexity. This approach provides a flexible alternative to designing a high order sharp filter for every set of specifications, for which each design takes valuable time in the context of an application.

In this section, the functional composition and decomposition view of filter sharpening will suggest a systematic method for its solution and yield improved results as compared to traditional ad hoc approaches. Filter sharpening will be extended to cases for which a subfilter to be sharpened is not prescribed a priori, i.e. when there is freedom to choose a subfilter to use in a modular structure. In addition, it will be shown that the composition point of view and the tools developed in that framework will allow using more general filters than real valued or linear phase FIR filters for designing modular filters, including the possibility to use continuous time subfilters. Finally, the sensitivity and stability of modular filters will be investigated.

6.1.1 Revisiting Filter Sharpening

Filter sharpening, as already discussed in Section 2.2.7, is the technique of reducing deviations from unity in the passband and from zero in the stopband of the frequency response of a particular filter $G(z)$ with a real frequency response by multiple passes through this filter. In the same section, a sharpening method proposed by Kaiser and Hamming [26] was also presented, which performs the weighting and the ordering of these passes as dictated by an amplitude change function $A(x)$, a polynomial with vanishing derivatives at $x = 0$ and $x = 1$. In this approach, the suppression of deviations from an ideal filter behavior improves with increasing number of derivatives vanishing at these two points.

The approach in [26] fails in suppressing large ripples since it is developed with a small deviation assumption from ideal filter response characteristics. Moreover, although this approach successfully suppresses small deviations, it does not provide any optimality guarantees with respect to norms such as l_2 or l_∞. A more systematic approach to obtaining a better amplitude change function $A(x)$ with the same order is possible by reinterpreting filter sharpening within the framework developed in this thesis for which optimality guarantees can be established with respect to the norms of interest. More specifically, the resulting filter $H(z)$ is the composition of the amplitude change function $A(x)$ and the filter to be sharpened, namely $G(z)$,

$$H(z) = A(G(z)) = \sum_{k=0}^{K} a_k G^k(z), \tag{6.1}$$

where K is the maximum number of passes allowed through $G(z)$, and a_k is the weight for the result of the k-th pass of an input sequence through it. Formulating the filter sharpening problem as

$$\begin{aligned} \underset{\mathbf{a}}{\text{minimize}} \quad & \Delta \\ \text{subject to} \quad & \left\| D(e^{j\omega}) - \sum_{k=0}^{K} a_k G^k(e^{j\omega}) \right\|_\infty \leq \Delta. \end{aligned} \tag{6.2}$$

and solving for the coefficients of the amplitude change function $a_k, k = 0, 1, \ldots, K$ provide an optimality guarantee with respect to the Chebyshev norm, which lacks in the traditional formulation by Kaiser and Hamming [26]. This is an instance of a frequency response decomposition problem for which methods were developed in Chapter 4, i.e. decomposing a desired frequency response

$$D(e^{j\omega}) = \begin{cases} 1, & \omega \in \Omega_P \\ 0, & \omega \in \Omega_S \end{cases} \tag{6.3}$$

into a polynomial $A(\cdot)$, corresponding to the amplitude change function in the filter sharpening context, and the filter to be sharpened $G(e^{j\omega})$. Ω_P and Ω_S correspond

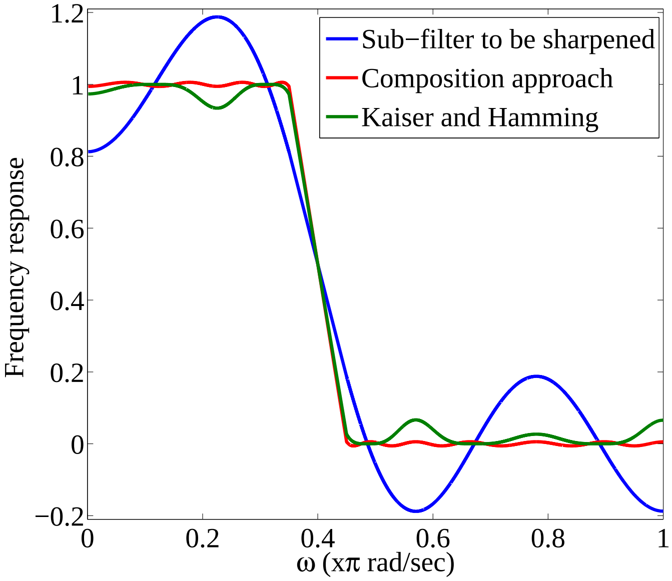


Figure 6-1: The frequency response of a 10^{th}-order filter $G(e^{j\omega})$ before and after sharpening with two different 7^{th}-order transformation polynomials. The polynomials were obtained using Kaiser and Hamming's proposed method in [26] and using frequency response decomposition method developed in Chapter 4.

to passband and stopband frequencies of the desired filter, respectively. The Parks-McClellan filter design becomes a special case of the filter sharpening problem with the subfilter $G(e^{j\omega}) = \cos\omega$. The solution is much easier for l_2 norm and does not require the methods developed in Chapter 4. However this is usually not a preferred optimality criterion in approximating filter responses.

Figure 6-1 illustrates the comparison of frequency responses of a 10^{th} order low-pass filter $G(e^{j\omega})$ obtained using the Parks-McClellan filter design algorithm with $\Omega_P = [0, 0.35\pi]$ and $\Omega_S = [0.45\pi, \pi]$, the response of the sharpened filter with a 7^{th}-order transformation using Kaiser and Hamming's method [26] and the sharpened filter with the optimal 7^{th} order polynomial obtained using freqeuncy response decomposition of the ideal filter response to minimize Chebyshev error. This example clearly shows that the proposed approach in the decomposition framework to the filter sharpening problem yields a better frequency response over the entire interval especially where the sub-filter exhibits large ripples.

In an off-line filtering application where an input signal $x[n]$ has been previously

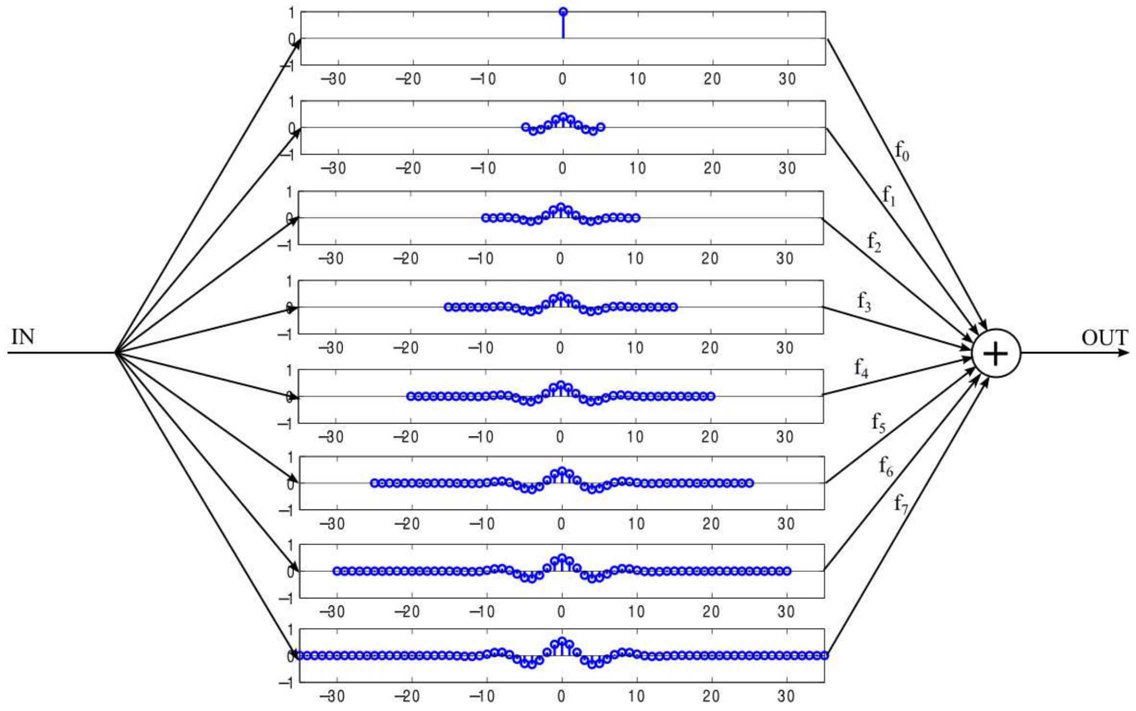


Figure 6-2: The time domain illustration of obtaining the impulse response in a filter sharpening context after determining the amplitude change function $F(\cdot)$ with coefficients f_k. The filter that was sharpened has the frequency response shown in Figure 6-1.

recorded, the implementation of filter sharpening can be performed with a single filter $G(z)$ in a rather straightforward manner if memory for caching a limited amount of intermediate signals is available. The result $y[n]$ is the weighted summation of the outputs of multiple passes through $G(z)$, where at each step this output is fed back to the filter for the next step. The weights in the summation are the coefficients of the amplitude change function $A(z)$. Since multiple stages of filtering through the filter $G(z)$ corresponds to an impulse response that is obtained by self convolutions of the sequence $g[n]$, the equivalent operation can be viewed as that in Figure 6-2, which illustrates the implementation of sharpening for the example filter given in Figure 6-1.

The method proposed for off-line applications does not apply to on-line applications where the output needs to be computed as the input arrives. The successive filtering of the entire input sequence by using a single filter is not desirable in this case

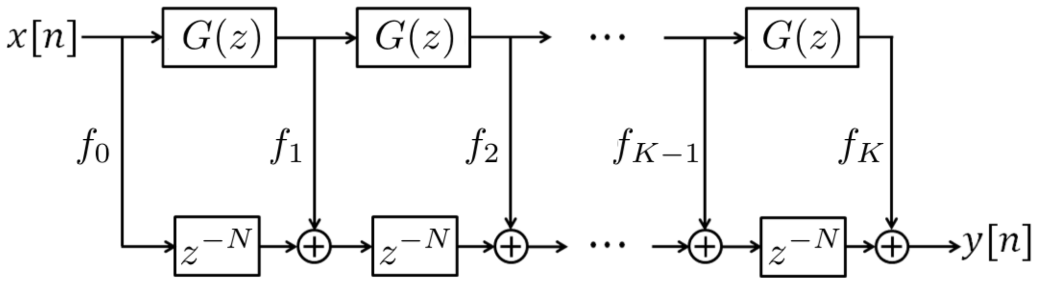


Figure 6-3: Tapped cascaded interconnection of identical subfilters $G(z)$ with appropriate time shifts after each tap to preserve causality [48].

as this introduces undesirable latency since the output cannot be computed until the entire input sequence is recorded. Therefore, one dedicated subfilter $G(z)$ is required to replace each of the K successive filtering operations; and the output of each such filter should be time delayed by half the order of the subfilter to preserve causality. For example, the implementation in Figure 6-3 was proposed in [48] to accomplish this for subfilters $G(z)$ of order $2N$, where it was referred to as a tapped cascaded interconnection of identical subfilters $G(z)$.

6.1.2 Two-Step Modular FIR Filter Design

For a real valued FIR subfilter $G(e^{j\omega})$ with an even order, the frequency response can be shown to be a polynomial $\tilde{G}(\cos\omega)$ in $\cos\omega$. Sharpening it with a polynomial $F(\cdot)$ leads to a filter with a frequency response

$$H(e^{j\omega}) = F(G(e^{j\omega})) = F(\tilde{G}(\cos\omega)). \tag{6.4}$$

In cases where a subfilter $G(e^{j\omega})$ is not pre-specified, it may still be desirable to design a high order filter in a modular form, which also requires choosing a suitable subfilter as a first step. Applying the transformation $x = \cos(\omega)$ on the interval $\omega \in [-\pi, \pi]$ to both $H(e^{j\omega})$ and the desired filter response $D(e^{j\omega})$ given in equation (6.3) yielding $\tilde{D}(x)$, the modular filter can be designed by finding the optimal pair of polynomials F and $\tilde{G}$ to approximate $\tilde{D}(x)$ on the interval $x \in [-1, 1]$ as $F(\tilde{G}(x))$. This is a different polynomial decomposition problem than those discussed in Chapter 3 where the approximation error was defined with respect to the coefficients of polynomials. In [51], it was shown that an optimal approximation with respect to the Chebsyhev norm of the form $F(\tilde{G}(x))$ with specified polynomial orders exists on a closed set $x \in [a, b]$. However such an optimal approximation does not have a characterization involving counting alternations unlike the alternation theorem for approximating by ordinary polynomials. Finding an optimal approximation is more formidable when the desired orders of the composing polynomials are not pre-specified.

A heuristic and greedy two-step approach to obtain such modular filter structures consists of obtaining a sub-filter G and then an optimal polynomial F to sharpen it rather than jointly optimizing them. This method starts with splitting the order P of $H(e^{j\omega})$ between these two components, where P can be viewed as the number of available distinct multipliers or degrees of freedom to design $H(e^{j\omega})$. The sub-filter is designed as the optimal zero-phase FIR filter with its N allocated degrees of freedom, and the remaining $M = P - N$ degrees of freedom are used to choose the coefficients of F to improve the frequency response characteristics of $G(e^{j\omega})$ by sharpening it. The best results using this heuristic algorithm seem to be obtained when the degrees of freedom for F and $G(e^{j\omega})$ are chosen close to each other for low-pass and high-pass linear phase FIR filters.

In Figure 6-4, a 24^{th}-order zero-phase Parks-McClellan low-pass filter with passband $[0, 0.30\pi]$ and stopband $[0.34\pi, \pi]$ is compared to a composition of a 7^{th} order polynomial F and a 10^{th} order sub-filter $G(e^{j\omega})$, where $G(e^{j\omega})$ is also a Parks-McClellan filter with the same passband and stopband edge frequencies as the 24^{th} order filter. Even-order Parks-McClellan filters can be implemented using distinct coefficients as many as one plus half of their order due to their coefficient symmetry, therefore both of these designs can be shown to be implemented using the same number of distinct multipliers. The compositional design yields a considerably superior frequency response characteristics. In general, the total number of multiplications per input sample in a compositional design, $M(N + 1) + (M + 1)$, is greater than a direct form implementation with the same number of distinct multipliers, which requires $M + N + 1$ multiplications. However, the increase in the number of multipli-

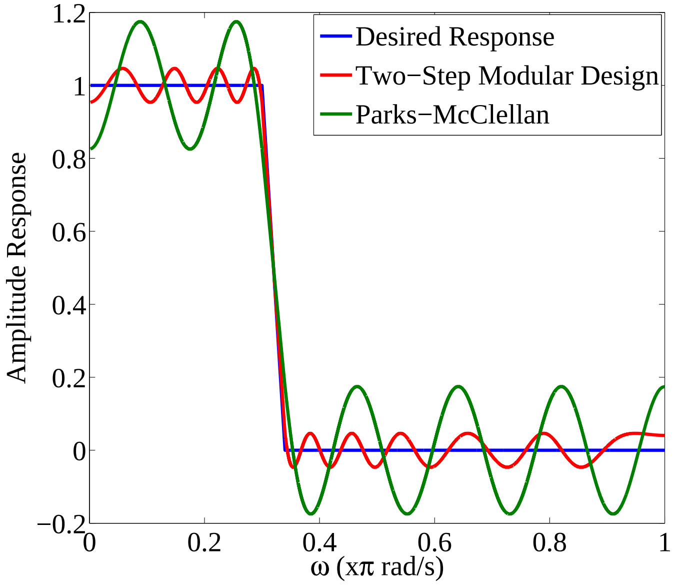
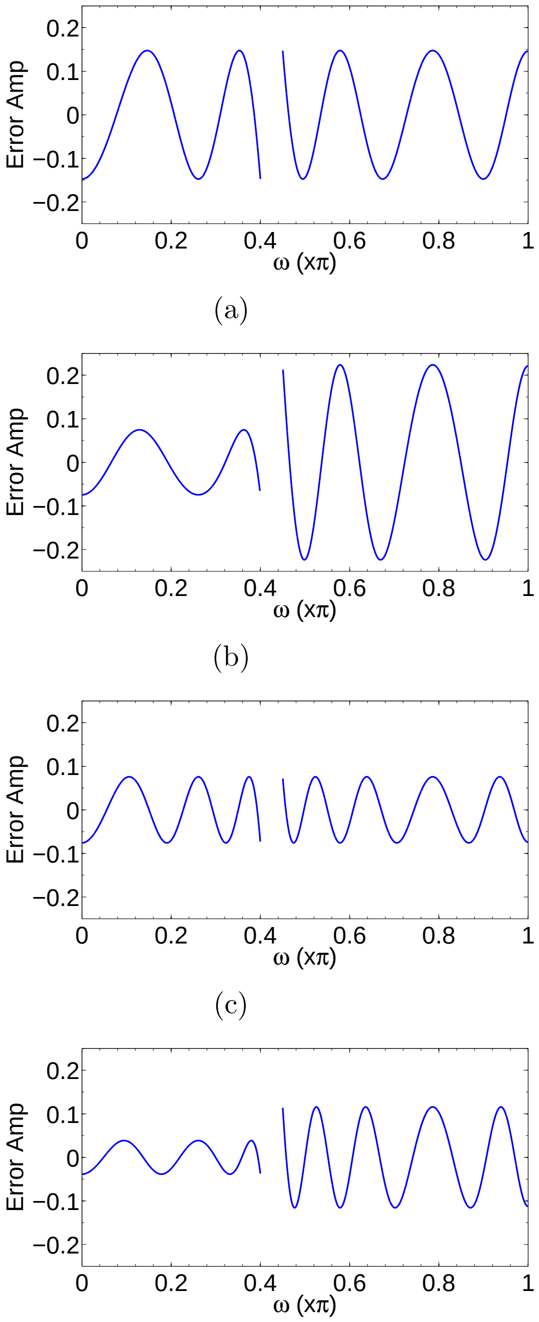


Figure 6-4: The comparison of a 24^{th}-order Parks-McClellan filter and a modular filter obtained using the two-step filter design algorithm with a 7^{th}-order polynomial and a 10^{th}-order Parks-McClellan filter. Both designs can be shown to have the same number of distinct multipliers while the modular design has superior frequency response characteristics although the number of multiplications per input sample is higher.

cations may be regarded as a negligible side effect compared to the potential benefits of the modularity it provides, especially if the filters are to be designed using VLSI technology.

As an example of the convenience of modularity that results from designing filters using the proposed two-step design procedure, consider the case in which a low order low pass filter $G(e^{j\omega})$ is obtained in the first step. A sharper low pass filter with the same passband and stopband edges can be obtained by choosing appropriate coefficients $f_k, k = 0, 1, \ldots, M$ that will minimize the maximum deviation from the desired response. If the need to have sharper characteristics in one of the bands than the other band arises during an application or equivalently, if an explicit weight function $W(\omega)$ is specified, it will suffice to re-compute the coefficients f_k consistent with the weight function without altering the filter $G(e^{j\omega})$. This can be done by modifying the desired response as $W(\omega)D(e^{j\omega})$ and the basis functions as $W(\omega)G^k(e^{j\omega})$ in the frequency decomposition algorithm. If even sharper characteristics are desired in both



(a)

(b)

(c)

(d)

Figure 6-5: The approximation errors to an ideal low pass filter with passband and stopband edge frequencies 0.40π and 0.45π, respectively. A 10^{th}-order Parks-McClellan filter $G(e^{j\omega})$ with the same passband and stopband edges is used repetitively in tapped line with tap coefficients $f_k, k = 0, 1, \ldots, M$: (a) $M = 4$ with uniform weight. (b) $M = 4$ with relative passband error weight of three. (c) $M = 6$ with uniform weight. (b) $M = 6$ with relative passband error weight of three.

bands, then new blocks of $G(e^{j\omega})$ and their corresponding coefficients f_k can simply be appended to the structure in Figure 6-3 to increase the overall filter order.

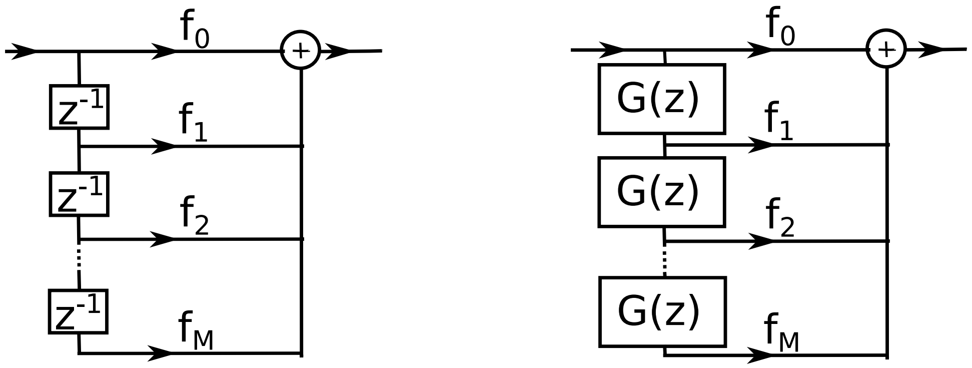
Figure 6-5 illustrates several different error profiles resulting from approximating an ideal low pass filter with a passband edge 0.40π and a stopband edge of 0.45π. In Figure 6-5a, the tap coefficients $f_k, k = 0, \ldots, 4$ are chosen to minimize the maximum error using four blocks of $G(e^{j\omega})$, where $G(e^{j\omega})$ is a 10^{th} order low pass filter with a passband edge 0.40π and a stopband edge of 0.45π. In the case where the passband error is assigned to have a weight of three times that of the error in the stopband, the tap coefficients $f_k, k = 0, 1 \ldots, 4$ can be changed to obtain the error profile given in Figure 6-5b. By allowing the use of two more blocks of $G(e^{j\omega})$, the coefficients $f_k, k = 0, 1, \ldots, 6$ can be chosen to obtain smaller errors with similar weight functions as illustrated in Figure 6-5c and 6-5d. Although in these examples the weighted errors have equi-oscillatory behavior, this is not necessarily the case for general frequency responses since, as already discussed, optimal approximation by polynomial composition is not in general characterizable by equi-oscillations [51].

6.1.3 Modular Filters with Complex-Valued Subfilter Responses

In the discussion of modular filters as an extension of filter sharpening in Sections 6.1.1 and 6.1.2, the subfilters were either specified or designed to have real valued frequency responses. Since FIR filters with real valued frequency responses are non-causal, they were shifted appropriately in time for real time applications by using multiple delays as illustrated in Figure 6-3. The restriction to real-valued responses was due to the original sharpening problem definition and solution by Kaiser and Hamming [26] since the proposed method relied on the subfilter having a real-valued frequency response. Authors who extended this work [12, 38, 48] also used the same assumption, and in fact some restricted the coefficient sequence of the amplitude change function to be even-symmetric around an integer [12,48] so that it corresponds to a “prototype filter” with a real valued frequency response. However, the frequency response decomposition methods developed in Chapter 4 are not restricted by such constraints, therefore it allows modular filter design using subfilters $G(z)$ with complex valued responses and a straightforward performance analysis within the functional composition framework. More specifically, the frequency decomposition algorithms can be applied to the problem (6.2) for any continuous real or complex valued desired function $D(e^{j\omega})$ and subfilter frequency response $G(e^{j\omega})$. FIR filter design becomes a special case of this approach with the subfilter $G(e^{j\omega}) = e^{-j\omega}$.

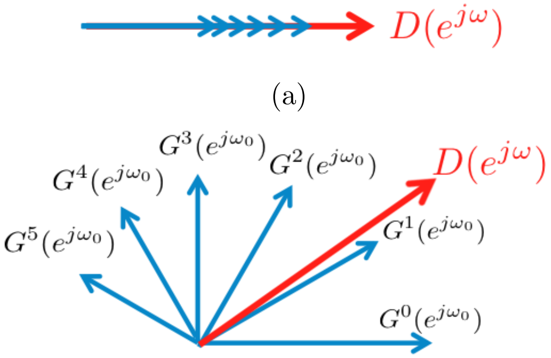
For obtaining modular filters, an alternative view that is more general than filter sharpening is to compose two filters, i.e, to replace every delay element in the implementation of $F(z)$ with $G(z)$ as depicted in Figure 6-6. The resulting structure is similar to that in Figure 6-3 with the exception of not requiring extra delays after each tap to preserve linear phase or causality as all the filters will be assumed to be causal, and having real valued or linear phase FIR subfilters is not required for the analysis in this generalized case. Figure 6-6a depicts the direct form implementation of an FIR filter $F(z)$ achieved by cascading delay units z^{-1} and tapping the output of each delay using a branch gain, where the gains correspond to the coefficients of the filter. This structure has traditionally been referred to as a tapped delay line. Substituting a subfilter $G(z)$ for the delay units in such structures leads to topologies as in Figure 6-6b, in which case the structure can be referred to as a generalized tapped delay line.

One caveat with using subfilters with complex valued frequency responses, as was also mentioned in Section 4.1.5, is the additional requirement of matching the phase of the desired filter $D(e^{j\omega})$ in addition to its magnitude, which was not a problem when both $D(e^{j\omega})$ and $G(e^{j\omega})$ are real valued functions. This is illustrated in Figure 6-7, which depicts the complex numbers $D(e^{j\omega_0})$ and $G^k(e^{j\omega_0}), k = 0, 1, \ldots, K$ at a particular frequency $\omega = \omega_0$ as vectors in the complex plane. In Figure 6-7a, the vector that corresponds to the desired real function to be matched is always aligned with those of the basis functions for every frequency as they are all real, a property that facilitates the approximation of $D(e^{j\omega})$ as a linear combination of the basis functions. On the other hand, in Figure 6-7b, the vectors for the basis functions have phases



(a) (b)

Figure 6-6: (a) The direct form implementation of an FIR filter $F(z)$ using a tapped delay line. (b) A generalized tapped delay line where the delays are replaced by another filter $G(z)$.



(a)

(b)

Figure 6-7: (a) Vectors corresponding to real values of the desired frequency response and the basis functions are always aligned (b) Vectors corresponding to complex values of the desired response and basis functions are not necessarily aligned, and for the case of integer powers of $G(e^{j\omega})$, the phases are integer multiples of the same angle at a given frequency. The figure illustrates an example alignment of vectors corresponding to real and complex numbers at a specific frequency ω_0.

that are different integer multiples of the same angle and even this angle changes for different frequency values. In cases where approximating a frequency response with complex valued basis functions is difficult and where it suffices to only approximate the magnitude instead, techniques developed in Section 4.2 can be used.

Continuous Time Modular Filters

Both the development of frequency response decomposition algorithms and its application to designing modular filters have so far been carried out in the context of discrete time filters. The same discussion and methods apply to designing continuous time filters by mapping the frequency response defined on $(-\infty, \infty)$ to the compact interval $[-\pi, \pi]$ as described in Section 4.1.5. This allows designing continuous time modular filters using low order filters similar to designing discrete time modular filters.

Common types of continuous time filters are Butterworth, Chebyshev and elliptic filters all of which have infinite impulse responses and complex valued frequency responses. Moreover, filter specifications for continuous filters are usually given as constraints on the magnitude response rather than on the frequency response as the phase is uniquely determined in these minimum phase filters. Although there is no guarantee that a modular continuous time filter obtained using a generalized tapped delay line with a continuous time subfilter $G(s)$ will remain minimum phase, it may be still desirable to approximate the desired filter response in magnitude only due to the additional phase matching requirement illustrated in Figure 6-7. An additional benefit that the functional composition framework introduces is that both passband and stopband edge frequencies can be specified for continuous time modular filters whereas traditionally only one of them is provided depending on the type of the filter as their design methods cannot accommodate constraints on the other band edges.

Figure 6-8 illustrates the comparison of a 4-th order elliptic bandpass filter $G(s)$ designed to approximate in magnitude the desired function

$$D(\Omega) = \begin{cases} 1, & 2\pi \times 6 \cdot 10^3 \leq \Omega \leq 2\pi \times 1 \cdot 10^4 \\ 0, & 0 \leq \Omega \leq 2\pi \times 5 \cdot 10^3 \text{ and } 1.1 \cdot 10^4 \leq \Omega \end{cases} \tag{6.5}$$

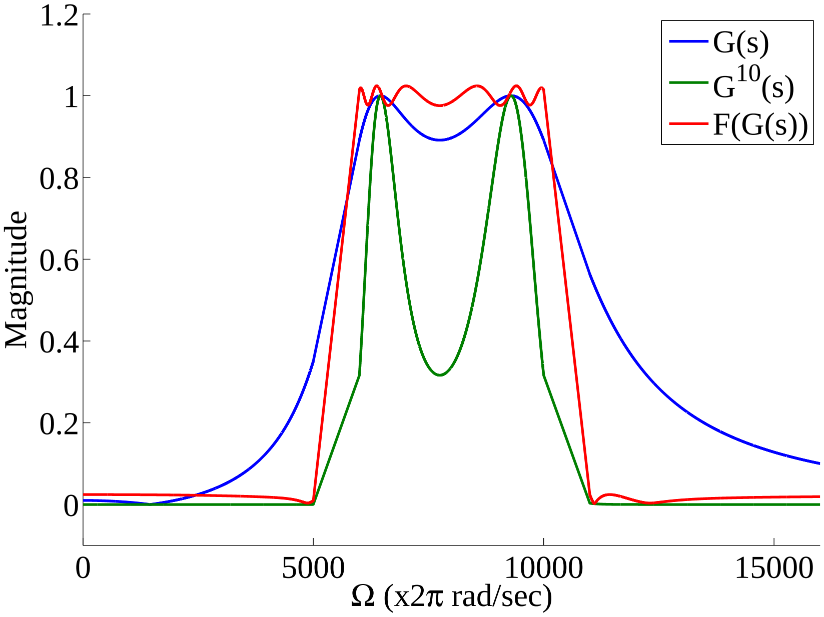


Figure 6-8: A 4-th order elliptic bandpass filter magnitude response, the magnitude response of its sharpened version obtained by simply cascading ten blocks and the magnitude response of the same order modular filter obtained within the decomposition framework.

More specifically, the maximum passband ripples were constrained to be 1dB and the minimum attenuation in the stopband was constrained to be 40dB. The same figure also illustrates the response of a filter obtained by cascading ten identical blocks of $G(s)$ to improve its response as well as its composition with a 10-th order polynomial $F(s)$, where $F(s)$ is obtained by decomposing $D(\Omega)$ using the magnitude decomposition algorithm developed in Chapter 4. It is clear that the magnitude response decomposition of $D(\Omega)$ yields a much better response than simple cascading.

Figure 6-9 illustrates the errors at each iteration during the computation of the optimal coefficients for f_k for the magnitude decomposition of $D(\Omega)$. Starting at two different initial guess points for the coefficient vector $\mathbf{f}$, both curves have a non-increasing trend consistent with the discussion in Section 4.2.1. This figure also shows that different initial guess points lead to different initial errors as well as final error levels. Therefore, in such problems, different initial guesses may be tried until a satisfactory error level is achieved with increasing number of iterations. The coefficients for F in Figure 6-8 were chosen as those corresponding to the smaller error curve in

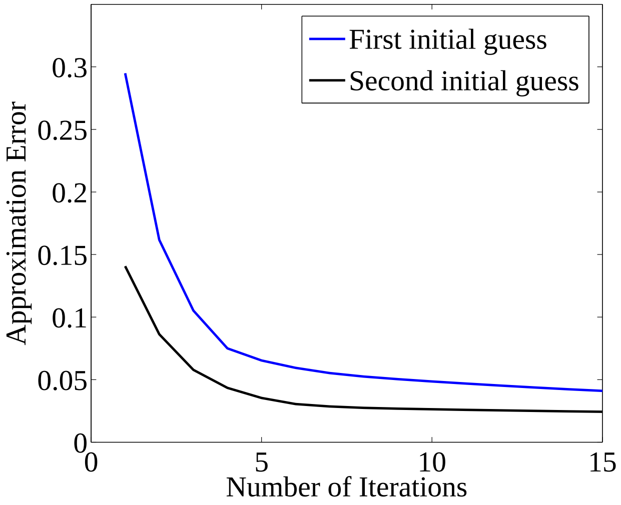


Figure 6-9: The resulting error values in the magnitude decomposition performed to obtain the modular filter in Figure 6-8 as they evolve with the number of iterations for two different initial guesses for $F(s)$.

Figure 6-9.

6.1.4 Sensitivity and Stability

The sensitivity of modular filters with respect to perturbations in the coefficients $\mathbf{f}$ of the polynomial $F(\cdot)$ can be seen to be quite low. Hence it is robust with respect to imperfections in the tap coefficients within the generalized tapped delay line. An error $\Delta\mathbf{f}$ in these coefficients result in a change in the modular filter response by $\Delta H(e^{j\omega})$ which satisfies

$$\begin{aligned} \left|\Delta H(e^{j\omega})\right| &= \left|\sum_{k=0}^{K} \Delta f_k G^k(e^{j\omega})\right| \\ &\le \sum_{k=0}^{K} |f_k| \left|G^k(e^{j\omega})\right| \\ &\le M_G \sum_{k=0}^{K} |f_k| \\ &= M_G \|\Delta\mathbf{f}\|_1, \end{aligned} \tag{6.6}$$

where M_G is the maximum absolute value that the basis functions $G^k(e^{j\omega})$ can take. The magnification of the l_1 norm of the perturbation $\mathbf{\Delta f}$ is therefore bounded by M_G,

$$S_{F\to H} = \frac{|\Delta H(e^{j\omega}|)}{\|\mathbf{\Delta f}\|_1} \leq M_G \tag{6.7}$$

which in turn is rather low because $|G(e^{j\omega})|$ takes values close to unity in the passbands and zero in the stopbands.

The sensitivity with respect to imperfections in $G(e^{j\omega})$ is slightly more complicated and the upper bound on the magnification is larger than that with respect to coefficients of $F(\cdot)$. A perturbation $\Delta G(e^{j\omega})$ results in $|\Delta H(e^{j\omega})|$ bounded as

$$\begin{aligned} \left|\Delta H(e^{j\omega})\right| &= \left|\sum_{k=0}^{K} f_k \left[(G(e^{j\omega}) + \Delta G(e^{j\omega}))^k - G^k(e^{j\omega})\right]\right| \\ &\approx \left|\sum_{k=1}^{K} f_k k G^{k-1}(e^{j\omega}) \Delta G(e^{j\omega})\right| \\ &\leq |\Delta G(e^{j\omega})| M_G \sum_{k=1}^{K} |f_k| k \\ &\leq |\Delta G(e^{j\omega})| M_G K \|\mathbf{f}\|_1, \end{aligned} \tag{6.8}$$

which leads to an upper bound on the magnification of $\Delta G(e^{j\omega})$ by

$$S_{G\to H} = \frac{|\Delta H(e^{j\omega}|)}{|\Delta G(e^{j\omega})|} \leq M_G K \|\mathbf{f}\|_1. \tag{6.9}$$

For low order F with small coefficients, this sensitivity will also be small, however this analysis implies the worst case sensitivities can be much higher with respect to the perturbations in the subfilter. This issue can be circumvented by designing the subfilters with greater precision as they are usually simple structures and can be designed off-line due to the flexibility of modular designs.

Another notable benefit of designing modular filters as generalized tapped delay lines is the fact that they do not compromise stability as the overall order of filter increases unlike the traditional designs of recursive filters which may become unstable with even the smallest perturbations in the coefficients, coefficient quantization errors or component imperfection in continuous time filters. For example, this can be analytically illustrated in a context of continuous time filters by first expressing the polynomial $F(s)$ using its roots s_k

$$F(s) = \prod_{k=1}^{K} (s - s_k), \tag{6.10}$$

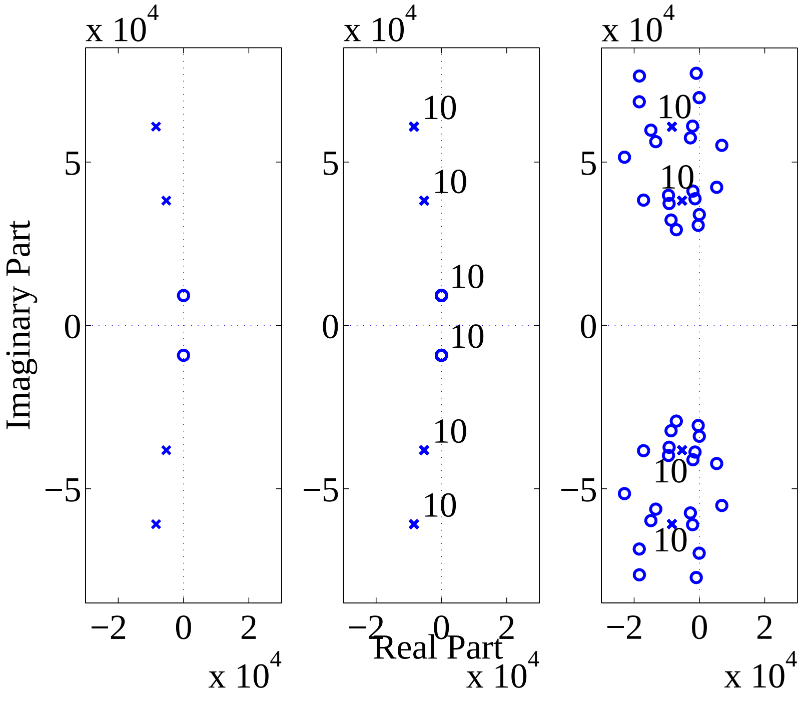
and then expressing its composition with $G(s) = \frac{N(s)}{Q(s)}$ as

$$F(G(s)) = \prod_{k=1}^{K} (G(s) - s_k) = \prod_{k=1}^{K} \left(\frac{N(s) - s_k Q(s)}{Q(s)} \right), \tag{6.11}$$

which implies that no additional poles are introduced by the composition. In other words, since $G(s)$ is already stable, the resulting modular filter will be stable regardless of the order of the polynomial $F(s)$ as no poles at new locations are introduced and simply the multiplicities of the existing ones are increased to K. Cascading the subfilter K times results in making the multiplicity of all the poles and zeros to be K whereas composition places the new zeros in a way to attain minimax optimality instead of just increasing their multiplicity. Figure 6-10 depicts the pole-zero diagrams of the filters $G(s)$, $G^{10}(s)$ and $F(G(s))$ the magnitude responses of which were given in Figure 6-8. Designing a higher order filter with sharper characteristics without using the modular structure may lead to poles that are close to the imaginary axis, which can compromise stability in case of perturbations.

6.2 Efficient Marginalization and Representation of Decomposable Functions

The sum-product algorithm used in inference problems is already known to be implemented efficiently if the corresponding joint probability density is factorable. This section starts with a brief review of the sum-product algorithm in the context of factorable functions, i.e., in the original context it was formulated. The computational



(a) (b) (c)

Figure 6-10: The pole-zero diagrams of the filters (a) $G(s)$, (b) $G^{10}(s)$ and (c) $F(G(s))$, the magnitude responses of which were given in Figure 6-8, respectively. The scale of the plots are relatively large, therefore zeros with relatively smaller magnitudes are not visible. The multiplicities of poles are zeros are marked by an integer next to them if it is greater than one.

efficiency obtained when the joint probability density function is factorable stems from the separation of its variables, which is visually captured by a factor graph, leading to local and more manageable computations.

The sum-product algorithm on factor graphs that represent a decomposable multivariate function cannot directly be carried out efficiently. However, the techniques developed in Chapter 5 will be used to approximately decompose a joint probability density function in Section 6.2.2 such that the corresponding factor graph lends itself to an efficient implementation of the sum-product algorithm due to the separation of the variables similar to the case of factorable functions.

6.2.1 Sum-Product Algorithm

As discussed in Section 5.3.6, factor graphs provide a visually convenient representation of multivariate functions that are product of functions of a smaller subset of variables where these are referred to as local functions [31]. In addition, when computing the marginal for a subset of variables, they also facilitate the interpretation of the results of intermediate summations as messages passed between nodes. Such an interpretation as well as an efficient way for bookkeeping the intermediate results are offered by the sum product algorithm. In this section, the sum-product algorithm will be defined and briefly reviewed following the notation in [31] in the context of a simple example to highlight its key aspects that will allow exploiting the decomposable functions similarly in Section 6.2.2.

The sum product algorithm provides a systematic and efficient framework for the computation of marginal functions on a factor graph by defining and computing messages to and from each node. More specifically, the marginal for a variable is the product of the incoming message to and the outgoing message from this variable on any of the edges that is incident on it in the factor graph. In order to illustrate how this algorithm works, consider the factor graph in Figure 6-11, which represents the joint probability distribution function of five random variables and given by

$$P(x_1, x_2, x_3, x_4, x_5) = P_A(x_1, x_3, x_5) P_B(x_2, x_3, x_4). \tag{6.12}$$

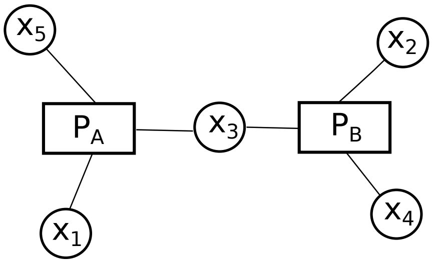


Figure 6-11: The factor graph representation of the joint distribution of five random variables as given in equation (6.12).

The marginal for x_2, for example, is given by the product

$$M(x_2) = \mu_{P_B \to x_2} \mu_{x_2 \to P_B}, \tag{6.13}$$

where $\mu_{P_B \to x_2}$ is the message function that is sent from the factor node P_B to the variable node x_2, and $\mu_{x_2 \to P_B}$ is the message function in the opposite direction. Both of these messages are functions of x_2 since, within this algorithm, each message is only a function of the variable connected to the edge on which it is computed.

The computation of each message is as follows. Each variable node computes the message that it will send to a factor simply by taking the product of all the incoming messages except the one from its destination node. For example,

$$\mu_{x_3 \to P_B} = \mu_{P_A \to x_3} \tag{6.14}$$

since this is the only incoming message from other nodes, whereas

$$\mu_{x_2 \to P_B} = 1 \tag{6.15}$$

since there are no incoming messages other than from the destination P_B. The computation of messages from a factor node to a variable node is slightly more complicated and requires a marginalization. More specifically, all the incoming messages to the factor node as well as the factor itself are multiplied and then this product is summed or marginalized over all the variables except the destination variable, giving the sum-product algorithm its name. For example, the message going from P_B to x_2 is given by

$$\mu_{P_B \to x_2} = \sum_{x_3} \sum_{x_4} P_B(x_2, x_3, x_4) \mu_{x_3 \to P_B} \mu_{x_4 \to P_B}. \tag{6.16}$$

These marginalizations always lead to a function of a single variable, namely the destination variable, consistent with the fact that messages can only be the function of the variable connected to the corresponding edge. Computing all the messages starting from leaves of the factor graph allows to evaluate the marginal function for each variable. Equations (6.13), (6.15) and (6.16) yield the marginal for x_2 as

$$\begin{aligned} M(x_2) &= \sum_{x_3} \sum_{x_4} \left(P_B(x_2, x_3, x_4) \left(\sum_{x_1} \sum_{x_5} P_A(x_1, x_3, x_5) \right) \right) \\ &= \sum_{\{x_1, x_3, x_4, x_5\}} P(x_1, x_2, x_3, x_4, x_5) \end{aligned} \tag{6.17}$$

as expected.

6.2.2 Inference with Decomposable Density Functions

The benefits of the sum-product formulation are the systematic manner in which local and thus simpler computations are combined as well as its ability to render the marginals for all the variables once the messages are computed and cached. The simplicity of the local computations stems from the fact that the local marginalizations are computed over a smaller subset of variables when the joint probability distribution is factorable. It also implies that the efficiency will be greater if the maximum number of variables connected to each factor is small. This observation underlies the benefit of decomposability as discussed in Chapter 5, where decomposition was related to factorization and consequently to the separation of variable nodes between local functions at the expense of increasing the dimensionality of the multivariate function.

In order to illustrate the computational efficiency of decomposing joint probability density functions in the context of an inference problem, the running times of two instances of the sum-product algorithm will be compared for the same problem in

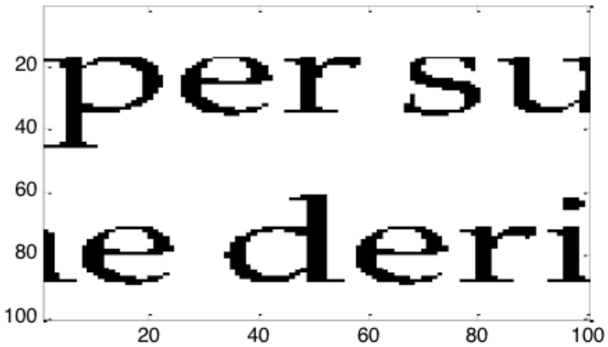


Figure 6-12: Clean binary text image.

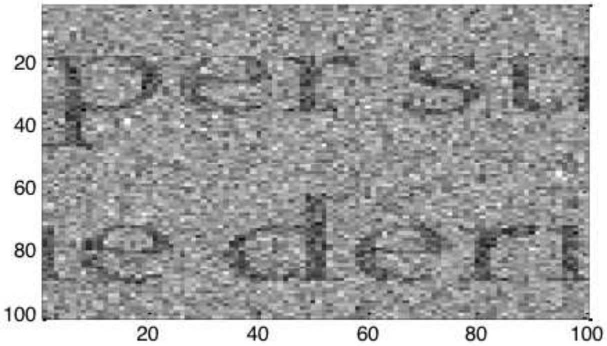


Figure 6-13: Noisy text image obtained by adding independent and identically distributed white Gaussian noise with zero mean and unit variance to the binary text image in Figure 6-12. The figure is displayed using `imagesc` function in MATLAB, which scales the image data to the full range of the colormap.

this section, one using a high dimensional joint probability density function and the other using its approximation as a decomposition into lower dimensional functions as described in Chapter 5.

Consider a binary image denoising application in which a text snippet is treated as a binary image with values $+1$ for white pixels and -1 for black pixels. An example of a clean image is given in Figure 6-12 and it consists of 100×100 pixels. This image is then contaminated by independent and identically distributed white Gaussian noise with mean $\mu = 0$ and variance $\sigma^2 = 1$, which is rather high as it is comparable to the absolute pixel values. The noisy image is given in Figure 6-13.

The noisy image is denoised by determining for each pixel which of the two values, $+1$ or -1, is more likely. This is performed by running the sum-product algorithm on the entire image by treating each pixel as a binary random variable, and providing a

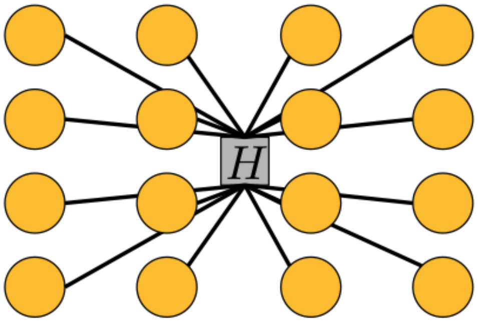


Figure 6-14: The factor graph corresponding to the joint probability density function for a block of 4 × 4 pixels in its original form, i,e., before decomposition.

joint probability density H for every 4×4 pixel block, where

$$H(x_1, x_2, \ldots, x_{16}) \tag{6.18}$$

is a 16 dimensional discrete multivariate function and is learned previously from training on a large set of binary text images. The algorithm also takes as input the likelihood for each pixel of having a value of +1 given by

$$P(+1|y) = \frac{e^{(y-1)^2/\sigma^2}}{e^{(y-1)^2/\sigma^2} + e^{(y-(-1))^2/\sigma^2}}, \tag{6.19}$$

where y is the observed pixel value. The factor graph for each 4×4 block is depicted in Figure 6-14. In this setup, each pixel will be a part of several neighboring 4×4 blocks. Therefore the factor graph corresponding to the entire image will lack the property of a tree and will have loops, therefore the sum-product algorithm will be an approximate solution and will need more than one iteration for a satisfactory result. The result of denoising using $H(x_1, x_2, \ldots, x_{16})$ in its original form in the sum-product algorithm with ten iteration yields the denoised image given in Figure 6-15 and the solution was obtained in 2561 seconds using Dimple, the open source probabilistic graphical model manipulation tool [25].

For computation time comparison, the joint probability density function H was decomposed using approximate general decomposition by introducing a latent variable

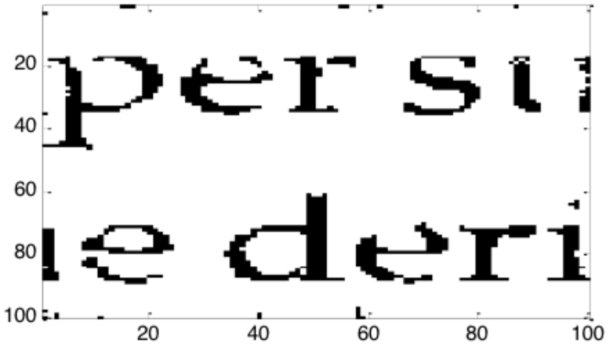


Figure 6-15: The denoised image obtained by running the sum-product algorithm with the joint density function in its original form $H(x_1, x_2, \ldots, x_{16})$.

c as discussed in Section 5.4.5 using nonnegative matrix factorization to obtain

$$\begin{aligned} H(x_1, x_2, \ldots, x_{16}) &= \sum_c \hat{H}(x_1, x_2, \ldots, x_{16}, c) \\ &= \sum_c F(x_1, x_2, \ldots, x_8, c)\tilde{G}(c, x_9, \ldots, x_{16}) \end{aligned} \tag{6.20}$$

where the alphabet size of c is restricted for efficiency by

$$D_c < \frac{D_{\{x_1,\ldots,x_8\}} D_{\{x_9,\ldots,x_{16}\}}}{D_{\{x_1,\ldots,x_8\}} + D_{\{x_9,\ldots,x_{16}\}}} = \frac{2^8 2^8}{2 \cdot 2^8} = 128 \tag{6.21}$$

as dictated by equation (5.19). More specifically, the alphabet size for c was chosen as 32 and supplied as a parameter to the nonnegative matrix factorization, where the Kullback-Leibler divergence was minimized for the approximation, a commonly preferred metric when working with probability distributions. The corresponding factor graph for a 4×4 pixel block after the decomposition is depicted in Figure 6-16. The sum-product algorithm with ten iterations on this factor graph yielded the denoised image given in Figure 6-17 and the computation was completed in 189 seconds suggesting an important improvement in the computation time using decomposition without compromising visual quality significantly.

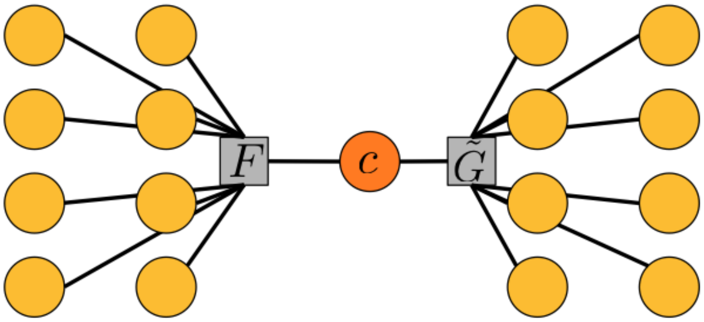


Figure 6-16: The factor graph corresponding to $\hat{H}$ given in equation (6.20).

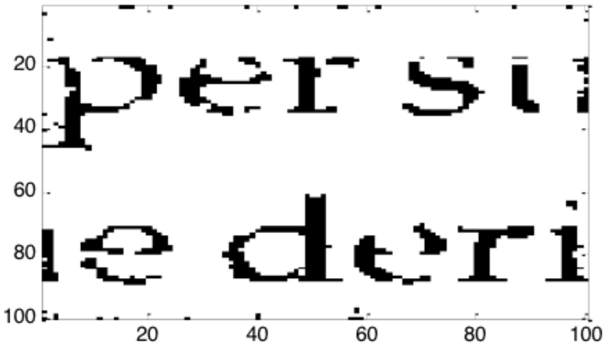


Figure 6-17: The denoised image obtained by running the sum-product algorithm using the decomposed joint density function $\hat{\hat{H}}$ in equation (6.20).

6.2.3 Compact Representations of Multivariate Functions

In addition to reducing the dimensionality over which marginalizations are performed and leading to a computational efficiency for MDF problems, the decomposition of a multivariate function $H(\mathbf{X_A}, \mathbf{X_B})$ as in equation (5.21), either exact or approximate, leads to its representation in terms of the functions F and $\tilde{G}$ in which the total number of parameters are less than that of H itself. Equivalently, the partition table $\mathbf{M_H}$ corresponding to H can be represented with fewer entries as a product of the matrices $\mathbf{M_F}$ and $\mathbf{M_{\tilde{G}}}$. This can be regarded as a more compact representation of H as a result of its decomposability.

6.3 Polynomial Decomposition for Compact Representations and Modularity

6.3.1 Decomposable Finite Sequences

A discrete sequence or a filter impulse response is typically considered to be sparse if it has few nonzero samples, i.e. if it can be represented with few parameters. Polynomial decomposition can also be considered useful in a signal processing context in which decomposability is exploited to reduce the number of required parameters to represent a signal. A finite length discrete time signal $h[n]$ the z-Transform $H(z)$ of which can be represented as the composition of two smaller order polynomials $F(z)$ and $G(z)$, i.e. $H(z) = F(G(z))$, has a length greater than the number of parameters required to represent it. Consider

$$\begin{aligned} F(z) &= f_0 + f_1 z^{-1} + f_2 z^{-2} + \cdots + f_M z^{-M} \\ G(z) &= g_0 + g_1 z^{-1} + g_2 z^{-2} + \cdots + g_N z^{-N}. \end{aligned} \tag{6.22}$$

The relationship between the degrees of these polynomials is $P = MN$ where P, M and N are the degrees of $H(z)$, $F(z)$ and $G(z)$, respectively. This implies that $h[n]$ can be represented indirectly by F and G using $(M+1)+(N+1)$ coefficients instead of being directly represented by H using $MN+1$ coefficients. In this case, $h[n]$ is a sparse signal, where sparsity corresponds to having a decomposable structure that can be represented with few parameters rather than having few nonzero coefficients. The sensitivity analysis in Section 3.5 illustrates how a perturbation on the coefficients of F and G affect the coefficients of H, and compositions with linear polynomials are shown as a means to reduce sensitivity with respect to these coefficients.

In the case where $H(z)$ is not an exact composition, approximate decomposition techniques can be used to decompose the signal $h[n]$. An example was given in Section 3.4.3, where a 12-th order decomposable polynomial was decomposed using the iterative approximate decomposition method due to Corless [16] and represented as the approximate composition of a 4-th order polynomial and a 3-rd order polynomial. This reduced the number of parameters required to represent this polynomial from 13 to 9. The savings increase with increasing polynomial orders. However, as already mentioned in Section 3.4.3, the approximate decomposition techniques are successful only for small orders. Moreover, an approximate decomposition that is satisfactorily close to the original polynomial does not always exist due to the radius of non-decomposability given in equation (3.17). In other words, there is always a high dimensional ball centered at a given non-decomposable polynomial with nonzero radius within which all the polynomials are also non-decomposable. This makes the problem of finding approximate decompositions difficult and often non-practical.

6.3.2 Modular Filter Design by Approximate Polynomial Decomposition

Modular filter design as a generalization of filter sharpening was discussed in Section 6.1, where a subfilter was either pre-specified or pre-selected at the first step of a two step modular filter design algorithm to approximate the overall filter specifications. An alternative to the latter approach is to use polynomial decomposition techniques to approximate the impulse response of FIR filters as the composition of two smaller order sequences without the requirement of a pre-specified subfilter. For example the

z-transform of an FIR filter $H(z)$ can be decomposed as $F(G(z))$ and the resulting filter structure can be implemented as a generalized tapped delay line given in Figure 6-6 leading to a modular design.

Consider a 30-th order Parks-McClellan low-pass filter with the passband and stopband edges of 0.20π and 0.24π, respectively. Figure 6-18a shows the impulse responses of the original filter and its approximate decomposition $F(G(z))$ obtained by Corless' method where

$$F(z) = -0.0526+0.0649z^{-1}-0.0359z^{-2}-0.0021z^{-3}+0.1160z^{-4}-0.0226z^{-5}+0.0049z^{-6} \tag{6.23}$$

and

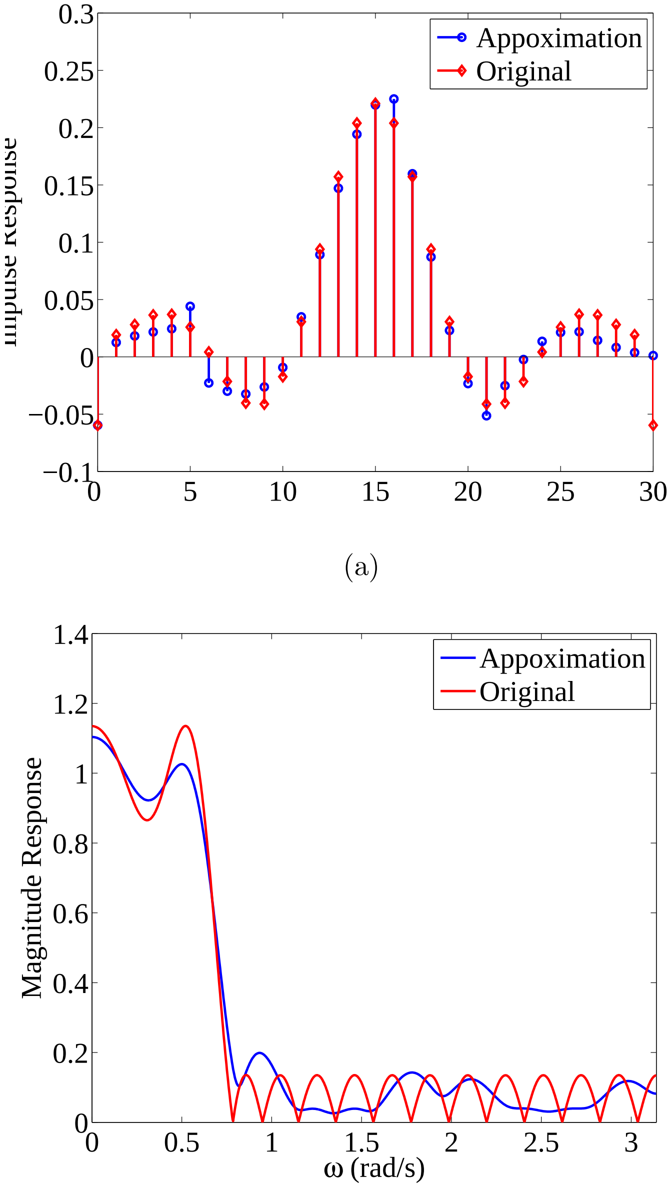
$$G(z) = -0.1037+0.1759z^{-1}+0.2667z^{-2}+0.3432z^{-3}+0.4321z^{-4}+0.7834z^{-5}. \tag{6.24}$$

Figure 6-18b depicts the corresponding magnitude responses. Although the approximate polynomial decomposition method optimizes the approximation with respect to the l_2 norm and the impulse responses differ significantly, the magnitude response of the approximation still exhibits the general characteristics of the original low-pass filter magnitude response. However, due to the difficulty of finding nearby decomposable polynomials in general, this similarity does not always hold. Moreover, the approximation does not have the symmetry in the coefficients losing the desirable linear phase property.

One approach to approximate FIR filters that are symmetric around an integer point consists of first expressing the original frequency response as a polynomial in $\cos\omega$ and then decompose this polynomial rather than decomposing the z-transform directly. More specifically, as given in equation 2.15, the Fourier transform of an even symmetric filter $H(e^{j\omega})$ with order $2L$ and symmetric around $n = L$ can be represented as

$$H_{shifted}(e^{j\omega}) = \sum_{n=-L}^{L} h[n]e^{-j\omega} = \sum_{n=0}^{L} h[n]\cos n\omega \tag{6.25}$$

after a time shift of L samples, where the time shift can be reversed once the fil-



(a)

(b)

Figure 6-18: The comparison of a 30-th order low-pass Parks-McClellan filter $H(z)$ with the passband and stopband edges of 0.20π and 0.24π, respectively, with its approximate decomposition $F(G(z))$: (a) the impulse responses (b) the magnitude responses.

ter is designed. Expanding each term using the Chebyshev polynomials leads to a polynomial in $\cos\omega$ as in equation (2.16), and provided here for convenience,

$$H_{shifted}(e^{j\omega}) = \sum_{n=0}^{L} b_n(\cos\omega)^n. \tag{6.26}$$

In other words, the frequency response of the time shifted filter becomes $B(\cos\omega)$, where B is a polynomial of order L. An approximate decomposition given by

$$B(x) \approx \hat{B}(x) = F(G(x)) \tag{6.27}$$

suggests a modular representation of the FIR filters as a generalized tapped delay line where coefficients of F are the tap coefficients and $G(\cos\omega)$ corresponds to an even symmetric subfilter. The frequency responses $B(\cos\omega)$ and $\hat{B}(\cos\omega)$ were significantly different in simulations even in cases where the coefficients of $\hat{B}(x)$ were a good approximation to those of $B(x)$. This is expected since, in general, the proximity of the coefficients of two polynomials with respect to the l_2 norm has implications only for their values on the unit circle due to Parseval's theorem, and not necessarily on the interval $[-1, 1]$ from which $\cos\omega$ assumes values.

In cases where the symmetry of a given filter is required to be preserved by the approximation, a third approach to perform the decomposition that also yields an acceptable approximation to the frequency response can be to split the impulse response before the decomposition into two subsequences which are related to each other through time reversal. More specifically, the z-transform of the time shifted filter can be expressed as

$$H_{shifted}(z) = C(z) + C(z^{-1}), \tag{6.28}$$

where coefficients of $C(z)$ are those of $h_{shifted}[n]$ for $n \geq 0$ with the exception that its constant term is $\frac{h_{shifted}[0]}{2}$. An approximate decomposition of $C(z)$ as in

$$C(z) \approx F(G(z)) \tag{6.29}$$

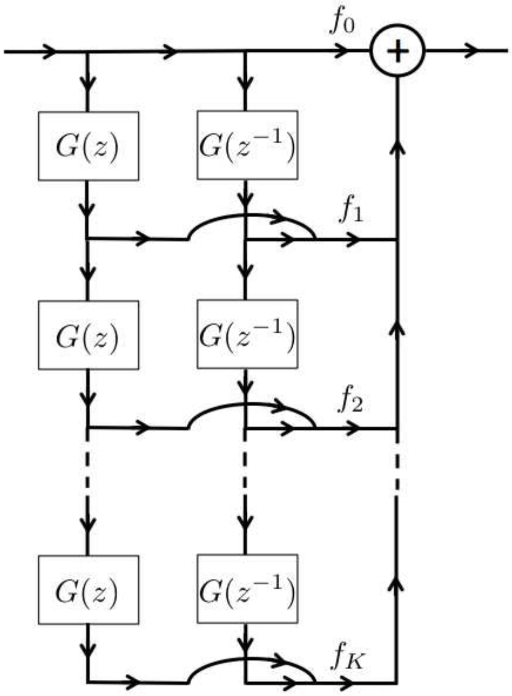


Figure 6-19: The implementation of an even symmetric FIR filter using the approximate decomposition given in equation 6.30.

yields

$$H_{shifted}(z) \approx F(G(z)) + F(G(z^{-1})) \tag{6.30}$$

the coefficients of which are guaranteed to be symmetric. The implementation of this decomposable approximation leads to the modular structure given in Figure 6-19. Although this implementation requires two different subfilters, namely $G(z)$ and $G(z^{-1})$, they are related through a time reversal which does not require the design of an additional subfilter. For on-line applications, this design can be used by introducing a buffer stage at the input to reinstate causality.

The method of symmetric decomposition in equation (6.30) was applied to the Parks-McClellan filter given in Figure 6-18. The polynomial $C(z)$ corresponding to this polynomial is given by

$$\begin{aligned}C(z) &= 0.1105 + 0.2039z^{-1} + 0.1572z^{-2} + 0.0939z^{-3} + 0.0307z^{-4} - 0.0173z^{-5} \\ &\quad - 0.0412z^{-6} - 0.0402z^{-7} - 0.0215z^{-8} + 0.0042z^{-9} + 0.0260z^{-10} \\ &\quad + 0.0370z^{-11} + 0.0364z^{-12} + 0.0281z^{-13} + 0.0192z^{-14} - 0.0597z^{-15},\end{aligned} \tag{6.31}$$

which was approximated as the composition of

$$F(z) = 0.1862 + 0.2261z^{-1} + 0.0020z^{-2} - 0.0068z^{-3} - 0.0132z^{-5} + 0.0097z^{-6} \tag{6.32}$$

and

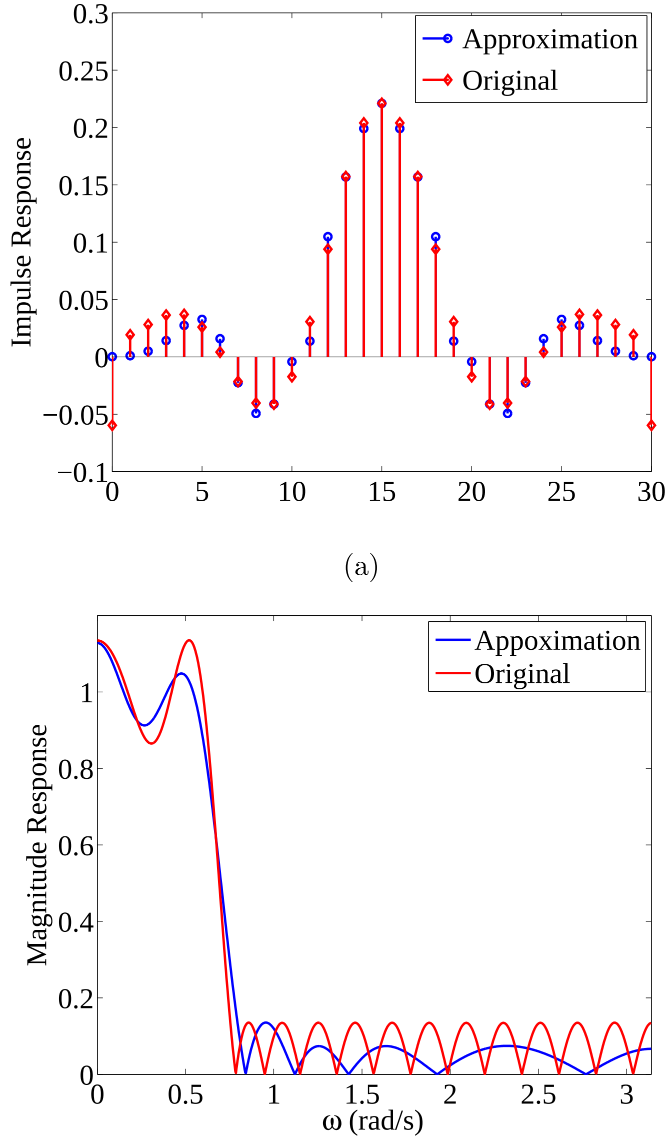
$$G(z) = -0.3359 + 0.8847z^{-1} + 0.7099z^{-2} + 0.4192z^{-3}. \tag{6.33}$$

Figure 6-20a illustrates the original response and its approximation obtained using this approach. The symmetry around $n = 15$ was preserved as desired. As seen in Figure 6-20b which depicts the corresponding magnitude responses, the low-pass characteristics of the original filter were also preserved in this example with a slight widening of the transition region.

6.4 Chapter Conclusions

In this chapter, examples illustrating the benefits of the functional composition and decomposition framework were discussed. Composition of frequency responses was shown to suggest modular design topologies that are desirable for filters as well as provide a systematic framework for an existing signal processing application, namely filter sharpening, in which improved results can be obtained as compared to the traditional ad hoc approaches. Moreover, this was shown to be easily extended to complex valued frequency responses such as causal IIR filters or continuous time filters, which was not possible using the previous methods.

Decomposability of multivariate functions proved to lead to efficient marginaliza-



(a)

(b)

Figure 6-20: The comparison of the 30-th order low-pass Parks-McClellan filter $H(z)$ with the passband and stopband edges of 0.20π and 0.24π, respectively, with its approximate decomposition $F(G(z)) + F(G(z^{-1}))$: (a) the impulse responses (b) the magnitude responses.

tions and an example was given in the context of the sum-product algorithm that was used to denoise a binary text image. Decomposable representations of multivariate functions and polynomials suggested an alternative way to represent multivariate signals and finite sequences with fewer parameters either exactly or approximately. Approximate polynomial decomposition was also shown to be possibly used as a means of designing discrete time modular FIR filters.

Chapter 7

Conclusions

In this thesis, the potential benefits of approaching signal processing applications from a functional composition and decomposition viewpoint were explored. The composition and decomposition methods for functions that are common in signal processing were identified from the current literature, implemented and compared. For certain classes of functions, new decomposition methods were proposed. Several signal processing contexts were revisited, re-interpreted and generalized in a systematic way within this framework equipped with these decomposition techniques. Moreover, new and interesting signal processing applications and approaches were developed.

The signal processing applications in the current literature that can be re-interpreted as a form of functional composition and decomposition as reviewed in Chapter 2 provides evidence that embedding signals and systems to form compositions has been recognized as to have benefits in different contexts. However, the breadth of these applications and the extent to which these operations were utilized were seen to be rather limited with an emphasis on time and frequency transformations. Other applications such as filter sharpening and filter design as a tapped cascaded interconnection of identical subfilters can be viewed as to have the potential to extend these benefits by reusing subfilters to build better filters. However, the general approach in these applications was rather ad hoc and their formulation was also confined within the frequency transformation viewpoint. These motivated the development of a more systematic and unified composition framework for signal processing within which further benefits can be recognized and exploited more conveniently. Such additional benefits that were explored in this thesis can be summarized as modularity, compactness and separability.

As one aspect of such a composition framework, certain composition and decomposition methods were reviewed from the current math literature and extended for an important class of functions for signal processing, namely polynomials, since polynomials often appear in the form of z-transforms when manipulating discrete time FIR signals and systems. The exact decomposition methods for the case in which a polynomial is known to be decomposable were overviewed and compared, which revealed that the results remain very accurate for low orders and deteriorate for higher orders, usually due to numerical issues. Approximate polynomial decomposition methods, including the one developed in this thesis based on Structured Total Least Norm (STLN) method, seemed to suffer from both this limitation as well as the difficult nature of the approximate polynomial decomposition problem. More specifically, the set of decomposable polynomials constitute only a small subset of the space of polynomials which makes the problem of finding a satisfactorily close decomposable approximation difficult even in cases where numerical problems were not an issue. Nevertheless, for the low orders that the methods remained viable, the examples of exact and approximate decompositions were shown to suggest more compact representations of discrete time FIR sequences and an opportunity to design modular filter structures without the requirement of specifying subfilters unlike in the case for filter sharpening. The sensitivity of polynomial composition and decomposition operations were also investigated and a method to decrease sensitivity using compositions with first order polynomials was introduced.

Modular filter design constituted an important class of applications that were identified to benefit from a functional composition and decomposition viewpoint. Development of a functional decomposition algorithm for frequency responses into a polynomial and a rational pre-specified frequency response allowed revisiting filter sharpening within the composition framework and extending its applicability to subfilters with complex valued frequency responses as special cases of modular filters.

This algorithm utilized the fact that the composition of a polynomial and a frequency response is a special case of approximating a target function using a linear combination of continuous subfunctions on a compact set, which accepts solutions from the mathematics literature such as the First Algorithm of Remez. The frequency response decomposition algorithm was further modified to accommodate approximation specifications given only for the magnitude of the frequency responses. The modular filters obtained this way were shown to have the same poles as the subfilter with increased multiplicities, which guaranteed stability even in the presence of perturbations in the coefficients of the composing polynomial.

The decomposition of multivariate discrete functions was explored in the context of applications that require marginalization over all or a subset of the variables of a multivariate function. These applications often arise in signal processing and machine learning, where factorability of the involved multivariate functions is already known to lead to efficient marginalizations. Decomposability was shown to be equivalent to factorability at a higher dimension. This was accomplished by introducing latent variables to the multivariate functions. An upper limit on the alphabet size of these latent variables were given to ensure decomposition indeed led to efficient computations. In cases where the alphabet size of the latent variables were large, approximate decompositions were obtained by enforcing this limit in the decomposition process. Mathematical methods such as nonnegative matrix factorization (NMF) and singular value decomposition (SVD) were utilized in the decomposition process as the decomposition problem was reduced to a form that is equivalent to a matrix factorization. More specifically, the multivariate function to be decomposed was unfolded into a matrix representation for which a rank deficient factorable approximation was found, sometimes referred to as rank factorization in the literature. The same approach also suggested a more compact representation for a given multivariate function, when it was either exactly or approximately decomposable.

The applications either formulated or revisited in this thesis are by no means a complete list as many others can possibly be formulated within the richness of signal processing. For example, one possible further application is a generalization of modular filter design, which in turn was shown to be a generalization of filter sharpening in Section 6.1.1. In that section, the subfilters used in the modular filters were assumed to be identical and the tap coefficients in the generalized tapped delay line were optimized by decomposing an ideal filter response using the frequency response decomposition algorithm developed in Chapter 4. Since this decomposition algorithm utilized the First Algorithm of Remez, which optimally solves the problem of approximation to a continuous function with a generalized polynomial, the method can be extended in a straightforward way to cases for which subfilters are non-identical. For example, consider non-identical subfilters $A_1(e^{j\omega}), \ldots, A_K(e^{j\omega})$. One way to design a filter the characteristics of which is sharper than any of these subfilters is to use them in a configuration as in Figure 7-1 and optimize the branch gains $f_k, k = 1, \ldots, K$ to approximate the ideal filter response $D(\omega)$ in the optimization problem

$$\begin{aligned} \underset{\mathbf{f}}{\text{minimize}} \quad & \Delta \\ \text{subject to} \quad & \left\| D(\omega) - \sum_{k=1}^{K} f_k A_k(e^{j\omega}) \right\|_{\infty} \leq \Delta \end{aligned} \tag{7.1}$$

The resulting filter will have either identical to or smaller minimax errors than that of each one of these subfilters since, otherwise, the optimal branch gain could be chosen to be unity for a filter that has a smaller approximation error and zero for others. Another approach to obtain such a modular filter using multiple non-identical subfilters $A_k(e^{j\omega})$ is to use them in a generalized tapped delay line as in Figure 7-2, in which case the basis functions to approximate the ideal filter response are the products of the subfilter responses along the tapped line. The order of the subfilters were chosen consistent with their indices in Figure 7-2 leading to the optimization problem

$$\begin{aligned} \underset{\mathbf{f}}{\text{minimize}} \quad & \Delta \\ \text{subject to} \quad & \left\| D(\omega) - \sum_{k=0}^{K} f_k \left(\prod_{k'=1}^{k} A_{k'}(e^{j\omega}) \right) \right\|_{\infty} \leq \Delta \end{aligned} \tag{7.2}$$

The quality of the approximation is likely to be different for distinct orderings of the subfilters along the tapped line. This imposes a difficulty on the decision as to which

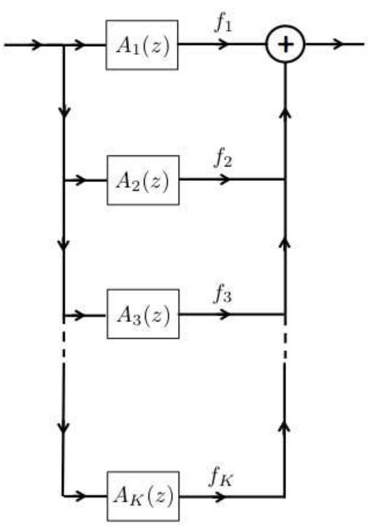


Figure 7-1: A modular filter structure formed with non-identical subfilters, where the branch gains f_k are optimized as in (7.1).

ordering yields a better approximation after the tap coefficients are optimized since there are $K!$ possible orderings. However, this also implies there are many other options to order them if a particular ordering did not yield a satisfactory approximation, a flexibility which was lacking in the case of identical subfilters.

The decomposition techniques that were developed or extended in this thesis were also limited to a few classes of functions that are ubiquitous in signal processing constituting only a part of a complete framework. For example, the decomposition of rational functions was excluded, which has already been explored in the mathematics literature to a certain extent [3, 56], and can be an important future addition to this framework. The exploration of rational function composition and decomposition may allow building modular filters that are not limited to embedding subfilters into FIR filters. More specifically, modular structures can be obtained by replacing delay elements in recursive filters by other subfilters. However, such an implementation requires extra care to avoid delay free loops. Another possible advantage of having rational function decomposition algorithms would be the ability to generalize the operations of decimation and expansion of discrete sequences by integers, which in turn may lead to the generalization of the polyphase representations. For example, the expansion of a discrete sequence $f[n]$ by an integer M, i.e. inserting $M-1$

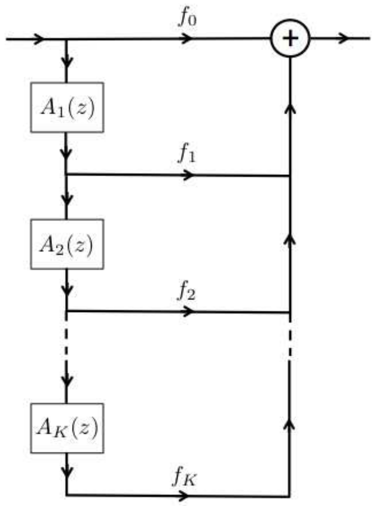


Figure 7-2: A modular filter structure formed with non-identical subfilters, where the branch gains f_k are optimized as in (7.2).

zeros after every sample of this sequence, corresponds to changing its z-Transform $F(z)$ to $F(z^M)$. This can be viewed as the composition of $F(z)$ with the M-th order polynomial $G(z) = z^{-M}$ in z^{-1}. The composition of a rational z-transform $F(z)$ of a discrete time sequence with another rational z-transform $G(z)$ can be viewed as a generalization of an expander. Conversely, the decimation of a sequence can be associated with the decomposition of its z-transform with more general rational functions $G(z)$ than the polynomial z^{-M}. However, the generalization in this latter case is less straightforward since not every sequence has a decomposable z-transform, which requires the definition of an approximate generalized decimation.

In Chapter 4, the approximation of frequency responses as a composition of functions more general than a polynomial and a rational function was not explored as it was currently unclear how to extend the methods developed in that chapter. The development of more general frequency response decomposition techniques is a promising future direction to explore as it suggests an alternative way to design recursive modular structures where the approximation specifications can be imposed directly on the frequency responses rather than discrete time samples. Similarly to the current frequency response decomposition algorithm, that can be extended to cases where specifications are given based on the magnitude responses.

The computational and representational efficiencies of multivariate discrete functions arise the question as to whether developing decomposition techniques for continuous multivariate functions leads to efficiency as well. Indeed, this would provide an extension of the computational efficiency in computing marginalizations of multivariate continuous functions, which motivates investigating such decomposition techniques as a future direction. Moreover, decomposability can potentially be utilized in a similar way to factorability for efficiently sampling both continuous and discrete joint probability density functions in the context of computationally intractable inference problems, for example when taking samples of high dimensional probability densities to compute approximate expectations.

Appendix A

A Convolution Inequality

Lemma: Denote $s_3[n]$ as the convolution of the finite length signals $s_1[n]$ which is non-zero only for $0 \leq n \leq L_1$ and $s_2[n]$ which is non-zero only for $0 \leq n \leq L_2$. Assume $L_1 \geq L_2$, then the energy of these signals satisfy

$$E_{s_3} \leq (L_2 + 1) E_{s_1} E_{s_2},$$

where the energy is given by $E_{s_i} = \sum_{n=-\infty}^{\infty} s_i^2[n],\ i = 1, 2, 3.$

Proof: For $0 \leq n \leq L_1 + L_2$, Cauchy-Schwarz inequality implies

$$\begin{aligned}
s_3^2[n] &= \left(\sum_{m=\max(0,n-L_2)}^{\min(L_1,n)} s_1[m] s_2[n-m] \right)^2 \\
&\leq \left(\sum_{m=\max(0,n-L_2)}^{\min(L_1,n)} s_1^2[m] \right) \left(\sum_{m=\max(0,n-L_2)}^{\min(L_1,n)} s_2^2[n-m] \right) \\
&\leq \left(\sum_{m=\max(0,n-L_2)}^{\min(L_1,n)} s_1^2[m] \right) E_{s_2}.
\end{aligned}$$

Summing for $n = 0, 1, \ldots, (L_1 + L_2)$ yields

$$\begin{aligned} E_{s_3} &\leq \sum_{n=0}^{L_1+L_2} \left(\sum_{m=\max(0,n-L_2)}^{\min(L_1,n)} s_1^2[m] \right) E_{s_2} \\ &= \sum_{m=0}^{L_1} \left(\sum_{n=m}^{m+L_2} s_1^2[m] \right) E_{s_2} = (L_2 + 1) E_{s_1} E_{s_2}, \end{aligned}$$

where the first equality is obtained through re-parametrization of the double summation bounds.

Appendix B

First Algorithm of Remez: Convergence and Optimality

In the first algorithm of Remez outlined in Algorithm 2, certain guarantees exist for the clustering points of the coefficients vectors and their optimality as well as the monotonicity of the the minimax approximation error on $\mathcal{S}$ at each iteration. Their proof by Cheney [13] is restated here with the notation of Chapter 4. Several definitions are given here for the formal statement of the theorem and its proof.

For a given coefficient vector $\mathbf{f} = [f_0, f_1, \ldots, f_K]^T$, the approximation error on $\mathcal{S}$ is given by

$$E(\mathbf{f}, \omega) = D(\omega) - \sum_{k=0}^{K} f_k U_k(\omega), \tag{B.1}$$

and the maximum approximation error is denoted by $\Delta(\mathbf{f})$,

$$\Delta(\mathbf{f}) = \|E(\mathbf{f}, \omega)\|_\infty = \max_{\omega \in \mathcal{S}} \left| D(\omega) - \sum_{k=0}^{K} f_k U_k(\omega) \right|. \tag{B.2}$$

The minimax error Δ_{opt} is defined as the minimum value of $\Delta(\mathbf{f})$ over all possible choices of $\mathbf{f}$,

$$\Delta_{opt} = \inf_{\mathbf{f} \in \mathcal{R}^{K+1}} \Delta(\mathbf{f}). \tag{B.3}$$

The discrete set of frequencies on which the optimization problem (4.3) will be solved at the i-th iteration is denoted as $\mathcal{S}^{[i]}$. For any $\mathbf{f}$, the maximum error on this set can be defined as

$$\Delta^{[i]}(\mathbf{f}) = \max_{\omega \in \mathcal{S}^{[i]}} |E(\mathbf{f}, \omega)| \tag{B.4}$$

which is minimized by $\mathbf{f}^{[i]}$, i.e.,

$$\mathbf{f}^{[i]} = \arg \min_{\mathbf{f} \in \mathcal{R}^{K+1}} \Delta^{[i]}(\mathbf{f}). \tag{B.5}$$

Since this is the minimizer for $\Delta^{[i]}(\mathbf{f})$, it satisfies

$$\Delta^{[i]}(\mathbf{f}^{[i]}) \leq \Delta^{[i]}(\mathbf{f}) \tag{B.6}$$

for every $\mathbf{f} \in \mathcal{R}^{K+1}$, where $\Delta^{[i]}(\mathbf{f}^{[i]}$ can be viewed as the minimax error at the i-th iteration. Once $\mathbf{f}^{[i]}$ is computed at the first step of Algorithm 2, the frequency $\omega^{[i]}$ at which the maximum approximation error on $\mathcal{S}$ occurs is determined at the second step as

$$\omega^{[i]} = \arg \max_{\omega \in \mathcal{S}} E(\mathbf{f}^{[i]}, \omega). \tag{B.7}$$

From the definition of $\Delta(\mathbf{f})$ in equation (B.2),

$$\Delta(\mathbf{f}^{[i]}) = |E(\mathbf{f}^{[i]}, \omega^{[i]})| = \Delta^{[i]}(\mathbf{f}^{[i]}). \tag{B.8}$$

Finally, $\omega^{[i]}$ is added to the set of points for which problem (4.3) will be solved at the next iteration,

$$\mathcal{S}^{[i+1]} = \mathcal{S}^{[i]} \cup \{\omega^{[i]}\}. \tag{B.9}$$

Theorem B.1. *[13]. The minimax error at the i-th iteration, $\Delta^{[i]}(\mathbf{f}^{[i]})$, approaches Δ_{opt} as the iterations continue, i.e.,*

$$\lim_{i \to \infty} \Delta^{[i]}(\mathbf{f}^{[i]}) = \Delta_{opt}. \tag{B.10}$$

Moreover, the sequence $\mathbf{f}^{[i]}$ is bounded and its cluster points all minimize the error.

Proof. First, the sequence $\mathbf{f}^{[i]}$ is shown to be bounded. Since the initial set of points

$\mathcal{S}^{[1]}$ are chosen such that the matrix $\mathbf{V}_{init}$ in equation (4.11) has full column rank, the number θ defined as

$$\gamma = \min_{\|\mathbf{f}\|_1=1} \max_{\omega \in \mathcal{S}^{[1]}} \left| \sum_{k=0}^{K} f_k U_k(\omega) \right| \tag{B.11}$$

is strictly positive, where $\|\cdot\|_1$ denotes l_1 norm of a vector. Therefore,

$$\begin{aligned}
\Delta^{[1]}(\mathbf{f}) &= \max_{\omega \in \mathcal{S}^{[1]}} \left| \sum_{k=0}^{K} f_k U_k(\omega) - D(\omega) \right| \\
&\geq \max_{\omega \in \mathcal{S}^{[1]}} \left| \sum_{k=0}^{K} f_k U_k(\omega) \right| - \max_{\omega \in \mathcal{S}^{[1]}} |D(\omega)| \\
&\geq \max_{\omega \in \mathcal{S}^{[1]}} \left| \sum_{k=0}^{K} f_k U_k(\omega) \right| - \|D(\omega)\|_\infty \\
&\geq \|\mathbf{f}\|_1 \left(\max_{\omega \in \mathcal{S}^{[1]}} \left| \sum_{k=0}^{K} \frac{f_k}{\|\mathbf{f}\|_1} U_k(\omega) \right| \right) - \|D(\omega)\|_\infty \\
&\geq \gamma \|\mathbf{f}\|_1 - \|D(\omega)\|_\infty,
\end{aligned}$$

where the first inequality is due to the triangular inequality for norms and the last inequality follows from the definition of γ. For any coefficient vector $\mathbf{f}$ with $\|\mathbf{f}\|_1 > \frac{2\|D(\omega)\|_\infty}{\gamma}$ and for any $i > 1$,

$$\Delta^{[i]}(\mathbf{f}) \geq \Delta^{[1]}(\mathbf{f}) > \|D(\omega)\|_\infty \geq \Delta^{[i]}(\mathbf{0}),$$

therefore such a vector $\mathbf{f}$ cannot be the minimizer of $\Delta^{[i]}(\mathbf{f})$, i.e., coefficient vectors with l_1 norms greater than $\frac{2\|D(\omega)\|_\infty}{\gamma}$ are never considered and the sequence $\mathbf{f}^{[i]} \in \mathcal{R}^{K+1}$ is bounded, which implies it has least one clustering point [46].

Second, it is shown that the minimax error $\Delta^{[i]}(\mathbf{f}^{[i]})$ approaches Δ_{opt} as iterations continue and the clustering points attain Δ_{opt}. Since for any i the inclusion $\mathcal{S}^{[i]} \subset \mathcal{S}^{[i+1]} \subset \mathcal{S}$ holds, the maximum errors on these sets for a given $\mathbf{f}$ satisfy

$$\Delta^{[i]}(\mathbf{f}) \leq \Delta^{[i+1]}(\mathbf{f}) \leq \Delta(\mathbf{f}).$$

Minimizing each error term with respect to $\mathbf{f}$ yields

$$\Delta^{[i]}(\mathbf{f}^{[i]}) \leq \Delta^{[i+1]}(\mathbf{f}^{[i+1]}) \leq \Delta_{opt} \tag{B.12}$$

since $\mathbf{f}^{[i]}$ is the minimizer of $\Delta^{[i]}(\mathbf{f})$ as defined in equation (B.5). This implies that $\Delta^{[i]}(\mathbf{f}^{[i]})$ is a monotonically non-decreasing sequence that is bounded above by Δ_{opt}, i.e.,

$$\lim_{i\to\infty} \Delta^{[i]}(\mathbf{f}^{[i]}) = \Delta_{opt} - \epsilon \tag{B.13}$$

for some nonnegative ϵ. It can be shown that $\epsilon = 0$ as follows. For any two coefficient vectors $\mathbf{a}$ and $\mathbf{b}$,

$$\begin{aligned} |E(\mathbf{a},\omega) - E(\mathbf{b},\omega)| &= \left|\sum_{k=0}^{K}(a_k - b_k)U_k(\omega)\right| \\ &\leq \sum_{k=0}^{K} |(a_k - b_k)| \cdot |U_k(\omega)| \\ &\leq M\sum_{k=0}^{K} |(a_k - b_k)| \\ &= M\|\mathbf{a} - \mathbf{b}\|_1, \end{aligned} \tag{B.14}$$

where M is maximum value attained by any of the basis functions $U_k(\omega)$ and is finite since the basis functions are all continous on the compact set $\mathcal{S}$ [46]. Triangular inequality can be applied to equation (B.14) to obtain

$$|E(\mathbf{a},\omega)| \leq |E(\mathbf{b},\omega)| + M\|\mathbf{a} - \mathbf{b}\|_1. \tag{B.15}$$

Furthermore, maximization of both sides of this inequality over ω yields

$$\Delta(\mathbf{a}) \leq \Delta(\mathbf{b}) + M\|\mathbf{a} - \mathbf{b}\|_1. \tag{B.16}$$

In order to show $\epsilon > 0$ leads to a contradiction, let $\mathbf{c}$ denote any clustering point of the sequence $\mathbf{f}^{[i]}$. For any $\delta > 0$, there exists an index i such that $\|\mathbf{c} - \mathbf{f}^{[i]}\|_1 < \delta$ and another index $j > i$ such that $\|\mathbf{c} - \mathbf{f}^{[j]}\|_1 < \delta$. Therefore

$$\|\mathbf{f}^{[j]} - \mathbf{f}^{[i]}\|_1 \leq 2\delta \tag{B.17}$$

and

$$\begin{aligned}
\Delta_{opt} &\leq \Delta(\mathbf{c}) \\
&\leq \Delta(\mathbf{f}^{[i]}) + M\delta \\
&= |E(\mathbf{f}^{[i]}, \omega^{[i]})| + M\delta \\
&\leq |E(\mathbf{f}^{[j]}, \omega^{[i]})| + 3M\delta \\
&\leq |E(\mathbf{f}^{[j]}, \omega^{[j]})| + 3M\delta \\
&= \Delta^{[j]}(\mathbf{f}^{[j]}) + 3M\delta \\
&\leq \Delta_{opt} - \epsilon + 3M\delta
\end{aligned} \tag{B.18}$$

where the first inequality follows from the definition of Δ_{opt} in equation (B.3), the second inequality follows from equation (B.16) for $\mathbf{c}$ and $\mathbf{f}^{[i]}$, the first equality follows from the definition of $\Delta(\mathbf{f}^{[i]})$ in equation (B.8), the third inequality follows from equation (B.15), the fourth inequality follows from the definition of $\omega^{[i]}$ in equation (B.7), the second equality follows from equation (B.8) and the last inequality follows from the monotonic convergence of $\Delta^{[i]}(\mathbf{f}^{[i]})$ to Δ_{opt} as implied by equations (B.12) and (B.13).

Since the relationships in (B.18) hold for any choice of δ, it leads to a contradiction if $\delta < \frac{\epsilon}{3M}$. Therefore, the initial hypothesis $\epsilon > 0$ is not true, i.e. $\epsilon = 0$. Moreover, the same relationships imply the clustering point $\mathbf{c}$ attains Δ_{opt}, i.e, $\Delta\mathbf{c} = \Delta_{opt}$.

□

Theorem B.1 does not imply the coefficient vector sequence $\mathbf{f}^{[i]}$ converges. It only states the minimax error sequence will converge and the clustering points of $\mathbf{f}^{[i]}$ will attain the minimax error.

Theorem B.2. *[13].If the basis functions $U_k(\omega)$ also satisfy the Haar condition, the coefficient vector sequence $\mathbf{f}^{[i]}$ will also converge to the unique optimum.*

Proof. The Haar condition implies there is a unique optimum $\mathbf{f}_{opt}$ suuch that $\Delta(\mathbf{f}_{opt}) = \Delta_{opt}$. The sequence $\mathbf{f}^{[i]}$ must have at least one clustering point since it is bounded in $\mathcal{R}^{K+1}$, and it cannot have more than one since that would imply the existence of more than one optimum. Since it is the only clustering point, $\mathbf{f}^{[i]}$ converges to this point. □

Bibliography

[1] S.M. Aji and R.J. McEliece. The generalized distributive law. *Information Theory, IEEE Transactions on*, 46(2):325–343, Mar 2000.

[2] V. Alagar and M. Thanh. Fast polynomial decomposition algorithms. In Bob Caviness, editor, *EUROCAL '85*, volume 204 of *Lecture Notes in Computer Science*, pages 150–153. Springer Berlin / Heidelberg, 1985.

[3] Gutierrez J. Alonso C. and Recio T. A rational function decomposition algorithm by near-separated polynomials. *Journal of Symbolic Computation*, 19(6):527 – 544, 1995.

[4] 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*, pages 55–58, 1999.

[5] Philippe Aubry and Annick Valibouze. Algebraic computation of resolvents without extraneous powers. *European Journal of Combinatorics*, 33(7):1369 – 1385, 2012.

[6] D. R. Barton and R. E. Zippel. A polynomial decomposition algorithm. In *Proceedings of the third ACM Symposium on Symbolic and Algebraic Computation*, SYMSAC '76, pages 356–358, New York, NY, USA, 1976. ACM.

[7] David R. Barton and Richard Zippel. Polynomial decomposition algorithms. *Journal of Symbolic Computation*, 1(2):159–168, June 1985.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[9] H.D. Block and H.P. Thielman. Commutative polynomials. *The Quarterly Journal of Mathematics*, 2(1):241–243, 1951.

[10] Marko Bohanec and Blaz Zupan. A function-decomposition method for development of hierarchical multi-attribute decision models. *Decision Support Systems*, 36(3):215 – 233, 2004.

[11] Brad Botting, Mark Giesbrecht, and John May. Using Riemannian SVD for problems in approximate algebra. In *Proceedings of the International Workshop of Symbolic-Numeric Computation*, pages 209–219, 2005.

[12] Charng-Kann and Chen. A method for synthesizing multiplierless FIR digital filters with narrow transition widths. *Signal Processing*, 62(3):351 – 360, 1997.

[13] E.W. Cheney. *Introduction to approximation theory*. McGraw-Hill, 1966.

[14] J.J. Clark, M. Palmer, and P. Lawrence. A transformation method for the reconstruction of functions from nonuniformly spaced samples. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 33(5):1151–1165, 1985.

[15] A.G. Constantinides. Spectral transformations for digital filters. *Electrical Engineers, Proceedings of the Institution of*, 117(8):1585 –1590, August 1970.

[16] R. M. Corless, M. W. Giesbrecht, D. J. Jeffrey, and S. M. Watt. Approximate polynomial decomposition. In *Proceedings of ISSAC '99*, pages 213–219, 1999.

[17] B. De Moor. Total least squares for affinely structured matrices and the noisy realization problem. *Signal Processing, IEEE Transactions on*, 42(11):3104 – 3113, nov 1994.

[18] Tony D. DeRose. Composing bezier simplexes. *ACM Trans. Graph.*, 7(3):198–221, July 1988.

[19] Charles B Dunham. The weakened first algorithm of remez. *Journal of Approximation Theory*, 31(1):97 – 98, 1981.

[20] M. D. Fried and R. E. MacRae. On curves with separated variables. *Mathematische Annalen*, 180:220–226, 1969.

[21] Michael Fried. On a conjecture of Schur. *Michigan Mathematical Journal*, 17:41–55, 1970.

[22] Shuhong Gao, Erich Kaltofen, John May, Zhengfeng Yang, and Lihong Zhi. Approximate factorization of multivariate polynomials via differential equations. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, ISSAC '04, pages 167–174, New York, NY, USA, 2004.

[23] Joachim von zur Gathen and J. Weiss. Homogeneous bivariate decompositions. *Journal of Symbolic Computation*, 19(5):409 – 434, 1995.

[24] M. Giesbrecht and J. May. New algorithms for exact and approximate polynomial decomposition. In *Proceedings of the SNC Workshop*, July 2005.

[25] Shawn Hershey, Jeffrey Bernstein, Bill Bradley, Andrew Schweitzer, Noah Stein, Theophane Weber, and Benjamin Vigoda. Accelerating inference: towards a full language, compiler and hardware stack. *CoRR*, abs/1212.2991, 2012.

[26] J. Kaiser and R. Hamming. Sharpening the response of a symmetric nonrecursive filter by multiple use of the same filter. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25(5):415 – 422, Oct 1977.

[27] Dan Kalman. A matrix proof of newton's identities. *Mathematics Magazine*, 73(4):313, 2000.

[28] E. Kaltofen and J. May. On approximate irreducibility of polynomials in several variables. In *Proceedings of the 2003 international Symposium on Symbolic and Algebraic Computation*, ISSAC '03, pages 161–168, New York, NY, USA, 2003. ACM.

[29] Erich Kaltofen, John P. May, Zhengfeng Yang, and Lihong Zhi. Approximate factorization of multivariate polynomials using singular value decomposition. *Journal of Symbolic Computation*, 43(5):359 – 376, 2008.

[30] D. Kozen and S. Landau. Polynomial decomposition algorithms. *Journal of Symbolic Computation*, 7:445–456, May 1989.

[31] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498 –519, Feb 2001.

[32] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[33] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *In NIPS*, pages 556–562. MIT Press, 2000.

[34] G. Meinardus. *Approximation of functions: theory and numerical methods.* Heidelberg, 1967.

[35] R. Mersereau, W. Mecklenbrauker, and Jr. Quatieri, T. Mcclellan transformations for two-dimensional digital filtering-Part I: Design. *Circuits and Systems, IEEE Transactions on*, 23(7):405 – 414, Jul 1976.

[36] Vassilis G. Mertzios and AnastasiosN. Venetsanopoulos. Block decomposition structures for the fast modular implementation of two-dimensional digital filters. *Circuits, Systems and Signal Processing*, 8(2):163–185, 1989.

[37] M. Minimair. *Resultants of Composed Polynomials. PhD Thesis.* North Carolina State University, Raleigh, NC, USA, 2000.

[38] S. Nakamura and S. K. Mitra. Design of FIR digital filters using tapped cascaded FIR subfilters. *Circuits, Systems, and Signal Processing*, 1:43–56, 1982.

[39] A. Oppenheim, D. Johnson, and K. Steiglitz. Computation of spectra with unequal resolution using the Fast Fourier Transform. *Proceedings of the IEEE*, 59(2):299 – 301, Feb. 1971.

[40] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[41] P. Pakzad and V. Anantharam. A new look at the generalized distributive law. *Information Theory, IEEE Transactions on*, 50(6):1132–1155, June 2004.

[42] T. Parks and J. McClellan. Chebyshev approximation for nonrecursive digital filters with linear phase. *Circuit Theory, IEEE Transactions on*, 19(2):189 – 194, Mar 1972.

[43] Lawrence R. Rabiner, Ronald W. Schafer, and Charles M. Rader. The chirp z-transform algorithm and its application. *Bell System Technical Journal*, 48(5):1249–1292, 1969.

[44] J. F. Ritt. Prime and composite polynomials. *Transactions of the American Mathematical Society*, 23(1):51–66, 1922.

[45] J. Rosen, H. Park, and J. Glick. Total least norm formulation and solution for structured problems. *SIAM Journal on Matrix Analysis and Applications*, 17(1):110–126, 1996.

[46] W. Rudin. *Principles of Mathematical Analysis.* International Series in Pure and Applied Mathematics. McGraw-Hill International, 1976.

[47] W.M. Ruppert. Reducibility of polynomials $f(x, y)$ modulo p. *Journal of Number Theory*, 77(1):62–70, July 1999.

[48] T. Saramaki. Design of FIR filters as a tapped cascaded interconnection of identical subfilters. *Circuits and Systems, IEEE Transactions on*, 34(9):1011 – 1029, Sep 1987.

[49] M. R. Schroeder. Natural Sounding Artificial Reverberation. *J. Aud. Eng. Soc.*, 10(3):219–223, 1962.

[50] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20(4):151–160, August 1986.

[51] Joseph Anthony Spuria. *Best Approximation by Polynomial Composition.* PhD thesis, Boston University, 1970.

[52] J. W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, 1977.

[53] Gerhard Turnwald. On schur's conjecture. *Journal of the Australian Mathematical Society (Series A)*, 58(03):312–357, 1995.

[54] M.M. Vai. *VLSI Design.* VLSI Circuits Series. CRC Press, 2001.

[55] D. Wei and A.V. Oppenheim. Sampling based on local bandwidth. In *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on*, pages 1103–1107, 2007.

[56] R. Zippel. Rational function decomposition. In *Proceedings of the 1991 international Symposium on Symbolic and Algebraic Computation*, ISSAC '91, pages 1–6, New York, NY, USA, 1991. ACM.

[57] Blaz Zupan, Ivan Bratko, Marko Bohanec, and Janez Demsar. Function decomposition in machine learning. In Georgios Paliouras, Vangelis Karkaletsis, and Constantine Spyropoulos, editors, *Machine Learning and Its Applications*, volume 2049 of *Lecture Notes in Computer Science*, pages 71–101. Springer Berlin / Heidelberg, 2001.

Epilogue

An ever increasing part of the research being performed in the Digital Signal Processing Group (DSPG) is initiated by the question "What can this inspire about signal processing?", which usually inquires about other fields, tools and concepts such as biology, circuits, quantum physics, conservation, solitons, fractals and chaos theory among many others. Asking this question as a first step is an early sign that the research and the thesis will probably not be traditional as in identifying a problem, searching for tools to solve it and conclude the job done at the end. Rather, it is an exciting warning to prepare yourself for and be open-minded about the inspirations that follow as you almost meditate into this question. These ideas and inspirations, as they arise, may and in most cases do present alternative and creative approaches for different signal processing applications. This approach aligns well with Al Oppenheim's semi-formal research group mission statement: having fun, chasing ideas, to find solutions in search of problems. I think my research has been a very typical example of this atypical approach and it investigated the inspirations from functional composition and decomposition operations for signal processing applications. With this inverse research approach, the development of tools and ideas emerged seemingly independent of each other. However, at later stages of the thesis development and especially in the writing phase, these pieces came together very nicely as parts of a bigger pattern. This unifying theme almost gives the impression that the whole encounter was planned as the thesis started, but in fact it evolved into what it is. This exciting process and the final product that unified the little pieces of research into one consistent story was described accurately and concisely by Al as being similar to making a "patchwork quilt". In this epilogue, I am hoping to tell the story of my patchwork quilt, i.e. the actual order of ideas and how they emerged, how they seemed to be independent or unrelated and how the whole unifying picture came together towards the end.

One great thing that made me click with DSPG after my master's degree in a totally unrelated area, the reliability of Gallium Nitride transistors, was that I had no idea about what would make a good thesis in signal processing. This was surprisingly a good thing in this group: you have freedom and actually are encouraged to explore things until you find something that is interesting as opposed to bringing along an ordinary problem to solve under the roof of DSPG. A few weeks after joining the group in the spring of 2010, I found myself in the group library going through old DSPG theses to see "which thesis I would have liked to be my own", per Al's suggestion, as the first stage of the exploration phase. Until the end of Spring 2011 semester, I read and thought about many small projects including zeros of random polynomials, Volterra series expansion, time-frequency analysis and Fractional Fourier Transform.

One thesis that particularly intrigued me was the SM thesis by Dennis Wei. He explored the idea of local bandwidth and efficient sampling of signals that were obtained by time-warping another bandlimited signal. I found the idea of time-warping interesting in itself, and wanted to understand whether the original bandlimited signal and the warping function are unique up to a linear scaling for a given signal, and how to recover them in such a case. Among the many keywords I used to locate previous work on this approach was functional decomposition, since time warping is the composition of a signal with another function of time. This is when I realized functional decomposition is a difficult but very interesting problem in mathematics. The ignition for my thesis topic was one of the regular research discussions with Al probably in early Fall 2011 during which he said the idea of functional composition was rather unheard of in signal processing contexts, and it would be interesting to see "what we can learn from functional composition and decomposition for signal processing".

I was not sure how to approach this question and even which classes of functions to consider. As a first cut, I decided to start looking into polynomials. The reason was two fold. First, there is an abundance of literature on their decomposition starting from early 1900s that I was aware of. Second, polynomials arise in many contexts in signal processing in the form of z-transforms. My first attempt was to understand which applications could exploit decomposable polynomials, which in turn required investigating the decomposition techniques for them. I was convinced that there is at least one possible application whether decomposition has a straightforward or difficult solution. It can be used to encrypt sequences and data if it is a hard problem. If an easy solution existed, it would provide opportunities for representing sequences with fewer degrees of freedom corresponding to sparsity from the perspective of parametric representations. I also explored the current literature to see which of the existing applications in signal processing can be viewed as a form of functional composition, whether or not they involved polynomials. I found quite a few examples, which constituted Chapter 2 of this thesis, and realized that they were usually in the context of time and frequency warping. These observations, the literature search and own efforts to understand opportunities presented by functional composition and decomposition constituted the promising pieces of my PhD thesis proposal at the end of Fall 2011.

I had plenty of material which was getting a handful, so I took Al's advice in Spring 2012 to target a conference paper on polynomial composition and decomposition to organize some of the thinking. I was still trying to understand the extent to which composition and decomposition remain reliable. Therefore, I started developing analytic expressions for the sensitivities of polynomial composition and decomposition operations. This turned into a conference paper with the simulations that we did with my great tireless officemate and colleague Guolong Su, which also eventually became the last part of Chapter 3 of my thesis. In the mean time, I found polynomial decomposition techniques from the mathematics and computer science literature, which became a part of Chapter 3 along with comparisons that we performed with Guolong. By the end of Spring 2012 and after the extensive literature search, development of own techniques, very useful discussions with Prof. Bjorn Poonen from the MIT Mathematics Department, several research meetings with Al and Guolong and Prof. Pablo Parrilo's semidefinite programming class including an intense discussion on polynomials, I could more or less frame the opportunities and the limitations of polynomial decomposition techniques. As this was already promising, I also started pondering on how to model nondecomposable polynomials as decomposable to extend the possible applications to these cases as well.

Just as I was thinking that I was spending way too much time on polynomials and was wondering decomposition of what other functions could be exploited, I received a very interesting question by Ben Vigoda in Summer 2012 during my internship interview talk at Analog Devices Lyric Labs. I was talking about how polynomial decomposition led to more compact and efficient representations for discrete sequences when he asked whether this could be applied to functions that are given as a look-up table for several different variables. Obviously, this would have helped them do certain operations efficiently in inference problems. It took me almost half a summer internship and several useful discussions with my colleague Theo Weber at Lyric to decrypt this question into a mathematical representation besides a few other small projects. I was convinced that I found a new class of functions that I could play with, namely multivariate functions consisting of discrete variables as they also appeared in many contexts in signal processing and machine learning. Theo proposed the idea to use the δ-functions to connect variables of subfunctions in factor graphs if the multivariate function is decomposable, which we later extended to functions more general than $\delta()$. The idea was neat and simple, yet I was not sure how to place this piece of insight into the composition framework and generalize to other applications not necessarily using factor graphs. Totally independent from this stream of work, starting from early stages of my PhD, Al and I occasionally discussed how raising problems to a higher dimensionality and solving it there could be easier or more useful. Although this seemed counter-intuitive, he gave the example where a one dimensional signal is raised to a higher dimensionality by taking an outer product by itself before multiplying a Volterra kernel and being projected back to its original dimension through integration. It was not until Fall 2013 that I realized introducing δ-functions with latent variables into decomposable multivariate functions was a great example of what we have been discussing with Al: raising the dimensionality of the function to make it factorable, a property known for its efficiencies in representations and computations. This manipulation allowed me to borrow matrix factorization algorithms from the literature to decompose multivariate functions, which turned into Chapter 5 of this thesis. Using an image denoising algorithm which was previously explored at Lyric, I was able to show the approach actually led to computational efficiency as discussed in Chapter 6.

In Fall 2012, I worked to finish what I started before the summer, namely the approximate decomposition of polynomials. I came across a very interesting set of papers which established a one-to-one relationship between the decomposability of a univariate polynomial and the factorability of an associated bivariate polynomial. The connection between decomposability of a lower dimensional function and the factorability of a higher dimensional associated function surprisingly manifested itself in the context of polynomials as well as the multivariate functions. Although I had been working on them for about one year, this parallel behavior between two very different functional forms did not become so clear until I was trying to build my "patchwork quilt" at the late stages of my thesis writing and defense preparation. Certain approximate polynomial decomposition methods based on this relationship that I found intriguing were formulated in terms of Ruppert matrices, which involved coefficients of the polynomial under question. I formulated a new approximate polynomial decomposition algorithm following other examples in the literature which utilized Ruppert matrices but using a different algorithm based on Structured Total Least Norm algorithm. This method as well as the review of the existing exact and approximate polynomial decomposition methods turned into a conference paper and the rest of Chapter 3.

In one of the conferences where I presented a poster, a professor was intrigued by the idea of obtaining equivalent compositions with lower sensitivities that were discussed in Chapter 3. He asked whether compositions with polynomials of order more than unity can be used to obtain equivalent compositions. As I tried to explain that only first order polynomials are invertible, he insisted that inversions over finite intervals can be considered. Although his suggestion eventually did not fit into our analyses and purposes, this gave me a new perspective to consider decomposability. More specifically, I started thinking for ways to exploit approximate decompositions of continuous functions only on intervals of interest rather than requiring the approximate decomposition to be close on the entire range of definition or with respect to their parameters such as coefficients in polynomials. After all, approximating functions on finite intervals had already proved useful in signal processing in the context of Parks-McClellan filters. I started inquiring whether the Remez Exchange Algorithm can be extended to approximations with compositions on intervals. In Spring 2013, my search led me to a 1970 PhD thesis from Boston University Mathematics Department, which I ended up printing using a microfilm viewer in the basement of MIT Hayden Library. This was a nice experience in itself, but also helpful in that this thesis saved me valuable research time. It had counterexamples that implied minimax approximations using polynomial compositions lacked a characterization in terms of alternations for which the Remez Exchange Algorithm would not apply. However, I found another algorithm due to Remez in Cheney's book on approximation theory that would allow me decompose a continuous function as a composition of a polynomial and any other pre-specified continuous function. I was able to, for example, specify a target frequency response and approximate it as the weighted sum of powers of another frequency response. This evolved into what I called as the decomposition of frequency responses discussed in Chapter 4. I had a solution, which was now "searching a problem". It turned out that filter sharpening, an application which I encountered two years before exploring this, was a perfect example that could exploit this approach, and extend itself to more general modular structures. This was one of the main applications in Chapter 6.

The seemingly independent pieces of this research had proved to bear signs of a story in the background such as the relationship between decomposability and factorability. However I had to zoom out and take a 30000-feet look to frame my work as an answer to the original question that got us into this path. What had we learned from functional composition and decomposition for signal processing? As an eye-candy for several posters and a simple visual to help introduce functional composition and decomposition to different audiences, we used Russian dolls. The idea of nesting dolls was similar to what we meant by functional composition, namely nesting functions. As I was starting to write my thesis, I stopped staring at the pile of work I had done so far and tried to answer for myself the implications of nesting signals and systems virtually as Russian dolls. Three implications came to me almost as a flash: compactness, separability and modularity. The Russian dolls took much less space when they were nested, implying a more compact form just like how decomposable polynomials required much fewer parameters to be represented. Moreover, separating or de-nesting dolls would make it easier to work on each of them individually, for example to dye them in a different color, which is similar to computations on functions becoming more manageable when they are separated. Finally, the dolls resembled each other and looked nicer when displayed side by side similar to better and more sophisticated structures obtained by reusing identical simpler modules as in modular filter design. I remembered my confusion when the research question was first raised, with no clue that it would lead here although it seems in this thesis that the goals were set on day one to exploit these three themes. It was both inspiring, exciting and rewarding to see how the little pieces naturally found their correct place on the patchwork quilt while I was busy with the details and technicalities.

Al suggested that every graduate of DSPG have a short story describing their thesis and research experience consisting of only six words, similar to what is commonly referred to as a “six-word novel”. I feel like my six words should reflect the nature of the process in which my thesis topic emerged and evolved into a quilt in this rather unanticipated manner while I was investigating different directions. Therefore, I chose them as “Search for it, it finds you”.