Polynomial Decomposition Algorithms in Signal Processing

by

Guolong Su

B.Eng., Electronic Engineering, Tsinghua University, China (2011)

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

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author .....
Department of Electrical Engineering and Computer Science
May 20, 2013

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

Accepted by .....
Professor Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Theses

Polynomial Decomposition Algorithms in Signal Processing

by

Guolong Su

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Polynomial decomposition has attracted considerable attention in computational mathematics. In general, the field identifies polynomials $f(x)$ and $g(x)$ such that their composition $f(g(x))$ equals or approximates a given polynomial $h(x)$. Despite potentially promising applications, polynomial decomposition has not been significantly utilized in signal processing. This thesis studies the sensitivities of polynomial composition and decomposition to explore their robustness in potential signal processing applications and develops effective polynomial decomposition algorithms to be applied in a signal processing context. First, we state the problems of sensitivity, exact decomposition, and approximate decomposition. After that, the sensitivities of the composition and decomposition operations are theoretically derived from the perspective of robustness. In particular, we present and validate an approach to decrease certain sensitivities by using equivalent compositions, and a practical rule for parameter selection is proposed to get to a point that is near the minimum of these sensitivities. Then, new algorithms are proposed for the exact decomposition problems, and simulations are performed to make comparison with existing approaches. Finally, existing and new algorithms for the approximate decomposition problems are presented and evaluated using numerical simulations.

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

Acknowledgments

I would like to first thank my research advisor Prof. Alan Oppenheim, whose remarkable wisdom and guidance for this thesis are significant in many ways. I am deeply impressed by Al's unconventional creativity, and I am particularly thankful for our research meetings that were not only academically invaluable but also emotionally supportive. Al is a great source of sharp and insightful ideas and comments, which make me always feel more energetic to explore more stimulating directions after meeting with him. I am also sincerely grateful to Al, a great mentor, for patiently improving my academic writing level and shaping me into a person with higher maturity. In addition to his tremendous intellectual support, I really appreciate Al's impressive warmheartedness and caring efforts in helping me with a number of personal issues to make my life comfortable at MIT and during the summer internship.

I would also like to thank Sefa Demirtas for his helpful contribution and our friendly research collaboration during the last two years. The topic of this thesis had been discovered by Al and Sefa before I joined MIT; Sefa has been a great close collaborator with a significant role in the development of many results in this thesis. In addition, I sincerely thank Sefa for carefully reviewing the thesis and patiently helping me with other English documents. I am also grateful to Sefa for his enthusiastic encouragement, guidance and support.

It has been an enjoyable and rewarding journey for me as a member of Digital Signal Processing Group (DSPG). I would like to sincerely thank the past and present DSPG members including: Tom Baran, Petros Boufounos, Sefa Demirtas, Dan Dudgeon, Xue Feng, Yuantao Gu, Zahi Karam, Tarek Lahlou, Martin McCormick, Milutin Pajovic, Charlie Rohrs, and Laura von Bosau. The collaborative research environment and the intriguing weekly brainstorming meetings make the group an enthusiastic, creative, and harmonious academic family. Especially, thanks to Laura for her responsible efforts to make the group function efficiently and constant readiness to help; thanks to Tarek for his enthusiastic conversation in the night when we both heard for the first time of the problem of polynomial decomposition as well as his

helpful comments as a native speaker on my English documents; thanks to Zahi for the helpful discussion on root-finding algorithms; thanks to Yuantao for the enthusiastic discussion on Vandermonde matrices as well as your valuable guidance and friendly mentorship during my time in Tsinghua University. In addition to group members, I would thank Yangqin for the helpful discussion on the Galois group.

I would also like to thank Davis Pan at Bose Corporation. Davis is a great and easygoing manager, who is truly caring about his interns even beyond the scope of the internship. The experience with Davis was really enjoyable both intellectually and personally.

I feel really fortunate to have met and been accompanied by Shengxi. I thank her for her companionship, kind heart, patience, as well as the love and happiness she has brought me. Special thanks for her encouragement for going to the gym and her correction of my English pronunciation.

I am deeply grateful to my parents for their unconditional love and unlimited support. Through the years, I know that they are always available to talk to for support and encouragement. Their guidance and wisdom in life have been essential for the development of my personality. My love and appreciation for my parents is beyond any words. Thanks also to my extended family for their love and support when I am far away from home.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Functional composition $(\alpha \circ \beta)(x)$ is defined as $(\alpha \circ \beta)(x) = \alpha(\beta(x))$, where $\alpha(x)$ and $\beta(x)$ are arbitrary functions. It can be interpreted as a form of cascading the two functions $\beta(\cdot)$ and $\alpha(\cdot)$. One application of functional composition in signal processing is in time warping [1–3]. The basic idea of time warping is to replace the time variable t with a *warping function* $\psi(t)$, so the time-axis is stretched in some parts and compressed in other parts. In this process, a signal $s(t)$ is time-warped to a new signal $s(\psi(t))$ in the form of functional composition. It is possible that the original signal $s(t)$ is non-bandlimited, while the composed signal $s(\psi(t))$ is band-limited [1–3]. For example, the chirp signal $s(t) = \cos(at^2)$ is non-bandlimited [4], but it can be warped into the band-limited signal $s(\psi(t)) = \cos(at)$ by the warping function $\psi(t) = \sqrt{|t|}$. For certain signals, if proper warping functions are chosen, time warping may serve as an anti-aliasing technique in sampling. In addition to its application in efficient sampling, time warping has also been employed to model and compensate for certain nonlinear systems [5]. Moreover, time warping may be utilized in speech recording to improve speech verification [6].

As a particular case of functional composition, polynomial composition may also find potentially beneficial applications in signal processing. The precise definition of polynomial composition is stated as follows with the symbols to be used throughout

this thesis. For polynomials

$$f(x) = \sum_{m=0}^{M} a_m x^m, \; g(x) = \sum_{n=0}^{N} b_n x^n, \tag{1.1}$$

their *composition* is defined as

$$h(x) = (f \circ g)(x) = f(g(x)) = \sum_{m=0}^{M} a_m (g(x))^m = \sum_{k=0}^{MN} c_k x^k. \tag{1.2}$$

If a polynomial $h(x)$ can be expressed in form (1.2), then it is *decomposable*; otherwise it is *indecomposable*. For simplicity, we assume that all polynomials $f(x)$, $g(x)$, and $h(x)$ have real coefficients; however, most results of this thesis also apply for complex polynomials.

The inverse process to polynomial composition is called *polynomial decomposition*, which generally means determining $f(x)$ and $g(x)$ given $h(x)$. Polynomial decomposition is potentially as useful as composition in signal processing applications. For example, polynomial decomposition may be employed in efficient representation of signals [7]. If a signal can be represented by a decomposable polynomial $h(x)$, then it can also be represented by its decomposition $(f \circ g)(x)$. Note that $h(x)$ has $(MN + 1)$ degrees of freedom, while $f(x)$ and $g(x)$ together have degrees of freedom $(M + N)$. [1] Thus, the decomposition representation of the signal has a reduction of $(MN + 1 - M - N)$ degrees of freedom and thus can potentially be used for signal compression. Another possible application of polynomial decomposition is an alternative implementation of decomposable FIR filters [7–9]. The z-transform [4] of an FIR filter $Q(z) = \sum_{n=0}^{K} q[n] z^{-n}$ is a polynomial in z^{-1}. Figure 1-1 (a) shows the direct form implementation [4] of a decomposable filter $H(z^{-1})$; an alternative implementation of this filter is presented in Fig. 1-1 (b) [7–9]. Comparing Fig. 1-1 (a) and Fig. 1-1 (b) shows that the alternative implementation of $H(z^{-1})$ substitutes the FIR filter $G(z^{-1})$ for each time delay in $F(z^{-1})$.

[1] The degrees of freedom of $f(x)$ and $g(x)$ are fewer than the total number of their coefficients, since the decomposition is not unique. Further discussion can be found in (4.31) in Section 4.2 and in the paragraph immediately above Section 5.1.

Another related problem is approximate decomposition, which determines $f(x)$ and $g(x)$ such that $h(x) \approx (f \circ g)(x)$ for an indecomposable polynomial $h(x)$. Approximate decomposition may have wider applications than exact decomposition, since most real signals are unlikely to be exactly decomposable. The above argument about reduction in degrees of freedom implies the low density of decomposable polynomials in the polynomial space. In particular, given M and N, all the decomposable polynomials are located on a manifold of dimensions $(M+N)$, while the whole space has $(MN+1)$ dimensions. As the length $(MN+1)$ of the polynomial increases, the reduction in degrees of freedom also grows, which makes decomposable polynomials less and less dense in the polynomial space. Thus, it is unlikely that an arbitrarily long signal will correspond to an exactly decomposable polynomial.

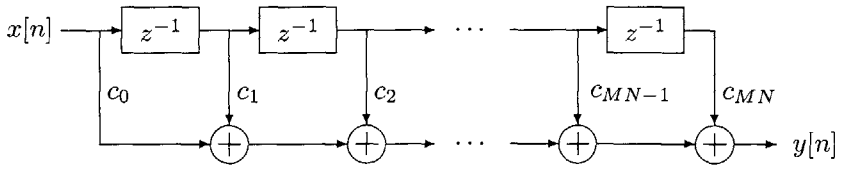
Indecomposable polynomials can possibly be represented by approximate decomposition. For example, if a signal corresponds to an indecomposable polynomial $h(x)$, the approximate decomposition method might be employed in compressing the signal into the composition of $f(x)$ and $g(x)$, with a decrease in degrees of freedom by $(MN+1-M-N)$ and possibly without much loss in quality. However, since exact decomposition can be thought of as a problem of identification while approximate decomposition corresponds to modeling, approximate decomposition appears much more challenging than exact decomposition.

1.2 Objective

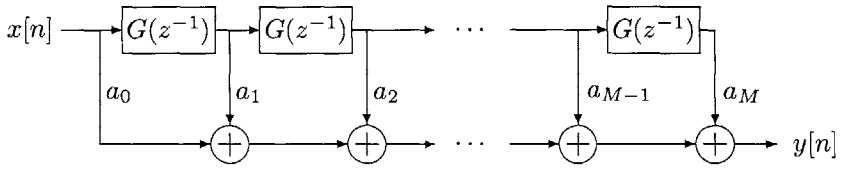
Many of the applications of functional composition are currently being explored by Sefa Demirtas [7]. The primary goals of this thesis are to theoretically evaluate the robustness of polynomial composition and decomposition as well as to develop effective algorithms for the decomposition problems in both the exact and the approximate cases. [2]

Robustness is characterized by sensitivities of composition and decomposition,

[2]Many of the results in this thesis are included in [10, 11] by S. Demirtas, G. Su, and A. V. Oppenheim.



(a) Direct Form [4]



(b) Alternative Implementation [7–9]

Figure 1-1: Two Implementations of a Decomposable FIR Filter where $H(z^{-1}) = (F \circ G)(z^{-1})$

where the sensitivities represent the maximum relative magnification of the energy among all small perturbations. Lower sensitivity indicates higher robustness and higher reliability in applications. Equivalent compositions are shown to be effective to decrease certain sensitivities, especially when the degree of $h(x)$ is high.

New algorithms are proposed for both the exact and the approximate decomposition problems. We propose two types of decomposition algorithms: those with polynomial coefficients as input and those with polynomial roots as input. Different algorithms have different capabilities to decompose high order polynomials and different robustness to noise.

The remainder of this thesis is organized as follows. Chapter 2 briefly summarizes the basic properties and existing work on polynomial decomposition. Chapter 3 states the precise definition of the problems that will be explored in this thesis. The sensitivities are theoretically studied in Chapter 4, where we also develop an approach to decrease certain sensitivities by equivalent compositions. The algorithms for the exact and the approximate decomposition problems are presented and evaluated with numerical simulation in Chapter 5 and 6, respectively. Chapter 7 concludes this thesis and proposes potential problems for future work.

Chapter 2

Background

2.1 Polynomial Composition Properties

A number of basic properties that will be utilized about polynomial composition are briefly stated in this section. The proofs of these properties are omitted and can be found in the references [12, 13].

1. Polynomial composition is linear with respect to $f(x)$ but not to $g(x)$. Namely, $(f_1 + f_2) \circ g = f_1 \circ g + f_2 \circ g$ always holds, but generally $f \circ (g_1 + g_2) \neq (f \circ g_1) + (f \circ g_2)$.

2. Polynomial composition satisfies the associative law, i.e., $(f \circ g) \circ p = f \circ (g \circ p)$.

3. Polynomial composition generally does not satisfy the commutative law, i.e., $(f \circ g) \neq (g \circ f)$ in general. However, two special situations are worthy of notice [12]. The *cyclic* polynomials, which have only a single power term x^n, satisfy $x^{MN} = x^M \circ x^N = x^N \circ x^M$. Similarly, $T_{MN}(x) = (T_M \circ T_N)(x) = (T_N \circ T_M)(x)$, where $T_n(x) = \cos(n \arccos(x))$ is the nth-order Chebyshev Polynomial.

4. Polynomial composition is not unique in the following three situations. First, it holds in general that $f \circ g = (f \circ q^{-1}) \circ (q \circ g)$, where $q(x) = q_1 x + q_0$ is a first-degree polynomial and $q^{-1}(x) = (x - q_0)/q_1$ is the inverse function of $q(x)$ under composition. Second, $x^{MN} \left(v(x^M)\right)^M = x^M \circ \left(x^N v(x^M)\right) = \left(x^N (v(x))^M\right) \circ x^M$

where $v(x)$ is an arbitrary polynomial. Third, as stated above, the Chebyshev polynomials can have more than one decomposition. These three scenarios of non-unique decomposition include all possible situations: if there are two ways to decompose a given polynomial into indecomposable components, then the two ways of decomposition can differ only in the above three situations, as is described more precisely in [13] on this topic.

5. If $h(x)$ is decomposable, the flipped polynomial $h_{flip}(x) = \sum_{k=0}^{MN} c_{MN-k}x^k$ is not necessarily decomposable.

6. Similar to a minimum phase filter [4], we refer to $h(x)$ as a *minimum phase polynomial* if all the roots are inside the unit circle. If $f(x)$ and $g(x)$ are both minimum phase, the composition $h(x)$ is not necessarily minimum phase. However, as is shown in Appendix A, if a decomposable polynomial $h(x)$ is minimum phase, then there always *exists* a non-trivial minimum phase decomposition. More precisely, if a minimum phase polynomial $h(x)$ is decomposable into two components with degrees M and N ($M > 1$ and $N > 1$), then we can construct an equivalent composition of $h(x)$, the components in which have the same degrees and are both minimum phase polynomials. The proof in Appendix A also provides the construction of a minimum phase decomposition from a non-minimum phase decomposition of a minimum phase decomposable polynomial, which may have potential application in implementation of minimum phase decomposable filters.

7. Similar to a linear phase filter [4], we refer to $h(x)$ as a *linear phase polynomial* if the coefficients have odd or even symmetry. If $f(x)$ and $g(x)$ are both linear phase, the composition $h(x)$ is not necessarily linear phase. If a decomposable $h(x)$ is linear phase, there may not exist a non-trivial decomposition with linear phase components, where *non-trivial* means the degrees of the components are both larger than one.

2.2 Review of Existing Polynomial Decomposition Algorithms

This section briefly summarizes the existing polynomial decomposition algorithms.[1] In mathematics and symbolic algebraic computation, polynomial decomposition has been an important topic for decades [14–24]. The first algorithm for the exact decomposition problem was proposed by Barton and Zippel in [14]. This algorithm is based on deep mathematical results to convert the univariate polynomial decomposition problem into the bivariate polynomial factorization problem [15, 19] and has exponential-time complexity. Recently, Barton's algorithm has been improved to polynomial-time complexity in [16], and extended into the approximate decomposition problem [16]. An alternative decomposition algorithm was proposed by Kozen and Landau in [17], which requires the degrees M and N of $f(x)$ and $g(x)$, respectively. Compared with Barton's method, Kozen's algorithm is much more straightforward to implement. A third type of algorithm was based on the algorithm that Aubry and Valibouze described in [20], which explores the relationship between the coefficients and the roots of a polynomial. Kozen's algorithm is theoretically equivalent to Aubry's algorithm; however, they may show significantly different robustness in numerical computation.

Approximate decomposition algorithms fall into two main categories. The first category is to find a locally optimal solution based on the assumption that the input polynomial $h(x)$ is the sum of a decomposable polynomial and a small perturbation. The algorithm proposed by Corless et al. in [18] belongs to this category; this algorithm employs the result in [17] as an initial value and proceeds iteratively to find a locally optimal approximate solution. However, the assumption that $h(x)$ is a nearly decomposable polynomial would not hold in most cases, and this fact in general constrains the applicability of Corless's algorithm. Moreover, there is no general guarantee for the convergence of this algorithm, nor the global optimality of the result.

[1] Much of this background was uncovered by Sefa Demirtas [11].

The second category of approximate decomposition algorithms is a generalization of Barton's algorithm [14] that employs bivariate polynomial factorization [15, 19]. This approach has a deep theoretical foundation and makes no assumption that $h(x)$ is nearly decomposable. As an example, Giesbrecht and May [16] employ the theoretical results in [19, 25] and convert the approximate decomposition problem into a special case of the Structured Total Least Squares (STLS) problem [21, 24]. There are a number of heuristic algorithms to solve the STLS problem, such as the Riemannian Singular Value Decomposition (RiSVD) algorithm [21] and the weighted penalty relaxation algorithm [22, 26]. However, none of these heuristic algorithms guarantees convergence or global optimality in a general setting, which may be a disadvantage of this type of approach. The Ruppert matrix [25], which is critical in the corresponding STLS problem, has such high dimension that the numerical accuracy and efficiency may become problematic. In summary, determining the optimal approximate decomposition of an arbitrary polynomial still remains a challenging problem.

Theoretically, it does not appear to be possible to determine the distance from the given $h(x)$ to the nearest decomposable polynomial. Although there is a lower bound on this distance in [16], this bound may not be sufficiently tight.

Chapter 3

Problem Definition

In this chapter, we state the problems to be explored in this thesis. The goal of this thesis is to theoretically study the robustness of polynomial composition and decomposition, and to design polynomial decomposition algorithms for both the exact and the approximate cases.[1] The robustness is characterized by the sensitivities in Section 3.1. For both the exact decomposition in Section 3.2 and the approximate decomposition in Section 3.3, there are two problems defined with different input information.

3.1 Sensitivity Analysis

In polynomial composition, a perturbation of the components will typically result in a corresponding perturbation of the composed polynomial, and vice versa for polynomial decomposition. For example, if the component $f(x)$ has a perturbation of $\Delta f(x)$, then there is a corresponding perturbation $\Delta h(x) = ((f + \Delta f) \circ g)(x) - (f \circ g)(x)$ in the composed polynomial $h(x) = (f \circ g)(x)$. However, the energy of perturbation can be significantly different between the components and the composed polynomial. Sensitivities for the composition operation describe the maximal extent to which the small perturbation of the polynomial components is magnified in the perturbation of the composed polynomial [10], and sensitivities for the decomposition operation

[1] Many of the results were developed in collaboration with Sefa Demirtas [7].

describe the inverse maximum magnification.

3.1.1 Sensitivities of the Coefficients

In this section, we consider the sensitivities of the coefficients of the polynomials. In the composition $h(x) = (f \circ g)(x)$, the sensitivity from $f(x)$ to $h(x)$ is defined as [10]

$$S_{f \to h} = \max_{\|\mathbf{\Delta f}\|_2 = \kappa} \left(\frac{R_{\mathbf{\Delta h}}}{R_{\mathbf{\Delta f}}} \right), \tag{3.1}$$

where $R_{\mathbf{\Delta h}}$ and $R_{\mathbf{\Delta f}}$ are defined as

$$R_{\mathbf{\Delta h}} = \frac{\|\mathbf{\Delta h}\|_2^2}{\|\mathbf{h}\|_2^2}, \text{ and } R_{\mathbf{\Delta f}} = \frac{\|\mathbf{\Delta f}\|_2^2}{\|\mathbf{f}\|_2^2}, \tag{3.2}$$

in which $\mathbf{f}$, $\mathbf{h}$, $\mathbf{\Delta f}$, and $\mathbf{\Delta h}$ are vectors of the coefficients of respective polynomials, $\|\cdot\|_2$ denotes the l_2-norm, and κ is the magnitude of the perturbation $\mathbf{\Delta f}$ which is constrained to be sufficiently small. To a first-order approximation, $S_{f \to h}$ and other sensitivities become independent of the specific value of κ when κ is sufficiently small. Both $R_{\mathbf{\Delta h}}$ and $R_{\mathbf{\Delta f}}$ are the ratios of the perturbation polynomial energy over the original polynomial energy, and they represent the *relative perturbation* of $h(x)$ and $f(x)$, respectively. If we consider the coefficients of polynomials as vectors, then the sensitivity $S_{f \to h}$ is the maximum magnification of the relative perturbation from $f(x)$ to $h(x)$, among all possible directions of perturbation. Since the sensitivity is defined in the worst case scenario, it serves as an upper bound on the ratio between relative perturbation $\frac{R_{\mathbf{\Delta h}}}{R_{\mathbf{\Delta f}}}$ when the perturbation $\mathbf{\Delta f}$ is small.

Similarly, we can define the sensitivity from $g(x)$ to $h(x)$ in the composition process [10],

$$S_{g \to h} = \max_{\|\mathbf{\Delta g}\|_2 = \kappa} \left(\frac{R_{\mathbf{\Delta h}}}{R_{\mathbf{\Delta g}}} \right), \tag{3.3}$$

where $R_{\mathbf{\Delta h}}$ is defined in (3.2) and $R_{\mathbf{\Delta g}}$ denotes

$$R_{\mathbf{\Delta g}} = \frac{\|\mathbf{\Delta g}\|_2^2}{\|\mathbf{g}\|_2^2}, \tag{3.4}$$

in which the magnitude κ of the perturbation $\mathbf{\Delta g}$ is sufficiently small. This sensitivity involves the worst direction of the perturbation of $g(x)$, in which the perturbation is maximally magnified after composition.

In the composition process, the resulting polynomial is of course decomposable even after perturbation of its components. In contrast, a decomposable polynomial can become indecomposable after an arbitrary perturbation. Consequently, components do not exist for the perturbed polynomial, and thus sensitivities are undefined in this scenario. In addition, even if the polynomial remains decomposable after perturbation, the degrees of the components may change, which again makes it difficult to assess the sensitivities. To avoid these situations, in our discussion of sensitivities in the decomposition operation, we consider only the perturbation after which the polynomial still remains decomposable and the degrees of the components remain the same. In such cases, the sensitivities of the decomposition process imply the extent to which an error is magnified from the composed polynomial to its components.

With the constraints specified above on the perturbation, the sensitivity from $h(x)$ to $f(x)$ is defined as [10]

$$S_{h\to f} = \max_{\|\mathbf{\Delta f}\|_2=\kappa} \left(\frac{R_{\mathbf{\Delta f}}}{R_{\mathbf{\Delta h}}} \right), \tag{3.5}$$

and the sensitivity from $h(x)$ to $g(x)$ is defined as [10]

$$S_{h\to g} = \max_{\|\mathbf{\Delta g}\|_2=\kappa} \left(\frac{R_{\mathbf{\Delta g}}}{R_{\mathbf{\Delta h}}} \right), \tag{3.6}$$

where perturbations $\mathbf{\Delta f}$ and $\mathbf{\Delta g}$ have sufficiently a small magnitude of κ.

In summary, the sensitivities within the *coefficient triplet* (f, g, h) are defined in Table 3.1.

3.1.2 Sensitivities of the Roots

Before we introduce the sensitivities of the roots, we first show the relationship of roots of the polynomials in the composition process. Denoting z_h as a root of $h(x)$,

Table 3.1: Definitions of the Sensitivities within the Coefficient Triplet (f, g, h)

From	To	Process	Sensitivity Definition
f	h	Composition	$S_{f\to h} = \max_{\|\mathbf{\Delta f}\|_2=\kappa} \left(\frac{R_{\mathbf{\Delta h}}}{R_{\mathbf{\Delta f}}}\right)$
g	h	Composition	$S_{g\to h} = \max_{\|\mathbf{\Delta g}\|_2=\kappa} \left(\frac{R_{\mathbf{\Delta h}}}{R_{\mathbf{\Delta g}}}\right)$
h	f	Decomposition	$S_{h\to f} = \max_{\|\mathbf{\Delta f}\|_2=\kappa} \left(\frac{R_{\mathbf{\Delta f}}}{R_{\mathbf{\Delta h}}}\right)$
h	g	Decomposition	$S_{h\to g} = \max_{\|\mathbf{\Delta g}\|_2=\kappa} \left(\frac{R_{\mathbf{\Delta g}}}{R_{\mathbf{\Delta h}}}\right)$

then $h(z_h) = f(g(z_h)) = 0$. Thus, if we define

$$z_f \triangleq g(z_h), \tag{3.7}$$

then $f(z_f) = 0$ and z_f is a root of $f(x)$. In other words, evaluating the polynomial $g(x)$ at the roots of $h(x)$ results in the roots of $f(x)$; equivalently, the roots z_h of the composed polynomial are the solutions to the equation (3.7) when z_f and $g(x)$ are both determined. As a result, the root relationship (3.7) can be regarded as a description of polynomial composition that is alternative to (1.2): while (1.2) characterizes the relationship among the coefficients f, g, and h in the composition process, (3.7) describes the relationship among z_f, g, and z_h.

We can also study the robustness of polynomial composition and decomposition from the perspective of the relationship among the *root triplet* (z_f, g, z_h) in (3.7). If z_f or $g(x)$ are perturbed, then there will typically be a corresponding perturbation in z_h to satisfy the constraint (3.7); similarly, perturbation in z_h will typically result in perturbations in z_f and $g(x)$, under the assumption of decomposability of the perturbed polynomial into components with unchanged degrees. As a result, the worst-case magnification of the magnitude of perturbation can be described by sensitivities.

In Section 3.1.1, we have shown sensitivities that are described within the *coefficient triplet* (f, g, h) and listed in Table 3.1; similarly, we can define four new sensitivities within the *root triplet* (z_f, g, z_h). The four new sensitivities are summa-

Table 3.2: Definitions of the Sensitivities within the Root Triplet $(z_f,\ g,\ z_h)$

From	To	Process	Sensitivity Definition
z_f	z_h	Composition	$S_{z_f \to z_h} = \max_{\|\mathbf{\Delta z_f}\|_2 = \kappa} \left(\frac{R_{\mathbf{\Delta z_h}}}{R_{\mathbf{\Delta z_f}}} \right)$
g	z_h	Composition	$S_{g \to z_h} = \max_{\|\mathbf{\Delta g}\|_2 = \kappa} \left(\frac{R_{\mathbf{\Delta z_h}}}{R_{\mathbf{\Delta g}}} \right)$
z_h	z_f	Decomposition	$S_{z_h \to z_f} = \max_{\|\mathbf{\Delta z_f}\|_2 = \kappa} \left(\frac{R_{\mathbf{\Delta z_f}}}{R_{\mathbf{\Delta z_h}}} \right)$
z_h	g	Decomposition	$S_{z_h \to g} = \max_{\|\mathbf{\Delta g}\|_2 = \kappa} \left(\frac{R_{\mathbf{\Delta g}}}{R_{\mathbf{\Delta z_h}}} \right)$

rized in Table 3.2, where perturbation $\mathbf{\Delta z_f}$ or $\mathbf{\Delta g}$ have sufficiently small magnitudes of κ.

It may seem asymmetric to include the coefficients of $g(x)$ rather than its roots z_g in the root triplet $(z_f,\ g,\ z_h)$; however, such a root triplet has higher mathematical simplicity: the coefficients of $g(x)$ have direct relationship with z_f and z_h as shown in (3.7), while we do not have such direct relationship for the roots z_g.

Till now, we have formulated both the coefficient sensitivities and the root sensitivities. In Chapter 4, we derive expressions and develop bounds for these sensitivities; we also compare the coefficient sensitivities and the root sensitivities; in addition, we explore the effects of equivalent compositions on sensitivities with first-degree polynomials.

3.2 Exact Decomposition

In the exact case, the input polynomial $h(x)$ is guaranteed to be decomposable. The approach to decomposition depends on the input information available. With different input information, two problems in the exact case are as follows.

Problem 1: Exact Decomposition with Coefficients as Input

Given the coefficients of $h(x)$ as well as $\deg(f(x)) = M$ and $\deg(g(x)) = N$ where $\deg(h(x)) = MN$, determine a choice for $f(x)$ and $g(x)$ such that $h(x) = (f \circ g)(x)$.

Problem 2: Exact Decomposition with Roots as Input

Given the roots of $h(x)$ as well as $\deg(f(x)) = M$ and $\deg(g(x)) = N$ where $\deg(h(x)) = MN$, determine a choice for $f(x)$ and $g(x)$ such that $h(x) = (f \circ g)(x)$.

The above two problems are of course closely related. On the one hand, they are theoretically equivalent, since the coefficients of a polynomial determine the roots, and vice versa. On the other hand, the two problems have considerable differences in numerical computation. Obtaining the roots from the coefficients is difficult for a polynomial whose degree is high or whose roots are clustered. Thus, an algorithm for Problem 1 may have better performance than an algorithm that first determines the roots from the coefficients and then solves Problem 2, since the roots may be numerically inaccurate. For similar reasons, an algorithm that directly works with roots may be more robust than an algorithm that first obtains the coefficients and then solves Problem 1. In fact, algorithms with polynomial coefficients as input and those with roots as input may have considerably different performance.

The solutions to both problems are potentially useful in signal processing. If we want to decompose a decomposable signal, the signal values naturally correspond to the coefficients of the z-transform, which is in the form of Problem 1. In addition to the coefficients, the roots of polynomials are also important in the z-transform of a filter and the pole-zero analysis of the transfer function of a system. With the knowledge of precise roots in such applications, Problem 2 is potentially useful.

In the statements of both problems, the degrees of components are included in the input information. However, we can also perform decomposition for a decomposable polynomial without knowing the degrees of the components. Since the product of the degrees of the components equals to the degree of the composed polynomial, we can perform the decomposition algorithms for each candidate pair of degrees of the components, and then we check whether the output of the algorithms is a valid decomposition. In this case, the computational complexity is higher than the scenario where the degrees of components are available.

3.3 Approximate Decomposition

In this class of problems, we consider indecomposable polynomials; i. e. $h(x) \neq (f \circ g)(x)$ for any non-trivial $f(x)$ and $g(x)$. In this case, our goal is to find a decomposable polynomial nearest in some sense to $h(x)$ as the approximate decomposition. The problems are defined with a one-to-one correspondence to those in the exact case.

Problem 3: Approximate Decomposition with Coefficients as Input

Given the coefficients of $h(x)$ as well as $\deg(f(x)) = M$ and $\deg(g(x)) = N$ where $\deg(h(x)) = MN$, determine a choice for $f(x)$ and $g(x)$ minimizing $\text{dist}(h(x), (f \circ g)(x))$.

Problem 4: Approximate Decomposition with Roots as Input

Given the roots of $h(x)$ as well as $\deg(f(x)) = M$ and $\deg(g(x)) = N$ where $\deg(h(x)) = MN$, determine a choice for $f(x)$ and $g(x)$ minimizing $\text{dist}(h(x), (f \circ g)(x))$.

Here $\text{dist}(s(x), t(x))$ is a distance measure between polynomials $s(x)$ and $t(x)$ with the same degree. As an example, [18] uses the l_2 norm corresponding to the energy of the residue error:

$$\|s(x) - t(x)\|_2 = \left(\sum_{k=0}^{P} |s_k - t_k|^2 \right)^{\frac{1}{2}}, \tag{3.8}$$

where $P = \deg(s(x)) = \deg(t(x))$.

Similar to the exact case, these two problems here are also related. Although the two problems are equivalent in theory, the algorithms for the two problems may have different numerical performance.

Chapter 4

Sensitivities of Polynomial Composition and Decomposition

This chapter explores the sensitivities of polynomial composition and decomposition to study their robustness, as proposed in Section 3.1. [1] Derivation and bounds for the sensitivities are developed in Section 4.1. The effects of equivalent compositions with first-degree polynomials on sensitivities are studied in Section 4.2, with the proposal of a rule for parameter selection to approach the minimum sensitivities. Simulation results are shown in Section 4.3.

4.1 Derivation of the Sensitivities

4.1.1 Sensitivities of the Coefficients

In this section, we derive the expressions and bounds for $S_{f\to h}$, $S_{g\to h}$, $S_{h\to f}$, and $S_{h\to g}$ that are defined in Table 3.1. First, we derive the sensitivities of $f(x)$ and $h(x)$ with respect to each other. The polynomial composition $h(x) = (f \circ g)(x)$ in (1.2) can be equivalently presented by the following linear equations:

$$\mathbf{h} = \mathbf{G}\mathbf{f}, \tag{4.1}$$

[1] Many of the results in this chapter are summarized in [10] by S. Demirtas, G. Su, and A. V. Oppenheim.

where the vectors $\mathbf{f} = [a_M, a_{M-1}, \ldots, a_0]^{\mathrm{T}}$ and $\mathbf{h} = [c_{MN}, c_{MN-1}, \ldots, c_0]^{\mathrm{T}}$ have elements as the coefficients of respective polynomials, and the matrix $\mathbf{G}$ is the self-convolution matrix of the polynomial $g(x)$

$$\mathbf{G} = [g^M, \ldots, g^2, g^1, g^0]. \tag{4.2}$$

From (4.1), it follows that the perturbations satisfy the linear relationship

$$\mathbf{\Delta h} = \mathbf{G\Delta f}. \tag{4.3}$$

Thus, the maximal magnification of perturbation energy between $\mathbf{\Delta f}(x)$ and $\mathbf{\Delta h}(x)$ is

$$S_{f\to h} = \max_{\|\mathbf{\Delta f}\|_2=\kappa} \left(\frac{R_{\mathbf{\Delta h}}}{R_{\mathbf{\Delta f}}}\right) = \frac{\|\mathbf{f}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot \max_{\|\mathbf{\Delta f}\|_2=\kappa} \left(\frac{\|\mathbf{G\Delta f}\|_2^2}{\|\mathbf{\Delta f}\|_2^2}\right) = \sigma_{\mathbf{G},\max}^2 \frac{\|\mathbf{f}\|_2^2}{\|\mathbf{h}\|_2^2}, \tag{4.4}$$

where $\sigma_{\mathbf{G},\max}$ is the largest singular value of the matrix $\mathbf{G}$. If the perturbation $\mathbf{\Delta f}$ is in the direction of the maximal magnification of the matrix $\mathbf{G}$, then the sensitivity above is achieved. In addition, the sensitivity in (4.4) does not depend on the magnitude κ of the perturbation, due to the linear relationship in (4.3).

For a fixed polynomial $g(x)$, we can obtain an upper bound for the sensitivity $S_{f\to h}$ over all possible polynomials $f(x)$ with degree M. Since $\mathbf{h} = \mathbf{Gf}$,

$$\sigma_{\mathbf{G},\min}^2 \leq \frac{\|\mathbf{h}\|_2^2}{\|\mathbf{f}\|_2^2} \leq \sigma_{\mathbf{G},\max}^2,$$

where $\sigma_{\mathbf{G},\min}$ is the smallest singular value of the matrix $\mathbf{G}$. Consequently, the sensitivity $S_{f\to h}$ is bounded by

$$1 \leq S_{f\to h} \leq \mathrm{cond}(\mathbf{G})^2, \tag{4.5}$$

where $\mathrm{cond}(\mathbf{G}) = \frac{\sigma_{\mathbf{G},\max}}{\sigma_{\mathbf{G},\min}}$ is the condition number of the matrix $\mathbf{G}$. The upper bound in (4.5) depends only on the polynomial $g(x)$ and the degree of $f(x)$, since the matrix $\mathbf{G}$ is the self-convolution matrix of $g(x)$ up to the power of M. The upper and lower bounds in (4.5) are both tight in theory; however, the sensitivity with a given $f(x)$

may be significantly lower than the upper bound. The upper bound is theoretically achieved when $f(x)$ and $\Delta f(x)$ are in the directions of minimal and maximal magnification of the matrix $\mathbf{G}$, respectively; the lower bound is achieved when $f(x)$ and $\Delta f(x)$ are both in the direction of maximal magnification. However, for a given $f(x)$, the sensitivity $S_{f\to h}$ is empirically shown to be significantly lower than the upper bound in a number of cases as shown in Section 4.3.1.

Using the same approach as above, the sensitivity $S_{h\to f}$ is:

$$S_{h\to f} = \max_{\|\mathbf{\Delta f}\|_2=\kappa} \left(\frac{R_{\mathbf{\Delta f}}}{R_{\mathbf{\Delta h}}}\right) = \left(\frac{\|\mathbf{f}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot \min_{\|\mathbf{\Delta f}\|_2=\kappa} \frac{\|\mathbf{G\Delta f}\|_2^2}{\|\mathbf{\Delta f}\|_2^2}\right)^{-1} = \left(\sigma^2_{\mathbf{G},\min}\frac{\|\mathbf{f}\|_2^2}{\|\mathbf{h}\|_2^2}\right)^{-1}, \quad (4.6)$$

which is also independent on the magnitude κ of the perturbation, due to the linear relationship in (4.3).

This sensitivity has the same bounds as in (4.5), specifically

$$1 \le S_{h\to f} \le \mathrm{cond}(\mathbf{G})^2. \quad (4.7)$$

The upper bound for $S_{h\to f}$ is achieved when $f(x)$ and $\Delta f(x)$ are in the directions of maximal and minimal magnification of the matrix $\mathbf{G}$, respectively. The lower bound is achieved when $f(x)$ and $\Delta f(x)$ are both in the direction of minimal magnification.

Next, we derive the sensitivities of $g(x)$ and $h(x)$ with respect to each other. In contrast to $f(x)$, the relationship between $h(x)$ and $g(x)$ is nonlinear. However, if the energy of perturbation is sufficiently small, then we can use the linear approximation [2]

$$\begin{aligned}(f\circ(g+\Delta g))(x) &= \sum_{m=0}^{M} a_m(g(x)+\Delta g(x))^m \\ &\approx \sum_{m=0}^{M} a_m\left((g(x))^m + m\cdot(g(x))^{m-1}\cdot\Delta g(x)\right) \\ &= (f\circ g)(x) + \Delta g(x)\cdot d(x), \end{aligned} \quad (4.8)$$

[2]Derivation similar to (4.8) and (4.11) also appears in [18].

where $d(x)$ is the composition of $f'(x)$ (the derivative of $f(x)$) and $g(x)$:

$$d(x) = \sum_{k=0}^{(M-1)N} d_k x^k = (f' \circ g)(x). \tag{4.9}$$

Consequently, we have

$$\Delta h(x) = (f \circ (g + \Delta g))(x) - (f \circ g)(x) \approx \Delta g(x) \cdot d(x), \tag{4.10}$$

which indicates that $\Delta h(x)$ is approximately the convolution of $\Delta g(x)$ and $d(x)$, when $\Delta g(x)$ is sufficiently small. Expressed in matrix form,

$$\mathbf{\Delta h} \approx \mathbf{D}\mathbf{\Delta g}, \tag{4.11}$$

where the matrix $\mathbf{D}$ is a $(MN+1) \times (N+1)$ Teoplitz matrix with the last column as $[0, 0, \ldots, 0, d_{(M-1)N}, d_{(M-1)N-1}, \ldots, d_0]^{\mathrm{T}}$. Consequently, the sensitivity $S_{g \to h}$ defined in (3.3) becomes

$$\begin{aligned} S_{g \to h} &= \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot \max_{\|\mathbf{\Delta g}\|_2 = \kappa} \left(\frac{\|\mathbf{\Delta h}\|_2^2}{\|\mathbf{\Delta g}\|_2^2} \right) \\ &= \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot \max_{\|\mathbf{\Delta g}\|_2 = \kappa} \left(\frac{\|\mathbf{D}\mathbf{\Delta g}\|_2^2}{\|\mathbf{\Delta g}\|_2^2} \right) \\ &= \sigma_{\mathbf{D},\max}^2 \cdot \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{h}\|_2^2}, \end{aligned} \tag{4.12}$$

where $\sigma_{\mathbf{D},\max}$ is the maximum singular value of the matrix $\mathbf{D}$, and the perturbation $\mathbf{\Delta g}$ has a sufficiently small magnitude of κ.

As developed in Appendix B, the sensitivity $S_{g \to h}$ has the upper bound

$$S_{g \to h} \leq (N+1)\|\mathbf{g}\|_2^2 \cdot \sigma_{\mathbf{T},\max}^2, \tag{4.13}$$

where $\sigma_{\mathbf{T},\max}$ is the maximum singular value of the matrix $\mathbf{T}$:

$$\mathbf{T} = \mathbf{G}\mathbf{V}(\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1}\mathbf{G}^{\mathrm{T}}. \tag{4.14}$$

The matrix $\mathbf{V}$ in (4.14) is a $(M+1) \times (M+1)$ matrix with sub-diagonal elements $M, M-1, M-2, \ldots, 2, 1$ and other elements as zero, i.e.,

$$\mathbf{V} = \begin{bmatrix} 0 & & & & & & \\ M & 0 & & & & 0 & \\ & M-1 & 0 & & & & \\ & & & \ddots & & & \\ & & & & 0 & & \\ & 0 & & & 2 & 0 & \\ & & & & & 1 & 0 \end{bmatrix}. \tag{4.15}$$

This upper bound holds for all possible polynomials $f(x)$ with a fixed degree M and a fixed polynomial $g(x)$. Although this bound is not always tight for any $g(x)$, numerical simulations presented in Section 4.3.1 indicate empirically that it is a reasonably good estimate.

The sensitivity $S_{h \to g}$ in the decomposition process can be derived in an approach similar to that used for $S_{g \to h}$:

$$\begin{aligned} S_{h \to g} &= \max_{\|\mathbf{\Delta g}\|_2 = \kappa} \left(\frac{R_{\mathbf{\Delta g}}}{R_{\mathbf{\Delta h}}} \right) \\ &= \left(\frac{\|\mathbf{g}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot \min_{\|\mathbf{\Delta g}\|_2 = \kappa} \left(\frac{\|\mathbf{\Delta h}\|_2^2}{\|\mathbf{\Delta g}\|_2^2} \right) \right)^{-1} \\ &= \left(\frac{\|\mathbf{g}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot \min_{\|\mathbf{\Delta g}\|_2 = \kappa} \left(\frac{\|\mathbf{D\Delta g}\|_2^2}{\|\mathbf{\Delta g}\|_2^2} \right) \right)^{-1} \\ &= \left(\sigma_{\mathbf{D},\min}^2 \cdot \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{h}\|_2^2} \right)^{-1}, \end{aligned} \tag{4.16}$$

where $\sigma_{\mathbf{D},\min}$ is the minimum singular value of the matrix $\mathbf{D}$, and the perturbation $\mathbf{\Delta g}$ has a sufficiently small magnitude of κ.

4.1.2 Sensitivities of the Roots

In this section, we analyze the sensitivities among the triplet $(z_f,\ g\ , z_h)$, which are defined in Table 3.2. Before the derivation of the sensitivities, we study the structure

of the roots of decomposable polynomials.

If we denote the roots of $f(x)$ and $h(x)$ as $z_f(i)$ $(1 \leq i \leq M)$ and $z_h(k)$ $(1 \leq k \leq MN)$, respectively, then the composed polynomial $h(x)$ can be factored in the following two ways:

$$h(x) = a_M \prod_{i=1}^{M} (g(x) - z_f(i)) = c_{MN} \prod_{k=1}^{MN} (x - z_h(k)), \tag{4.17}$$

where a_M and c_{MN} are the coefficients of the highest degree term in $f(x)$ and $h(x)$, respectively. If we denote

$$\tilde{g}_i(x) = g(x) - z_f(i), \quad 1 \leq i \leq M, \tag{4.18}$$

then (4.17) shows that the union of the roots of all $\tilde{g}_i(x)$ forms the roots of $h(x)$. Thus, we can partition the roots $z_h(k)$ $(1 \leq k \leq MN)$ into M *groups* A_i $(1 \leq i \leq M)$, where all the roots in the i-th group A_i satisfy $\tilde{g}_i(x) = 0$, i.e.,

$$A_i = \{k \mid g(z_h(k)) = z_f(i),\ 1 \leq k \leq MN\},\ 1 \leq i \leq M. \tag{4.19}$$

There are N elements in each group A_i; the roots in a group correspond to the same one root of $f(x)$.

To simplify the analysis of the sensitivity of the roots, we assume that the derivative of $g(x)$ is non-zero at the roots of $h(x)$, i.e., $g'(z_h(i)) \neq 0$ for $1 \leq i \leq MN$. In fact, this assumption holds in most scenarios: if $h(x)$ has only single roots, then it ensures $g'(z_h) \neq 0$, since $h'(z_h) \neq 0$ and $h'(x) = f'(g(x))g'(x)$.

First, we consider the sensitivities of z_f and z_h with respect to each other, when the polynomial $g(x)$ is fixed. Since the roots z_h in each group correspond to the same root of $f(x)$, the perturbation of $z_f(i)$ affects only those roots of $h(x)$ that correspond to $z_f(i)$, i.e., $z_h(k)$ where $k \in A_i$. For a sufficiently small perturbation $\Delta z_h(k)$ $(k \in A_i)$, we can employ the following linear approximation

$$g(z_h(k) + \Delta z_h(k)) \approx g(z_h(k)) + g'(z_h(k)) \cdot \Delta z_h(k), \quad k \in A_i,$$

so the the perturbation $\Delta z_f(i)$ is

$$\Delta z_f(i) = g\left(z_h(k) + \Delta z_h(k)\right) - g(z_h(k)) \approx g'(z_h(k)) \cdot \Delta z_h(k), \quad k \in A_i.$$

Consequently, if we perturb only one root $z_f(i)$, the ratio of perturbation energy is

$$\frac{\|\mathbf{\Delta z_h}\|_2^2}{\|\mathbf{\Delta z_f}\|_2^2} = \frac{\sum_{k\in A_i} |\Delta z_h(k)|^2}{|\Delta z_f(i)|^2} \approx \sum_{k\in A_i} \frac{1}{|g'(z_h(k))|^2}. \tag{4.20}$$

With the assumption that $g'(z_h) \neq 0$, the ratio in (4.20) is finite.

If two or more roots of $f(x)$ are perturbed, then each $z_f(i)$ affects the corresponding roots of $h(x)$ (i.e., $z_h(k)$, $k \in A_i$) independently. In this case, the ratio of perturbation energy is

$$\frac{\|\mathbf{\Delta z_h}\|_2^2}{\|\mathbf{\Delta z_f}\|_2^2} = \frac{\sum_{i=1}^{M} \left(\sum_{k\in A_i} |\Delta z_h(k)|^2\right)}{\sum_{i=1}^{M} |\Delta z_f(i)|^2} \approx \frac{\sum_{i=1}^{M} |\Delta z_f(i)|^2 \left(\sum_{k\in A_i} \frac{1}{|g'(z_h(k))|^2}\right)}{\sum_{i=1}^{M} |\Delta z_f(i)|^2}. \tag{4.21}$$

Since (4.21) is a weighted summation of (4.20) over the range of $i = 1, 2, \ldots, M$, we know

$$\min_{i\in\{1,2,\ldots,M\}} \sum_{k\in A_i} \frac{1}{|g'(z_h(k))|^2} \leq \frac{\|\mathbf{\Delta z_h}\|_2^2}{\|\mathbf{\Delta z_f}\|_2^2} \leq \max_{i\in\{1,2,\ldots,M\}} \sum_{k\in A_i} \frac{1}{|g'(z_h(k))|^2},$$

Consequently, the sensitivity $S_{z_f \to z_h}$ in the composition process can be derived:

$$\begin{aligned} S_{z_f\to z_h} &= \max_{\|\mathbf{\Delta z_f}\|_2=\kappa} \frac{R_{\mathbf{\Delta z_h}}}{R_{\mathbf{\Delta z_f}}} \\ &= \frac{\|\mathbf{z_f}\|_2^2}{\|\mathbf{z_h}\|_2^2} \cdot \max_{\|\mathbf{\Delta z_f}\|_2=\kappa} \frac{\|\mathbf{\Delta z_h}\|_2^2}{\|\mathbf{\Delta z_f}\|_2^2} \\ &= \frac{\|\mathbf{z_f}\|_2^2}{\|\mathbf{z_h}\|_2^2} \cdot \left(\max_{i\in\{1,2,\ldots,M\}} \sum_{k\in A_i} \frac{1}{|g'(z_h(k))|^2} \right), \end{aligned} \tag{4.22}$$

and the sensitivity $S_{z_h \to z_f}$ in the decomposition process is:

$$\begin{aligned} S_{z_h \to z_f} &= \max_{\|\mathbf{\Delta z_f}\|_2 = \kappa} \frac{R_{\mathbf{\Delta z_f}}}{R_{\mathbf{\Delta z_h}}} \\ &= \left(\min_{\|\mathbf{\Delta z_f}\|_2 = \kappa} \frac{R_{\mathbf{\Delta z_h}}}{R_{\mathbf{\Delta z_f}}} \right)^{-1} \\ &= \left(\frac{\|\mathbf{z_f}\|_2^2}{\|\mathbf{z_h}\|_2^2} \cdot \left(\min_{i \in \{1,2,\ldots,M\}} \sum_{k \in A_i} \frac{1}{|g'(z_h(k))|^2} \right) \right)^{-1}, \end{aligned} \tag{4.23}$$

where the perturbation $\mathbf{\Delta z_f}$ has a sufficiently small magnitude of κ.

It is shown in (4.22) and (4.23) that the sensitivities depend on the derivative of $g(x)$ at the roots of $h(x)$. This dependence is intuitive from the fact that $z_f = g(z_h)$: if the derivatives $g'(z_h)$ have small magnitudes, then a big perturbation on z_h may still result in a small perturbation on z_f, so the composition sensitivity $S_{z_f \to z_h}$ may be large while the decomposition sensitivity $S_{z_h \to z_f}$ may be small; in contrast, a large derivative could result in a small composition sensitivity $S_{z_f \to z_h}$ and a large decomposition sensitivity $S_{z_h \to z_f}$.

As a result, the clustered roots of $h(x)$ influence the robustness between z_f and z_h in composition and decomposition. For composition, if $h(x)$ has clustered roots in one group A_i, then the derivatives $g'(z_h)$ at those clustered roots have small values, so the composition operation is vulnerable to noise. In contrast, if $h(x)$ does not have clustered roots, or the clustered roots belong to different groups, then the composition operation is robust. For decomposition, if clustered roots appear in each group A_i, then the sensitivity $S_{z_h \to z_f}$ has a low value, so the decomposition has high robustness.

Next we derive the sensitivities of z_h and $g(x)$ with respect to each other, i.e., $S_{g \to z_h}$ in the composition process and $S_{z_h \to g}$ in the decomposition process. In contrast to the sensitivities between z_h and z_f where changing one root z_f affects only N roots of $h(x)$, a perturbation of $g(x)$ results in perturbation of all roots of $h(x)$. When the roots of $f(x)$ are all fixed, we have the following relationship between perturbations of $g(x)$ and the roots of $h(x)$:

$$g(z_h(k) + \Delta z_h(k)) + \Delta g(z_h(k) + \Delta z_h(k)) = z_f(i), \quad k \in A_i.$$

Applying linear approximation with small perturbations,

$$\begin{aligned} g(z_h(k) + \Delta z_h(k)) &\approx z_f(i) + g'(z_h(k)) \cdot \Delta z_h(k), \\ \Delta g(z_h(k) + \Delta z_h(k)) &\approx \Delta g(z_h(k)), \end{aligned}$$

the perturbation of $z_h(k)$ is derived as

$$\Delta z_h(k) \approx -\frac{\Delta g(z_h(k))}{g'(z_h(k))}.$$

The perturbations of all roots of $h(x)$ can be expressed in matrix form as:

$$\mathbf{\Delta z_h} \approx -\mathbf{Q W \Delta g} \tag{4.24}$$

where matrices $\mathbf{Q}$ and $\mathbf{W}$ are

$$\mathbf{Q} = \operatorname{diag}\left(\frac{1}{g'(z_h(1))}, \frac{1}{g'(z_h(2))}, \cdots, \frac{1}{g'(z_h(MN))}\right), \tag{4.25}$$

$$\mathbf{W} = \begin{bmatrix} z_h^N(1) & z_h^{N-1}(1) & \cdots & z_h(1) & 1 \\ z_h^N(2) & z_h^{N-1}(2) & \cdots & z_h(2) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ z_h^N(MN) & z_h^{N-1}(MN) & \cdots & z_h(MN) & 1 \end{bmatrix}. \tag{4.26}$$

Consequently, we can derive the sensitivities $S_{g \to z_h}$ in the composition process and $S_{z_h \to g}$ in the decomposition process:

$$S_{g \to z_h} = \max_{\|\mathbf{\Delta g}\|_2 = \kappa} \left(\frac{R_{\mathbf{\Delta z_h}}}{R_{\mathbf{\Delta g}}}\right) = \sigma^2_{\mathbf{QW},\max} \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{z_h}\|_2^2}, \tag{4.27}$$

$$S_{z_h \to g} = \max_{\|\mathbf{\Delta g}\|_2 = \kappa} \left(\frac{R_{\mathbf{\Delta g}}}{R_{\mathbf{\Delta z_h}}}\right) = \left(\sigma^2_{\mathbf{QW},\min} \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{z_h}\|_2^2}\right)^{-1}, \tag{4.28}$$

where $\sigma_{\mathbf{QW},\max}$ and $\sigma_{\mathbf{QW},\min}$ are the maximum and minimum singular values of the matrix $\mathbf{Q} \cdot \mathbf{W}$, respectively; the perturbation $\mathbf{\Delta g}$ has a sufficiently small magnitude of κ.

4.2 Sensitivities of Equivalent Compositions with First-Degree Polynomials

As mentioned in Section 2.1, a composed polynomial may have equivalent compositions when first-degree polynomials are used. Specifically, if we denote

$$\hat{f}(x) = \left(f \circ q^{-1}\right)(x), \tag{4.29}$$

$$\hat{g}(x) = \left(q \circ g\right)(x), \tag{4.30}$$

where $q(x) = q_1 x + q_0$ is a first-degree polynomial, then we have

$$h(x) = (f \circ g)(x) = \left(\hat{f} \circ \hat{g}\right)(x). \tag{4.31}$$

However, these equivalent compositions may have different sensitivities. In this section, we show the effects of equivalent compositions on sensitivities, and we propose a practical rule to choose the parameters of the first-degree polynomial to get to a point that is near the minimum of certain sensitivities.

First, we analyze the sensitivities between the coefficients of $\hat{f}(x)$ and $h(x)$. Applying (4.1) to the equivalent composition (4.31), we have

$$\mathbf{h} = \hat{\mathbf{G}}\hat{\mathbf{f}}, \tag{4.32}$$

where the matrix $\hat{\mathbf{G}}$ has columns as the self-convolutions of the new polynomial $\hat{g}(x)$. The self-convolution $(\hat{g}(x))^n$ can be regarded as a composition

$$(\hat{g}(x))^n = (q_1 g(x) + q_0)^n = \left(s_n \circ g\right)(x), \tag{4.33}$$

where the polynomial $s_n(x) = (q_1 x + q_0)^n$. Connecting (4.33) with the matrix formulation in (4.1), we have

$$[(\hat{g}(x))^n] = \mathbf{G}\mathbf{s_n},$$

where $[(\hat{g}(x))^n]$ is the corresponding vector of the polynomial $(\hat{g}(x))^n$. As a result, we

can establish the relationship between the self-convolution matrices $\mathbf{G}$ and $\hat{\mathbf{G}}$,

$$\hat{\mathbf{G}} = \left[(\hat{g}(x))^M,\ (\hat{g}(x))^{M-1}, \ldots, (\hat{g}(x))^0\right] = \mathbf{G}\left[\mathbf{s}_M,\ \mathbf{s}_{M-1}, \ldots, \mathbf{s}_0\right] = \mathbf{GA}, \tag{4.34}$$

where the matrix $\mathbf{A}$ is the self-convolution matrix of the first-degree polynomial $q(x) = q_1 x + q_0$. Combining (4.1), (4.32), and (4.34), we can know

$$\hat{\mathbf{f}} = \mathbf{A}^{-1}\mathbf{f}. \tag{4.35}$$

Consequently, the composition sensitivity $S_{\hat{f}\to h}$ becomes

$$S_{\hat{f}\to h} = \max_{\|\boldsymbol{\Delta}\hat{\mathbf{f}}\|_2=\kappa}\left(\frac{R_{\boldsymbol{\Delta}\mathbf{h}}}{R_{\boldsymbol{\Delta}\hat{\mathbf{f}}}}\right) = \frac{\|\mathbf{A}^{-1}\mathbf{f}\|_2^2}{\|\mathbf{h}\|_2^2}\cdot\max_{\|\boldsymbol{\Delta}\hat{\mathbf{f}}\|_2=\kappa}\left(\frac{\|\mathbf{GA}\boldsymbol{\Delta}\hat{\mathbf{f}}\|_2^2}{\|\boldsymbol{\Delta}\hat{\mathbf{f}}\|_2^2}\right) = \frac{\|\mathbf{A}^{-1}\mathbf{f}\|_2^2}{\|\mathbf{h}\|_2^2}\cdot\sigma^2_{\hat{\mathbf{G}},\max}, \tag{4.36}$$

and the decomposition sensitivity $S_{h\to\hat{f}}$ becomes

$$S_{h\to\hat{f}} = \max_{\|\boldsymbol{\Delta}\hat{\mathbf{f}}\|_2=\kappa}\left(\frac{R_{\boldsymbol{\Delta}\hat{\mathbf{f}}}}{R_{\boldsymbol{\Delta}\mathbf{h}}}\right) = \left(\frac{\|\mathbf{A}^{-1}\mathbf{f}\|_2^2}{\|\mathbf{h}\|_2^2}\cdot\min_{\|\boldsymbol{\Delta}\hat{\mathbf{f}}\|_2=\kappa}\left(\frac{\|\mathbf{GA}\boldsymbol{\Delta}\hat{\mathbf{f}}\|_2^2}{\|\boldsymbol{\Delta}\hat{\mathbf{f}}\|_2^2}\right)\right)^{-1} = \left(\frac{\|\mathbf{A}^{-1}\mathbf{f}\|_2^2}{\|\mathbf{h}\|_2^2}\cdot\sigma^2_{\hat{\mathbf{G}},\min}\right)^{-1}, \tag{4.37}$$

where $\sigma_{\hat{\mathbf{G}},\max}$ and $\sigma_{\hat{\mathbf{G}},\min}$ are the maximum and minimum singular value of the matrix $\hat{\mathbf{G}}$, respectively.

Utilizing (4.36) and (4.37), we explore how to choose an appropriate first-degree polynomial to efficiently enhance the robustness between $\hat{f}(x)$ and $h(x)$. The optimal parameter choice for q_1 and q_0 to minimize $S_{\hat{f}\to h}$ or $S_{h\to\hat{f}}$ is not obvious, since the sensitivities have complicated dependence on both $f(x)$ and $g(x)$. However, combining (4.36) and (4.37), we note that

$$S_{\hat{f}\to h}\cdot S_{h\to\hat{f}} = \mathrm{cond}(\hat{\mathbf{G}})^2, \tag{4.38}$$

i.e., the product of the sensitivities results in the squared condition number of the matrix $\hat{\mathbf{G}}$, which is independent of $f(x)$ as long as its degree is M. If we want both sensitivities to be small, then (4.38) implies the condition number $\mathrm{cond}(\hat{\mathbf{G}})$ has to

be small. In addition, as shown in (4.5) and (4.7), the condition number cond($\hat{\mathbf{G}}$) is an upper bound for both sensitivities $S_{\hat{f}\to h}$ and $S_{h\to\hat{f}}$, so a small condition number ensures that these sensitivities are simultaneously small.

To increase robustness, we are interested in the optimal parameters (q_1^*, q_0^*) that minimize cond($\hat{\mathbf{G}}$), for a given polynomial $g(x)$ and a given degree M.[3] It is still not obvious how to obtain the optimal parameters or to prove the convexity of the condition number cond($\hat{\mathbf{G}}$) with respect to q_1 and q_0; however, we have the following parameter selection rule that may approach the minimum value of cond($\hat{\mathbf{G}}$).

Approximate Parameter Selection Rule for $q(x)$: *Given a polynomial $g(x)$ and a degree M, the first-degree polynomial $\tilde{q}(x) = \tilde{q}_1 x + \tilde{q}_0 = \tilde{q}_1(x + \tilde{q}_r)$ with parameters*

$$\tilde{q}_r = \arg\min_{q_r} \|(g(x) + q_r)^M\|_2^2, \tag{4.39}$$

$$\tilde{q}_1 = \left(\|(g(x) + \tilde{q}_r)^M\|_2^2\right)^{-\frac{1}{2M}}, \tag{4.40}$$

$$\tilde{q}_0 = \tilde{q}_1 \cdot \tilde{q}_r, \tag{4.41}$$

results in a corresponding matrix $\hat{\mathbf{G}}$ whose condition number is near the minimum among all first-degree polynomials.

The development of the approximate rule is in Appendix C. The function $\|(g(x)+q_r)^M\|_2^2$ in (4.39) is convex towards q_r, so the parameter $\tilde{q}_r$ can be computed efficiently, and then $\tilde{q}_1$ and $\tilde{q}_0$ are obtained.

The approximate rule can be intuitively explained as follows. If we consider $\hat{\mathbf{G}}$ as a geometric mapping from the vector spaces of $\hat{\mathbf{f}}$ to that of $\mathbf{h}$, then the condition number cond($\hat{\mathbf{G}}$) is the ratio between the lengths of the longest and the shortest vectors that are the images of unit vectors. In particular, each unit vector on a coordinate axis is mapped to a corresponding column of the matrix $\hat{\mathbf{G}}$. Thus, if the columns of the matrix $\hat{\mathbf{G}}$ vary significantly in energy, then the condition number is high. In addition, if two columns of the matrix $\hat{\mathbf{G}}$ are relatively very close in space, then their difference is a vector with low magnitude, which also leads to a high condition number. Thus,

[3] Although the matrix $\hat{\mathbf{G}}$ is independent of coefficients of $f(x)$, the degree of $f(x)$ influences the size of $\hat{\mathbf{G}}$.

in order to have a small condition number, the columns of $\hat{\mathbf{G}}$ should be relatively similar in energy, and they should not be highly correlated. The columns of $\hat{\mathbf{G}}$ are the coefficients of self-convolutions of $g(x)$; the rule above may keep relatively similar energy among the self-convolutions and may avoid high correlation among them. As a result, the rule above may achieve an approximately minimum condition number of the associated matrix $\hat{\mathbf{G}}$.

The heuristic rule above cannot guarantee to obtain the minimum condition number $\text{cond}(\hat{\mathbf{G}})$ among all first-degree polynomials. However, empirically the condition number with the rule above may achieve near the actual minimum.

Next, we derive the sensitivities $S_{\hat{g}\to h}$ and $S_{h\to\hat{g}}$. After composing the first-degree polynomial, the polynomial $d(x)$ in (4.9) becomes

$$\hat{d}(x) = (\hat{f}' \circ \hat{g})(x) = \left((f \circ q^{-1})' \circ q \circ g\right)(x) = \frac{1}{q_1}(f' \circ g)(x) = \frac{1}{q_1}d(x),$$

where in the third step, we use the fact that $(f \circ q^{-1})'(x) = ((q^{-1})')(x) \cdot (f' \circ q^{-1})(x) = \frac{1}{q_1}(f' \circ q^{-1})(x)$. Thus, the sensitivities become

$$S_{\hat{g}\to h} = \frac{\|q_1\mathbf{g} + q_0\mathbf{e}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot \max_{\|\boldsymbol{\Delta}\hat{\mathbf{g}}\|_2=\kappa} \left(\frac{\|\frac{1}{q_1}\mathbf{D}\boldsymbol{\Delta}\hat{\mathbf{g}}\|_2^2}{\|\boldsymbol{\Delta}\hat{\mathbf{g}}\|_2^2} \right) = S_{g\to h} \cdot \frac{\|\mathbf{g} + \frac{q_0}{q_1}\mathbf{e}\|_2^2}{\|\mathbf{g}\|_2^2}, \tag{4.42}$$

$$S_{h\to\hat{g}} = \left(\frac{\|q_1\mathbf{g} + q_0\mathbf{e}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot \min_{\|\boldsymbol{\Delta}\hat{\mathbf{g}}\|_2=\kappa} \left(\frac{\|\frac{1}{q_1}\mathbf{D}\boldsymbol{\Delta}\hat{\mathbf{g}}\|_2^2}{\|\boldsymbol{\Delta}\hat{\mathbf{g}}\|_2^2} \right) \right)^{-1} = S_{h\to g} \cdot \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{g} + \frac{q_0}{q_1}\mathbf{e}\|_2^2}, \tag{4.43}$$

where the vector $\mathbf{e} = [0, 0, \ldots, 0, 1]^{\mathrm{T}}$ corresponds to the constant term in the polynomial, and the perturbation $\boldsymbol{\Delta}\hat{\mathbf{g}}$ has a sufficiently small magnitude of κ.

With respect to the sensitivities between $\hat{g}(x)$ and $h(x)$, the parameters of the first-degree polynomial should depend on the application, especially due to the following tradeoff. Combining (4.42) and (4.43), we notice that the product of $S_{\hat{g}\to h}$ and $S_{h\to\hat{g}}$ remains a constant regardless of the choice of the first-degree polynomial:

$$S_{\hat{g}\to h} \cdot S_{h\to\hat{g}} = \frac{\sigma_{\mathbf{D},\max}^2}{\sigma_{\mathbf{D},\min}^2}.$$

Consequently, these two sensitivities cannot be reduced simultaneously by the same first-degree polynomial; a decrease in one sensitivity always results in an increase in the other. Furthermore, we observe that only the ratio $q_r \triangleq \frac{q_0}{q_1}$ affects the sensitivities between $\hat{g}(x)$ and $h(x)$ but not the individual q_0 or q_1; the sensitivity $S_{\hat{g}\to h}$ decreases first and then increases with q_r, and the ratio to minimize $S_{\hat{g}\to h}$ is $q_r = -b_0$ where b_0 is the constant term in $g(x)$. In addition, for a fixed ratio $\frac{q_0}{q_1}$ that achieves good sensitivities between $\hat{g}(x)$ and $h(x)$, there is still freedom to adjust the values of q_0 (or q_1) to decrease the sensitivities between $\hat{f}(x)$ and $h(x)$.

Third, we consider the effects of equivalent composition on sensitivities of the roots. After the composition with the first-degree polynomial in (4.29), the roots z_h remain the same, but the roots of $\hat{f}(x)$ become

$$z_{\hat{f}} = q(z_f) = q_1 z_f + q_0,$$

where z_f are the roots of the original polynomial $f(x)$. In a derivation similar to the above, we finally obtain the sensitivities of the roots for the equivalent compositions:

$$S_{z_{\hat{f}}\to z_h} = S_{z_f\to z_h} \cdot \frac{\|\mathbf{z_f} + \frac{q_0}{q_1}\|_2^2}{\|\mathbf{z_f}\|_2^2}, \tag{4.44}$$

$$S_{z_h\to z_{\hat{f}}} = S_{z_h\to z_f} \cdot \frac{\|\mathbf{z_f}\|_2^2}{\|\mathbf{z_f} + \frac{q_0}{q_1}\|_2^2}, \tag{4.45}$$

$$S_{\hat{g}\to z_h} = S_{g\to z_h} \cdot \frac{\|\mathbf{g} + \frac{q_0}{q_1}\mathbf{e}\|_2^2}{\|\mathbf{g}\|_2^2}, \tag{4.46}$$

$$S_{z_h\to\hat{g}} = S_{z_h\to g} \cdot \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{g} + \frac{q_0}{q_1}\mathbf{e}\|_2^2}, \tag{4.47}$$

where $S_{z_f\to z_h}$, $S_{z_h\to z_f}$, $S_{g\to z_h}$, and $S_{z_h\to g}$ are in (4.22), (4.23), (4.27), and (4.28), respectively.

The same as the sensitivities between $\hat{g}(x)$ and $h(x)$, the sensitivities of the roots have the following two properties. First, the product of two corresponding sensitivities in the composition and decomposition processes remains a constant for all equivalent compositions, so it is impossible to decrease both of them simultaneously; second, the sensitivities of the roots are affected only by the ratio $q_r = \frac{q_0}{q_1}$ rather than the

individual values of q_1 and q_0. Consequently, the optimal choice of parameters has a tradeoff and depends on the application. In addition, after the determination of the ratio $\frac{q_0}{q_1}$ that has acceptable sensitivities of the roots, it is possible to further improve $S_{\hat{f}\to h}$ and $S_{h\to\hat{f}}$ by adjusting q_0 (or q_1). As for the tendency, we may see that both $S_{z_{\hat{f}}\to z_h}$ and $S_{\hat{g}\to z_h}$ decreases first and then increases with q_r; the ratio to minimize $S_{\hat{g}\to z_h}$ is $q_r = -b_0$, which is the same as the ratio to minimize $S_{\hat{g}\to h}$, but the ratio q_r to minimize $S_{z_{\hat{f}}\to z_h}$ is usually different.

4.3 Simulation Results

In this section, the results of simulations are presented to evaluate sensitivity in different contexts. Specifically, simulations are shown to evaluate each sensitivity with polynomials of different degrees, to compare the sensitivities of the coefficients and those of the roots, and to demonstrate the effectiveness of decreasing sensitivities with equivalent compositions.

The data in the simulation are generated with the following parameters: The degrees of both polynomial $f(x)$ and $g(x)$ vary from 2 to 15. For each degree, we create 100 random samples of $f(x)$ and $g(x)$, respectively. For each sample polynomial, the coefficients are first generated from i.i.d. standard normal distribution, and then the polynomial is normalized to have unit energy.

4.3.1 Evaluation of the Sensitivities

At each degree of $f(x)$ and $g(x)$, we compose each of the 100 samples of $f(x)$ and each of the 100 samples of $g(x)$, and then evaluate all the sensitivities for all the $10,000$ compositions. The results are shown in Fig. 4-1 to Fig. 4-8; each figure shows a certain sensitivity. In these figures, the continuous curve indicates the median of the sensitivity among the $10,000$ compositions at that degree, and each vertical bar shows the maximum and the minimum of the sensitivity obtained at that degree in the simulation.

The first two figures show the sensitivities between the coefficients of $f(x)$ and

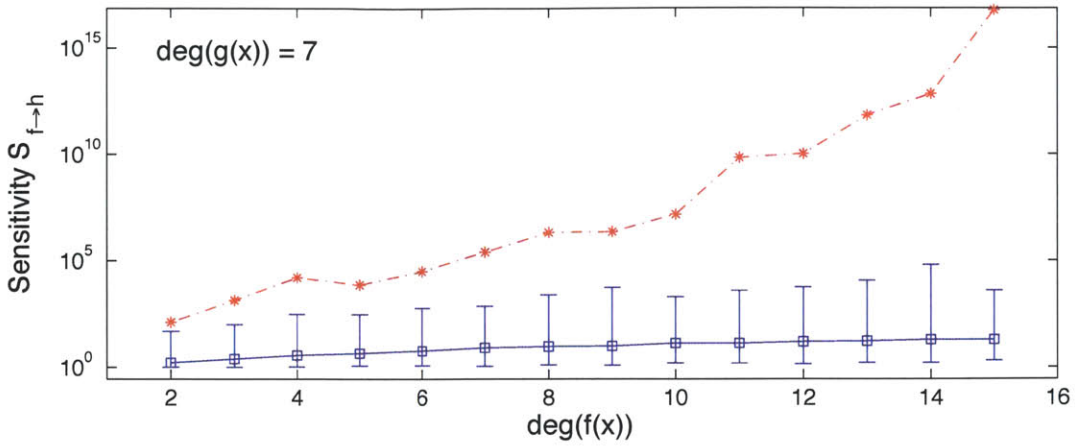


Figure 4-1: Coefficient Sensitivity from $f(x)$ to $h(x)$.

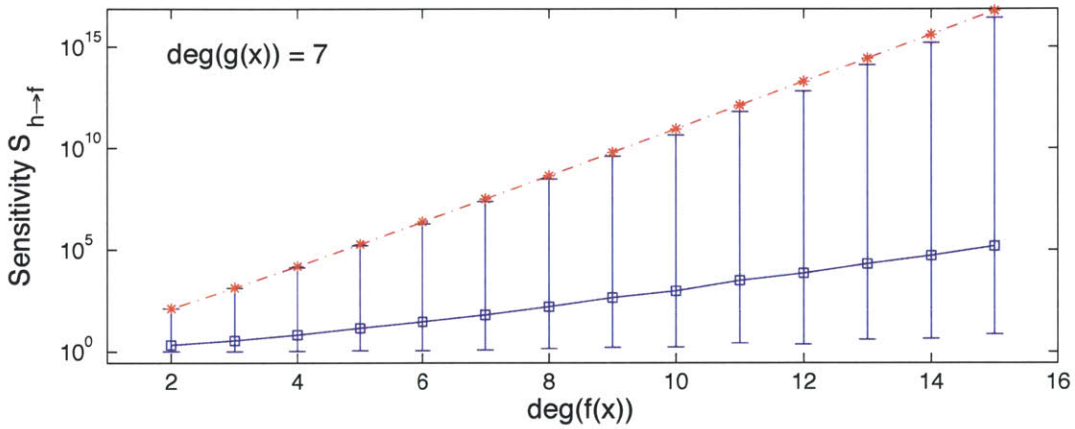


Figure 4-2: Coefficient Sensitivity from $h(x)$ to $f(x)$.

$h(x)$: the composition sensitivity $S_{f\to h}$ in (3.1) and the decomposition sensitivity $S_{h\to f}$ in (3.5) are shown in Fig. 4-1 and 4-2, respectively. The degree of $g(x)$ is fixed to 7, and the degree of $f(x)$ varies from 2 to 15 as indexed by the x-axis. In each figure, the continuous curve is the median of the sensitivity, and the dashed curve is the upper bound in (4.5) or (4.7) evaluated with the instance of $g(x)$ that achieves the maximum sensitivity at each degree. The simulation results demonstrate that the sensitivities satisfy the theoretical bounds in (4.5) and (4.7). We notice that there is a considerably large gap between the upper bound for the composition sensitivity $S_{f\to h}$ and its empirical maximum obtained in the simulation, which indicates the upper bound in (4.5) is tight in theory but possibly conservative in practice. As for

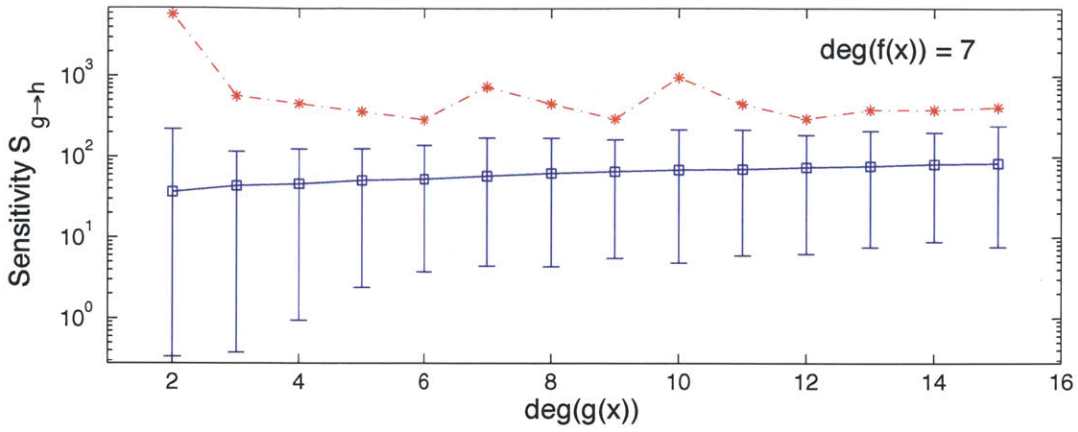


Figure 4-3: Coefficient Sensitivity from $g(x)$ to $h(x)$.

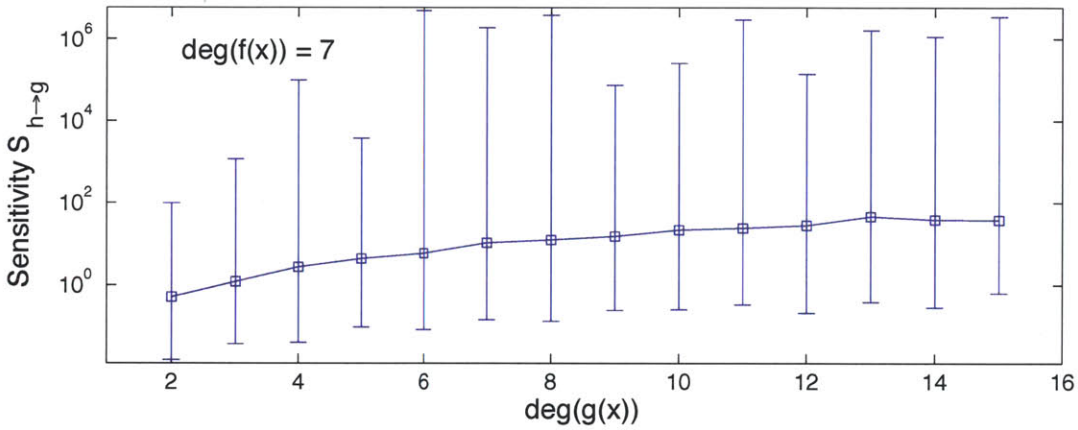


Figure 4-4: Coefficient Sensitivity from $h(x)$ to $g(x)$.

the tendency of the sensitivities, both sensitivities increase with the increase of the degree of $f(x)$. In addition, the decomposition sensitivity $S_{h\to f}$ is significantly larger than the composition sensitivity $S_{f\to h}$ in the simulation, which indicates the composition process is likely to be more robust than the decomposition process. Although the sensitivities are large in Fig. 4-1 and 4-2, however, as will be shown in the Section 4.3.3, the sensitivities between the coefficients of $f(x)$ to $h(x)$ can be decreased simultaneously using equivalent compositions, and the robustness can be improved significantly.

The next two figures correspond to the coefficient sensitivities between $g(x)$ and $h(x)$: the composition sensitivity $S_{g\to h}$ in (3.3) and the decomposition sensitivity

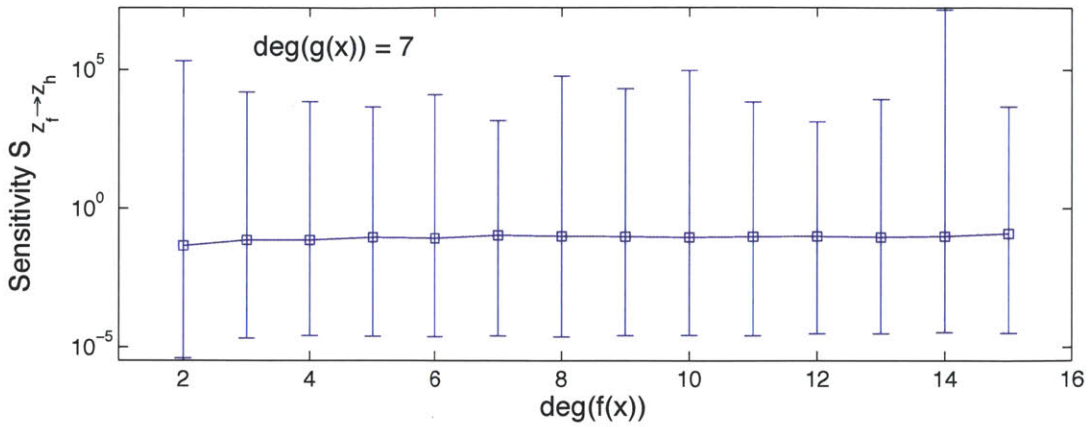


Figure 4-5: Root Sensitivity from z_f to z_h.

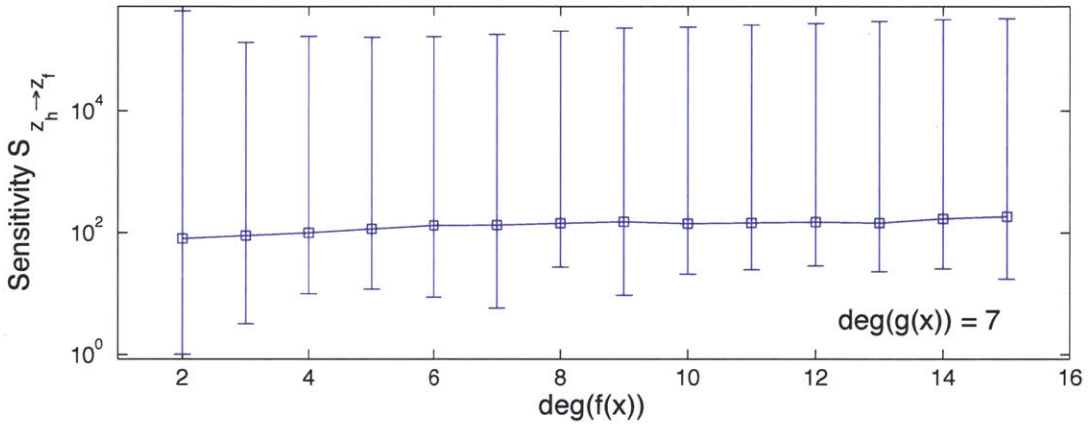


Figure 4-6: Root Sensitivity from z_h to z_f.

$S_{h\to g}$ in (3.6) are shown in Fig. 4-3 and 4-4, respectively. The degree of $f(x)$ is fixed to 7, and the degree of $g(x)$ varies from 2 to 15. The dashed curve in Fig. 4-3 is the upper bound in (4.13), where $g(x)$ is chosen as the instance that achieves the maximum sensitivity at each degree. The simulation results show that the upper bound is satisfied and empirically tight. Furthermore, the decomposition sensitivity $S_{h\to g}$ is generally larger and increases faster with the degree of $g(x)$ than the composition sensitivity $S_{g\to h}$. This indicates the composition is more robust than the decomposition for $g(x)$.

The subsequent two figures show the root sensitivities between $f(x)$ and $h(x)$: Fig. 4-5 shows the composition sensitivity $S_{z_f\to z_h}$, and Fig. 4-6 shows the decomposition

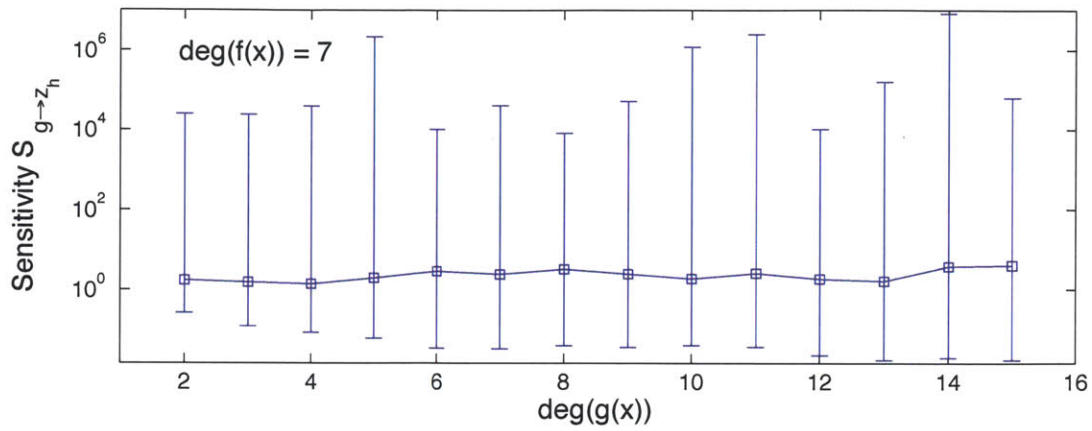


Figure 4-7: Root Sensitivity from $g(x)$ to z_h.

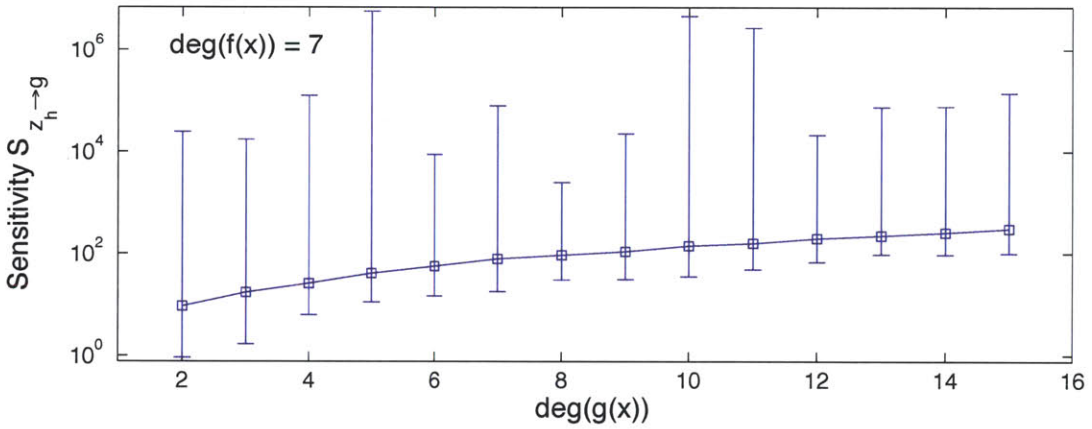


Figure 4-8: Root Sensitivity from z_h to $g(x)$.

sensitivity $S_{z_h \to z_f}$. The degree of $f(x)$ varies from 2 to 15 while the degree of $g(x)$ is fixed at 7. In contrast to the coefficient sensitivities between $f(x)$ and $h(x)$ that increase fast with the degree of $f(x)$, the median root sensitivities between z_f and z_h have only little increase. This phenomenon indicates potential benefit to use the roots rather than the coefficients for better robustness in polynomial composition and decomposition where $f(x)$ has a high degree. The root sensitivities between $f(x)$ and $h(x)$ is generally more homogeneous and less dependent on the degree of $f(x)$ than the coefficient sensitivities. We may see this difference from the following example [4]:

[4]Although $h(x)$ has multi-roots in this example, however, as long as $g(x)$ has only single roots, then the multi-roots do not belong to the same group, so the sensitivities $S_{z_f \to z_h}$ and $S_{z_h \to z_f}$ are still finite.

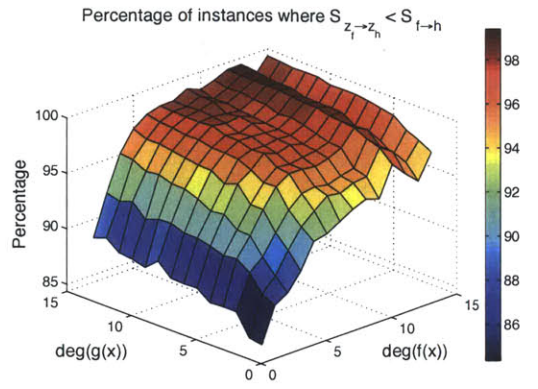
if $f(x) = x^M$, then we can verify the root sensitivities $S_{z_f \to z_h}$ and $S_{z_h \to z_f}$ are the same value regardless of the degree M, since the $\|\mathbf{z_f}\|_2^2$ and $\|\mathbf{z_h}\|_2^2$ are both proportional to M^2; however, in the coefficient sensitivities, the size of the matrix $\mathbf{G}$ depends on M, so the singular values of $\mathbf{G}$ may be significantly affected when M increases, which may result in an increase in the coefficient sensitivities.

The last two figures correspond to the root sensitivities between the coefficients of $g(x)$ and the roots z_h: Fig. 4-7 shows the composition sensitivity $S_{g \to z_h}$, and Fig. 4-8 shows the decomposition sensitivity $S_{z_h \to g}$. The degree of $g(x)$ varies from 2 to 15 while the degree of $f(x)$ is fixed at 7. The decomposition sensitivity $S_{z_h \to g}$ increases with the degree of $g(x)$, while there does not seem to be such an obviously increasing tendency for the composition sensitivity $S_{g \to z_h}$.

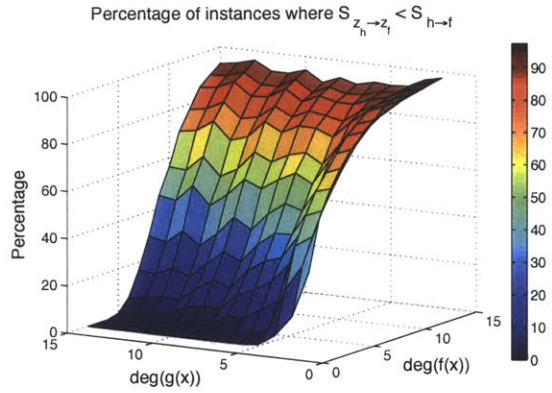
4.3.2 Comparisons of the Sensitivities

This section shows simulation results comparing the coefficient sensitivities with the root sensitivities. We perform comparison on sensitivities in four pairs, namely $S_{f \to h}$ vs $S_{z_f \to z_h}$, $S_{h \to f}$ vs $S_{z_h \to z_f}$, $S_{g \to h}$ vs $S_{g \to z_h}$, and $S_{h \to g}$ vs $S_{z_h \to g}$; each pair contains a coefficient sensitivity and a root sensitivity corresponding to the same polynomials involved. At each degree of $f(x)$ and $g(x)$, we compare the sensitivities within each pair for each of the $10,000$ composition instances, then we record the percentage of instances where the root sensitivity is smaller than the coefficient sensitivity. The results for the four pairs of sensitivities are plotted in Fig. 4-9.

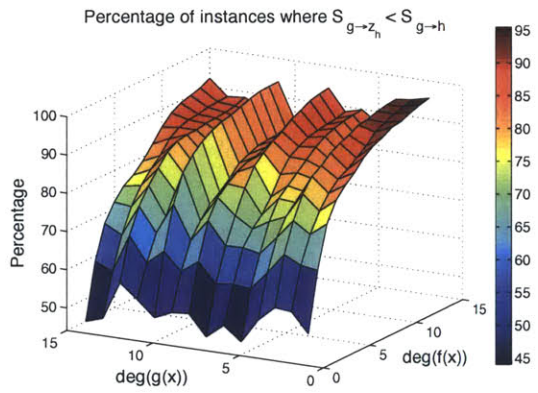
The results seem to support that composition and decomposition using the root triplet $(z_f,\ g,\ z_h)$ are likely to be more robust than using the coefficient triplet $(f,\ g,\ h)$, when the degrees of polynomials are high. As the degrees of $f(x)$ and $g(x)$ increase, there are more instances in our simulation where the root sensitivity is smaller than the corresponding coefficient sensitivity. As we mentioned in Section 4.3.1, between the polynomials $f(x)$ and $h(x)$, since the relationship of the coefficients in (4.1) involves self-convolution of the polynomial $g(x)$, a perturbation may be magnified; however, the root sensitivities between z_f and z_h seem to be more homogeneous. However, we cannot conclude for every polynomial that the root triplet



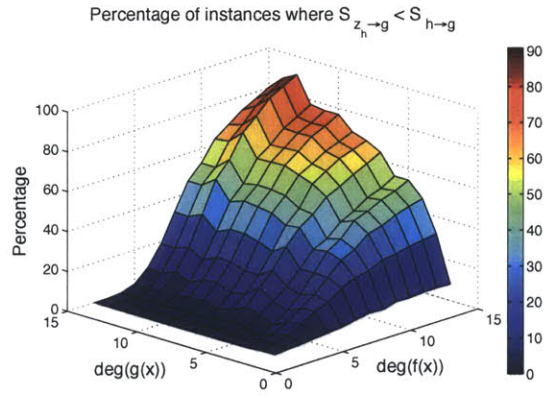
(a): Sensitivities from $f(x)$ to $h(x)$, i.e., $S_{f\to h}$ vs $S_{z_f\to z_h}$



(b): Sensitivities from $h(x)$ to $f(x)$, i.e., $S_{h\to f}$ vs $S_{z_h\to z_f}$



(c): Sensitivities from $g(x)$ to $h(x)$, i.e., $S_{g\to h}$ vs $S_{g\to z_h}$



(d): Sensitivities from $h(x)$ to $g(x)$, i.e., $S_{h\to g}$ vs $S_{z_h\to g}$

Figure 4-9: Comparison between Corresponding Coefficient Sensitivities and Root Sensitivities.

has lower sensitivities than the coefficient triplet, since certain multi-roots of $h(x)$ result in infinite root sensitivities, while all coefficient sensitivities are finite.

4.3.3 Sensitivities of Equivalent Compositions

This section presents simulation results to illustrate the effects of equivalent compositions on the sensitivities. In particular, we validate the effectiveness of equivalent

composition in reducing the sensitivities $S_{\hat{f}\to h}$ and $S_{h\to\hat{f}}$, and we show the performance of the approximate rules (4.39)-(4.41) of choosing the first-degree polynomial.

In Fig. 4-10 - Fig. 4-14, we show the dependence of the condition number cond($\hat{\mathbf{G}}$) and all the sensitivities on the parameters of the first-degree polynomial. The degree of $g(x)$ is 7; polynomial $g(x)$ is chosen as the instance that achieves the maximum condition number of $\mathbf{G}$ among the 100 random samples (without composing with first-degree polynomials), which are generated in previous simulations in Section 4.3.1. The degree of $f(x)$ is chosen as $M = 15$; $f(x)$ is the polynomial that has the highest sensitivity $S_{f\to h}$ with the $g(x)$ above (without composing with first-degree polynomials) among the 100 randomly generated instances in previous simulations. In the previous section, we derive that the sensitivities $S_{\hat{g}\to h}$, $S_{h\to\hat{g}}$, $S_{z_{\hat{f}}\to z_h}$, $S_{z_h\to z_{\hat{f}}}$, $S_{\hat{g}\to z_h}$, and $S_{z_h\to\hat{g}}$ depend only on the ratio $q_r = \frac{q_0}{q_1}$ of the first-degree polynomial. Thus, the x-axis is q_r in Fig. 4-12 - Fig. 4-14 for these sensitivities. In contrast, cond($\hat{\mathbf{G}}$), $S_{\hat{f}\to h}$, and $S_{h\to\hat{f}}$ depend on both q_1 and q_0. For consistency with the other sensitivities, we plot cond($\hat{\mathbf{G}}$), $S_{\hat{f}\to h}$, and $S_{h\to\hat{f}}$ with respect to q_1 and $q_r = \frac{q_0}{q_1}$. The range of q_1 and q_r are $[0.9, 1.9]$ and $[-1.4, -0.4]$, respectively.

Fig. 4-10 indicates that there is an optimal $q^*(x)$ that achieves the minimum

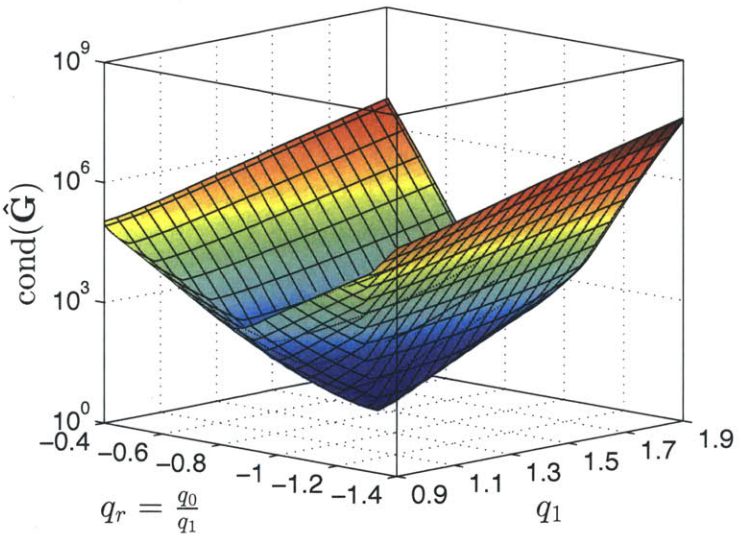
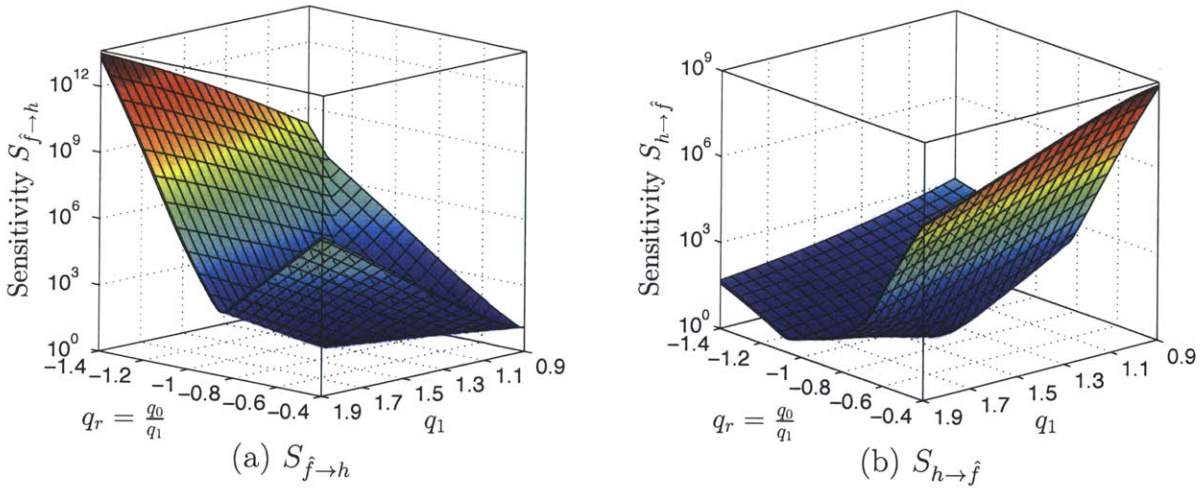


Figure 4-10: The Condition Number cond($\hat{\mathbf{G}}$) with Different q_1 and q_r, where $q_r = \frac{q_0}{q_1}$.



(a) $S_{\hat{f}\to h}$ (b) $S_{h\to\hat{f}}$

Figure 4-11: The Sensitivities $S_{\hat{f}\to h}$ and $S_{h\to\hat{f}}$ with Different q_1 and q_r.

condition number cond($\hat{\mathbf{G}}$); in this example, the optimal first-order polynomial has parameters as $q_r^* = -0.8563$, $q_1^* = 1.3973$, and $q_0^* = -1.1965$. As the parameters q_1 and q_r deviate from the optimal point, the condition number increases rapidly. In Fig. 4-11, the sensitivities $S_{\hat{f}\to h}$ and $S_{h\to\hat{f}}$ also have their respective minimum points, which are near $q^*(x)$. Thus, these figures show that we can choose a proper $q(x)$ to have low sensitivities in both the composition and the decomposition operations, between the coefficients of $f(x)$ and $h(x)$.

In contrast, in each pair of sensitivities in Fig. 4-12 - Fig. 4-14, an decrease in one sensitivity results in an increase in the other; this phenomenon is consistent with our derivation in Section 4.2: the product of the two sensitivities remains a constant regardless of q_r. In addition, as discussed in Section 4.2, the composition sensitivities decrease at first and then increase (the range of q_r does not include the minimum point for $S_{z_{\hat{f}}\to z_h}$ in Fig. 4-13); the sensitivities $S_{\hat{g}\to h}$ and $S_{\hat{g}\to z_h}$ share the same optimal value for q_r.

Fig. 4-15 shows the condition number with polynomials of different degrees, and tests the performance of the approximate rules in (4.39)-(4.41). In Fig. 4-15 (a) and (b), the polynomial $g(x)$ has degree 7, and we use the 100 randomly generated instances of $g(x)$ in previous sections; the degree[5] of $f(x)$ varies from 2 to 15. The three

[5]Although the matrix $\hat{\mathbf{G}}$ is independent of coefficients of $f(x)$, the degree of $f(x)$ influences the size of $\hat{\mathbf{G}}$.

curves in 4-15 (a) are obtained as follows: for each degree M, we pick the polynomial $g(x)$ that achieves the maximum original condition number $\text{cond}(\mathbf{G})$ within the 100 samples (without composing with first-degree polynomials); with the instance of $g(x)$ that we pick, the dash-dot line denotes the original condition number, the dotted line denotes the minimum condition number by composing with the optimal first-degree polynomial $q^*(x)$, and the continuous line denotes the condition number by composing with the first-degree polynomial $\tilde{q}(x)$ proposed by the approximate rules (4.39)-(4.41). For the instance of $g(x)$ at each degree, the optimal first-degree polynomial $q^*(x)$ to minimize $\text{cond}(\hat{\mathbf{G}})$ is obtained by brute force search. To show the performance of the approximate rule clearly, we magnify Fig. 4-15 (a) into Fig. 4-15 (b), but without the curve of the original condition number. The above description also applies to Fig. 4-15 (c) and (d), but the polynomial $g(x)$ has degree 15 rather than 7 for these figures.

The figures demonstrate that the equivalent composition efficiently reduces the condition number $\hat{\mathbf{G}}$, and the approximate rules are considerably effective to achieve a nearly minimum condition number. In Fig. 4-15 (a) and (c), compared with the rapid growth of the original condition number with the degree M, the corresponding minimum condition number has only considerably small increase. At each degree M in Fig. 4-15 (b) and (d), the condition number of equivalent composition by composing with $\tilde{q}(x)$ that is proposed by the approximate rules is considerably near the actual minimum value. Thus, the approximate rules (4.39)-(4.41) are considerably effective in practice.

As shown in (4.5) and (4.7), the squared condition number of $\hat{\mathbf{G}}$ is an upper bound for both sensitivities $S_{\hat{f}\to h}$ and $S_{h\to\hat{f}}$. In our simulation, since the minimum condition number can be reduced to less than 2.5 with proper equivalent composition, these two sensitivities also become considerably low, which indicates an improvement of robustness.

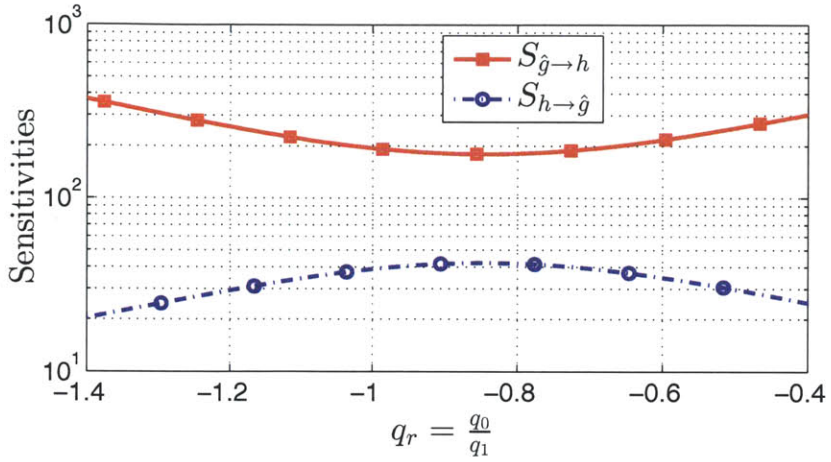


Figure 4-12: The Sensitivities $S_{\hat{g}\to h}$ and $S_{h\to\hat{g}}$ with Different q_r.

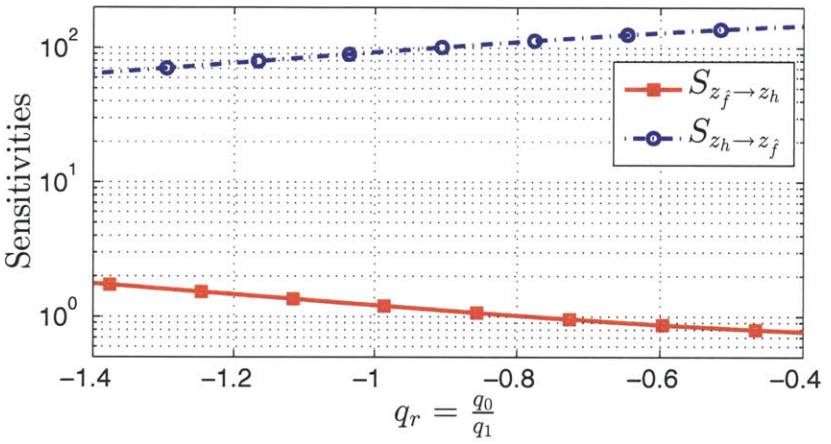


Figure 4-13: The Sensitivities $S_{z_{\hat{f}}\to z_h}$ and $S_{z_h\to z_{\hat{f}}}$ with Different q_r.

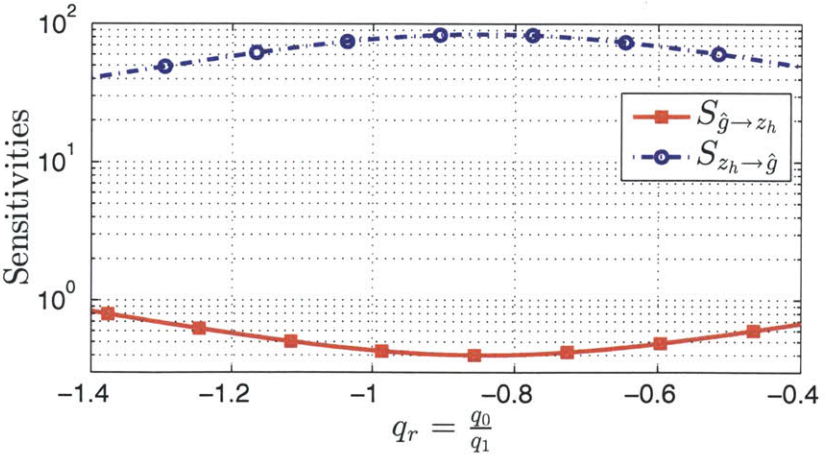
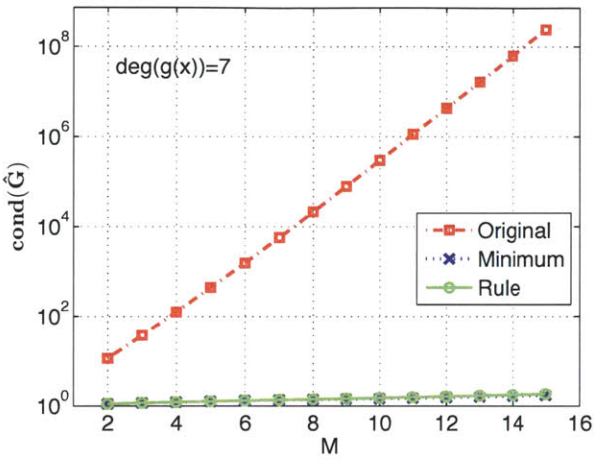
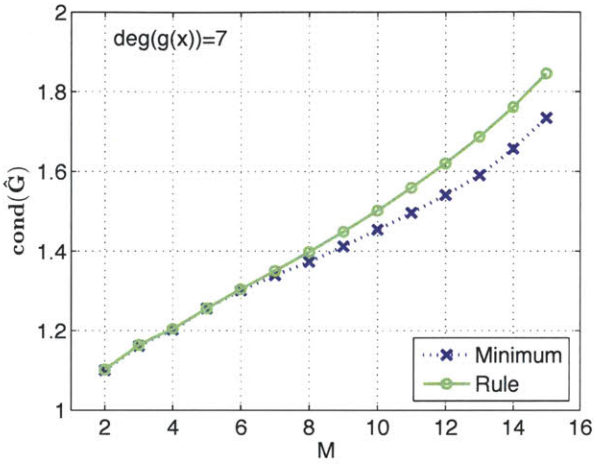


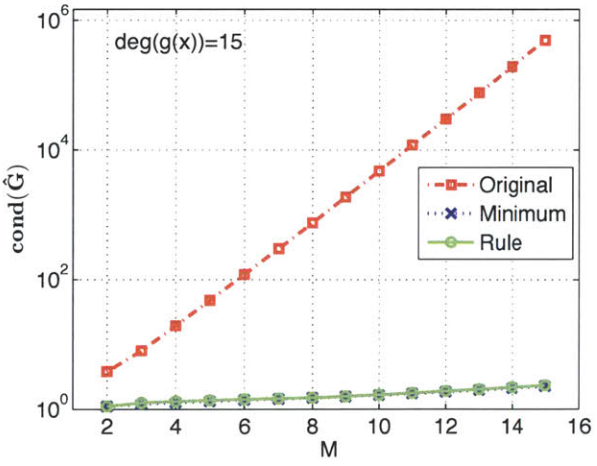
Figure 4-14: The Sensitivities $S_{\hat{g}\to z_h}$ and $S_{z_h\to\hat{g}}$ with Different q_r.



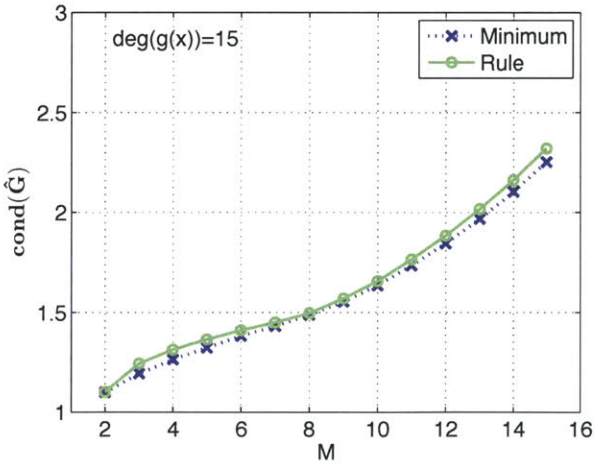
(a): $\deg(g) = 7$



(b): $\deg(g) = 7$ (Magnified)



(c): $\deg(g) = 15$



(d): $\deg(g) = 15$ (Magnified)

Figure 4-15: Comparison of the Condition Number of $\hat{\mathbf{G}}$ among the Original Value, the Minimum Value, and the Value Achieved with the Approximate Rules (4.39)-(4.41).

Chapter 5

Exact Decomposition Algorithms

In this chapter, algorithms are discussed for the exact decomposition for polynomials that are known to be decomposable. The associated simulation results are also presented. Three algorithms are shown for the problems as defined in Section 3.2: one of the algorithms has coefficients as input corresponding to Problem 1, while the other two have roots as input corresponding to Problem 2. Simulations are performed to compare the new algorithm developed in Section 5.2.3 with two other existing algorithms. [1]

In the development of the methods, an assumption is made without loss of generality. Specifically, in this chapter, we assume that all the polynomials $f(x)$, $g(x)$, and $h(x)$ are monic, and the constant term in $g(x)$ is zero; i.e.,

$$a_M = b_N = c_{MN} = 1, \quad \text{and} \quad b_0 = 0, \tag{5.1}$$

where a_i, b_i, and c_i are the coefficients of the term x^i in $f(x)$, $g(x)$, and $h(x)$, respectively.

The validity of this assumption results from the equivalent compositions with first-degree polynomials. As mentioned in Sections 2.1 and 4.2, for an arbitrary first-degree polynomial $q(x) = q_1 x + q_0$, it holds that $f \circ g = \hat{f} \circ \hat{g}$, where $\hat{f}(x) = (f \circ q^{-1})(x)$

[1] The first two algorithms in this chapter and the associated simulation results are included in [11] by S. Demirtas, G. Su, and A. V. Oppenheim.

and $\hat{g}(x) = (q \circ g)(x)$. Thus, by choosing a proper $q(x)$, we can set $\hat{g}(x)$ as a monic polynomial with a constant term of zero. Consequently, there always exists a way of decomposing $h(x)$ as $(\hat{f} \circ \hat{g})(x)$ such that $b_N = 1$ and $b_0 = 0$. This case implies that $c_{MN} = a_M b_N^M = a_M$, so the leading coefficients of $\hat{f}(x)$ and $h(x)$ are the same. Thus, $\hat{f}(x)$ and $h(x)$ can be scaled to monic polynomials simultaneously. This concludes the validation of our assumption.

As a byproduct, equivalent compositions with linear functions also imply that the degrees of freedom of decomposable $h(x)$ (with fixed M and N) are at most $(M+1) + (N+1) - 2 = M + N$, which is two fewer than the total number of coefficients in $f(x)$ and $g(x)$.

5.1 Problem 1: Exact Decomposition with Coefficients as Input

An algorithm for exact decomposition with coefficients as input was proposed by Kozen and Landau in [17]. This algorithm employs the fact that the N highest degree terms of $h(x)$ depend only on the coefficients of $g(x)$ and the highest degree term of $f(x)$. By solving a system of equations iteratively, the coefficients of $g(x)$ can be obtained in the descending order of the degrees of terms. After obtaining $g(x)$, the coefficients of $f(x)$ can be solved by a projection method due to the linear relationship between $h(x)$ and $f(x)$ as shown in (4.1).

The coefficients of several highest degree terms in $h(x)$ have the expressions:

$$
\begin{aligned}
c_{MN} &= a_M b_N^M \\
c_{MN-1} &= \binom{M}{1} a_M b_N^{M-1} b_{N-1} \\
c_{MN-2} &= \binom{M}{2} a_M b_N^{M-2} b_{N-1}^2 + \binom{M}{1} a_M b_N^{M-1} b_{N-2} \\
c_{MN-3} &= \binom{M}{3} a_M b_N^{M-3} b_{N-1}^3 + \binom{M}{1}\binom{M-1}{1} a_M b_N^{M-2} b_{N-1} b_{N-2} + \binom{M}{1} a_M b_N^{M-1} b_{N-3}
\end{aligned}
$$

These equations imply the theorem that is observed by Kozen and Landau [17]:

Theorem 5.1. [17]. *For $1 \leq k \leq N-1$, the coefficient c_{MN-k} in $h(x)$ satisfies*

$$c_{MN-k} = \mu_k\left(a_M, b_N, b_{N-1} \ldots b_{N-k+1}\right) + \binom{M}{1} a_M b_N^{M-1} b_{N-k}, \tag{5.2}$$

where $\mu_k\left(a_M, b_N, b_{N-1} \ldots b_{N-k+1}\right)$ is a polynomial of a_M and of the coefficients of terms with degrees higher than $b_{N-k}x^{N-k}$ in $g(x)$.

This theorem is directly implied by the fact that the terms $c_{MN-k}x^{MN-k}$ $(1 \leq k \leq N-1)$ in $h(x)$ can be generated only by $a_M g(x)^M$, i.e., the highest degree term in $f(x)$. Using multinomial expansion, we can further see that c_{MN-k} is independent of b_{N-j} where $k < j \leq N$. A detailed discussion can be found in [17], which has the same core idea as the fact above.

Theorem 5.1 shows clearly that, after $a_M, b_N, b_{N-1} \ldots b_{N-k+1}$ are obtained, a linear equation can be constructed with respect to b_{N-k}. Thus, the coefficients of $g(x)$ can be obtained one by one from the highest degree term to the lowest; in this process of *coefficient unraveling*, only the N highest degree terms in $h(x)$ are used.

After $g(x)$ is obtained, the determination of $f(x)$ can be accomplished by the solution to the linear equations (4.1):

$$\mathbf{f} = \mathbf{G}^{\dagger}\mathbf{h}, \tag{5.3}$$

where $\mathbf{G}^{\dagger} = (\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1}\mathbf{G}^{\mathrm{T}}$ is the pseudo-inverse matrix of $\mathbf{G}$, and the matrix $\mathbf{G}$ is the self-convolution matrix as defined in (4.2).

In summary, the algorithm for exact decomposition with coefficients as input is stated as follows:

Coefficient-Unraveling Algorithm [17]

(1) for $k = 1$ to $N-1$

(2) Calculate $\mu_k\left(a_M, b_N, b_{N-1}, \ldots, b_{N-k+1}\right)$

(3) Obtain $b_{N-k} = \left(c_{MN-k} - \mu_k\left(a_M, b_N, b_{N-1}, \ldots, b_{N-k+1}\right)\right)/(Ma_M)$

(4) endfor

(5) Construct the matrix $\mathbf{G}$ as defined in (4.2).

(6) Solve the coefficients of $f(x)$ by (5.3).

5.2 Problem 2: Exact Decomposition with Roots as Input

This section presents two algorithms for exact decomposition with roots as input: the first algorithm is based on the algorithm by Aubry and Valibouze [20]; we present a new algorithm based on grouping the roots of the composed polynomial. Both algorithms use the structure of the roots of a decomposable polynomial, as well as the relationship between the coefficients and the roots of a polynomial. The properties of the roots of a decomposable polynomial are first stated in Section 5.2.1, and then the two algorithms are presented in Sections 5.2.2 and 5.2.3, respectively.

5.2.1 Properties of Roots of a Decomposable Polynomial

As mentioned in Section 4.1.2, we can partition all the roots of $h(x)$ into M groups A_1, A_2, ..., A_M, and the roots in each group correspond to the same root of $f(x)$. The same as in Section 4.1.2, we denote the roots of $f(x)$ and $h(x)$ as $z_f(i)$ $(1 \leq i \leq M)$ and $z_h(k)$ $(1 \leq k \leq MN)$, respectively. Then, the roots $z_h(k)$ in the group A_i all satisfy $\tilde{g}_i(z_h(k)) = 0$, where $\tilde{g}_i(x) = g(x) - z_f(i)$ is a polynomial with a different constant term from $g(x)$. Without loss of generality, we assume all polynomials involved are monic.

Since each $\tilde{g}_i(x)$ has the same coefficients except for the constant term, Theorem 5.2 is implied by Vieta's theorem that states the relationship between the coefficients and the roots of a polynomial [27].

Theorem 5.2. **[27]**. *For each $j = 1, 2, ..., N-1$, the j-th elementary symmetric function $\sigma_j(\cdot)$ has the same value on all groups A_i $(1 \leq i \leq M)$, where $\sigma_j(A_i)$ is*

defined as

$$\sigma_j(A_i) = \sum_{k_1<k_2<\cdots<k_j\in A_i} z_h(k_1)z_h(k_2)z_h(k_3)\cdots z_h(k_j), \quad 1 \le j \le N-1. \tag{5.4}$$

Newton's identities [28], which show the relationship between the elementary symmetric functions and the power summations of the roots of a polynomial, imply the validity of Theorem 5.3 from Theorem 5.2 [20].

Theorem 5.3. **[20]**. *For each $j = 1, 2, ..., N-1$, the j-th power summation of roots $s_j(\cdot)$ has the same value on all groups A_i $(1 \le i \le M)$, where $s_j(A_i)$ is defined as*

$$s_j(A_i) = \sum_{k\in A_i} (z_h(k))^j, \quad 1 \le j \le N-1. \tag{5.5}$$

5.2.2 Root-Power-Summation Algorithm

Based on Theorem 5.3, an algorithm was proposed in [20] in order to determine a polynomial $g(x)$ from its self-convolution $(g(x))^N$. In fact, this algorithm can also be applied to the general problem of determining $g(x)$ from the composition $(f \circ g)(x)$.

The algorithm in [20] has the following principles. Since the power summations $s_j(A_i)$ in (5.5) are equal for each group A_i $(i = 1, 2, ..., M)$, they can be directly computed by

$$s_j \triangleq s_j(A_i) = \frac{1}{M}\sum_{k=1}^{MN} (z_h(k))^j, \quad 1 \le j \le N-1. \tag{5.6}$$

Then, the coefficients of $g(x)$ are obtained by using Newton's identities [20, 28].

Next, we need to determine the component $f(x)$. An elementary way is to first obtain the roots of $f(x)$ by clustering $g(z_h(k))$ for $k = 1, 2, \ldots, MN$ and using the largest M clusters; then, $f(x)$ is solved by $f(x) = \prod_{i=1}^{M}(x - z_f(i))$. However, since numerical errors in the input roots may be magnified and cause the final reconstruction of $f(x)$ to be inaccurate, the description above is just a basic implementation of the

root-power-summation algorithm. We make an improvement to this algorithm in our implementation to enhance precision; however, for easier and more clear description, we discuss the improvement at the end of Section 5.2.3 rather than in this section.

The root-power-summation algorithm is summarized as follows.

Root-Power-Summation Algorithm

(1) Compute s_j for $j = 1, 2, ..., N-1$ by (5.6) [20].

(2) Compute the coefficients of $g(x)$ using Newton's identities [20].

(3) Compute $g(z_h(k))$ and construct $z_f(i)$.

(4) Construct $f(x)$ by $f(x) = \prod_{i=1}^{M}(x - z_f(i))$.

5.2.3 Root-Grouping Algorithm

In this section, we propose a new algorithm that uses the root-grouping information for decomposition. For each root $z_h(k)$ $(1 \leq k \leq MN)$, denote β_k as the index of the root of $f(x)$ such that $z_f(\beta_k) = g(z_h(k))$. Then, the mapping property in (3.7) is expressed in the following matrix form (5.7). In contrast to the coefficients of $h(x)$ that have a complex nonlinear dependence on $g(x)$, we can form the following linear equations with respect to $g(x)$ with the roots z_f and z_h:

$$\begin{bmatrix} z_h^N(1) & z_h^{N-1}(1) & \cdots & z_h(1) & 1 \\ z_h^N(2) & z_h^{N-1}(2) & \cdots & z_h(2) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ z_h^N(MN) & z_h^{N-1}(MN) & \cdots & z_h(MN) & 1 \end{bmatrix} \begin{bmatrix} b_N \\ b_{N-1} \\ \vdots \\ b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} z_f(\beta_1) \\ z_f(\beta_2) \\ \vdots \\ z_f(\beta_{MN}) \end{bmatrix}. \quad (5.7)$$

The core problem in this approach is to determine the grouping information β_k, since this information directly leads to the solution to (5.7). The grouping information is theoretically difficult to obtain, since the total number of possible grouping patterns is extremely large. Equation (4.17) implies that each root z_f corresponds to N roots of z_h, so we want to partition z_h into M groups, each of which has N elements. The total number of such partitions is $\frac{(MN)!}{M!(N!)^M}$, which increases extremely fast with M

and N. Thus, searching all possible grouping patterns is impractical due to its high computational complexity.

However, Theorem 5.3 constrains the possible grouping patterns and effectively decreases the computational complexity in practice. We propose an approach that formulates the grouping pattern as a mixed integer program. There are M steps in this approach; in each step, we determine the N roots in a group, and then we remove them from the roots that remain to be grouped. To determine the roots that form a new group, we introduce binary indicators δ_k for each $z_h(k)$ that has not been grouped yet, and the following mixed integer program (MIP) is constructed:

$$\begin{aligned} \min \quad & 0 && (5.8)\\ \text{s. t.} \quad & \sum_{k\in\mathcal{S}} \delta_k \cdot (z_h(k))^j = s_j, \quad \forall j = 1, 2, ..., N-1, && (5.9)\\ & \sum_{k\in\mathcal{S}} \delta_k = N, && (5.10)\\ & \delta_k \in \{0, 1\}, \quad \forall k \in \mathcal{S}. \end{aligned}$$

The set $\mathcal{S}$ is those roots $z_h(k)$ that have not been grouped. If $\delta_k = 1$, then $z_h(k)$ is in the newly constructed group; otherwise, $z_h(k)$ remains not grouped. The constraint (5.9) is due to Theorem 5.3, and the constraint (5.10) results from the fact that each group has N roots. The values of s_j in (5.9) are calculated by (5.6). Due to numerical errors in implementation, the constraint (5.9) can be relaxed to

$$\left| \sum_{k\in\mathcal{S}} \delta_k \cdot (z_h(k))^j - s_j \right| \leq \epsilon |s_j|, \quad \forall j = 1, 2, ..., N-1, \tag{5.11}$$

for a small ϵ; furthermore, since the roots $z_h(k)$ are mostly complex numbers, the left-hand-side of (5.11) for each j is implemented for the real part and the imaginary part, respectively. [2] Since we are interested only in the binary points in the feasible region, the cost function can be arbitrary and we set it as 0 for simplicity. After

[2] In contrast, the right-hand-side of (5.11) is a constant for the optimization problem, so we do not separate the real and imaginary parts of it; in addition, $|s_j|$ in (5.11) is the magnitude of the complex number s_j.

the mixed integer optimization problem in (5.8) has been solved for M times, the grouping information can be fully determined.

An improvement is performed on the procedure above to decrease time complexity for obtaining the grouping information. In fact, we do not need to solve the optimization problem in (5.8) for M times; instead, we only solve (5.8) for one time and get one group of roots. Then we can construct $g(x)$ as follows:

$$g(x) = \prod_{k \in A_1} (x - z_h(k)) - (-1)^N \prod_{k \in A_1} z_h(k), \tag{5.12}$$

Although numerical errors may cause (5.12) to be a rough reconstruction, however, we may still use (5.12) to determine the grouping information. The roots z_h in one group should have the same value of $g(z_h)$, so we can cluster $g(z_h)$ to obtain the grouping information of z_h.

After obtaining the full grouping information, we consider the construction of $g(x)$ and z_f. Theoretically, we can construct $g(x)$ in (5.12) with any one group, then the roots of $f(x)$ are computed by $z_f(i) = g(z_h(k)),\ k \in A_i,\ 1 \leq i \leq M$. However, accumulation of numerical error may cause the direct expansion (5.12) to be inaccurate. To enhance robustness and precision, we utilize the linear relationship in (5.7) and form the following linear program to solve $z_f(i)$ and $g(x)$:

$$\min_{b_j, z_f(i)} \quad \sum_{i=1}^{M} \sum_{k \in A_i} |\psi_{i,k}| \tag{5.13}$$

$$\text{s. t.} \quad \psi_{i,k} = \sum_{j=1}^{N-1} (z_h(k))^j \cdot b_j + (z_h(k))^N - z_f(i),$$

$$\text{for } i = 1, 2, ..., M,\ k \in A_i. \tag{5.14}$$

The cost function for this linear program is the total deviation from $g(z_h(k))$ to $z_f(i)$ where $z_h(k)$ belongs to the group A_i; the deviation should be zero in theory without numerical error. The grouping information A_i has been obtained by solving the mixed integer program in (5.8). Since the roots $z_h(k)$ and $z_f(i)$ are mostly complex numbers, we implement (5.14) for the real and imaginary parts, respectively; then, the terms

in the cost function are implemented as $|\psi_{i,k}| \triangleq |\mathrm{Re}\{\psi_{i,k}\}| + |\mathrm{Im}\{\psi_{i,k}\}|$. In addition to the constraints listed above, we also constrain that the roots z_f that correspond to conjugate pairs of z_h are also in a conjugate pair, which ensures that $f(x)$ has real coefficients.

The complete algorithm is summarized as follows:

Root-Grouping Algorithm

(1) Set $\mathcal{S} = \{1, 2, ..., MN\}$, and compute s_j $(1 \leq j \leq N-1)$ from (5.6).

(2) Solve the integer program in (5.8).

(3) Construct the first group $A_1 = \{k \in \mathcal{S} \mid \delta_k = 1\}$.

(4) Obtain a rough reconstruction of $g(x)$ from (5.12).

(5) Determine the full grouping information A_i $(1 \leq i \leq M)$ by clustering $g(z_h)$.

(6) Construct precise $g(x)$ and $z_f(i)$ from linear optimization (5.13).

(7) Construct $f(x)$ by $f(x) = \prod_{i=1}^{M}(x - z_f(i))$.

Compared to the high complexity of a number of integer programming problems, the efficiency in practice of formulation (5.8) is usually high.

As a last comment, the technique of reconstructing $g(x)$ and z_f from the linear program in (5.13) is also applicable to the root-power-summation algorithm, which can improve the overall precision. In that algorithm, we have a rough reconstruction of $g(x)$ using the power summation of roots (5.6) and Newton's identity [20]. Then, we can obtain the grouping information of roots z_h by clustering $g(z_h)$. With the grouping information, we finally use the linear program in (5.13) to solve $g(x)$ and z_f, which enhances numerical performance.

5.3 Evaluation of the Exact Decomposition Algorithms

This section presents a comparison between the three exact decomposition algorithms with respect to the success rates of $f(x)$, $g(x)$, and $h(x)$ in the decomposition. From the coefficients or the roots of a decomposable polynomial $h(x) = (f \circ g)(x)$, the

three algorithms obtain the components $\bar{f}(x)$ and $\bar{g}(x)$, and then we compose $\bar{f}(x)$ and $\bar{g}(x)$ into $\bar{h}(x) = (\bar{f} \circ \bar{g})(x)$. The errors in this decomposition process are defined as

$$err_p(x) = p(x) - \bar{p}(x), \qquad \text{for } p(x) = f(x),\ g(x),\ \text{or } h(x).$$

The signal to error ratios (SER) are defined as

$$SER_p \triangleq 20 \log_{10} \left(\frac{\|p(x)\|_2}{\|err_p(x)\|_2} \right), \qquad \text{for } p(x) = f(x),\ g(x),\ \text{or } h(x).$$

The criterion of successful decomposition is chosen as $SER_p \geq 80$dB in the simulation, for $p(x) = f(x),\ g(x),\ \text{or } h(x)$.

In the simulation, the degrees of $f(x)$ and $g(x)$ are equal and vary from 5 to 75 with increments of 5; the corresponding degrees of $h(x)$ vary from 25 to 5625. For each fixed degree, we generate 100 samples of $h(x)$ by composing monic polynomial components $f(x)$ and $g(x)$ with coefficients of i.i.d. standard Gaussian distribution (except for the leading terms and the constant terms: $f(x)$ and $g(x)$ are both monic polynomials, and $g(x)$ has a constant term of zero). The roots z_f and z_h are computed, and then z_h are sorted into random order. Then all three algorithms perform decomposition on these samples. The details of parameter setting for the algorithms are as follows. In the algorithms working with roots, two *reconstructed roots* $g(z_h)$ are considered to belong to one cluster (i.e., correspond to the same z_f) if the distance between them is lower than a threshold, for which we use 10^{-3}. In the root-grouping algorithm, the mixed integer optimization problem (5.8) is solved by the CPLEX software where we set the time limit for the MIP as 2 minutes, and the parameter ϵ in (5.11) is chosen as 10^{-11} in our simulation. For each algorithm, we record its successful decomposition rates within the sample polynomials according to the criterion above; the results for $f(x)$, $g(x)$, and $h(x)$ are plotted in Fig. 5-1 (a), (b), and (c), respectively.

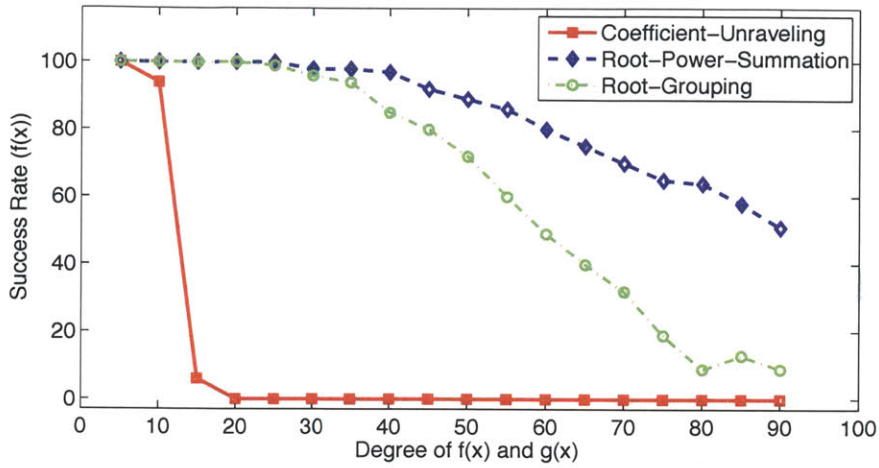
Figure 5-1 indicates that among these three algorithms, the root-power-summation algorithm has the best performance, followed by the root-grouping algorithm; the coefficient-unraveling algorithm has the lowest successful decomposition rate. For example, when $M = N = 50$, the coefficient-unraveling algorithm fails to decompose

any sample polynomial; the root-grouping algorithm achieves successful decomposition on 72, 76, and 75 samples of $f(x)$, $g(x)$, and $h(x)$, respectively; the root-power-summation algorithm succeeds in obtaining 89, 94, and 93 samples of $f(x)$, $g(x)$, and $h(x)$, respectively. Since the root-power-summation algorithm and the root-grouping algorithm work with roots, while the coefficient-unraveling algorithm works with coefficients, we can conclude that in our simulation, the exact decomposition with roots as input is more robust than with the coefficients as input. The reasons are two-fold. First, the coefficient-unraveling algorithm uses only the coefficients of the N highest degree terms in $h(x)$ to obtain $g(x)$, which does not make full use of the input data. In contrast, the root-power-summation algorithm and the root-grouping algorithm use all the MN roots to obtain $g(x)$ and z_f, so they have better performance. Second, the iterative method in the coefficient-unraveling algorithm to obtain the coefficients of $g(x)$ accumulates numerical errors and may expand them exponentially. In contrast, in our simulation, the algorithms working with roots seem not to expand numerical errors so significantly.

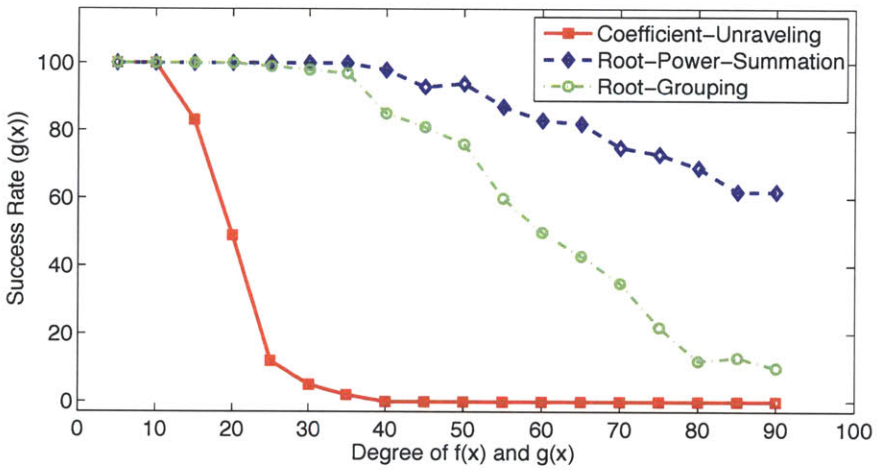
An interesting observation for the algorithms working with roots is that the successful decomposition rates of $f(x)$, $g(x)$, and $h(x)$ are generally similar for a fixed degree in our simulation. In fact, if these algorithms obtain the correct grouping information, then the reconstruction of $g(x)$ and z_f in our simulation is reasonably precise using the linear program in (5.13). In contrast, for the coefficient-unraveling algorithm, $g(x)$ in the simulation usually has much higher success rate than $f(x)$ for degrees under 35 (above 40, both of the success rates drop to zero). In this algorithm, $g(x)$ is first obtained and then used to determine $f(x)$ in the subsequent steps; thus, the failure to determine $g(x)$ usually leads to failure in $f(x)$. As a result, $g(x)$ usually has higher success rate. In addition, to determine $f(x)$, the coefficient-unraveling algorithm uses the least square projection that minimizes the error in $h(x)$, so the reconstructed $h(x)$ is possibly successful even if the obtained $f(x)$ is already inaccurate. Thus, for the coefficient-unraveling algorithm, the success rate of $h(x)$ is also normally higher than that of $f(x)$ for a fixed degree.

While the main difficulty of the coefficient-unraveling algorithm and the root-

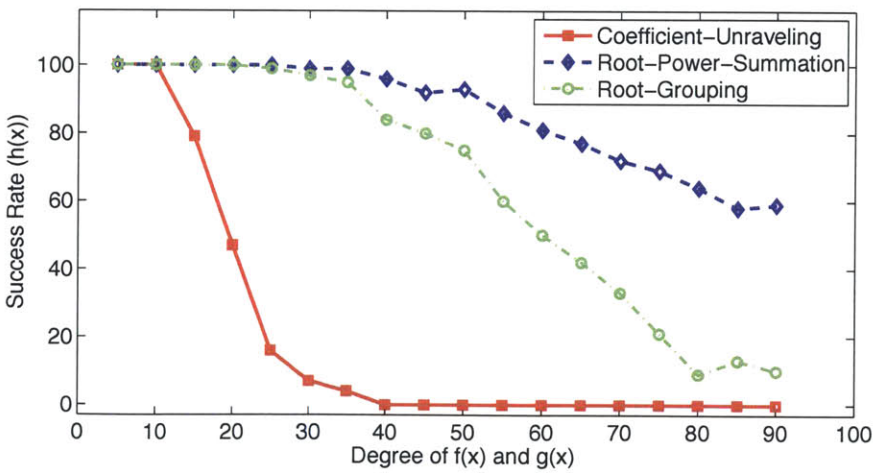
power-summation algorithm is numerical errors, the main challenge for the root-grouping algorithm is solving the mixed integer program in (5.8). The success rates of all algorithms generally decrease as the degrees M and N increase, since both numerical errors in all algorithms and the scale of MIP in the root-grouping algorithm increase with the degrees. However, the solution to the MIP problem is considerably efficient as compared with general MIP problems, especially when we take its scale into account. For example, when $M = N = 30$, there are 900 binary variables in the MIP problem, and there are 9.80×10^{55} possible patterns for the first group without considering the constraints on the power summations. However, the constraints from Theorem 5.3 efficiently shrink the feasible region and lower down the complexity so that the grouping information is obtained for 97 samples within the time limit of 2 minutes. Moreover, the efficiency of the MIP formulation depends on individual samples of polynomial; in general, we speculate that our MIP formulation in (5.8) may be more efficient if the absolute values of roots $|z_h|$ do not have a large range. This speculation results from the constraints (5.11). If there are two roots with a very large and a very small absolute value, respectively, then the high powers of the two roots have significantly different magnitudes. Thus, the power of the small root is susceptible to numerical errors and does not have much influence on the power summations when j is large. Consequently, if the large root is in the new group, it is not effective to decide whether the small root belongs to the group using the constraint (5.11) with a large j. In other words, such constraints may become ineffective to shrink the feasible region, and the computational complexity may not get efficiently decreased. In contrast, if all the roots have similar magnitudes, then it is likely that the power summation constraint (5.11) for each power j effectively shrinks the binary points in the feasible region, which results in higher overall efficiency.



(a) $f(x)$



(b) $g(x)$



(c) $h(x)$

Figure 5-1: Comparison between the three exact decomposition algorithms on the Success Rates of (a) $f(x)$, (b) $g(x)$, and (c) $h(x)$.

Chapter 6

Approximate Decomposition Algorithms

In this chapter, we present and evaluate algorithms for approximate polynomial decomposition. In contrast to exact decomposition which can be thought of as a problem of identification, approximate decomposition corresponds to modeling. Four algorithms are explored for the problems defined in Section 3.3: three of the algorithms are for Problem 3 for which the input is the coefficients, and the fourth is for Problem 4 for which the input is the roots. Simulation results are presented to evaluate the algorithms. [1]

6.1 Problem 3: Approximate Decomposition with Coefficients as Input

This section presents three algorithms for the approximate decomposition problem with polynomial coefficients as input, as stated in Section 3.3. The algorithms belong to two categories: the iterative mean square approximation algorithm and algorithms based on the Ruppert matrix.

[1]The first three algorithms in this chapter and the associated simulation results are included in [11] by S. Demirtas, G. Su, and A. V. Oppenheim.

6.1.1 Iterative Mean Square Approximation Algorithm

This approach was proposed by Corless et al. in [18]; it updates the polynomials $f(x)$ and $g(x)$ iteratively, where the update is achieved by approximate linearization and mean square projection. Since the composition is linear with respect to $f(x)$, if we fix $g(x)$ and want to obtain an approximate polynomial $f(x)$ to minimize the error energy $\|h(x) - (f \circ g)(x)\|_2^2$, then the optimal $f(x)$ is the mean square projection as given in (5.3). However, if we fix $f(x)$ and want to derive the optimal $g(x)$ to minimize the error energy, the problem is much more challenging since the composition is non-linear with respect to $g(x)$. If the input polynomial is near a decomposable one, then we may apply the Taylor approximation

$$h(x) - (f \circ (g + \Delta g))(x) \approx h(x) - (f \circ g)(x) - (f' \circ g)(x) \cdot \Delta g(x), \tag{6.1}$$

where $f'(x)$ is the derivative of $f(x)$. If we denote $r(x) = h(x) - (f \circ g)(x)$, then (6.1) can be presented in an equivalent matrix formulation

$$\mathbf{D} \cdot \mathbf{\Delta g} \approx \mathbf{r}, \tag{6.2}$$

where the matrix $\mathbf{D}$ is the Teoplitz matrix in (4.11). Thus, we can apply the mean square projection to obtain $\Delta g(x)$, where $g(x) + \Delta g(x)$ approximately minimizes the error energy $\|h(x) - (f \circ (g + \Delta g))(x)\|_2^2$:

$$\mathbf{\Delta g} = \mathbf{D}^{\dagger}\mathbf{r}. \tag{6.3}$$

If the input polynomial is near a decomposable polynomial, then the coefficient-unraveling algorithm for exact decomposition [17] in Section 5.1 would possibly have outputs $f(x)$ and $g(x)$ that are sufficiently near the optimal polynomials. Thus, the coefficient-unraveling algorithm can provide initial values for $f(x)$ and $g(x)$. In the iteration, we iteratively use linear projections (5.3) and (6.3) to update $f(x)$ and $g(x)$, respectively. The algorithm is summarized as follows:

Iterative Mean Square Approximation Algorithm [18]

(1) Obtain the initial guess $g^{(0)}(x)$ from the algorithm in Section 5.1.

(2) In the k-th iteration:

(3) Obtain $f^{(k)}(x)$ from (5.3), where the matrix $\mathbf{G}$ is constructed with $g^{(k-1)}(x)$ from the last iteration.

(4) Obtain $\Delta g^{(k)}(x)$ from (6.3), where the matrix $\mathbf{D}$ and the vector $\mathbf{r}$ are computed with $f^{(k)}(x)$ and $g^{(k-1)}(x)$.

(5) Let $g^{(k)}(x) = g^{(k-1)}(x) + \Delta g^{(k)}(x)$

(6) Continue until the $\|h(x) - (f^{(k)} \circ g^{(k)})(x)\|_2$ is sufficiently small or k attains the limit on iteration steps.

This algorithm has low computational complexity and considerably good empirical performance when the input polynomial is sufficiently near a decomposable polynomial. However, there is no guarantee on the convergence of this algorithm in general; since linearization is performed, the algorithm may converge into a local minimum rather than the global minimum. In addition, the initialization step utilizes the algorithm in Section 5.1 that uses only the N highest degree terms in $h(x)$ to obtain the component $g^{(0)}(x)$; if the highest degree terms in $h(x)$ have much noise, then the initial guess $g^{(0)}(x)$ may be significantly noisy, which possibly leads to divergence of the algorithm or convergence to a local minimum.

6.1.2 Algorithms Based on the Ruppert Matrix

This type of approximate decomposition algorithms [16] is based on a mathematical theory that establishes an equivalence between the decomposability of a polynomial and the rank deficiency of a corresponding matrix, which is named the Ruppert matrix. Consequently, the approximate decomposition problem is converted to determining a structured rank deficient approximation for a given Ruppert matrix. In this section, the theoretical principles proposed in [15, 19, 25] are briefly described; then we present the formulation by Giesbrecht et al. [16] of the decomposition problem

into a Structured Total Least Square (STLS) problem [21,24]; finally, we present two algorithms proposed in [16,21,23,24] to solve the converted STLS problem.

The decomposability of polynomials is converted to the rank deficiency of a Ruppert matrix via two steps [16]. In the first step, the univariate polynomial decomposition problem is transformed to corresponding bivariate polynomial factorization [15,19], as the following theorem states.

Theorem 6.1. [19]. *For a polynomial $h(x)$ that is defined on $x \in \mathbb{C}$ and has a non-prime degree, $h(x)$ is indecomposable if and only if the bivariate polynomial*

$$\Phi_h(x_1, x_2) = \frac{h(x_1) - h(x_2)}{x_1 - x_2} \tag{6.4}$$

is absolutely irreducible.

In the second step, the bivariate polynomial factorization problem is further transformed into a partial differential equation problem with the following theorem [25].

Theorem 6.2. [25]. *Suppose a bivariate polynomial $H(x_1, x_2)$ has bi-degree (P, Q), which means $\deg_{x_1}(H) = P$ and $\deg_{x_2}(H) = Q$. Then $H(x_1, x_2)$ is absolutely irreducible if and only if the partial differential equation*

$$H\frac{\partial U}{\partial x_2} - U\frac{\partial H}{\partial x_2} + V\frac{\partial H}{\partial x_1} - H\frac{\partial V}{\partial x_1} = 0 \tag{6.5}$$

has no nontrivial solution $U(x_1, x_2)$, $V(x_1, x_2)$ with degree constraints

$$\deg(U) \leq (P-1, Q), \quad \deg(V) \leq (P, Q-2). \tag{6.6}$$

Since (6.5) is linear with respect to the coefficients of polynomials $U(x_1, x_2)$ and $V(x_1, x_2)$ when $H(x_1, x_2)$ is fixed, the partial differential equation (6.5) is equivalent to the following linear equation [25]

$$\mathbf{Rup}(H) \cdot \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \mathbf{0}, \tag{6.7}$$

where $\mathbf{Rup}(H)$ is the Ruppert matrix [25] of the polynomial $H(x_1, x_2)$; both $\mathbf{u}$ and $\mathbf{v}$ are vectors with elements as the coefficients of polynomials $U(x_1, x_2)$ and $V(x_1, x_2)$, respectively. The Ruppert matrix $\mathbf{Rup}(H)$ represents the polynomial multiplication in (6.5): each row corresponds to the coefficient of a term $x_1^i x_2^j$ in the left-hand-side of (6.5), and each column represents the factors to be multiplied by a term in $U(x_1, x_2)$ or $V(x_1, x_2)$. The size of the Ruppert matrix is $(4PQ - 2P) \times (2PQ + Q - 1)$; the elements of $\mathbf{Rup}(H)$ depend only on the coefficients in $H(x_1, x_2)$ as well as the degrees P and Q. The degree constraints (6.6) are incorporated into (6.7) by the size (and indexing) of the vectors $\mathbf{u}$ and $\mathbf{v}$. The linear system (6.7) has a non-trivial solution, if and only if the Ruppert matrix $\mathbf{Rup}(H)$ is rank deficient.

The theorems above show that for a polynomial $h(x)$ with a non-prime degree, $h(x)$ is decomposable if and only if the corresponding Ruppert matrix $\mathbf{Rup}\left(\Phi_h(x_1, x_2)\right)$ is rank deficient [16].

Next, we present the method in [16] to formulate the approximate decomposition problem into an STLS problem with (6.7). For an indecomposable $h(x)$, our goal is to determine a decomposable polynomial $\hat{h}(x) = (f \circ g)(x)$ that is close to $h(x)$. Then we know that $\mathbf{Rup}\left(\Phi_h(x_1, x_2)\right)$ has full rank, while $\mathbf{Rup}\left(\Phi_{\hat{h}}(x_1, x_2)\right)$ is rank deficient. Thus, the approximate decomposition problem becomes determining a rank deficient Ruppert matrix $\hat{\mathbf{R}} = \mathbf{Rup}\left(\Phi_{\hat{h}}(x_1, x_2)\right)$ that is close to $\mathbf{R} = \mathbf{Rup}\left(\Phi_h(x_1, x_2)\right)$ [16]. After the solution of the Ruppert matrix $\hat{\mathbf{R}}$, the approximate decomposition $\hat{h}(x) = (f \circ g)(x)$ can be obtained directly from the elements of the matrix.

It should be noticed that the rank deficient matrix $\hat{\mathbf{R}} = \mathbf{Rup}\left(\Phi_{\hat{h}}(x_1, x_2)\right)$ is not an arbitrary Ruppert matrix; the construction of the bivariate polynomial in (6.4) introduces certain constraints on the *structure* of the Ruppert matrix [16]. Using (6.4) and the linearity of Ruppert matrix to the bivariate polynomial, we can see that

$$\hat{\mathbf{R}} = \mathbf{Rup}\left(\Phi_{\hat{h}}(x_1, x_2)\right) = \mathbf{Rup}\left(\sum_{k=1}^{MN} \hat{h}_k \sum_{j=0}^{k-1} x_1^j x_2^{k-1-j}\right) = \sum_{k=1}^{MN} \hat{h}_k \mathbf{R}_k, \tag{6.8}$$

where the matrices $\mathbf{R}_k$ are

$$\mathbf{R}_k = \mathbf{Rup}\left(\sum_{j=0}^{k-1} x_1^j x_2^{k-1-j}\right), \quad 1 \le k \le MN. \tag{6.9}$$

With the structure in (6.8), the approximate decomposition problem has been transformed into the following optimization problem [16], which is an instance of the Structured Total Least Square (STLS) problem [21, 24].

$$\min_{\hat{\mathbf{h}}} \quad \|\hat{\mathbf{h}} - \mathbf{h}\|_2, \tag{6.10}$$

$$\text{s. t.} \quad \hat{\mathbf{R}} = \sum_{k=1}^{MN} \hat{h}_k \mathbf{R}_k \text{ is rank deficient.} \tag{6.11}$$

Finally, we turn to the solution of the STLS problem in (6.10). There is no exact solution to the STLS problem, but there are heuristic algorithms [21–24, 26]. Here we show two algorithms, namely a heuristic algorithm based on Riemannian singular vector decomposition [21] and the STLS relaxation algorithm [22, 26]. We can note that the constant term in $h(x)$ does not influence its decomposability, so we only care about the other terms in $h(x)$ in the development of the following algorithms.

A Heuristic Based on Riemannian Singular Vector Decomposition

The problem in (6.10) is a special instance of the Riemannian Singular Vector Decomposition (RiSVD) framework [21], and the RiSVD problem can be solved by an iterative heuristic algorithm proposed in [21].

The RiSVD formulation for the STLS problem in (6.10) aims to obtain vectors $\mathbf{p}$, $\mathbf{q}$, and the smallest scalar τ such that

$$\mathbf{Rq} = \mathbf{D_q p}\tau, \tag{6.12}$$

$$\mathbf{R}^{\mathrm{T}}\mathbf{p} = \mathbf{D_p q}\tau, \tag{6.13}$$

$$\mathbf{p}^{\mathrm{T}}\mathbf{D_q p} = \mathbf{q}^{\mathrm{T}}\mathbf{D_p q} = 1, \tag{6.14}$$

where $\mathbf{R}$ is the Ruppert matrix, and the matrices $\mathbf{D_p}$ and $\mathbf{D_q}$ are

$$\mathbf{D_p} = \sum_{k=1}^{MN} \mathbf{R}_k^{\mathrm{T}}\mathbf{p}\left(\mathbf{R}_k^{\mathrm{T}}\mathbf{p}\right)^{\mathrm{T}}, \tag{6.15}$$

$$\mathbf{D_q} = \sum_{k=1}^{MN} \mathbf{R}_k\mathbf{q}\left(\mathbf{R}_k\mathbf{q}\right)^{\mathrm{T}}. \tag{6.16}$$

After the solution to the RiSVD problem, the approximate decomposition result $\hat{h}(x)$ is obtained by:

$$\hat{h}_k = h_k - \mathbf{p}^{\mathrm{T}}\mathbf{R}_k\mathbf{q}\tau, \quad k = 1, 2, \ldots, MN. \tag{6.17}$$

Although there is no exact algorithm for the RiSVD problem, there is a heuristic solution that is referred to as the inverse iteration algorithm [21, 24]. The algorithm is described as follows.

Inverse Iteration Algorithm for the RiSVD Formulation [21, 24]

(1) Perform the QR decomposition of the matrix $\mathbf{R}$, i.e., $[\mathbf{C}_1\ \mathbf{C}_2]\begin{bmatrix}\mathbf{S}\\ \mathbf{0}\end{bmatrix} = \mathbf{R}$.

(2) Perform the SVD of the matrix $\mathbf{R}$, and obtain the smallest singular value $\tau^{(0)}$ and the associated singular vectors $\mathbf{p}^{(0)}$ and $\mathbf{q}^{(0)}$.

(3) Compute $\gamma^{(0)} = ((\mathbf{q}^{(0)})^{\mathrm{T}}\mathbf{D}_{\mathbf{p}^{(0)}}\mathbf{q}^{(0)})^{\frac{1}{4}}$.

(4) Normalize $\mathbf{p}^{(0)} = \frac{\mathbf{p}^{(0)}}{\gamma^{(0)}}$, $\mathbf{q}^{(0)} = \frac{\mathbf{q}^{(0)}}{\gamma^{(0)}}$.

(5) In the k-th iteration:

(6) Compute $\mathbf{z}^{(k)} = (\mathbf{S}^{-1})^{\mathrm{T}}\mathbf{D}_{\mathbf{p}^{(k-1)}}\mathbf{q}^{(k-1)}\tau^{(k-1)}$.

(7) Compute $\mathbf{w}^{(k)} = -(\mathbf{C}_2^{\mathrm{T}}\mathbf{D}_{\mathbf{q}^{(k-1)}}\mathbf{C}_2)^{-1}(\mathbf{C}_2^{\mathrm{T}}\mathbf{D}_{\mathbf{q}^{(k-1)}}\mathbf{C}_1)\mathbf{z}^{(k)}$.

(8) Compute $\mathbf{p}^{(k)} = \mathbf{C}_1\mathbf{z}^{(k)} + \mathbf{C}_2\mathbf{w}^{(k)}$.

(9) Compute $\mathbf{q}^{(k)} = \mathbf{S}^{-1}\mathbf{C}_1^{\mathrm{T}}\mathbf{D}_{\mathbf{q}^{(k-1)}}\mathbf{p}^{(k)}$.

(10) Normalize $\mathbf{q}^{(k)} = \frac{\mathbf{q}^{(k)}}{\|\mathbf{q}^{(k)}\|_2}$.

(11) Compute $\gamma^{(k)} = ((\mathbf{q}^{(k)})^{\mathrm{T}}\mathbf{D}_{\mathbf{p}^{(k)}}\mathbf{q}^{(k)})^{\frac{1}{4}}$.

(12) Renormalize $\mathbf{p}^{(k)} = \frac{\mathbf{p}^{(k)}}{\gamma^{(k)}}$, $\mathbf{q}^{(k)} = \frac{\mathbf{q}^{(k)}}{\gamma^{(k)}}$.

(13) Compute $\tau^{(k)} = (\mathbf{p}^{(k)})^{\mathrm{T}}\mathbf{R}\mathbf{q}^{(k)}$.

(14) Continue until the associated Ruppert matrix is numerically rank deficient or k attains the limit on iteration steps.

(15) Obtain the final approximate decomposition result $\hat{h}(x)$ from (6.17).

STLS Relaxation

This algorithm is based on the algorithms in [22, 26]. It relaxes the constraint of rank deficiency of the Ruppert matrix into a penalty term in the cost function. The Ruppert matrix $\hat{\mathbf{R}}$ in (6.11) is rank deficient if and only if one column is a linear combination of the other columns. By switching two columns, we can always make the last column linearly dependent on other columns. Then the problem of (6.10) is equivalent to

$$\min_{\hat{\mathbf{h}},\mathbf{y}} \quad \|\hat{\mathbf{h}} - \mathbf{h}\|_2, \tag{6.18}$$

$$\text{s. t.} \quad \hat{\mathbf{R}} \begin{bmatrix} \mathbf{y} \\ -1 \end{bmatrix} = \mathbf{0}, \tag{6.19}$$

where the vector $[\mathbf{y}^{\mathrm{T}}, -1]^{\mathrm{T}}$ is in the null space of the matrix $\hat{\mathbf{R}}$. The constraint in (6.19) is nonlinear with respect to the pair $(\hat{\mathbf{h}}, \mathbf{y})$, since the matrix $\hat{\mathbf{R}}$ depends on the coefficients of $\hat{\mathbf{h}}$. A relaxation of (6.18) is [22, 26]

$$\min_{\hat{\mathbf{h}},\mathbf{y}} \quad C(\hat{\mathbf{h}}, \mathbf{y}) \triangleq \|\hat{\mathbf{h}} - \mathbf{h}\|_2^2 + \lambda \left\| \hat{\mathbf{R}} \begin{bmatrix} \mathbf{y} \\ -1 \end{bmatrix} \right\|_2^2, \tag{6.20}$$

where the parameter λ balances the deviation of polynomial $\hat{h}(x)$ from $h(x)$ and the energy of the residue vector $\hat{\mathbf{R}}[\mathbf{y}^{\mathrm{T}}, -1]^{\mathrm{T}}$.

The choice of λ is important for the quality of the relaxation problem (6.20). On one hand, if λ is large, the residue vector has low energy, but the polynomial $\hat{h}(x)$ may not be a close approximation to $h(x)$. On the other hand, if λ is small, the polynomial $\hat{h}(x)$ may be near the original $h(x)$, but the rank deficiency of the Ruppert matrix may be significantly violated.

The direct solution to the relaxation problem (6.20) is not obvious, so we would like to introduce an iterative method with linear approximation. Suppose in the i-th iteration step, we already have a polynomial $\hat{\mathbf{h}}^{(i)}(x)$ (which is not necessarily decomposable) and an associated vector $\mathbf{y}^{(i)}$, and we would like to make small adjustments to them in order to decrease the cost function $C(\hat{\mathbf{h}}, \mathbf{y})$ as defined in (6.20). If we denote the adjustments as $\boldsymbol{\Delta}\hat{\mathbf{h}}^{(i)}$ and $\boldsymbol{\Delta}\mathbf{y}^{(i)}$, then we can perform the following linearization:

$$
\begin{aligned}
& C(\hat{\mathbf{h}}^{(i)} + \boldsymbol{\Delta}\hat{\mathbf{h}}^{(i)}, \mathbf{y}^{(i)} + \boldsymbol{\Delta}\mathbf{y}^{(i)}) \\
= \; & \left\| \left(\hat{\mathbf{h}}^{(i)} - \mathbf{h} \right) + \boldsymbol{\Delta}\hat{\mathbf{h}}^{(i)} \right\|_2^2 + \lambda \left\| \left(\sum_{k=1}^{MN} \hat{h}_k^{(i)} \mathbf{R}_k + \sum_{k=1}^{MN} \Delta\hat{h}_k^{(i)} \mathbf{R}_k \right) \cdot \begin{bmatrix} \mathbf{y}^{(i)} + \boldsymbol{\Delta}\mathbf{y}^{(i)} \\ -1 \end{bmatrix} \right\|_2^2 \\
\approx \; & \left\| \left(\hat{\mathbf{h}}^{(i)} - \mathbf{h} \right) + \boldsymbol{\Delta}\hat{\mathbf{h}}^{(i)} \right\|_2^2 + \lambda \left\| \hat{\mathbf{R}}^{(i)} \cdot \begin{bmatrix} \mathbf{y}^{(i)} \\ -1 \end{bmatrix} + \mathbf{J}^{(i)} \cdot \begin{bmatrix} \boldsymbol{\Delta}\hat{\mathbf{h}}^{(i)} \\ \boldsymbol{\Delta}\mathbf{y}^{(i)} \end{bmatrix} \right\|_2^2 . \qquad (6.21)
\end{aligned}
$$

In the linearization above, we neglect the second order term $\left(\sum_{k=1}^{MN} \Delta\hat{h}_k^{(i)} \mathbf{R}_k \right) \left[(\boldsymbol{\Delta}\mathbf{y}^{(i)})^{\mathrm{T}}, 0 \right]^{\mathrm{T}}$. The matrix $\mathbf{J}^{(i)}$ is

$$
\mathbf{J}^{(i)} = \left[\mathbf{j}_{MN}^{(i)}, \; \mathbf{j}_{MN-1}^{(i)}, \; \cdots, \; \mathbf{j}_1^{(i)}, \; \mathbf{B}^{(i)} \right], \qquad (6.22)
$$

in which the matrix $\mathbf{B}^{(i)}$ consists of the columns of $\hat{\mathbf{R}}^{(i)}$ except for the last column, and the vectors $\mathbf{j}_k^{(i)} = \mathbf{R}_k \cdot \begin{bmatrix} \mathbf{y}^{(i)} \\ -1 \end{bmatrix}$, $(1 \le k \le MN)$.

After the linear approximation, (6.21) becomes a positive semi-definite quadratic form with respect to the vectors $\boldsymbol{\Delta}\hat{\mathbf{h}}^{(i)}$, $\boldsymbol{\Delta}\mathbf{y}^{(i)}$, so the minimum point could be directly obtained. The optimal solution for $\boldsymbol{\Delta}\hat{\mathbf{h}}^{(i)}$ and $\boldsymbol{\Delta}\mathbf{y}^{(i)}$ is

$$
\begin{bmatrix} \boldsymbol{\Delta}\hat{\mathbf{h}}^{(i)} \\ \boldsymbol{\Delta}\mathbf{y}^{(i)} \end{bmatrix} = - \left(\boldsymbol{\Psi}^{(i)} \right)^{-1} \mathbf{m}^{(i)}, \qquad (6.23)
$$

where

$$\mathbf{\Psi}^{(i)} = \lambda(\mathbf{J}^{(i)})^{\mathrm{T}}\mathbf{J}^{(i)} + \mathbf{\Theta}^{\mathrm{T}}\mathbf{\Theta}, \tag{6.24}$$

$$\mathbf{m}^{(i)} = \lambda(\mathbf{J}^{(i)})^{\mathrm{T}} \cdot \hat{\mathbf{R}}^{(i)} \cdot \begin{bmatrix} \mathbf{y}^{(i)} \\ -1 \end{bmatrix} + \mathbf{\Theta}^{\mathrm{T}} \left(\hat{\mathbf{h}}^{(i)} - \mathbf{h} \right), \tag{6.25}$$

where the matrix $\mathbf{\Theta}$ is

$$\mathbf{\Theta} = [\mathbf{I}_{MN \times MN}\ \mathbf{0}]. \tag{6.26}$$

In summary, the STLS relaxation algorithm can be stated as follows:

STLS Relaxation Algorithm

(1) Obtain the column of $\mathbf{R}$ that has the minimum residue error when mean-square approximated as a linear combination of the other columns, and determine the coefficients in the linear combination of the other columns as $\mathbf{y}^{(0)}$.
Let $\hat{\mathbf{h}}^{(0)} = \mathbf{h}$.
Choose $\lambda = 1/\sigma^2_{\mathbf{K},\min}$, i.e., the inverse of the squared minimum singular value of the matrix $\mathbf{K} = \left[\mathbf{j}^{(0)}_{MN},\ \mathbf{j}^{(0)}_{MN-1},\ \ldots,\ \mathbf{j}^{(0)}_{1}\right]$.

(2) In the i-th iteration:

(3) Compute $\mathbf{J}^{(i)}$ in (6.22) and the Ruppert matrix $\hat{\mathbf{R}}^{(i)}$ associated with $\hat{\mathbf{h}}^{(i)}$.

(4) Obtain the adjustments $\mathbf{\Delta}\hat{\mathbf{h}}^{(i)}$ and $\mathbf{\Delta}\mathbf{y}^{(i)}$ in (6.23).

(5) Update $\hat{\mathbf{h}}^{(i+1)} = \hat{\mathbf{h}}^{(i)} + \mathbf{\Delta}\hat{\mathbf{h}}^{(i)}$, $\mathbf{y}^{(i+1)} = \mathbf{y}^{(i)} + \mathbf{\Delta}\mathbf{y}^{(i)}$.

(6) Continue until the Ruppert matrix $\hat{\mathbf{R}}^{(i)}$ is numerically rank deficient or i attains the limit on iteration steps.

6.2 Problem 4: Approximate Decomposition with Roots as Input

This section proposes an algorithm for approximate decomposition, where the input is the roots of an indecomposable polynomial that is close to a decomposable one. This algorithm can be regarded as an extension of the root-grouping algorithm for

exact decomposition. Since the polynomial is indecomposable, the *groups* of roots are actually not fully defined and not unique. However, if we view the roots of an indecomposable polynomial as the roots of a decomposable polynomial with some perturbation, then we may use the grouping information of the nearby decomposable polynomial for decomposition.

The algorithm consists of iterations, in each of which there are three phases: the first phase is to determine the grouping information using mixed integer programming; the second phase is to determine whether to terminate the iteration by obtaining a decomposable polynomial and measuring its approximation quality to the input polynomial from the perspective of roots; the third phase is to make small adjustments to the roots of an indecomposable polynomial to approach a decomposable polynomial. Only the third phase updates the roots of the indecomposable polynomial; the roots of a decomposable polynomial obtained in the second phase of an iteration are for the only purpose of determining whether the termination criterion has been met, and they are not used in the third phase or any future iterations; the reason for this is explained in the description of the second phase.

In the first phase of each iteration, the grouping information is obtained with a formulation of a mixed integer program, which is similar to that in the root-grouping algorithm for exact decomposition. Since the roots are of an indecomposable polynomial, the equality of the power summations (5.9) does not hold and can no longer be directly applied to determine the grouping information; however, if the perturbation is small, there should not be significant differences among the power summations of the roots $s_j(A_i)$ in each group $1 \leq i \leq M$ for a given order $1 \leq j \leq N-1$, where $s_j(A_i)$ is defined in (5.5). As a result, we consider the most likely grouping pattern to be the one that minimizes the total difference among the power summations of the roots in each group. In particular, we formulate a mixed integer program as follows:

$$\min_{\delta_{k,i},\hat{s}_j,\hat{\varepsilon}_{i,j}} \quad \sum_{i=1}^{M}\sum_{j=1}^{N-1} |\hat{\varepsilon}_{i,j}| \tag{6.27}$$

$$\text{s. t.} \quad \hat{\varepsilon}_{i,j} = \sum_{k=1}^{MN} \delta_{k,i} \cdot (z_h(k))^j - \hat{s}_j, \text{ for } 1 \le i \le M,\ 1 \le j \le N-1, \tag{6.28}$$

$$\sum_{k=1}^{MN} \delta_{k,i} = N, \quad \text{for } 1 \le i \le M, \tag{6.29}$$

$$\sum_{i=1}^{M} \delta_{k,i} = 1, \quad \text{for } 1 \le k \le MN, \tag{6.30}$$

$$\delta_{k,i} \in \{0,1\}, \quad \text{for } 1 \le k \le MN,\ 1 \le i \le M.$$

This optimization problem has the following variables: binary variables $\delta_{k,i}$ that indicate whether the root $z_h(k)$ belongs to the group A_i, continuous variables $\hat{s}_j$ that are the *standard root power summation* of order j, and continuous variables $\hat{\varepsilon}_{i,j}$ that are the deviation from the root power summation of order j in the i-th group to the corresponding standard root power summation $\hat{s}_j$. Since the roots $z_h(k)$ are mostly complex numbers, the deviations $\hat{\varepsilon}_{i,j}$ are respectively implemented for the real part and the imaginary part, and each term $|\hat{\varepsilon}_{i,j}|$ in the cost function (6.27) is implemented as $|\hat{\varepsilon}_{i,j}| \triangleq |\mathrm{Re}\{\hat{\varepsilon}_{i,j}\}| + |\mathrm{Im}\{\hat{\varepsilon}_{i,j}\}|$. For decomposable polynomials with real coefficients, the standard root power summations $\hat{s}_j$ always have real values due to Newton's identities [28]; thus, in our implementation, we constrain that $\hat{s}_j$ are all real. In this way, the formulation above is in the framework of mixed integer optimization. The constraints (6.29) and (6.30) ensure that each group has N roots z_h and that each z_h belongs to one group, respectively. The cost function is the total difference between the power summations in each group and the corresponding standard power summations.

The formulation above can obtain the correct grouping pattern if the perturbation is small. If there is no perturbation, the formulation in (6.27) obtains the correct grouping information with a minimum cost of zero, since all the differences in power summations are zero due to Theorem 5.3. If the perturbation gradually increases from

zero but is sufficiently small, the differences of power summations $\hat{\varepsilon}_{i,j}$ also gradually increase, and the correct grouping information $\delta_{k,i}$ still has lower cost to the problem (6.27) than any incorrect grouping patterns; thus, the solution to (6.27) leads to the true grouping pattern if the perturbation is sufficiently small. This conclusion does not hold for the large perturbation scenario, where an incorrect grouping pattern may achieve a lower cost than the correct one.

The formulation (6.27) for the approximate decomposition algorithm has higher complexity than the formulation (5.8) for the exact decomposition algorithm. In the exact decomposition algorithm, the grouping process can be divided into multi-steps, each of which solves a mixed integer program and determines only one new group of roots using Theorem 5.3. However, for the approximate decomposition algorithm, all the groups are determined simultaneously in one step rather than multi-stages, since Theorem 5.3 does not hold for the approximate case and the cost function in (6.27) is globally dependent on all the groups. As a result, the dimension and the complexity of the optimization problem increase for approximate decomposition.

In the second phase of each iteration, we obtain the roots of a decomposable polynomial and determine whether to terminate the iteration. With the current roots z_h which typically do not correspond to a decomposable polynomial, we first utilize the linear program in (5.13) to determine the roots z_f and the coefficients of $g(x)$, and then we construct $f(x)$ from z_f. Last, by solving (3.7) with the obtained z_f and $g(x)$, we reconstruct the $\widehat{z_h}$ as the roots of $(f \circ g)(x)$, which are guaranteed to correspond to a decomposable polynomial. When the total deviation from the perturbed input roots to the roots $\widehat{z_h}$ is sufficiently small, we consider the algorithm as convergent and terminate the iterations; if not, we continue to the third phase and the next iteration. We want to clarify that the roots $\widehat{z_h}$ in an iteration are not used in the following third phase or in future iterations: although $\widehat{z_h}$ correspond to a decomposable polynomial, however, $\widehat{z_h}$ may be significantly deviated from the input roots due to the way they are obtained; in contrast, if we make small adjustments on the roots z_h of the indecomposable polynomial (in the following third phase) and obtain a new set of $\widehat{z_h}$ from the adjusted roots (in the second phase of the next

iteration), it is possible that the new set of $\widehat{z}_h$ is closer to the input roots than the previous set. Since we want to find a decomposable polynomial whose roots are good approximation to the input roots, we do not use the roots $\widehat{z}_h$ as the starting point of the adjustment in the third phase. As a result, we only update the roots z_h in the third phase, although z_h typically do not correspond to a decomposable polynomial.

In the third phase of each iteration, we adjust the roots z_h (not the roots $\widehat{z}_h$ that are obtained in the second phase) with Taylor approximation to approach a decomposable polynomial. We aim to approximately minimize the total difference of power summations with proper adjustment of the roots. Specifically, if the roots $z_h(k)$ are slightly adjusted by $\Delta z_h(k)$, then the j-th power is adjusted by $j\,(z_h(k))^{j-1}\cdot\Delta z_h(k)$ to the first-order Taylor approximation. As a result, a linear program is formulated as follows.

$$\min_{\Delta z_h(k),\tilde{s}_j,\tilde{\varepsilon}_{i,j}} \quad \sum_{i=1}^{M}\sum_{j=1}^{N-1}|\tilde{\varepsilon}_{i,j}| + W\cdot\sum_{k=1}^{MN}|\Delta z_h(k)| \tag{6.31}$$

$$\text{s. t.} \quad \tilde{\varepsilon}_{i,j} = \sum_{k\in A_i}^{MN}\Big((z_h(k))^j + j\,(z_h(k))^{j-1}\cdot\Delta z_h(k)\Big) - \tilde{s}_j, \quad \text{for } 1\le i\le M,\ 1\le j\le N-1. \tag{6.32}$$

Similar with the first phase, $\tilde{s}_j$ and $\tilde{\varepsilon}_{i,j}$ in (6.32) represent the standard power summations and the approximate deviations in power summations, respectively; in addition, $\tilde{s}_j$ are constrained to have real values. The same as (6.28) in the first phase, $\tilde{\varepsilon}_{i,j}$ in the constraint (6.32) are implemented for the real part and imaginary part, respectively, and in the cost function $|\tilde{\varepsilon}_{i,j}| \triangleq |\mathrm{Re}\{\tilde{\varepsilon}_{i,j}\}| + |\mathrm{Im}\{\tilde{\varepsilon}_{i,j}\}|$. The groups A_i in (6.32) are obtained from the results in the first phase, i.e., $A_i = \{k|\delta_{k,i} = 1\}$. In the cost function, W is a weight factor to balance between achieving a small total difference of power summations and lowering down the adjustments of the roots. Since $\Delta z_h(k)$ are probably complex numbers, we implement $|\Delta z_h(k)| \triangleq |\mathrm{Re}\{\Delta z_h(k)\}| + |\mathrm{Im}\{\Delta z_h(k)\}|$. In addition, we have two details for implementation: first, we constrain that the adjustments of conjugate roots are also conjugate to ensure that $h(x)$ is a real-coefficient polynomial after the adjustments of the roots; second, we set a upper bound for the

real and imaginary parts of each $\Delta z_h(k)$ to avoid huge adjustments which may decrease the precision of the Taylor approximation. Since Taylor expansion is used in (6.32), the adjusted roots from linear program in the third phase typically do not correspond to a decomposable polynomial.

An iteration with the three phases above may approach a decomposable polynomial. As we mentioned in the second phase, when the total deviation from the perturbed input roots to the roots $\widehat{z_h}$ of the decomposable polynomial in the iteration is sufficiently small, we consider the algorithm as convergent. However, we do not have a guarantee for convergence or global optimality. The approximate decomposition algorithm with roots as input is summarized as follows.

Approximate Root-Grouping Algorithm

(1) In the i-th iteration:

(2) Phase 1: Solve the mixed integer optimization (6.27).

(3) Phase 2: Solve (5.13) and obtain the coefficients $g^{(i)}(x)$ and the roots $z_f^{(i)}(k)$, $(1 \leq k \leq M)$. Then obtain $f^{(i)}(x) = \prod_{k=1}^{M} \left(x - z_f^{(i)}(k)\right)$. Solve (3.7) to obtain the roots of the decomposable polynomial $(f^{(i)} \circ g^{(i)})(x)$, denoted as $\widehat{z_h}^{(i)}(k)$ $(1 \leq k \leq MN)$.

(4) Phase 3: Solve the linear optimization (6.31) to get $\Delta z_h^{(i)}(k)$. Adjust the roots $z_h^{(i)}(k) = z_h^{(i-1)}(k) + \Delta z_h^{(i)}(k)$, for $1 \leq k \leq MN$.

(5) Continue until the total deviation from the perturbed input roots to the roots $\widehat{z_h}^{(i)}(k)$ obtained in step (3) is sufficient small, or i attains the limit on iteration steps.

6.3 Evaluation of the Approximate Decomposition Algorithms

This section presents the evaluation of the four algorithms for approximate decomposition. In the simulation, we vary the degrees of $f(x)$ and $g(x)$; for each degree

pair (M, N), 100 samples of $f(x)$ and $g(x)$ are generated, respectively, and the composed polynomial $h(x)$ and its roots z_h are obtained. The coefficients of $f(x)$ and $g(x)$ are generated from i.i.d. standard Gaussian distribution except for the leading coefficients which are fixed to one. Then, we utilize i.i.d. Gaussian noise to make the polynomial indecomposable: For the first three algorithms that work on coefficients, noise is added to the coefficients of $h(x)$. For the last algorithm, Gaussian perturbation is added to the roots z_h for the real part and imaginary part, respectively; the perturbation on a conjugate pair of roots is also in a conjugate pair, and the perturbation on real roots is real, in order to ensure the perturbed polynomial still has real coefficients. The *signal* to noise ratio is at 40dB, where the energy of *signal* is the total energy of the coefficients for the first three algorithms or the total energy of the roots for the last algorithm. For clarity, the generated decomposable polynomial without noise is denoted as $h^{cl}(x)$ with roots z_h^{cl}, while the noisy polynomial and its roots are denoted as $h^{in}(x)$ and z_h^{in}, respectively.

Since it is not known how to determine the decomposable polynomial that is the closest to an indecomposable one, the criterion for successful approximate decomposition is not obvious and may vary due to different structures of the algorithms. As a result, we present the results for each algorithm separately.

6.3.1 Iterative Mean Square Algorithm

This algorithm [18] works with coefficients and updates $f(x)$ as well as $g(x)$ directly. Thus, the output of each iteration is guaranteed a decomposable polynomial $h^{(k)}(x) = (f^{(k)} \circ g^{(k)})(x)$, where $f^{(k)}(x)$ and $g^{(k)}(x)$ are the results in the k-th iteration. The criterion for success is that the decomposable polynomial in the k-th iteration is closer to the input noisy polynomial than the initially generated noiseless polynomial, i.e., the energy of current deviation $\|h^{(k)}(x) - h^{in}(x)\|_2^2$ in the k-th iteration is lower than the original additional noise energy $\|h^{cl}(x) - h^{in}(x)\|_2^2$ in the data generation process. A sample is considered unsuccessful if the criterion above has not been satisfied after 100 iterations. The percentage of the successful samples for each degree pair is shown in Table 6.1.

Table 6.1: Success Rate of the Iterative Mean Square Algorithm (%)

$\deg(f)$	$\deg(g)$	$\deg(f \circ g)$	Success Rate
2	2	4	100.0
3	3	9	100.0
3	4	12	98.0
4	4	16	94.0
4	5	20	84.0
5	5	25	67.0
5	6	30	37.0
6	6	36	26.0
6	7	42	12.0
7	7	49	12.0

The iterative mean square algorithm achieves success in all but 2 samples with degrees below 12 in our simulation. For higher degrees, it has unsuccessful samples, either because the algorithm diverges or it converges to a local minimum. In conclusion, the iterative mean square algorithm is a practical and efficient approach if the input polynomial is close to a decomposable polynomial and its degree is not high, although there is no guarantee of the convergence to the global minimum.

6.3.2 RiSVD Heuristic and STLS Relaxation

The goal of both RiSVD and STLS relaxation algorithms is to determine a rank deficient Ruppert matrix that is close to the full rank initial Ruppert matrix. Thus, these algorithms are considered successful when the Ruppert matrix is numerically rank deficient, which is determined by the singular values in our simulation. In particular, a Ruppert matrix is considered rank deficient if the maximum gap between consecutive singular values among the smallest 20 singular values is larger than 100 times that of the initial Ruppert matrix or larger than 10^4. A sample is considered unsuccessful if the criterion above has not been satisfied after 100 iterations. The success rates of the RiSVD and STLS algorithms are listed in Table 6.2.

In Table 6.2, the success rate is the ratio between total successful samples and the number of samples where the initial Ruppert matrix is not numerically rank deficient

Table 6.2: Success Rate of the Approximate Decomposition Methods that are Based on Ruppert Matrix (%)

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

(i.e., the maximum gap between consecutive singular values among the smallest 20 singular values is smaller than 10^4 for the initial matrix). The success rate of the STLS relaxation algorithm is higher than that of the RiSVD algorithm, which shows the STLS algorithm performs better in generating numerically rank deficient Ruppert matrices.

However, complications involving numerical accuracy are encountered for these two algorithms. In contrast to the iterative mean square method [18] and the approximate root-grouping decomposition method where the polynomial in each iteration is guaranteed decomposable, the polynomial corresponding to the Ruppert matrix in each iteration of RiSVD and SLTS algorithms is generally indecomposable. Even if both RiSVD and STLS converge under our criterion of numerical rank deficiency, the output polynomials of the RiSVD or STLS algorithms are possibly unable to be faithfully decomposed with the coefficient-unraveling algorithm for exact decomposition [17]. In addition, the polynomials obtained by RiSVD may have better performance than those by STLS, when they are decomposed by the coefficient-unraveling algorithm. This implies that our criterion for rank deficiency is not necessarily compatible with the coefficient-unraveling algorithm. Although rank deficiency of Rup-

pert matrix is theoretically equivalent to decomposability of polynomial, the large dimensions of Ruppert matrices may cause numerical errors that lead to imprecision in the singular value calculation in MATLAB, so our rank deficient criterion may not be determined precisely.

6.3.3 Approximate Root-Grouping Algorithm

This section presents the simulation results for the approximate root-grouping algorithm. In the data generation process, perturbation with 40dB SNR is added to the roots of decomposable polynomials; the perturbation on a conjugate pair of roots is also in a conjugate pair, and the perturbation on real roots is real, so the perturbed polynomial still has real coefficients. The weight factor in (6.31) is chosen as $W = 10^{-2}$, and we set an upper bound of 10^{-3} for the real and imaginary parts of each adjustment $\Delta z_h(k)$ in the third phase of each iteration. As we discussed in Section 6.2, the output $\widehat{z_h}$ in the second phase of each iteration is guaranteed to correspond to a decomposable polynomial. Consequently, we choose the criterion for success as that the roots $\widehat{z_h}$ of the decomposable polynomial in the k-th iteration are closer to the perturbed input roots than the roots of the initially generated decomposable polynomial are; in other words, we find a better decomposable approximation to the input polynomial than the original one in the data generation process, from the perspective of the energy of the differences of roots. A sample is considered unsuccessful if the criterion above has not been satisfied after 100 iterations. The successful percentage for each degree pair is shown in Table 6.3.

In Table 6.3, the column of *correct grouping information* shows the results for the first phase of the algorithm, which determines the grouping information; the column of *successful decomposition* shows the percentage of the successful samples according to the criterion above. If $h(x)$ has a degree that is below 20, the approximate root-grouping algorithm in our simulation achieves considerable success. All grouping information is correctly obtained when the degree of the polynomial $h(x)$ is below 12; as the degree increases, there are cases where the grouping information is not correctly determined; since the total number of possible grouping patterns increases with the

Table 6.3: Success Rate of the Root Grouping Algorithm for Approximate Decomposition (%)

$\deg(f)$	$\deg(g)$	$\deg(f \circ g)$	Correct Grouping Information	Successful Decomposition
2	2	4	100.0	100.0
3	3	9	100.0	99.0
3	4	12	100.0	96.0
4	3	12	100.0	98.0
4	4	16	97.0	96.0
4	5	20	98.0	92.0
5	4	20	100.0	97.0
5	5	25	95.0	91.0
3	9	27	78.0	76.0
9	3	27	52.0	52.0
5	6	30	83.0	85.0
6	5	30	83.0	74.0

degree, the possibility that an incorrect grouping pattern achieves lower cost in (6.27) than the correct one also increases. In general, the algorithm as a whole works for most samples in our simulation; for those unsuccessful samples, the algorithm may diverge or converge to a local minimum.

As a result, the approximate root-grouping algorithm is a practical and efficient approach if the input roots are close to those of a decomposable polynomial and the degree of the polynomial is not high; however, we cannot guarantee its convergence to the global minimum.

Chapter 7

Conclusions and Future Work

This thesis studies the sensitivities of polynomial composition and decomposition in order to characterize their robustness with respect to perturbations in coefficients and roots. It also presents algorithms for both exact and approximate polynomial decomposition. Since both the coefficients and the roots of decomposable polynomials are potentially useful in signal processing applications, we explore polynomial decomposition with inputs of both coefficients and roots.

For sensitivity analysis, we have derived the expressions and developed bounds for the sensitivities. An empirical comparison shows that composition and decomposition using the root triplet $(z_f,\ g,\ z_h)$ is likely to be more robust than using the coefficient triplet $(f,\ g,\ h)$, when the degrees of polynomials are sufficiently high. Simulation results demonstrate that the sensitivities $S_{\hat{f}\to h}$ and $S_{h\to\hat{f}}$ can be significantly reduced by utilizing equivalent compositions with first-degree polynomials; in addition, our heuristic rule for parameter selection is shown to be efficient in approaching the minimum values for these sensitivities.

Three algorithms are presented for the exact decomposition problem, in which the polynomial $h(x)$ is ensured to be decomposable into polynomials with specified degrees. These algorithms all work in theory but have different numerical performances. Simulation results show that the algorithms with roots as input are more robust to numerical errors and can decompose polynomials with much higher degrees than the algorithm with coefficients as input. Specifically, the root-power-summation

algorithm has the highest successful decomposition rate; the root-grouping algorithm has a step of mixed integer programming, which may have high complexity but is empirically shown much more efficient than general integer programming problems; the coefficient-unraveling algorithm does not use all the coefficients of the input polynomial in the step to get $g(x)$ and easily accumulates numerical error due to its structure.

Four algorithms are shown for the approximate decomposition problem, for which we want to approximate $h(x)$ with a decomposable polynomial. Three algorithms work with coefficients: one is an iterative mean square method, and the other two are based on obtaining a rank deficient Ruppert matrix that approximates that of the indecomposable polynomial. The fourth algorithm has roots as input. Although each algorithm may be effective for certain polynomials, none of these algorithms is guaranteed to converge in general settings. The iterative mean square method is a practical and efficient algorithm if the input polynomial is near a decomposable one, but it may converge into a local minimum. The algorithms based on the Ruppert matrix may obtain a numerically rank deficient Ruppert matrix, but they encounter numerical problems in computation with the high-dimension Ruppert matrix and in the determination of the rank, so the output polynomials of these algorithms are possibly unable to be faithfully decomposed with the coefficient-unraveling algorithm for exact decomposition; in addition, the choice for an parameter in SLTS algorithm is not clear. The approximate root-grouping algorithm is effective when the input roots are near those of a decomposable polynomial, but it may also converge to a local minimum and the optimal values of parameters in this algorithm are not obvious.

Future work would mainly focus on further development and improvement of the approximate decomposition algorithms. For these existing algorithms, the convergence criteria may be derived to understand the conditions under which the algorithms converge to the globally optimal decomposable approximation. In addition, accurate numerical methods to determine the rank of a high-dimension matrix may improve the termination criteria of the RiSVD and the STLS algorithms as well as enable these algorithms to work on polynomials with higher degrees. With potential

improvements on numerical accuracy, efficient criteria for comparison among algorithms also need to be developed and verified.

New algorithms could also be developed with potentially deeper exploration in theory or practice. As an example in theory, the RiSVD and STLS algorithms are both for the general STLS problem but do not make full use of the special structure of the Ruppert matrix; thus, further study on the structure of the Ruppert matrix may lead to invention of algorithms with higher efficiency. In practice, exploration of the properties of the signals to be approximately decomposed may constrain the range of the problem and result in more specific but more efficient algorithms.

In addition to algorithms, tighter lower bounds may be developed on the distance from an indecomposable polynomial to its nearest decomposable approximation, which may serve as a fundamental limit and be used to evaluate the room for improvement of approximate decomposition algorithms.

Appendix A

Minimum Phase Decomposition for a Minimum Phase Decomposable Polynomial

In this appendix, we proof the statement in Item 6 in Section 2.1: if a decomposable $h(x)$ is minimum phase, then there always *exists* a non-trivial minimum phase decomposition. In other words, if we know that a minimum phase $h(x)$ has the composition

$$h(x) = (f \circ g)(x), \tag{A.1}$$

where $\deg(f(x)) = M$ and $\deg(g(x)) = N$ ($M > 1$ and $N > 1$), then we can construct an equivalent composition of $h(x)$ into minimum phase components $\hat{f}(x)$ and $\hat{g}(x)$ with degrees M and N, respectively. Here *minimum phase polynomials* refer to the polynomials the roots of which are all inside the unit circle. We assume $f(x)$, $g(x)$ and $h(x)$ are all real polynomials, and we require that both $\hat{f}(x)$ and $\hat{g}(x)$ have only real coefficients.

Since minimum phase depends on the radius of the roots, we first study the structure of the roots of a decomposable polynomial.[1] If we denote the roots of $f(x)$

[1]Similar discussion about the structure of roots of a decomposable polynomial is also presented in Section 4.1.2.

and $h(x)$ as $z_f(i)$ $(1 \leq i \leq M)$ and $z_h(k)$ $(1 \leq k \leq MN)$, respectively, then there are two ways to factor the composed polynomial $h(x)$:

$$h(x) = a_M \prod_{i=1}^{M} (g(x) - z_f(i)) = c_{MN} \prod_{k=1}^{MN} (x - z_h(k)) . \tag{A.2}$$

where a_M and c_{MN} are the coefficients of the highest degree term in $f(x)$ and $h(x)$, respectively. Denote

$$g_1(x) \triangleq g(x) - z_f(1), \tag{A.3}$$

then (A.2) implies that all the roots of $g_1(x)$ are included in the roots of $h(x)$. Since $h(x)$ is minimum phase, all its roots are in the unit circle, so all the roots of $g_1(x)$ are in the unit circle. If $z_f(1)$ is a real number, then $g_1(x)$ is a real-coefficient polynomial; otherwise, $g_1(x)$ has a complex constant term. We consider the following two cases, which depend on whether $f(x)$ has at least a real root.

Case 1: $f(x)$ has at least a real root.

Without loss of generality, we can assume $z_f(1)$ is a real root of $f(x)$. Then (A.2) becomes

$$h(x) = a_M \prod_{i=1}^{M} (g_1(x) - (z_f(i) - z_f(1))) = (f_1 \circ g_1)(x), \tag{A.4}$$

where

$$f_1(x) = a_M \prod_{i=1}^{M} (x - (z_f(i) - z_f(1))) . \tag{A.5}$$

The complex roots among $z_f(i)$ are in conjugate pairs since $f(x)$ is real; in addition, since $z_f(1)$ is real, we know the complex roots in (A.5) are also in conjugate pairs, so $f_1(x)$ is a real polynomial.

The polynomial $f_1(x)$ is not necessarily minimum phase; however, we can construct a minimum phase polynomial $\hat{f}(x)$ by scaling the roots:

$$\hat{f}(x) = a_M \cdot \theta_1^M \cdot \prod_{i=1}^{M} \left(x - \frac{z_f(i) - z_f(1)}{\theta_1} \right), \tag{A.6}$$

where

$$\theta_1 = 2 \cdot \max_{1 \le i \le N} |z_f(i) - z_f(1)|. \tag{A.7}$$

The roots of $\hat{f}(x)$ are $z_{\hat{f}}(i) = \frac{z_f(i) - z_f(1)}{\theta_1}$, and we can verify that these roots have radius

$$|z_{\hat{f}}(i)| \le \frac{\max_{1 \le i \le N} |z_f(i) - z_f(1)|}{\theta_1} = \frac{1}{2}.$$

Thus, all the roots of $\hat{f}(x)$ are in the unit circle, and $\hat{f}(x)$ is a minimum phase polynomial. In addition, $\hat{f}(x)$ is a real polynomial since its complex roots are in conjugate pairs.

To compensate the scaling in $\hat{f}(x)$, we need to scale $g_1(x)$ into $\hat{g}(x)$ correspondingly:

$$\hat{g}(x) = \frac{1}{\theta_1} \cdot g_1(x) = \frac{1}{\theta_1} \left(g(x) - z_f(1) \right). \tag{A.8}$$

Since $\hat{g}(x)$ has the same roots as $g_1(x)$ and $g_1(x)$ is minimum phase, we know $\hat{g}(x)$ is also minimum phase. Since $g_1(x)$ is real, we know $\hat{g}(x)$ has also real coefficients.

The composition $(\hat{f} \circ \hat{g})(x)$ yields

$$\begin{aligned} (\hat{f} \circ \hat{g})(x) &= a_M \cdot \theta_1^M \cdot \prod_{i=1}^{M} \left(\hat{g}(x) - \frac{z_f(i) - z_f(1)}{\theta_1} \right) \\ &= a_M \cdot \prod_{i=1}^{M} \left(g(x) - z_f(i) \right) \\ &= h(x). \end{aligned} \tag{A.9}$$

Thus, $(\hat{f} \circ \hat{g})(x)$ is an equivalent composition of $h(x)$, and both $\hat{f}(x)$ and $\hat{g}(x)$ are minimum phase and real polynomials. This completes our proof for the first case where $f(x)$ has at least a real root.

Case 2: $f(x)$ has only complex conjugate roots.

If $f(x)$ has only complex roots in conjugate pairs, we know $g_1(x) = g(x) - z_f(1)$ and $g_1^*(x) = g(x) - z_f^*(1)$ are both complex-coefficient minimum phase polynomials. Then,

we have the following lemma:

Lemma A.1. *Let $\eta(z) = \sum_{i=0}^{N} \eta_i z^i$ be a real-coefficient polynomial with degree N. If both $(\eta(z) + j\alpha)$ and $(\eta(z) - j\alpha)$ are minimum phase polynomials for a real number $\alpha > 0$, then $\eta(z)$ is also minimum phase.*

Proof. First, we show that if $|\gamma| \geq 1 + \sum_{i=0}^{N} |\eta_i|$, then $(\eta(z) - j\gamma)$ has no root in the unit circle. This can be shown by

$$|\eta(z) - j\gamma| \geq |\gamma| - |\eta(z)| \geq |\gamma| - \sum_{i=0}^{N} |\eta_i| \geq 1 \neq 0,$$

for any complex number z in the unit circle.

Next, we show the curve $\zeta = \{u \in \mathbb{C} | u = \eta(z),\ |z| = 1\}$ has at most $2N$ intersections with the imaginary axis on the complex plane. Intuitively, the curve ζ is the image of the unit circle when it is mapped by the polynomial $\eta(z)$. For any intersection of ζ and the imaginary axis, we have $\mathrm{Re}\{\eta(z)\} = 0$ for some z on the unit circle (i.e., $|z| = 1$); this is equivalent to $\eta(z) + \eta^*(z) = 0$ for some z with $|z| = 1$. Since $\eta(z)$ has real coefficients, for any z on the unit circle, we have $\eta^*(z) = \eta(z^*) = \eta(1/z)$. Thus, the corresponding values of z of all the intersections of ζ and the imaginary axis satisfy $\eta(z) + \eta(1/z) = 0$, which can be arranged into a polynomial of z with the degree of $2N$. Thus, the number of such intersections does not exceed $2N$ (since we additionally require $|z| = 1$).

Finally, we show the relationship between the curve ζ on the complex plane and the number of roots of $(\eta(z) - j\gamma)$ that are inside the unit circle. Obviously, $(\eta(z) - j\gamma)$ has root(s) on the unit circle if and only if $j\gamma$ is on the curve ζ. Thus, as γ varies, the roots of $(\eta(z) - j\gamma)$ move continuously, and the number of roots in the unit circle changes by one each time when $j\gamma$ crosses the curve ζ (including multiplicity). Since $(\eta(z) - j\alpha)$ are minimum phase, it has all N roots in the unit circle; let γ moves from α to $1 + \max\{\alpha, \sum_{i=0}^{N} |\eta_i|\}$, then $j\gamma$ crosses the curve ζ for at least N times, since the number of roots of $(\eta(z) - j\gamma)$ that are in the unit circle decreases from N to 0. Similarly, let γ moves from $-\alpha$ to $-1 - \max\{\alpha, \sum_{i=0}^{N} |\eta_i|\}$, then $j\gamma$ crosses the curve

ζ for at least another N times. Since there are at most $2N$ intersections of ζ and the imaginary axis $j\gamma$, there is no intersection between $-\alpha$ and α. So the number of roots in the unit circle does not vary as γ changes from $-\alpha$ to α, i.e., all roots of $(\eta(z) - j\gamma)$ are in the unit circle for $-\alpha \leq \gamma \leq \alpha$. Specially, when $\gamma = 0$, $\eta(z)$ has all roots in the unit circle and thus is minimum phase.

This accomplishes the proof for Lemma A.1. □

With the lemma above, we may construct a minimum phase polynomial $g_2(x)$:

$$g_2(x) = g(x) - \mathrm{Re}\{z_f(1)\}. \tag{A.10}$$

Since $g_2(x)$ has real-coefficients, we may construct $\hat{f}(x)$ and $\hat{g}(x)$ in a way similar to case 1:

$$\hat{f}(x) = a_M \cdot \theta_2^M \cdot \prod_{i=1}^{M} \left(x - \frac{z_f(i) - \mathrm{Re}\{z_f(1)\}}{\theta_2} \right), \tag{A.11}$$

$$\hat{g}(x) = \frac{1}{\theta_2} \left(g(x) - \mathrm{Re}\{z_f(1)\} \right), \tag{A.12}$$

where the scaling factor is

$$\theta_2 = 2 \cdot \max_{1 \leq i \leq N} |z_f(i) - \mathrm{Re}\{z_f(1)\}|. \tag{A.13}$$

In similar procedures with case 1, we may verify that both $\hat{f}(x)$ and $\hat{g}(x)$ are minimum phase and real polynomials, and their composition $(\hat{f} \circ \hat{g})(x)$ yields $h(x)$. This completes our proof for the second case where $f(x)$ has only complex conjugate roots.

Combining cases 1 and 2, we have proved the statement about the existence of minimum phase decomposition for minimum phase decomposable polynomials.

Appendix B

Derivation of Upper Bound (4.13)

In this appendix, we derive the upper bound (4.13) for the sensitivity $S_{g\to h}$. First, we show the relationship between the polynomials $d(x)$ and $h(x)$. The definition of the polynomial $d(x)$ in (4.9) implies

$$\mathbf{d} = \mathbf{G} \cdot \begin{bmatrix} 0 \\ Ma_M \\ (M-1)a_{M-1} \\ \vdots \\ a_1 \end{bmatrix} = \mathbf{GVf} = \mathbf{GVG}^{\dagger}\mathbf{h}, \tag{B.1}$$

where the last step uses (5.3); $\mathbf{G}$ and $\mathbf{V}$ are defined in (4.2) and (4.15), respectively; $\mathbf{G}^{\dagger} = (\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1}\mathbf{G}^{\mathrm{T}}$ is the pseudo-inverse matrix of $\mathbf{G}$.

Next, we want to bound the energy of $\Delta h(x)$ with the energy of $\Delta g(x)$. Since (4.8) indicates $\Delta h(x)$ is approximately the convolution of $\Delta g(x)$ and $d(x)$, we want to bound the energy of the output signal with the two input signals of the convolution. In general, we have the following lemma on the energy of the signals in a convolution.

Lemma B.1. *Denote $s_3[n]$ as the convolution of the finite length signals $s_1[n]$ that is non-zero only for $0 \le n \le L_1$ and $s_2[n]$ that is non-zero only for $0 \le n \le L_2$. Assume $L_1 \ge L_2$, then the energy of these signals satisfy*

$$E_{s_3} \le (L_2 + 1)E_{s_1}E_{s_2}, \tag{B.2}$$

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 that

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

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

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

This accomplishes the proof for Lemma B.1. □

Applying Lemma B.1 to (4.8), we know

$$E_{\Delta h} \leq (N+1) E_{\Delta g} E_d, \tag{B.3}$$

where E denotes the energy of the signals. A combination of (3.3) and (B.3) shows

$$S_{g \to h} = \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot \max_{\mathbf{\Delta g}} \left(\frac{\|\mathbf{\Delta h}\|_2^2}{\|\mathbf{\Delta g}\|_2^2} \right) \leq \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot (N+1) \|\mathbf{d}\|_2^2, \tag{B.4}$$

where $\mathbf{\Delta g}$ is a sufficiently small perturbation. Incorporating (B.1) into (B.4), we have

$$S_{g \to h} \leq \frac{\|\mathbf{g}\|_2^2}{\|\mathbf{h}\|_2^2} \cdot (N+1) \|\mathbf{G V G}^\dagger \mathbf{h}\|_2^2 \leq (N+1) \|\mathbf{g}\|_2^2 \cdot \sigma_{\mathbf{T},\max}^2, \tag{B.5}$$

where $\mathbf{T} = \mathbf{G V G}^\dagger = \mathbf{G V} (\mathbf{G}^{\mathrm{T}} \mathbf{G})^{-1} \mathbf{G}^{\mathrm{T}}$ is the matrix in (4.14), and $\sigma_{\mathbf{T},\max}$ is its maximum singular value. Thus, we have completed the proof for (4.13).

Appendix C

Explanation of the Approximate Rules (4.39)-(4.41) for Parameter Selection

In this appendix, we develop the approximate rules (4.39)-(4.41) for parameter selection for the first-degree polynomials in Section 4.2. In order to decrease the sensitivities between the coefficients of $\hat{f}(x)$ and $h(x)$, the rules propose values for the parameters in the first-degree polynomial $q(x) = q_1 x + q_0$ to approximately minimize the condition number of the associated matrix $\hat{\mathbf{G}}$ in (4.34). In addition, we show the function $\|(g(x) + q_r)^M\|_2^2$ in (4.39) is convex towards q_r, so its minimum point can be obtained efficiently. Moreover, we analyze the limit of the approximately optimal parameters when M approaches infinity.

Before developing the rules, we first show a simple fact about the condition number. The condition number is the ratio between the maximum and the minimum magnification of vector norm. Thus, for arbitrary vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ that have dimension $M + 1$, the condition number satisfies

$$\mathrm{cond}(\hat{\mathbf{G}}) = \frac{\max_{\mathbf{f}}(\|\hat{\mathbf{G}}\mathbf{f}\|_2/\|\mathbf{f}\|_2)}{\min_{\mathbf{f}}(\|\hat{\mathbf{G}}\mathbf{f}\|_2/\|\mathbf{f}\|_2)} \geq \frac{\|\hat{\mathbf{G}}\mathbf{f}_1\|_2/\|\mathbf{f}_1\|_2}{\|\hat{\mathbf{G}}\mathbf{f}_2\|_2/\|\mathbf{f}_2\|_2}. \tag{C.1}$$

As a result, the right-hand-side of (C.1) with any vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ can serve as a lower bound to the condition number.

The approximate rules will be explained with certain choices of $\mathbf{f}_1$ and $\mathbf{f}_2$ in (C.1). For simplicity, we use an alternative expression of the first-degree polynomial $q(x) = q_1(x + q_r)$, where $q_r = \frac{q_0}{q_1}$. We first show how to select q_r with a fixed q_1, and then we consider a reasonable choice of q_1.

Now we show reasons for the rule (4.39) for q_r, when q_1 is fixed. If we let $\mathbf{f}_1 = [1, 0, \ldots, 0]^{\mathrm{T}}$ and $\mathbf{f}_2 = [0, \ldots, 0, 1]^{\mathrm{T}}$, then (C.1) implies

$$\mathrm{cond}(\hat{\mathbf{G}}) \geq |q_1|^M \cdot \|(g(x) + q_r)^M\|_2. \tag{C.2}$$

When q_1 is fixed, the q_r to minimize the right hand side in (C.2) is given by

$$\tilde{q}_r = \arg\min_{q_r} \|(g(x) + q_r)^M\|_2, \tag{C.3}$$

which is equivalent to the rule in (4.39).

Then, we consider how to select q_1, when q_r is chosen according to (C.3). Denote $\mathbf{e}_i$ $(0 \leq i \leq M)$ as the vector with 1 in the i-th element and 0 in the other elements. By sweeping both of $\mathbf{f}_1$ and $\mathbf{f}_2$ over every $\mathbf{e}_i$ $(0 \leq i \leq M)$, we may know

$$\mathrm{cond}(\hat{\mathbf{G}}) \geq \frac{\max_{i=0,1,\ldots,M} |q_1|^i \cdot \|(g(x) + q_r)^i\|_2}{\min_{i=0,1,\ldots,M} |q_1|^i \cdot \|(g(x) + q_r)^i\|_2} \triangleq R(q_1). \tag{C.4}$$

The right-hand-side of (C.4) is a function of q_1, which is denoted as $R(q_1)$. In fact, $R^2(q_1)$ is the ratio between the maximum and minimum energy among the self-convolutions of $\hat{g}(x) = (q \circ g)(x)$ up to M-th degree, so the minimization of $R(q_1)$ aims to minimize the energy variation among the self-convolutions. We claim that $R(q_1)$ is minimized when the energy of the polynomial $(\hat{g}(x))^M$ equals 1, i.e.,

$$\tilde{q}_1 = \left(\|(g(x) + \tilde{q}_r)^M\|_2^2\right)^{-\frac{1}{2M}}, \tag{C.5}$$

which is the rule in (4.40).

To justify the claim in (C.5), we first demonstrate the energy of self-convolutions $E(i) = \|(\hat{g}(x))^i\|_2^2$ $(i = 0, 1, \ldots, M)$ is a convex function towards i with a fixed $\hat{g}(x)$. For the length-$(N+1)$ signal $\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_N$ where $\hat{b}_i$ is the coefficient of the term x^i in $\hat{g}(x)$, we can zero-pad it and obtain its $(MN+1)$-points Discrete Fourier Transform (DFT) [4], which is denoted as $\hat{\mathcal{G}}[k]$ $(k = 0, 1, \ldots, MN)$. The convolution theorem [4] implies that the DFT of the self-convolution $(\hat{g}(x))^i$ $(i \leq M)$ is $\hat{\mathcal{G}}^i[k]$ $(k = 0, 1, \ldots, MN)$, since the degree of $(\hat{g}(x))^i$ $(i \leq M)$ does not exceed MN. As shown by the Parseval's theorem [4], the energy of self-convolution satisfies

$$E(i) = \|(\hat{g}(x))^i\|_2^2 = \frac{1}{MN+1}\sum_{k=0}^{MN}|\hat{\mathcal{G}}^i[k]|^2 = \frac{1}{MN+1}\sum_{k=0}^{MN}|\hat{\mathcal{G}}[k]|^{2i},\ i = 0, 1, \ldots, M. \tag{C.6}$$

Since each term $|\hat{\mathcal{G}}[k]|^{2i}$ is convex with respect to i, the summation $E(i)$ is a convex function of i with any fixed polynomial $\hat{g}(x)$.

Then, we will show the function $R(q_1)$ decreases with q_1 when $0 < q_1 < \tilde{q}_1$, and it increases with q_1 when $q_1 > \tilde{q}_1$, which proves the claim that $\tilde{q}_1$ is the minimum point for $R(q_1)$. When $0 < q_1 < \tilde{q}_1$, we know the energy $E(M) = \|(q_1(g(x) + q_r))^M\|_2^2 < 1$; since $E(0) = 1$ always holds, the convexity of $E(i)$ implies $E(i) < 1 = E(0)$ for $i = 1, 2, \ldots, M$. As a result, the square of $R(q_1)$ becomes

$$R^2(q_1) = \frac{\max\limits_{i=0,1,\ldots,M} E(i)}{\min\limits_{i=0,1,\ldots,M} E(i)} = \frac{1}{E(m^*)} = \frac{1}{\|(g(x) + q_r)^{m^*}\|_2^2} \cdot q_1^{-2m^*},$$

where $m^* = \arg\min\limits_{i=1,\ldots,M} E(i)$. Thus, $R^2(q_1)$ is monotonically decreasing with q_1 when $0 < q_1 < \tilde{q}_1$. When $q_1 > \tilde{q}_1$, we know $E(M) > 1 = E(0)$, so $E(i) < E(M)$ for $i = 0, 1, \ldots, M-1$. Thus,

$$R^2(q_1) = \frac{\max\limits_{i=0,1,\ldots,M} E(i)}{\min\limits_{i=0,1,\ldots,M} E(i)} = \frac{E(M)}{E(m^*)} = \frac{\|(g(x) + q_r)^M\|_2^2}{\|(g(x) + q_r)^{m^*}\|_2^2} \cdot q_1^{2(M-m^*)},$$

where $m^* = \arg\min\limits_{i=0,\ldots,M-1} E(i)$. Thus, it is shown $R^2(q_1)$ is monotonically increasing with q_1 when $q_1 > \tilde{q}_1$. This analysis completes the proof for the claim that $\tilde{q}_1$ in (C.5)

is the optimal value to minimize $R(q_1)$ in (C.4).

At this point, we have shown the reasons for the approximate rules (4.39)-(4.41) for the parameters of the first-degree polynomial.

Next, we show the function $\|(g(x)+q_r)^M\|_2^2$ is convex towards q_r, which guarantees the efficiency of obtaining $\tilde{q}_r$ in (4.39). If we let $\hat{g}(x) = g(x)+q_r$ in the above analysis of energy with DFT, then (C.6) becomes

$$\|(g(x)+q_r)^M\|_2^2 = \frac{1}{MN+1}\sum_{k=0}^{MN}|\mathcal{G}[k]+q_r|^{2M},$$

where $\mathcal{G}[k]$ is the $(MN+1)$-point DFT of the coefficients of $g(x)$. It can be verified that the second derivative of each term $|\mathcal{G}[k]+q_r|^{2M}$ towards q_r is non-negative, so the summation is a convex function of q_r. As a result, we may obtain $\tilde{q}_r$ in (4.39) efficiently due to the convexity of $\|(g(x)+q_r)^M\|_2^2$.

Finally, we analyze the behaviors of $\tilde{q}_r$ and $\tilde{q}_1$ in the limit scenario where M approaches infinity. For $g(x) = \sum_{n=0}^{N} b_n x^n$, the discrete-time Fourier transform [4] of the sequence $b_0, b_1, \ldots, b_N$ is $g(e^{-j\omega}) = \sum_{n=0}^{N} b_n e^{-jn\omega}$. By the convolution theorem and Parseval's theorem [4], we can know the energy of the self-convolution $(g(x)+q_r)^M$ is

$$\|(g(x)+q_r)^M\|_2^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi}|g(e^{-j\omega})+q_r|^{2M}\mathrm{d}\omega.$$

Thus, the rule for $\tilde{q}_r$ in (4.39) becomes

$$\tilde{q}_r = \arg\min_{q_r}\|(g(x)+q_r)^M\|_2^2 = \arg\min_{q_r}\frac{1}{2\pi}\int_{-\pi}^{\pi}|g(e^{-j\omega})+q_r|^{2M}\mathrm{d}\omega = \arg\min_{q_r}\|g(e^{-j\omega})+q_r\|_{2M}, \tag{C.7}$$

where $\|F(\omega)\|_{2M} = \left(\int_{-\pi}^{\pi}|F(\omega)|^{2M}\mathrm{d}\omega\right)^{\frac{1}{2M}}$ denotes the $2M$-norm of a function on the interval $[-\pi,\pi]$. As $M\to\infty$, we know $\|F(\omega)\|_{2M}$ approaches the infinity-norm $\|F(\omega)\|_\infty = \max_\omega|F(\omega)|$. Hence, the rule for $\tilde{q}_r$ has the following limit as M approaches infinity

$$\tilde{q}_r \to \arg\min_{q_r}(\max_{\omega}|g(e^{-j\omega})+q_r|). \tag{C.8}$$

Similar to the derivation above, the rule for $\tilde{q}_1$ in (4.40) is equivalent to

$$\tilde{q}_1 = \left(\frac{1}{2\pi}\int_{-\pi}^{\pi} |g(e^{-j\omega}) + \tilde{q}_r|^{2M} \mathrm{d}\omega\right)^{-\frac{1}{2M}} = \left(\frac{1}{2\pi}\right)^{-\frac{1}{2M}} \cdot \left(\|g(e^{-j\omega}) + \tilde{q}_r\|_{2M}\right)^{-1}. \quad \text{(C.9)}$$

When M approaches infinity, $\tilde{q}_1$ has the limit

$$\tilde{q}_1 \to \left(\max_{\omega} |g(e^{-j\omega}) + \tilde{q}_r|\right)^{-1}. \quad \text{(C.10)}$$

An intuitive explanation of the results in the limit scenario is: the term $\tilde{q}_r$ smoothes the spectrum of $(\tilde{q} \circ g)(e^{-j\omega})$ since it reduces the maximum peak, and the term $\tilde{q}_1$ normalizes the peak of the spectrum in order to avoid significant expansion or shrink of the energy of the self-convolutions with the increase of the degree M.

In addition, the limit scenario analysis implies: when the order of $f(x)$ is sufficiently large, we may also use the results in (C.8) and (C.10) to construct a first-degree polynomial to efficiently reduce the condition number $\mathrm{cond}(\hat{\mathbf{G}})$ as well as the sensitivities $S_{\hat{f}\to h}$ and $S_{h\to\hat{f}}$. In addition, the evaluation of (C.8) and (C.10) does not depend on M; thus, compared with the rules (4.39) and (4.40), they may be more computationally efficient if the value of M may vary, at the expense of potentially lower approximation quality to the optimal first-degree polynomial that actually minimizes the condition number.

Bibliography

[1] D. Wei and A. V. Oppenheim, "Sampling based on local bandwidth," in *Signals, Systems and Computers (ASILOMAR), 2007 Conference Record of the Forty First Asilomar Conference on*, 2007, pp. 1103–1107.

[2] D. Wei, "Sampling based on local bandwidth," Master's thesis, Massachusetts Institute of Technology, 2007.

[3] 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*, vol. 33, no. 5, pp. 1151–1165, 1985.

[4] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010.

[5] M. Blanco and F. Hill Jr, "On time warping and the random delay channel," *Information Theory, IEEE Transactions on*, vol. 25, no. 2, pp. 155–166, 1979.

[6] R. Lummis, "Speaker verification by computer using speech intensity for temporal registration," *Audio and Electroacoustics, IEEE Transactions on*, vol. 21, no. 2, pp. 80–89, 1973.

[7] S. Demirtas, "Functional composition and decomposition in signal processing," Ph. D. Thesis Proposal, Massachusetts Institute of Technology, 2012.

[8] 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*, vol. 25, no. 5, pp. 415–422, 1977.

[9] T. Saramaki, "Design of fir filters as a tapped cascaded interconnection of identical subfilters," *Circuits and Systems, IEEE Transactions on*, vol. 34, no. 9, pp. 1011–1029, 1987.

[10] S. Demirtas, G. Su, and A. V. Oppenheim, "Sensitivity of polynomial composition and decomposition for signal processing applications," in *Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on*, 2012, pp. 391–395.

[11] S. Demirtas, G. Su, and A. V. Oppenheim, "Exact and approximate polynomial decomposition methods for signal processing applications," to appear in the Proceeding of IEEE International Conference on Acoustics, Speech and Signal Processing, 2013.

[12] H. D. Block and H. P. Thielman, "Commutative polynomials," *The Quarterly Journal of Mathematics*, vol. 2, no. 1, pp. 241–243, 1951.

[13] J. F. Ritt, "Prime and composite polynomials," *Transactions of the American Mathematical Society*, vol. 23, no. 1, pp. 51–66, 1922.

[14] D. Barton and R. Zippel, "A polynomial decomposition algorithm," in *Proceedings of the third ACM symposium on Symbolic and algebraic computation.* ACM, 1976, pp. 356–358.

[15] M. Fried and R. MacRae, "On the invariance of chains of fields," *Illinois journal of mathematics*, vol. 13, pp. 165–171, 1969.

[16] M. Giesbrecht and J. May, "New algorithms for exact and approximate polynomial decomposition," *Symbolic-Numeric Computation*, pp. 99–112, 2007.

[17] D. Kozen and S. Landau, "Polynomial decomposition algorithms," *Journal of Symbolic Computation*, vol. 7, no. 5, pp. 445–456, 1989.

[18] R. Corless, M. Giesbrecht, D. Jeffrey, and S. Watt, "Approximate polynomial decomposition," in *Proceedings of the 1999 international symposium on Symbolic and algebraic computation.* ACM, 1999, pp. 213–219.

[19] G. Turnwald, "On schur's conjecture," *Journal of the Australian Mathematical Society-Series A*, vol. 58, no. 3, pp. 312–357, 1995.

[20] P. Aubry and A. Valibouze, "Algebraic computation of resolvents without extraneous powers," *European Journal of Combinatorics*, vol. 33, no. 7, pp. 1369 – 1385, 2012.

[21] B. De Moor, "Total least squares for affinely structured matrices and the noisy realization problem," *Signal Processing, IEEE Transactions on*, vol. 42, no. 11, pp. 3104–3113, 1994.

[22] S. Van Huffel, H. Park, and J. Rosen, "Formulation and solution of structured total least norm problems for parameter estimation," *Signal Processing, IEEE Transactions on*, vol. 44, no. 10, pp. 2464–2474, 1996.

[23] B. Botting, "Structured total least squares for approximate polynomial operations," Master's thesis, University of Waterloo, 2004.

[24] P. Lemmerling, "Structured total least squares: analysis, algorithms and applications," Ph.D. dissertation, K. U. Leuven (Leuven, Belgium), 1999.

[25] W. Ruppert, "Reducibility of polynomials $f(x, y)$ modulo p," *Journal of Number Theory*, vol. 77, no. 1, pp. 62–70, 1999.

[26] J. B. Rosen, H. Park, and J. Glick, "Total least norm formulation and solution for structured problems," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 1, pp. 110–126, 1996.

[27] J. Rickards, "When is a polynomial a composition of other polynomials?" *American Mathematical Monthly*, vol. 118, no. 4, pp. 358–363, 2011.

[28] D. Kalman, "A matrix proof of newton's identities," *Mathematics Magazine*, pp. 313–315, 2000.

Epilogue

The development of this thesis contains interesting stories and experiences which are not revealed in the technical chapters. The topic of polynomial decomposition had already been discovered by Al and Sefa before I joined the group; however, in the process of developing the thesis, there were shifts of focus and discovery of new problems, which made up a short but interesting "intellectual adventure."

This thesis started from an informal talk in one of the earliest 6.341 office hours in fall 2011, when Sefa put forth the question of polynomial decomposition to Tarek and me. After one evening's discussion, we came up with a solution that almost worked except for the constant term. On the next day, we talked to Sefa about our discovery, and the problem of constant term was solved from his previous observation. Then, after several research meetings with Al, we decided that polynomial decomposition for both exact and approximate cases would be a stimulating direction to explore and had the potential to result in my master's thesis. Not long after our discovery, Sefa found a paper [17] which had proposed the coefficient-unraveling algorithm – nearly the same as our discovery – at the time when I was one year old. Although at that time I was not so happy with this fact, looking back now, I think such a "rediscovery" may be a very common situation. In one meeting with Al near the end of the first semester, we discussed linear phase decomposition and minimum phase decomposition, which generated some interesting results as listed in Section 2.1. Meanwhile, I played with the roots of the polynomial and proposed an elementary algorithm to get the roots of $f(x)$ with available $g(x)$ from the coefficient-unraveling method. In order to obtain the roots precisely, Al mentioned Burrus' root-finding algorithm in our discussion, and I had an interesting talk with Zahi afterwards; however, we shifted to more interesting directions before we fully combined Burrus' algorithm with polynomial decomposition. In addition, although Sefa sent me a paper [27] introducing Theorem 5.2, I had no idea how that property could help with the decomposition until I made a related guess (Theorem 5.3, but already proposed in [20]) a year later.

The second semester had two main parts: the first part was writing my master's

thesis proposal, and the second part was developing the sensitivity analysis. With Al's patient teaching and guidance, the master's thesis proposal was a good and effective exercise for me to improve my technical writing skills, although the content in the proposal was considerably less than the thesis – the sensitivity analysis and the decomposition with input as roots were out of the scope of the proposal. Later, the sensitivity analysis was proposed by Al and Sefa, which was intended to understand the robustness to perturbations, since our early simulations had already revealed serious numerical errors when the degrees of polynomials were high. For the process of collaboratively developing our paper [10], my deepest impression is perhaps how productive we three were in the last several days before the deadline (in a good way); the content of the paper got changed and improved to a large extent over the last weekend before the deadline. The content of [10] and some follow-up work are summarized in Chapter 4.

In the third semester, I worked on the roots of polynomials, for which one of Al's predictions got validated. In the semester before, Al had once commented on my master's thesis proposal that the roots seemed to be intriguing and there should be something to discover. Frankly speaking, at that time I did not know how to explore more about the roots except for a simple brute-force-search method, due to the complexity of Theorem 5.2 [27]. In a group meeting in the third semester, Al made a comment that $f(x)$ was easier to obtain due to the linear relationship in (4.1); inspired by this comment, I thought that the mapping property between roots in (5.7) seemed linear with respect to $g(x)$, which might lead to some results. After discussions with Al and Sefa, I started to explore the roots more deeply with the possibility of developing algorithms working on roots. Using part of Theorem 5.2, I first considered the knapsack problem and dynamic programming, which turned out to be too high memory complexity. Then, by observing a kind of symmetry within Theorem 5.2, I proposed a guess that the power sums should be equal among all groups up to power $N-1$ (i.e., Theorem 5.3), which turned out to be correct although I did not think about the proof in the beginning. With this guess and inspired by the course Optimization Methods that I was taking, I formulated the

mixed integer program and developed the root-grouping algorithm in Section 5.2.3, which in our simulation had much better performance than the coefficient-unraveling algorithm [17]. In order to put the root-grouping algorithm in the collaborative ICASSP paper [11] (which finally did not happen), we needed to prove my guess (i.e., Theorem 5.3) and we found a proof with Newton's identities. Later (but before the deadline of ICASSP), searching the literature with more key words, I came across the paper [20]; although the title and abstract of this paper [20] seemed unrelated to my problem in the beginning, I finally realized that it had already proposed Theorem 5.3 and the root-power-summation algorithm (the part to get $g(x)$) in Section 5.2.2, which had even higher efficiency. "Rediscovery" happened again for Theorem 5.3. At that point, we could be sure that my thesis would include decomposition algorithms with input as roots, and Al's prediction became true.

Another big topic in the third semester was approximate decomposition algorithms. In IAP 2012, Sefa sent me a paper [16] proposing approximate decomposition algorithms based on the Ruppert matrix, which became the topic of several meetings with Al and Sefa afterwards. In fall 2012, we focused on the Ruppert-matrix-based algorithms with a number of heated discussions from framework to implementation details; the results are summarized in the collaborative paper [11] and in Section 6.1.2 of this thesis. The transformation from polynomial decomposition to determining a rank deficient Ruppert matrix was mathematically deep and interesting; however, after implementation and extensive trials, we realized that the high dimension of the Ruppert matrix might be a numerical challenge. I still think the direction of developing and improving algorithms that are based on determining a rank-deficient approximation of the Ruppert matrix is worth more exploration and may potentially lead to better and more promising results.

In the fourth semester, my main focus was writing the thesis, for which Al and Sefa offered significant help in improving the quality of the thesis. In addition to writing, I extended the root-grouping algorithm to approximate decomposition with input as roots, which is summarized in Section 6.2, but I believe there is considerable room for improvements since I did not have sufficient time to work on it.