2020-08

Decoupling Methods for the Identification of Polynomial Nonlinear Autoregressive Exogenous Input Models

Karami, Kiana

UNIVERSITY OF CALGARY

Decoupling Methods for the Identification of Polynomial Nonlinear Autoregressive Exogenous Input Models

by

Kiana Karami

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA
AUGUST, 2020

Abstract

Developing a mathematical model of the system to be controlled is a significant part of a control design process, the more accurate the model, the more precise the control design can be. Since most of the systems around us behave nonlinearly, linear models are not always adequate. Therefore, nonlinear models should be considered. Many different model structures have been used in the literature such as the Volterra series, block-structured models, state-space representations, or nonlinear input-output models. Each of these structures has the ability to represent a large class of nonlinear systems but with its own drawbacks. Many are black-box modeling approaches and do not provide any intuition regarding the system. Many also suffer from the curse of dimensionality, becoming overly complex as the severity of the nonlinearity increases. It would be more practical if a nonlinear system can be approximated by a simpler model that is more accessible and understandable.

During this research, the focus was on one of the widely used nonlinear input-output models known as the polynomial Nonlinear Autoregressive eXogenous input (NARX) model, as it represents a large class of nonlinear systems. A decoupling approach is proposed for polynomial NARX models. This technique replaces the multivariate polynomial that characterizes the NARX model with a decoupled model comprising a mixing matrix followed by a bank of univariate polynomials and a summation. While the proposed decoupling algorithm reduces the number of parameters significantly, performing the decoupling involves solving a non-convex optimization, which must be solved iteratively. Different initialization techniques are proposed for this optimization. In addition, identification algorithms are developed in both prediction error and simulation error minimization frameworks.

The results of the decoupling approach are verified on two nonlinear identification benchmark problems and show promising outcomes since the number of parameters decreases significantly while the model accuracy remains high. Also, the decoupled model is capable of providing some insight into the identified model, as it is no longer a black-box.

Acknowledgements

I would never have been able to finish my Ph.D. journey without the guidance of my supervisor, help from my friends, and support from my family.

Foremost, I would like to express my sincere gratitude to my supervisor, Dr. David Westwick, for his continuous support of my research, his patience, motivation, enthusiasm and immense knowledge. His guidance helped me in all the time of research and writing of this thesis; without his help, patience and expertise, this thesis would not have been possible. I believe working with Dr. Westwick has helped me grow into a better version of me. One simply could not wish for a better or more supportive supervisor.

Besides my supervisor, I would like to thank Dr. Johan Schoukens (Vrije Universteit Brussel) for providing critical and valuable discussions regarding the ideas developed in this manuscript.

I would also like to thank my thesis committee members. Also, thank you to all my teachers and mentors over the years, who helped shape me into the person I am today.

Also I was blessed to have the support of my father, Dr. Ezatollah Karami, whose guidance and support made my education and this work possible. I will be forever indebted to him and could never thank him enough for his love and devotion.

I dedicate this thesis to my father, Ezatollah, and my sister, Saina, who have always supported me no matter what path I chose to take. Also, to my mother, Kozet, who will forever be with me.

Table of Contents

List of Figures and Illustrations

List of Tables

List of Symbols, Abbreviations and Nomenclature

Acronyms

ALS	Alternating Least Squares
ARX	Auto Regressive eXogenous input model
BLA	Best Linear Approximation
CPD	Canonical Polyadic Decomposition
DP-NARX	Decoupled Polynomial Nonlinear AutoRegressive eXogenous Input Model
FIR	Finite Impulse Response
IID	Independent Identically Distributed
LTI	Linear Time Invariant
LTV	Linear Time Variant
MIMO	Multi-Input Multi-Output
MISO	Multi-Input Single-Output
NARX	Nonlinear AutoRegressive eXogenous Input Model
NLSS	NonLinear State-Space
PEM	Prediction Error Minimization
P-NARX	Polynomial Nonlinear AutoRegressive eXogenous Input Model
PNLSS	Polynomial NonLinear State-Space
RMS	Root Mean Square
SEM	Simulation Error Minimization
SISO	Single-Input Single-Output
SLS	Separable Least Squares
SNR	Signal to Noise Ratio
SVD	Singular Value Decomposition

Notation Conventions

a, A	Scalar
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix
$\boldsymbol{\mathcal{A}}$	Tensor
$u, \mathbf{u}$	Input of a system
$y, \mathbf{y}$	Output of a system
$\hat{a}$	Estimation of variable a
$\|\mathbf{a}\|$	Euclidian norm of vector $\mathbf{a}$
$\otimes$	Kronecker product
$\times_n$	Mode-n tensor product

Symbols

N	Number of data points
n_u	Number of past inputs
n_y	Number of past outputs
n_k	Number of input delays
m	Number of past inputs+outputs
M	Polynomial Order
r	Number of branches
$\mathbf{V}$	Mixing matrix
$\mathbf{C}$	Polynomial coefficients matrix

Epigraph

Everything we think we know about the world is a model. Our models do have a strong congruence with the world. Our models fall far short of representing the real world fully.

- Donella H. Meadows, *Thinking in Systems: A Primer*

Chapter 1

Introduction

According to the 11th Century C.E. Persian mathematician, astronomer, philosopher, and poet Omar Khayyam, "By learning the simple language of algebra, mathematical models of real-world situations can be created and solved."[1] The passion to use mathematics to solve problems has always been the concern of great minds. If in the ancient world, building mathematical models was an intellectual curiosity, our modern technological society cannot survive without it. Modeling is an indispensable tool for understanding real-world problems and preparing mathematical formulations that provide insight and solutions. One can hardly find an area of science and/or engineering that has not been touched by an application of mathematical modeling. In areas as diverse as archaeology, artificial intelligence, astronomy, biology, economics, engineering, medicine, and space sciences, mathematical modeling is used to understand the behaviors of both natural and human-made systems and to assess the model's ability to simulate important features of those systems.

The procedure of building mathematical models is very essential in science and engineering. In the control systems area, "System Identification" is the term that is used for the process of finding the mathematical model of a system. In system identification, the goal is to find a mathematical description of a dynamic system from the observation of the input

[1]Sh. Khalid, (2019, March 7). Omar Khayyam and his stupendous role in Algebra. Retrieved 6/25/2020, from Greater Kashmir: www.greaterkashmir.com/news/opinion/omar-khayyam-and-his-stupendous-role-in-algebra/

and output data.

As shown in Fig. 1.1, in general there are 3 components in a process, the input that drives the system, the system itself, and the output that the system generates. Any of these three components can be unknown at any point in time, depending which part is unknown to us, a new problem arises that needs to be solved.

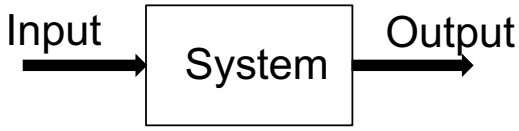

Figure 1.1: Schematic of a General Process

Most of the times, the mathematical model of the system is unknown to us, this brings up the **system identification** problem. Therefore, we have to determine an appropriate mathematical model for the system under test using available information from its inputs and outputs. The next problem rises up when the output of the system is the concern, hence it is called the **simulation** problem. Predicting how the output will behave when the model of the system and its input are known defines the simulation problem. Finally, if we know the model of the system, as well as how we want the output of the system to behave, we have to figure out the appropriate input signal. Hence, the **control** problem deals with finding the input to the system.

So to design a control framework, normally one has to develop a mathematical model for the system and solve the simulation problem as well. As Fig. 1.2 illustrates, usually all three problems must be solved more than once as the design is refined.

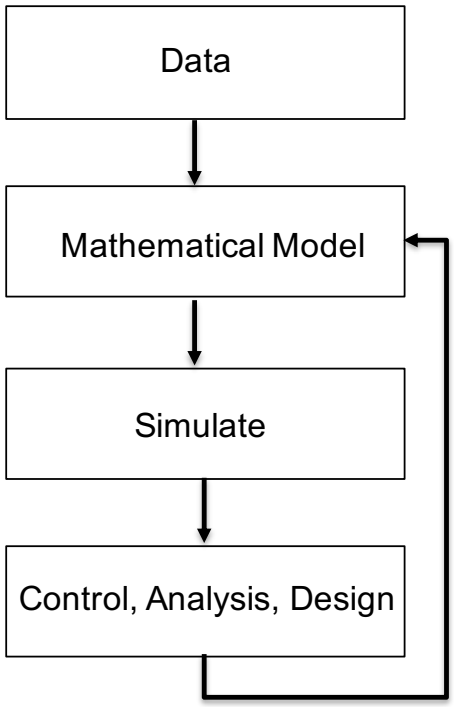

Figure 1.2: Schematic of Control Design Framework

In the real world, the signals that we measure are likely to contain noise, and the system itself may be excited by unmeasured disturbances as shown in Fig. 1.3. Then the challenge becomes solving each of the three mentioned problems so as to minimize the influence of these noise/disturbance sources.

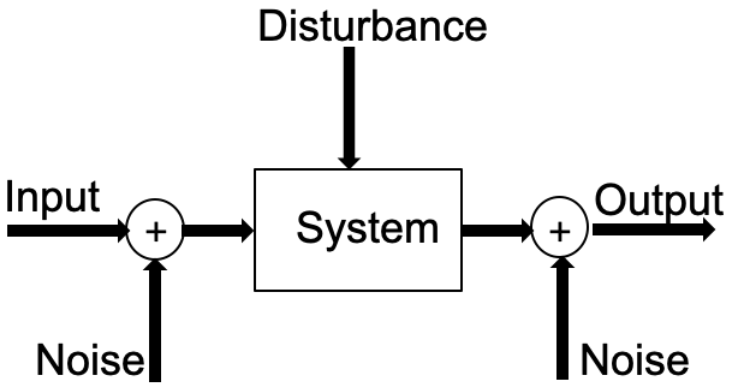

Figure 1.3: Schematic of a General Process with Noise and Disturbance

1.1 General Context of the Thesis

It is impossible to ignore the fact that almost all of the systems around us possess at least some sort of nonlinear behavior. Some of the examples of nonlinear systems that we encounter in everyday life are the weather system, salary tax rating system, your automobile, audio amplifiers and etc. As it is suggested in the literature, it is more appropriate to find a nonlinear mathematical representation of a nonlinear system rather than just a linear model. The area of nonlinear system identification has received lots of attention in past few decades and it has been a very active area these days as well.

There exist many different model structures for modeling nonlinear systems such as the Volterra series, block-structured models, state-space representations or nonlinear difference equation models such as the NARX and NARMAX models. Each of these models is capable of representing some class of nonlinear systems with their own advantages and disadvantages. One of the main disadvantages that is common between these models is that they are all black-box modeling approaches. Also the growth of the number of parameters is another issue, especially for higher orders of nonlinearity the computation will become very expensive.

As engineers, the first step in dealing with a nonlinear system often involves approximating it with a linear model. The class of nonlinear systems whose linearizations contain poles that move when the input is changing, cannot be represented using Volterra series or block-structured model. Both state-space and NARX/NARMAX models are capable of representing this class of systems but for higher order nonlinearities (higher than 3) it is almost impossible to use these representations due to the extensive growth of the number of parameters. Therefore, these complex models need to be modified and approximated by simpler models to eliminate the curse of dimensionality.

This thesis is focused on the system identification problem. More specifically, the field of nonlinear system identification has been studied in this thesis. the question that motivated this work is:

Can one propose a simpler model for dynamic nonlinear systems that is still very accurate? In particular, can one develop an identification method that is able to handle non-Volterra systems, whose linearizations contain moving poles, but without suffering from the curse of dimensionality, while it is capable of providing some insight into the model?

In this thesis, a decoupling method is developed for polynomial nonlinear auto regressive exogenous input models. This technique not only reduces the number of parameters significantly but also provides an insight into the model that can be used, for example, to detect the occurrence of extrapolation. Furthermore, the accuracy of the decoupled model is comparable to that of the full NARX model.

The proposed technique is applied to two different nonlinear benchmark problems: the Silver-Box benchmark and the Bouc-Wen benchmark. The Silver-Box is an electronic system that provides measurement data. The Bouc-Wen hysteresis benchmark is a simulated system that is implemented as a set of coupled nonlinear differential equations. This benchmark was chosen because hysteresis appears in wide range of engineering problems and it is a very nonlinear phenomenon. In particular,the nonlinear behavior depends on an unmeasureable internal variable so that identifying such a system using only input-output data is very challenging. Indeed, hysteresis can not be modeled by any of the Volterra series based models.

1.2 Outline of the Thesis

- **Chapter 2:** This chapter provides a background review on system identification with the emphasis on nonlinear system identification techniques. Different nonlinear model structures have been discussed as well as their best linear approximations.

- **Chapter 3:** This chapter discusses a decoupling algorithm for the polynomial NARX model that substitutes the multivariate polynomial with a transformation matrix followed by a bank of univariate polynomials. This approach decreases the number of

model parameters significantly and also imposes structure on the black-box NARX model. The decoupling technique results in a substantial reduction in the number of parameters, and allows some insight into the nature of the nonlinearities in the system. The proposed decoupling technique involves solving a non-convex optimization that is sensitive to initial starting points.

- **Chapter 4:** In this chapter, three different initialization techniques for the optimization problem in decoupling polynomial NARX model are proposed. Since the proposed decoupling approach results in a non-convex optimization problem, initialization is an important factor to consider. Solving non-convex optimization problems has always been challenging due to the possibility of getting trapped in a sub-optimal local optimum. As a result, these kinds of optimization problems are sensitive to the initial solution. Providing an appropriate initial solution can increase the likelihood of finding the globally optimal solution.

- **Chapter 5:** Prediction Error Minimization (PEM) is a classical approach for the identification of both linear and nonlinear systems. Models trained using PEM may not be suitable for system simulation, where the model only has access to the system's inputs. In this chapter, an identification method based on Simulation Error Minimization (SEM) for decoupled polynomial NARX models is proposed. The proposed algorithm is applied to data from the two nonlinear system identification benchmarks discussed previously and its performance is compared to that of the PEM based algorithm developed in previous chapters.

- **Chapter 6:** This chapter concludes this thesis and also introduces some ideas for possible future works.

1.3 Contributions

A list of publications resulting from this research is as follows:

1.3.1 Journal Publication

- **Kiana Karami**, David T. Westwick, Johan Schoukens. "Applying Polynomial Decoupling Methods to the Polynomial NARX Model". Mechanical Systems and Signal Processing Journal, Elsevier, Accepted.

1.3.2 International Conferences

- **Kiana Karami**, David T. Westwick. "Initialization Approach for Decoupling Polynomial NARX Model Using Tensor Decomposition". 21st IFAC World Congress 2020, Berlin, Germany, Accepted.

- **Kiana Karami**, David T. Westwick, Johan Schoukens. "Identification of Decoupled Polynomial NARX Model Using Simulation Error Minimization". 2019 American Control Conference (ACC), pp. 4362-4367, IEEE, 2019.

- David T. Westwick, Gabriel Hollander, **Kiana Karami**, Johan Schoukens. "Using Decoupling Methods to Reduce Polynomial NARX Models". IFAC PapersOnLine 51-15 (2018): 796–801, 18th IFAC Symposium on System Identification, SYSID 2018. Stockholm, Sweden, July 9-11.

- **Kiana Karami**, and David T. Westwick. "Identification of Stretch Reflex Dynamics from Closed-Loop Data". IFAC-PapersOnLine 48.28 (2015): 1397-1402, 17th IFAC Symposium on System Identification, SYSID 2015. Beijing, China, October 19-21.

1.3.3 International Workshops with Abstract

- 4th Edition of the Workshop on Nonlinear System Identification Benchmarks, Eindhoven, Netherlands, April 2019, Oral presentation: Titled "Polynomial Constrained Factoring in P-NARX Identification", Authors: **Kiana Karami**, David T. Westwick.

- 3rd Edition of the Workshop on Nonlinear System Identification Benchmarks, Liege, Belgium, April 2018, Oral presentation: Titled "Identification of Decoupled Polynomial NARX Models Using Simulation Error Minimization", Authors: **Kiana Karami**, David T. Westwick, Johan Schoukens.

Chapter 2

Background

2.1 System Identification

System identification is a diverse field that can be presented in many different ways [1, 2, 3]. As an interface between the world of mathematical abstractions, control theory, and the real world of application, system identification finds application in a variety of fields. At its core, system identification uses statistical methods to build mathematical models of dynamical systems from measured data. The first system identification methods were developed decades ago as powerful techniques to analyze system configurations and approximate real system behavior [4]. System identification is a very broad topic, with different techniques that depend on the characteristic of the models to be estimated: linear, nonlinear, continuous, discrete, hybrid, non-parametric and etc. [5].

The approach in system identification is to construct a mathematical model of a system such as in Fig. 2.1 that captures the relationship between the input and output as accurately as possible in the presence of noise and disturbances

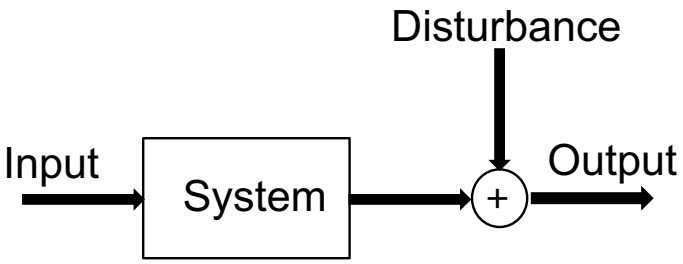

Figure 2.1: General System

System identification tries to find the "best" model that describe input-output data, where I will define what is meant by "best" later in this chapter. Being able to obtain the best model, provides us with a better knowledge about the system. One of the main factors in the success of control theories is that the design methods are accompanied with powerful system identification techniques, so the control design is based on an accurate system model [6]. This is true for any field of study, from process industries like chemical or mechanical plants [7, 8], study of human being [9] to study of biological systems [10, 11].

However it is important to note that the system identification process faces challenges such as finding an appropriate model structure, rejecting disturbances in the recorded data, dealing with the effects of time-varying and nonlinear systems and the difficulty of measuring some variables that may be central to the model.

2.1.1 Models of Linear Systems

Systems may be either linear or nonlinear with respect to their inputs. A system that is linear with respect to its input, usually simply called a linear system, obeys the superposition and scaling principles. Any system that does not have these properties is said to be nonlinear. Due to the relative simplicity of linear models, linear system identification can be used as a preliminary step for the purpose of control design. Numerous works have been done in identifying linear systems, for example [2, 12, 13, 14, 15]. Various types of linear models have been use in literature such as:

- Impulse Response: which is defined as a response of a system at time $t = T$ to a single impulse applied at a time $t = T_1$. Assuming $u(t) = \delta(t - T_1)$ is the input to a linear system ($\delta(.)$ is the Dirac delta function), the corresponding response of the system would be $y(t)|_{t=T} = h(T, T_1)$. Thus the function $h(.)$ is called the impulse response of the system. Using the impulse response to compute the output of a linear, continuous time system:

$$y(t) = \int_{-\infty}^{t} h(t, T)u(T)dT = \int_{-\infty}^{\infty} h(t, T)u(T)dT \tag{2.1}$$

The impulse response can be either LTV or LTI. For a time-invariant system, $h(.)$ is only dependent on the time difference, $\tau = t - T$, so it is also possible to write the relationship between the input and output as:

$$y(t) = \int_{-\infty}^{\infty} h(\tau)u(t - \tau)d\tau = \int_{o}^{\infty} h(t - \tau)u(\tau)d\tau \tag{2.2}$$

- Transfer Function: for the system with input $u(t)$ and output $y(t)$, where $U(s) = \mathcal{L}\{u(t)\}$ and $Y(s) = \mathcal{L}\{y(t)\}$ are the Laplace transforms of the input and output respectively, $H(s)$ which is called the transfer function, is a linear mapping between the Laplace transforms of the input and output:

$$H(s) = \frac{Y(s)}{U(s)} \tag{2.3}$$

- State-Space: this kind of representation is a mathematical model that relates the output and the input using, first order differential equations and state variables.

$$\begin{cases} \dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t) \\ \mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t) \end{cases} \tag{2.4}$$

where $\mathbf{u}(t)$, $\mathbf{y}(t)$ and $\mathbf{x}(t)$ are input, output and state vectors, respectively. $A(.)$, $B(.)$, $C(.)$ and $D(.)$ are called state, input, output and feed-through matrices.

2.1.2 Identification of Linear Systems

One of the main approaches in fitting the models that are linear in their parameters is linear regression [16]. In this approach a linear predictor function is used whose parameters are estimated from the given data. This approach is used extensively is practical applications. Assuming $\mathbf{y} = [y_1, y_2, \cdots, y_k]$ is the response variable and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k]$ represents the dependent variables, a general model relating these variables is:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} = \mathbf{x}_i^T \boldsymbol{\beta} \qquad \Longrightarrow \qquad \mathbf{y} = \mathbf{X}\boldsymbol{\beta} \tag{2.5}$$

then the least square estimation technique can be used to find the unknown variables.

Ordinary Least Squares

Ordinary least squares is a method used to estimate the unknown parameters in a linear regression model. This technique chooses the parameters least squares principle which is minimizing the sum of squares of the differences between the true and estimated variable [17, 18].

Consider the system that has n linear equations and p unknowns, $\beta_1, \cdots, \beta_p$ where $n > p$

$$\sum_{j=1}^{p} X_{i,j}\beta_j = y_i \quad \Longrightarrow \quad \mathbf{X}\boldsymbol{\beta} = \mathbf{y} \tag{2.6}$$

where

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1p} \\ X_{21} & X_{22} & \cdots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{np} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \tag{2.7}$$

usually there is no exact solution for such a system, so the goal is to find an optimal solution for β by minimizing the cost function:

$$S(\boldsymbol{\beta}) = ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 \quad \Longrightarrow \quad \hat{\boldsymbol{\beta}} = \arg\min_{\beta} S(\boldsymbol{\beta}) \tag{2.8}$$

This minimization has a unique solution by solving

$$(\mathbf{X}^T\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}^T\mathbf{y} \tag{2.9}$$

Finally the unknown parameters can be expressed as:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.10}$$

Nonetheless, linear regression is commonly used in system identification – and models that are linear in the variables are often preferred simply because they can be estimate using OLS; but in many cases time varying linear systems, with very slow variations, provide sufficient accuracy without the need to invoke a nonlinear model. Also with respect to theory and identification methods of linear systems, in particular main stream identification, still forms the core part of system identification [19]. Thus, starting from experimental data, a linear dynamic time-invariant model is first identified to describe the relationship between the reference signal and the output of the system [20]. If it is sufficiently accurate for the application at hand, the linear model in then used. If a more sophisticated model is required

to describe the data, then, and only then, a nonlinear and/or time-varying model will be fitted. Least squares technique is usually used to fit the linear regression models such as Finite Impulse Response (FIR) or Auto Regressive eXogenous input (ARX) models.

FIR Models

Suppose we have a system:

$$y(t) = G(q)u(t) + e(t) \tag{2.11}$$

where $G(q)$ is a finite impulse response filter,

$$G(q) = \sum_{k=0}^{T} g(k)u(t-k) \tag{2.12}$$

then the parameter vector would be:

$$\boldsymbol{\beta} = \begin{bmatrix} g(0) & g(1) & g(2) & \cdots & g(T) \end{bmatrix}^T \tag{2.13}$$

where

$$\begin{aligned} \boldsymbol{x}(t) &= \begin{bmatrix} u(t) & u(t-1) & u(t-2) & \cdots & u(t-T) \end{bmatrix} \\ \boldsymbol{X} &= \begin{bmatrix} x(0)^T & x(1)^T & \cdots \end{bmatrix}^T \end{aligned} \tag{2.14}$$

ARX Models

Consider the following model:

$$A(q)y(t) = B(q)u(t) + e(t) \quad \Longrightarrow \quad y(t) = \frac{B(q)}{A(q)}u(t) + \frac{1}{A(q)}e(t) \tag{2.15}$$

which is usually known as Auto Regressive eXogenous model, in this case the parameter vector is:

$$\boldsymbol{\beta} = \begin{bmatrix} a_1 & a_2 & \cdots & a_{n_a} & b_0 & b_1 & \cdots & b_{n_b} \end{bmatrix}^T \tag{2.16}$$

where a_is are the coefficients of $A(q)$ and b_is are the coefficients of $B(q)$. Then the regression vector would be

$$\begin{aligned} \boldsymbol{x}(t) &= \begin{bmatrix} -y(t-1) & -y(t-2) & \cdots & -y(t-n_a) & u(t) & u(t-1) & u(t-2) & \cdots & u(t-n_b) \end{bmatrix}^T \\ \boldsymbol{X} &= \begin{bmatrix} x(0)^T & x(1)^T & \cdots \end{bmatrix}^T \end{aligned} \tag{2.17}$$

As with the FIR filter, we can solve the optimization using a linear regression. Note that if $A(q) = 1$, we are back to the FIR filter.

2.1.3 Systems and Model Approximation

Schoukens et al., [20] discussed the use of linear identification methods in order to identify linear approximations of nonlinear systems. With regards to identifying nonlinear systems using linear identification methods it is necessary to make some assumptions. Now the question remains here is that with all of these assumptions, how accurate is the identified model? Since a linear model cannot capture the nonlinear distortions, in order to have a more useful and reliable model it is better to identify a nonlinear model, but this will require more data and is more involved than a linear identification. Part of the problem is that methods for nonlinear identification have not reached the maturity level of their linear counterparts [11, 21, 22]. Consider these three general scenarios:

- Linear Systems - Linear Models: Linear systems represent the most developed area in the field of systems identification. Perhaps fitting data of a linear system into a linear model is the simplest type of identification and will result in very accurate

estimation, but in the real world, the vast majority of systems are not purely linear and so nonlinearity will likely be evident in the data. As a result, an alternative approach is required to model practical problems.

- Nonlinear Systems - Linear Models: Approximation of a linear model for a nonlinear system, which is very common in practice, requires more complicated and strenuous effort. In [20], there is an example that shows that even in the presence of significant nonlinear distortions, it is still possible to have a good linear estimate but the model may be reliable only under certain conditions. Most often, if one knows the error bounds, imperfect linear models can still be useful to design controllers that meet the required specifications.

- Nonlinear Systems - Nonlinear Models: From the results that exist in the literature, like Schoukens et al., [20], it is obvious that it is better to estimate a complete nonlinear model in the presence of nonlinear distortion even if it is possible to build a linear model. But this is a choice with its own disadvantages. Most of these are due to the added complexity in the models, and in the algorithm used to identify them.

Choosing a linear or nonlinear system identification method is mostly a user's decision. Certainly, there are problems in using linear models in presence of nonlinearity [10] but they can also be helpful since they are much easier to deal with. On there other hand, linear models are not always sufficiently accurate and/or flexible, in which case nonlinear models must be identified and then used.

2.2 Nonlinear System Identification

Identification of nonlinear models is probably the most active area in system identification research today [23]. Many more nonlinear model structures are available than there are for linear models. How to choose a proper model is a big challenge especially for those ones

who are new to the field. The choice should be made based on system behavior prospective (hystersis, chaos, fading memory) and the user point of view (physical, black-box model, model that is linear in the parameters) [24].

The main pieces involved in a hypothetical (non)linear system identification process are:

- Data: Every model is just an approximation of the true system. Obtaining a good estimate of your system depends on how well your measured data reflects the behavior of the system. Higher Signal to Noise Ratio (SNR) is preferred since it means that the signal level is much higher than the noise level.

- Model: In case of linear identification there are limited choices for model selection (state-space, transfer function), but in case of nonlinear system identification since there are varieties of options, it is tough to pick an appropriate model structure. Schoukens in [24] discuss extensively how choose the proper nonlinear model based on different aspects such as user preference and system behavior.

- Estimation Method: Having the data set and the model structure in hand, we are trying to find the best model that captures the information in the data set. Most of estimation approaches trying to minimize the error between the estimated output and the system output.

- Validation Process: In order to make sure that the estimated model is reliable, the model will be checked using a validation data set to make sure that the estimate meets the user expectations.

The next two sections will describe two approaches for modeling nonlinear systems: the Volterra Series, and Block-Structured models. The Volterra series provides a convenient theoretical framework, but on the other hand, since the series itself is cumbersome to work with, not many applications exist. Block-structured models are more convenient, but less general than the Volterra series [25], but have found extensive applications [26].

2.2.1 Volterra Series

The Volterra series provides a general representation of nonlinear systems provided that they have a fading memory, as defined by Boyd and Chua [27]. It consists of a series of multidimensional generalized convolutions between multidimensional generalizations of the impulse response and multiple copies of the system input. Several algorithms have been developed for its identification [11, 28, 29, 30, 31, 32, 33]. Identification of Volterra series and their kernels can be described as optimization problem in which parameters will be approximated in a way that minimizes the error between the actual output and the output of Volterra series representation.

Mathematical Framework for Volterra Kernels

The Volterra kernels are extensions of the FIR filter model that is used to represent linear time invariant (LTI) systems as described in Section 2.1.1. The output of a discrete-time LTI system can be computed as follows:

$$y(t) = \sum_{\tau=0}^{T-1} h(\tau)u(t-\tau) \tag{2.18}$$

where $u(t)$ is input and $h(t)$ is the impulse response. Note that this is a discrete time model. The Volterra series extends this structure by adding terms that take the interactions between inputs into account

$$y(t) = h_0 + \sum_{\tau=0}^{T-1} h_1(\tau)u(t-\tau) + \sum_{\tau_1=0}^{T-1}\sum_{\tau_2=0}^{T-1} h_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2) + \cdots \tag{2.19}$$

so

$$y(t) = \sum_{j=0}^{J}\sum_{\tau_1=0}^{T-1}\cdots\sum_{\tau_j=0}^{T-1} h^{(j)}(\tau_1,\cdots\tau_j)u(t-\tau_1)\cdots u(t-\tau_j) \tag{2.20}$$

where $u(t)$, $y(t)$ and $h(.)$ represent the input and output data sequence and the kernels, respectively. It can be shown that an order J Volterra series with a memory length of T samples includes $\frac{(T+J)!}{T!J!}$ parameters [11]. This will be excessively large if either the system order or the system memory length increases. The possibility of finding a good estimates steps down and becomes computationally complicated as the order of the Volterra representation and the number of input time delays increase [30]. So it is clear the Volterra series model is limited to systems with relatively short memory lengths, and to low-order nonlinearities. Note that, (2.20) is linear in the variables, so the identification can be done by solving very large ordinary least squares problem. Different techniques have been presented in the literature in order to solve this problem more efficiently such as Fast Orthogonal Algorithm [34], which solve the regression using efficient numerical techniques, or Laguerre expansion of kernels [29], which uses a basis expansion to reduce the number of parameters, or even more recent approaches that use regularization methods to improve the conditioning of the problem [32, 35].

2.2.2 Nonlinear Block-Structured Models

As mentioned previously, using Volterra series to represent highly nonlinear system is impractical and computationally costly. Another simpler possibility involves building nonlinear systems models as interconnections of zero-memory nonlinearities and dynamic linear systems. The most common of these so-called block-structured models are the Wiener and Hammerstein models [26].

Wiener Model

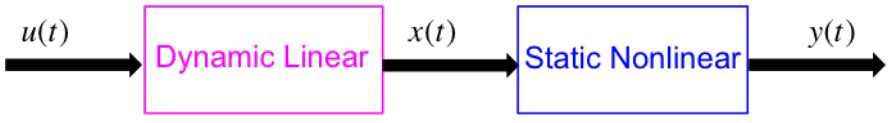

Figure 2.2: Block Diagram of Wiener Model

The block diagram of a Wiener model is shown in Figure 2.2. As the figure illustrates, the model structure is a dynamic linear element followed by a static nonlinearity. This model is widely used in the literature [11, 36, 37].

For example if the static nonlinearity is represented by a power series

$$m(x(t)) = \sum_{j=0}^{J} c^{(j)} x^j(t) \tag{2.21}$$

the output of the Wiener model will be

$$y(t) = \sum_{j=0}^{J} c^{(j)} \Big(\sum_{\tau=0}^{T-1} h(\tau) u(t-\tau) \Big)^j = \sum_{j=0}^{J} c^{(j)} \Big(\sum_{\tau_1=0}^{T-1} \cdots \sum_{\tau_j=0}^{T-1} h(\tau_1) \cdots h(\tau_j) u(t-\tau_1) \cdots u(t-\tau_j) \Big) \tag{2.22}$$

Note that the output is a linear function of the polynomial coefficients, but it is a nonlinear function of the elements of the impulse response (and of course the inputs as well). Comparing (2.20) and (2.22), it can be seen that the order j Volterra kernel of a Wiener system is proportional to the product of j copies of the impulse response function of its linear element:

$$\mathbf{h}^{(j)}(\tau_1, \cdots, \tau_j) = c^{(j)} h(\tau_1) \cdots h(\tau_j) \tag{2.23}$$

The total gain of the Wiener model depends on the polynomial coefficients and the gain of the linear element. Any scaling of the impulse response (by some non-zero constant κ) can be removed by scaling the polynomial coefficients (by $\left(\frac{1}{\kappa}\right)^j$). As the result of this cancellation, the Volterra kernels will remain constant which means there is one degree of freedom that does not affect the input - output relationship. So it might be difficult to compare models from different data sets or different operating conditions, as the linear elements may be scaled differently. As a result, it is common to normalize the elements of the Wiener model. There are several possibilities for this normalization: one way is to set the norm of the linear element to 1 [11], or set the first element of $h(t)$ to be 1, but in this case you need to know

the delay structure before you can do anything, or alternately set the linear coefficient in the polynomial to 1, which makes sense if you are thinking of a linearized system, but if there is no linear term it can cause numerical problems.

Hammerstein Model

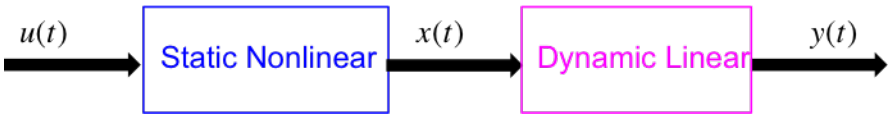

Figure 2.3: Block Diagram of Hammerstein Model

Figure 2.3 shows the static nonlinearity block followed by a dynamic linear element, which is the structure of a Hammerstein model. As shown for the Wiener model, if the nonlinearity is represented by a power series, the output of a Hammerstein model can be formulated as:

$$y(t) = \sum_{\tau=0}^{T-1} h(\tau) \Big(\sum_{j=0}^{J} c^{(j)} u^j (t-\tau) \Big) \tag{2.24}$$

expanding the powers of the input:

$$u^j(t-\tau) = \sum_{\tau_2=0}^{T-1} \cdots \sum_{\tau_j=0}^{T-1} u(t-\tau)u(t-\tau_2)\cdots u(t-\tau_j)\delta_{\tau,\tau_2}\cdots\delta_{\tau,\tau_j} \tag{2.25}$$

where $\delta_{i,j}$ is a Kronecker delta. By substituting the expanded input into the Hammerstein output equation:

$$y(t) = \sum_{j=0}^{J} c^{(j)} \Big(\sum_{\tau_1=0}^{T-1} \cdots \sum_{\tau_j=0}^{T-1} h(\tau_1)u(t-\tau_1)\cdots u(t-\tau_j)\delta_{\tau_1,\tau_2}\cdots\delta_{\tau_1,\tau_j} \Big) \tag{2.26}$$

The jth Voltera kernel of the Hammerstein cascade is calculated as:

$$\mathbf{h}^{(j)}(\tau_1, \cdots, \tau_j) = c^{(j)} h(\tau_1)\delta_{\tau_1,\tau_2}\cdots\delta_{\tau_1,\tau_j} \tag{2.27}$$

From the Volterra kernels of the Hammerstein model it is obvious that they are zero everywhere besides their diagonals, so they can only represent limited category of nonlinear systems. Parallel Hammerstein models can't do much better since their kernels will also be zero for all off diagonals[11]. The extra degree of freedom also exists in this model similar to the Wiener model, if the the polynomial coefficients are scaled by κ, the impulse response will be scaled by $\frac{1}{\kappa}$, therefore, the input - output relationship will remain the same.

Additional Structures

Another method for modeling nonlinear systems using nonlinear block structures is by combining the Wiener and Hammerstein models in series. In that case we have either a Wiener-Hammerstein model or a Hammerstein-Wiener model depending on the order. Speaking of the degree of freedom, there are additional degrees of freedom in these structures, just as there were in the Wiener or Hammerstein models. For the Hammerstein-Wiener model, there are 2 gains that can be chosen arbitrarily. With the Wiener-Hammerstein model, in addition to the two gains, if there are any delay in the system, it can also be assigned arbitrarily to either one of the linear elements.

Keep in mind that most systems cannot be represented accurately by a simple cascade model. For example a block - structure model of human ankle stiffness dynamic [10, 38] is shown in Fig. 2.4. This model which includes two parallel pathways: a linear pathway and a nonlinear pathway, cannot be modeled with any of the previously mentioned models (Wiener, Hammerstein or Wiener-Hammerstein). In cases like this, a parallel cascade model will be used. In case of parallel cascade model, the overall system response is represented as the sum of the outputs of parallel set of simple block - structured models.

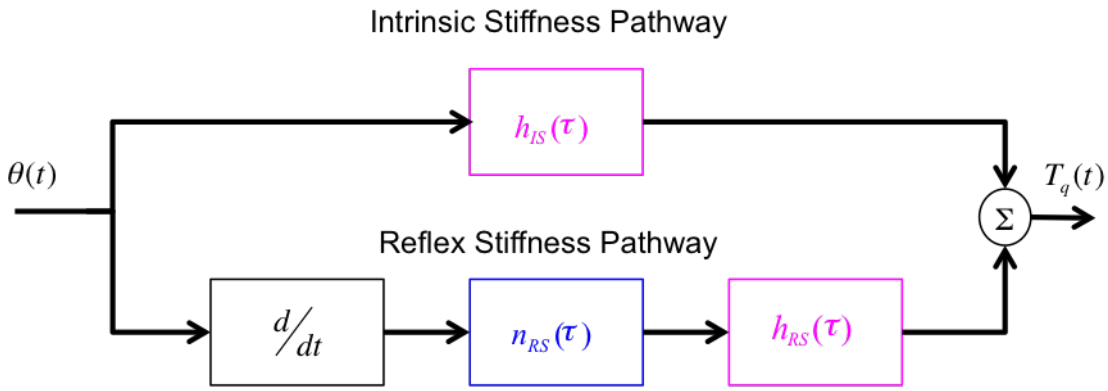

Figure 2.4: Block Diagram of Human Ankle Stiffness Block-Structure Model

Palm in [39] and [40], shows that any nonlinear system that can be approximated with arbitrarily small error by a Volterra series can also be represented by parallel sums of cascades of linear systems and no memory nonlinearities. Based on the work of Palm and others, Korenberg [34] showed that any discrete-time finite-memory nonlinear system having a finite order Volterra series representation can be exactly represented by a finite number of parallel Wiener cascade paths. Each Wiener path consists of a dynamic linear system followed by a static nonlinearity.

2.2.3 Identification of Block-Structured Models

The objective is to find the parameters that minimize the error between the measured output and the model output. For linear in the variables models, this optimization can simply be solved using linear regression. However, block-Structured models are nonlinear on some of their parameters so ordinary least squares cannot be used. One away of finding the unknown parameters for a model that is nonlinear in its parameters, is to use an iterative optimization. The basic idea, is to start with an initial parameter estimate, $\boldsymbol{\theta}(0)$, and then update it according to

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mu_k \boldsymbol{d}_k \tag{2.28}$$

where the vector $\boldsymbol{d}_k$ is the search direction for the kth step, and μ_k is the corresponding step length. The simplest approach, is to follow the direction of steepest descent.

$$\begin{aligned}\boldsymbol{d}_k &= -\frac{\partial V_N(\boldsymbol{\theta}}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = \frac{-1}{2N}\sum_{t=1}^{N}\frac{\partial}{\partial \boldsymbol{\theta}}\epsilon^2(t,\boldsymbol{\theta}) \\ &= \frac{1}{N}\sum_{t=1}^{N}\epsilon(t,\boldsymbol{\theta})\frac{\partial \hat{y}(t,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = J^T(\boldsymbol{\theta}_k)\epsilon(\boldsymbol{\theta}_k)\end{aligned} \tag{2.29}$$

where ϵ is a vector containing the errors,

$$\epsilon(t,\boldsymbol{\theta}_k) = y(t) - \hat{y}(t,\boldsymbol{\theta}_k) \tag{2.30}$$

and J is the Jacobian matrix,

$$J(t,j) = \frac{\partial \hat{y}(t,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}(j)}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} \tag{2.31}$$

with one row per time-point, and one column per parameter. The Jacobian is usually relatively cheap to compute, so the updates in the gradient descent algorithm are also cheap to compute. With a gradient descent iteration, the step size is limited by the tightest curvature in the error surface, however the convergence rate is limited by the step size and the most gradual curvature. The curvature (i.e. second derivative) of the error surface is used to adjust the step direction. Conceptually, the step size increases in the direction of the least curvature, and reduces in directions of high curvature.

Analytically, expanding the error surface, about $\boldsymbol{\theta}_k$, as a Taylor series, and truncating the series after the second order term:

$$V_N(\boldsymbol{\theta}_k + \delta) = V_N(\boldsymbol{\theta}_k) + \left(\frac{\partial V_N}{\partial \boldsymbol{\theta}}\right)^T \delta + \frac{1}{2!}\delta^T \frac{\partial^2 V_N}{\partial \boldsymbol{\theta}^2}\delta + \cdots \tag{2.32}$$

Lets define the Hessian which contains the second order partial derivatives and also is a

function of the current parameters:

$$H(i,j) = \frac{\partial^2 V_N(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}(i) \partial \boldsymbol{\theta}(j)} \tag{2.33}$$

Then (2.32) can be rewritten using (2.31) and (2.33) as:

$$V_N(\boldsymbol{\theta}_k + \delta) = V_N(\boldsymbol{\theta}_k) - \frac{1}{N}(\epsilon^T J)\delta + \frac{1}{2!}\delta^T H \delta \tag{2.34}$$

Finally, by minimizing V_N, we get

$$\delta = \frac{1}{N} H^{-1} J^T \epsilon \tag{2.35}$$

If the optimization was a linear in the variables problem, so that J was a matrix of constants and

$$H = \frac{-1}{N} J^T (\frac{\partial \epsilon}{\partial \boldsymbol{\theta}}) + \frac{-1}{N} (\frac{\partial J^T}{\partial \boldsymbol{\theta}}) \epsilon \tag{2.36}$$

where $\frac{\partial \epsilon}{\partial \boldsymbol{\theta}} = J$ and the second term would be zero, then:

$$H = \frac{1}{N} J^T J \tag{2.37}$$

and the optimal second order step would be

$$\delta = (J^T J)^{-1} J^T \epsilon \tag{2.38}$$

which is just the normal equation solution. However, if the problem was linear in the variables, then the second order Taylor expansion would be exact, and the minimum would be reached in a single step. Of course, we are interested in nonlinear problems, so the second order Taylor series won't be exact, so a single step will not, in general, reach the actual

optimum.

The Gauss-Newton Method

The Gauss-Newton algorithm is used to solve nonlinear least squares problems. This method uses an approximation to the Hessian, to reduce the computational cost of each step. Note that near the optimum, ϵ will be small, and can be expected to be independent of the input. Thus, the second term in the Hessian should be negligible. Far away from the optimum, one will be taking relatively large steps, so the second order approximation probably is not very good.

The advantage of the Gauss-Newton method is that the second derivatives, which can be challenging to compute, are not required. The Gauss-Newton method approximates the Hessian with

$$\hat{H} = \frac{1}{N} J^T J \tag{2.39}$$

so that the Gauss-Newton search direction is given by:

$$\boldsymbol{d}_k = (J^T J)^{-1} J^T \epsilon \tag{2.40}$$

and, as before, the step size can be chosen adaptively. Note that there is no guarantee that the approximate Hessian is nonsingular, and that if the approximate Hessian becomes singular, there is little that can be done, other than backing up a step or two, and readjusting the step size, which may or may not work.

The Levenberg-Marquardt Method

The biggest disadvantage with the Gauss-Newton approach, is that the approximate Hessian can become either singular, or very badly conditioned. The Levenberg-Marquardt method addresses these problems with the approximate Hessian by adding a diagonal regularization

term to the Hessian. The Levenberg-Marquardt approach is more robust than the Gauss-Newton, which means that in many cases it finds a solution even if it starts very far off the final minimum. The Levenberg-Marquardt step is given by:

$$\hat{H} = \frac{1}{N} J^T J + \mu_k I_M \tag{2.41}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + (\frac{1}{N} \hat{H}^{-1} J^T \epsilon) \tag{2.42}$$

Notice that there is no explicit step size parameter. Rather, the size of the diagonal ridge, μ_k, controls the step size. If μ_k is large, the ridge will dominate the approximate Hessian, and the Levenberg-Marquardt method will behave like a gradient descent, with a step size of $1/\mu_k$. On the other hand, if μ_k is small, then the Levenberg-Marquardt approach will behave like the Gauss-Newton optimization. Thus, after every successful update, μ_k is reduced, whereas it is increased after each attempted update that increases V_N. Since $J^T J$ is non-negative definite, and adding the diagonal increases all of the eigenvalues by μ_k, the approximate Hessian will always be positive definite and hence nonsingular.

Although these structures are nonlinear in some of their parameters, they are always linear in some of their parameters as well, therefore Separable Least Squares is also an option in order to identify these systems.

Separable Least Squares

In the cases that the model is a linear function of nonlinear variables we can use the SLS, which is sometimes called separable nonlinear least squares (SNLS), technique to solve for the unknown parameters. Using the variable projection technique will eliminate the linear variables and change the problem into one with only nonlinear parameters [41]. This technique is useful in reducing the number of parameters and also obtains a better conditioned problem. It has been shown that the reduced problem and original problem will both result in the same minima, provided that the linear parameters were recovered by solving an

appropriate linear least squares problem such as ordinary least squares.

The key observation behind SLS optimization is the parameters that appear linearly, can be written in closed-form as functions of the remaining parameters and then be solved using ordinary least squares methods. I have to note that, this is much more complicated than the methods presented for linear system identification.

2.2.4 Best Linear Approximation - BLA

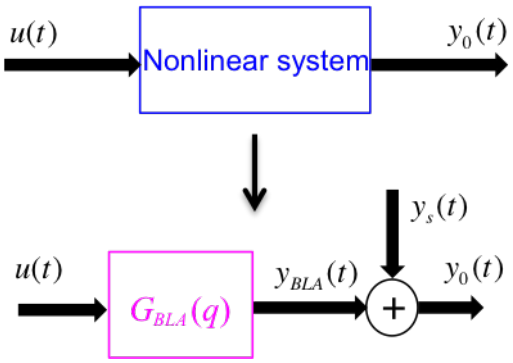

Figure 2.5: The Nonlinear System and Its Best Linear Approximation Diagram

In this section, the effort is to approximate a nonlinear system with a linear model. The linear model that best fits the nonlinear data, in the minimum mean squared error sense, is called Best Linear Approximation or BLA. The BLA can be represented by any kind of linear model, such as an impulse response, transfer function, etc. In particular the best linear approximation is defined by minimizing the mean squared error between the measured and model outputs:

$$G_{BLA}(q) = arg \min_{G} E\{|y_0(t) - G(q)u(t)|^2\} \tag{2.43}$$

where q is the forward shift operator, $q^{-1}u(t) = u(t-1)$, $G(q)$ is a polynomial or rational function in the backward shift, q^{-1}, and $E\{.\}$ denotes the expected value operator. In [14] it is shown that the properties of the filter $G(q)$ are quite analogous to those of $G(z)$ in the z-domain. The best linear approximation of a nonlinear system always depends on the input

signal. The principle is almost the same as the linearization around an operating point, as the operating point changes so does the linearization. A block diagram of a nonlinear system and its linear approximation is shown in Fig. 2.5, where $y_s(t)$ is an error term that captures all unmodeled nonlinearities:

$$y_s(t) = y_0(t) - G_{BLA}(q)u(t) \tag{2.44}$$

Some studies were undertaken in recent years in the area of best linear approximation by Schoukens, Pintelon, Enqvist, Ljung and others [2, 42, 43, 44]. The majority of these studies have been conducted in the frequency domain. Hence, this section will cover the best linear approximation subject with an approach to frequency domain.

In [2, 43, 44], it is shown that the minimizer of the optimization equation, G_{BLA}, can easily be obtained as follows:

$$G_{BLA}(e^{j\omega T_s}) = \frac{S_{Y_0U(e^{j\omega T_s})}}{S_{UU(e^{j\omega T_s})}} \tag{2.45}$$

where $S_{Y_0U(e^{j\omega T_s})}$ and $S_{UU(e^{j\omega T_s})}$ represent the cross power spectrum of y and u and the auto power spectrum of u, respectively. This is equivalent to the least squares solution in the time domain (so really any standard linear identification algorithm should give you the BLA).

Best Linear Approximation of a Wiener Model

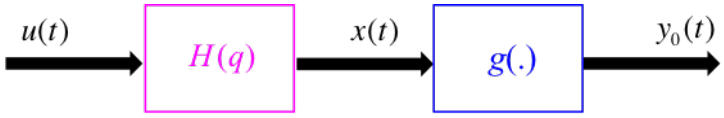

Figure 2.6: The Wiener System: $H(q)$ Is a Linear System and $g(.)$ Is Its Static Nonlinearity

For the Wiener system illustrated in Fig. 2.6, its output can be computed as:

$$\begin{aligned} y_0(t) &= g(x(t)) \\ x(t) &= H(q)u(t) \end{aligned} \tag{2.46}$$

The static nonlinearity $g(.)$ is assumed to be a linear combination of basis functions and the linear system $H(q)$ is assumed to be a rational function of the backward shift operator q^{-1}:

$$H(q) = \frac{b_0 + b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}}{a_0 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_a} q^{-n_a}} \tag{2.47}$$

In [42] it is proved that the best linear approximation of the Wiener model excited by an input signal belonging to the Riemann equivalence class of asymptotically normally distributed excitation signals [2, 45], can be given by:

$$G_{BLA}(e^{j\omega T_s}) = \alpha H(e^{j\omega T_s}) \tag{2.48}$$

The coefficient α depends on many factors: the linear system $H(q)$, the static nonlinear systems $g(x(t))$, the power spectrum of the input signal $u(t)$, and therefore as well on the variance of the input signal $u(t)$. Hence, the BLA of a Wiener system has the same zeros and poles as the linear part regardless of the input characteristics.

A random signal $u(t)$ fits in the Riemann equivalence class if it obeys by any of the following statements:

1. $u(t)$ is a Gaussian noise signal.

2. $u(t)$ is a random multi-sine or random phase multi-sine[2] with the conditions discussed in [42].

This is an extension to Bussgang's theorem [46], which considered only Gaussian excitation.

Best Linear Approximation of Parallel Wiener Model

Schoukens in [42] shows that the best linear approximation of a parallel Wiener system is a linear combination of the linear dynamics that represent each different linear block.

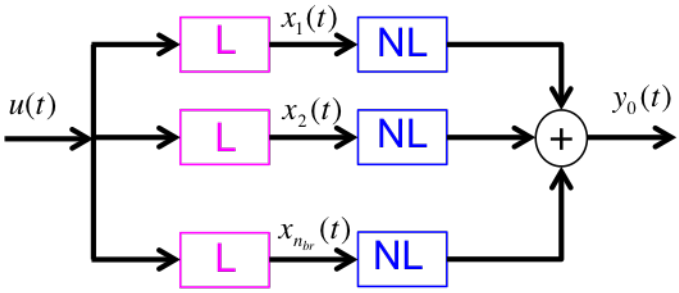

Figure 2.7: The Parallel Wiener System

The output y_0 of the Wiener parallel model is the sum of the output of each separate parallel Wiener branch y_k. So the BLA of the parallel Wiener model can be represented by the sum of the BLAs of each branch. Building on the equation of the best linear approximation of the Wiener model, the BLA of parallel Wiener model would be written as:

$$G_{BLA}(e^{j\omega T_s}) = \sum_{m=1}^{n_{br}} \alpha_m H^{[m]}(e^{j\omega T_s}) \tag{2.49}$$

Since the best linear approximation of the parallel Wiener model of a nonlinear system is a weighted summation of the linear components of the branches (i.e. the BLAs of each path), $H^{[m]}(q)$, the denominator of the BLA of parallel Wiener model will be the product of the denominators of all of the BLAs of the paths. (2.50) is obtained from (2.49) by bringing all the terms over a common denominator, which is the product of all of the $A^m(q)$.

$$G_{BLA}(q) = \frac{\sum_{m=1}^{n_{br}} \alpha_m B^{[m]}(q) \prod_{j=1, j\neq m}^{n_{br}} A^{[j]}(q)}{\prod_{m=1}^{n_{br}} A^{[m]}(q)} \tag{2.50}$$

where

$$H^{[m]}(q) = \frac{B^{[m]}(q)}{A^{[m]}(q)} \tag{2.51}$$

Schoukens [42] noted that the numerator of the best linear approximation will depend on each of the input dependent gains α_m, therefore the zeros of the BLA will change with change in the input, whereas the poles in the denominator that does not depend on the gains α_m and therefore does not depend on the input.

Since the parallel Wiener model is universal approximator for Volterra systems, the BLA of a Volterra system must have these properties as well. Therefore the Volterra analysis is not appropriate for class of systems with variant denominator, since they cannot be approximated by a Volterra series in the first place. In order to model this class of systems, other more appropriate approaches need to be implemented.

2.2.5 Nonlinear Feedback Model

In the case of some practical systems like stiffness dynamics of the human ankle, [10, 38], the best way to represent them is a closed-loop model with a static nonlinearity in the feedback path, as shown in Fig. 2.8.

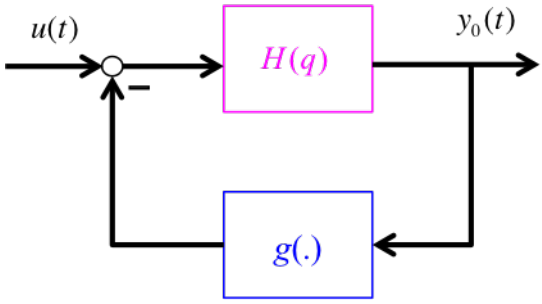

Figure 2.8: Nonlinear Feedback Model

Talking in terms of the best linear approximation, the static nonlinearity could be replaced by a gain, α, provided that the input to the nonlinearity, $y_0(t)$, is in the Riemann class of signals. Taking into account that we cannot control $y_0(t)$ directly, and the input to the linear element also contains the output of the nonlinearity, hence the input to the linear element is probably not in the Riemann class (at least not exactly), therefore this is only an approximation. This gain depends on the probability distribution of the input. As the

input, $u(t)$ changes, the distribution of $y_0(t)$ changes, and so does the BLA, α. So,

$$G_{BLA}(e^{j\omega T_s}) = \frac{H(e^{j\omega T_s})}{1 + \alpha H(e^{j\omega T_s})} \tag{2.52}$$

As it is obvious, the input dependent gain, α, shows in the denominator in this case. The drawback to this situation is that the poles of the best linear approximation will change with changes in the input. In this case, the Volterra representation is not possible, since all Volterra systems can be exactly represented by a parallel Wiener system, whose poles are invariant to changes in the input characteristics. Therefore, we need a to find more appropriate methods for identification of these kind of systems, which will be covered in next section.

2.3 Nonlinear State-Space Models

Another approach for modeling nonlinear systems is the nonlinear state-space model [47] which is a common way to represent systems with multiple inputs and outputs. Equation (2.53) shows the general expression of a nonlinear state-space model.

$$\begin{cases} \mathbf{x}(t+1) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), \theta) \\ \mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), \theta) \end{cases} \tag{2.53}$$

where $\mathbf{x}(t)$ is the state vector and $\mathbf{u}(t)$ and $\mathbf{y}(t)$ are input and output vectors respectively. $\mathbf{g}(.)$ and $\mathbf{h}(.)$ are two nonlinear functions and vector θ includes the parameters to be identified in the model. The first equation in the model, called the state equation, describes the dynamic behavior of the system and the second equation, which is the output equation, relates the current system state and input to the output. One of the downsides of state-space models is that they are not unique, using a similarity transform (2.53) can be converted to a new model that represents exactly the same input-output relation. Assume $\mathbf{w}(t) = T^{-1}\mathbf{x}(t)$,

where T is an arbitrary nonsingular square matrix, Equations (2.53) and (2.54) exhibit the same system dynamic.

$$\begin{cases} \mathbf{w}(t+1) = T^{-1}\mathbf{g}(T\mathbf{w}(t), \mathbf{u}(t), \theta) \\ \mathbf{y}(t) = \mathbf{h}(T\mathbf{w}(t), \mathbf{u}(t), \theta) \end{cases} \tag{2.54}$$

The nonlinear functions $\mathbf{g}(.)$ and $\mathbf{h}(.)$ in general can be expanded using various basis functions. One of the options for these basis functions is to use a polynomial representation. Polynomial expansions are linear in their parameters and can be extended to multivariate cases easily [48]. But there are limitations to this approach. As such, this model is a good tool when a black-box model of a system is required. Since the identified parameters do not have meaningful interpretations, there is no physical intuition of the model whereas the white-box identification provides meaningful information about the physical dynamic of the system.

Esfahani et al., [49, 50] proposed a method that starts from the polynomial nonlinear state-space model and uses the tensor factorization-based polynomial decoupling technique proposed by Dreesen et al. [51] in order to reduce the number of parameters used to represent the nonlinear functions $\mathbf{g}(.)$ and $\mathbf{h}(.)$. This results in a state-space description in which the nonlinearities are transferred into univariate (one-to-one) polynomials that operate on a linear combinations of the state variables and the input, rather than multivariate polynomials involving all state variables and the input to all state updates and the output.

The decoupling method proposed by Dreesen et al. [51] produces a decoupled representation using the Canonical Polyadic Decomposition (CPD) of a 3-dimensional structure known as a tensor [52, 53, 54]. This method will results in a new nonlinear state-space model as in

Equation (2.55):

$$\begin{cases} \mathbf{x}(t+1) = A\mathbf{x}(t) + B\mathbf{u}(t) + W_x\mathbf{f}\left(V^T \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{bmatrix}\right) \\ \mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t) + W_y\mathbf{f}\left(V^T \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{bmatrix}\right) \end{cases} \tag{2.55}$$

where the Ws are linear transformation matrices that transform the outputs of the decoupled nonlinear functions in the modified PNLSS and V is the matrix that transforms the states and inputs. Fig. 4.1 illustrates the polynomial decoupling, and shows how the MIMO polynomial is replaced by a bank of single-input single-output (SISO) polynomials sandwiched between two transformation matrices.

As noted above, the algorithm proposed by Dreesen et al. [51] is based on a standard tensor factorization technique. In the case of MIMO polynomials, a tensor will be constructed from polynomial's Jacobian matrix that is evaluated at several measurement points. The tensor factorization yields the two transformation matrices, W and V, as well as evaluations of the derivatives of the SISO nonlinearities at the operating points. In addition to its application to the nonlinear functions in the PNLSS, this approach has been used to decouple the polynomials that arise in several different systems: parallel Wiener-Hammerstein model [55] and MIMO systems in control design [56].

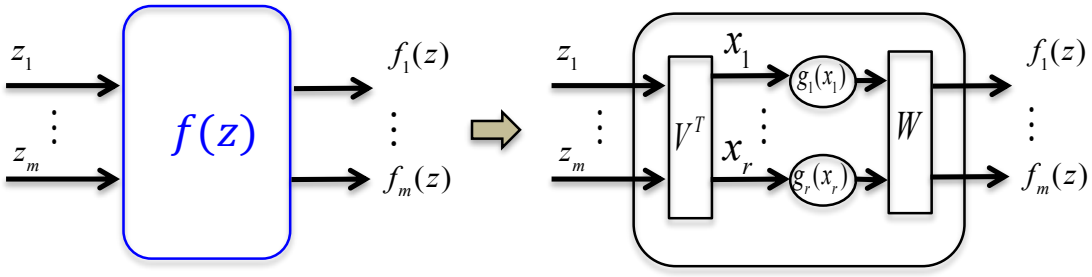

Figure 2.9: Decoupling a MIMO Nonlinearity into SISO Polynomial Branches and a Transformation Matrices

A major advantage of the decoupling method is that the number of parameters now increases linearly with system degree instead of growing combinatorially.

2.4 Nonlinear Auto Regressive eXogenus Input Model

Another time series modeling approach is the nonlinear auto regressive model with exogenous input known as the NARX model. NARX models characterize a large group of nonlinear systems, hence they are a very common class of nonlinear systems [57]. Even though NARX models are not as general as nonlinear state-space models, the advantage is that the NARX model is only based on the knowledge of the inputs and outputs and unlike the state-space model no information of the internal states is needed. Also the NARX model is linear in its parameters which is a major advantages over both NLSS and NARMAX models. Billings [3] developed the NARMAX model which also includes past prediction errors in the nonlinear function. Just like the NLSS depends on an internal state, the output of the NARMAX depends on past prediction errors which adds complexity. The NARX model which is a considerable simplification of that model has been used in literature extensively [58, 59, 60, 61]. In addition, the output equation contains an error term which is assumed to be a sequence of independent identically distributed (IID) random variables. A NARX model can be represented as:

$$y(t) = f(y(t-1), \cdots, y(t-n_y), u(t), u(t-1), \cdots, u(t-n_u)) + e(t) \tag{2.56}$$

where t is the discrete-time index, $u(t)$, $y(t)$ and $e(t)$ are the input, output and equation error, respectively and n_y and n_u are the maximum lags for the system output and input and f is a multiple-input single-output (MISO) nonlinear function.

Many different model structures can be used as basis functions for the MISO nonlinearity at the core of the NARX model, such as sigmoidal neural networks [62], splines [63], polynomials [64, 65] and so on.

2.4.1 Polynomial NARX Models

Polynomial basis functions have been used extensively [65] since they are linear in the parameters. Generally, when the nonlinear function in (2.56), f, is a linear combination of known basis functions [66] such as polynomials, the output is a linear function of the parameters, as in (2.57), which is a linear regression model.

$$f(z_1, z_2 \cdots, z_m) = \sum_{m=0}^{M} \sum_{k_1=1}^{m} \sum_{k_2=k_1}^{m} \cdots \sum_{k_m=k_m-1}^{m} \left[c_m(k_1, k_2, \cdots, k_m) \prod_{i=1}^{m} z_i \right] \tag{2.57}$$

where $c_m(k_1, k_2, \cdots, k_m)$ are the mth degree polynomial coefficients that corresponding to the mth input. Hence the parameters can be estimated using ordinary least squares techniques. Since no more iterative optimization is needed, the global optimal solution is guaranteed. NARX models are accepted by industry, as evident by their relatively common use, and have been successfully used in industrial problems because of this reason.

Another reason that makes polynomial functions so common to use is that some times polynomial terms may lead into physical interpretation [65]. Nonlinear frequency analysis tools have been developed for some classes of nonlinear models to analyze NARX models (or the deterministic components in NARMAX models) in the frequency domain [64, 67, 68].

2.4.2 Problems of NARX Models

Unfortunately, the are some drawbacks to the NARX model, such as:

- The number of parameters used in NARX model grows rapidly with the number of past inputs and outputs ($m = n_u + n_y$) and the degree of nonlinearity. This is actually a worst problem than what happens with the Volterra Series. The Volterra Series is just a polynomial in delayed inputs, whereas the NARX model uses both past inputs and outputs.

- The NARX model is a black-box modeling approach and hardly provides any physical

insight into the system.

- Any estimated model will be unreliable when it is used to extrapolate – to predict outcomes were the inputs are outside of the region probed by the estimation data – this is particularly bad with polynomials. If the function is defined on a multidimensional domain, it is also difficult to characterize the region that has been probed by the estimation (or identification) data. As a result, when the model is being used to process new data, it is difficult to detect when the data are pushing the model outside of its valid region.

  The polynomial NARX model combines both of these difficulties (i.e. a multidimensional domain, and a polynomial). Therefore, they are very poor in extrapolation, and the key issue is the detection of the extrapolation in the identification process.

In order to overcome the drawbacks of the polynomial NARX models, a decoupling approach is proposed and examined in the following chapters of this thesis. This approach brings its own advantages and limitations. Keep in mind that since any model is only an approximation, and not an exact representation, of a system, we should ask ourselves "is the model good enough for the particular application?" instead of "what is the best model?".

Chapter 3

Decoupling Polynomial NARX Models [1]

In this chapter a decoupling approach for polynomial NARX models is proposed. The main idea of the decoupling approach is to decompose a multivariate polynomial into a mixing matrix followed by a bank of univariate polynomials, whose outputs are summed to produce the final output. Later in this chapter, the proposed algorithm will be applied to data from two benchmark problems to examine and validate its performance.

3.1 Decoupling Method

Recall that the output of the NARX model is given by (2.56), repeated here

$$y(t) = f(y(t-1), \cdots, y(t-n_y), u(t), u(t-1), \cdots, u(t-n_u)) + e(t) \tag{3.1}$$

where $e(t)$ is the so-called equation error, an IID sequence of zero-mean random variables. In the polynomial NARX model, the function $f(.)$ is represented by a polynomial (2.57),

[1] The contents of this chapter were presented in the 18th IFAC Symposium on System Identification, SYSID 2018, Stockholm, Sweden [69].

thus:

$$f(y(t-1),\cdots,y(t-n_y),u(t),u(t-1),\cdots,u(t-n_u))$$
$$=\sum_{m=0}^{M}\sum_{p=0}^{m}\sum_{k_1=1}^{n_y}\cdots\sum_{k_p=k_p-1}^{n_y}\sum_{k_p+1=1}^{n_u}\cdots\sum_{k_m=k_m-1}^{n_u}\left[c_{p,q}(k_1,k_2,\cdots,k_m)\prod_{i=a}^{p}y(t-k_i)\prod_{i=p+1}^{m}u(t-k_i)\right] \tag{3.2}$$

where M is the degree of the polynomial, n_u and n_y are number of past inputs and past outputs respectively. The polynomial coefficient, $c_{p,q}(k_1,\cdots,k_{p+q})$, multiplies the product of p delayed outputs, $y(t-k_1)y(t-k_2)\cdots y(t-k_p)$ and $q=m-p$ delayed inputs, $u(t-k_{p+1})\cdots u(t-k_m)$.

Note that, the 1-step ahead prediction is given by $\hat{y}(t|t-1)=f(.)$, since all of the arguments are known, and the current value of $e(t)$ can only be replaced by its expected value which is 0.

Weinstein et al., [70] proved that any function that is continuous in a bounded interval, may be well approximated using polynomials in that interval (Weierstrass's Approximation Theorem). Therefore, using polynomials as basis functions can be useful in the approximation of a wide range of dynamic systems.

One of the major advantages of the polynomial NARX model is that the output is a linear function of the parameters. Hence, the coefficients, $c_{p,q}(k_1,k_2,\cdots,k_m)$, can be approximated simply using linear regression. On the other hand, one major disadvantage of this model is that the number of parameters grows combinatorially as the number of the inputs to the system, in this case number of lagged input and output values, and the polynomial degree, M, increase. In the case of polynomial NARX models, assuming we are interested in M degree nonlinearities, the number of parameters is given by (3.3)

$$P_{full}=\sum_{i=0}^{M}\binom{(m+i-1)}{i}=1+m+\frac{m(m+1)}{2}+\frac{m(m+1)(m+2)}{6}+\cdots \tag{3.3}$$

where m is the number of past inputs and outputs together, $(m = n_u + n_y)$. As a result, the linear regression can become very large and possibly ill-conditioned. The regular approach is to use an orthogonal forward regression algorithm to incrementally build up the model until a desired accuracy has been achieved [3].

Another major disadvantages of the NARX models in general, is that they are black-box models. Even though they are very useful for either simulating or predicting the output of the system, they are lacking in providing insight into the system's structure or internal function. Additional tools, such as a transformation to the frequency domain [71, 72], or the decoupling methods developed herein, are needed to provide this insight.

To tackle either of these problems, I proposed a decoupling approach for multi-input single-output (MISO) polynomials, in particular for polynomial NARX models. To simplify the presentation, I will consider a generic input, $\mathbf{z} = [\ z_1 \quad z_2 \quad \cdots \quad z_m\ ]^T$, that contains all the inputs to the polynomial as Fig. 3.1 shows. In this approach the function, f, that is a multivariate polynomial, will be replaced with a transformation matrix followed by a bank of several univariate polynomials where the model output, y is the summation of the outputs of all branches. For NARX models, the the input will be

$$\mathbf{z}(t) = [\ y(t-1), \cdots, y(t-n_y), u(t), u(t-1), \cdots, u(t-n_u)\ ]^T$$

by considering a generic input vector $\mathbf{z}$, we avoid splitting it into the past input and output terms, and no longer need to consider the time dependence.

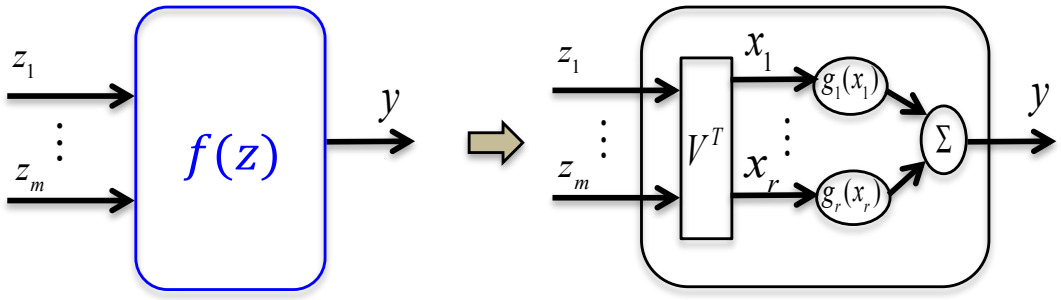

Figure 3.1: Decoupling a MISO Nonlinearity into SISO Polynomial Branches and a Transformation Matrix

Then the number of parameters will be given by:

$$P_{decoupled} = ((m + M) \times r) + 1 \tag{3.4}$$

note that r is the number of branches in decoupled model. Provided r is relatively small, this will be less for the full model – and the polynomial order can be increased since the number of parameters grows linearly with the polynomial order.

The output of the system on the left side of Fig. 3.1, can be expressed as:

$$y(\mathbf{z}) = f(z_1, \cdots, z_m) = \sum_{\ell=0}^{M} \sum_{k_1=0}^{m} \sum_{k_2=k_1}^{m} \cdots \sum_{k_n=k_n-1}^{m} \left[c_\ell(k_1, k_2, \cdots, k_\ell) \prod_{i=1}^{m} z_i \right] \tag{3.5}$$

where c_ℓ are the polynomial coefficients of degree ℓ. This coupled polynomial is a multivariate polynomial that might include cross-terms embracing several inputs. Thus, writing out the terms of orders 1 through M:

$$y(\mathbf{z}) = y_0 + \sum_{i_1=1}^{m} c_{i_1} z_{i_1} + \sum_{i_1=1}^{m} \sum_{i_2=i_1}^{m} c_{i_1,i_2} z_{i_1} z_{i_2} + \sum_{i_1=1}^{m} \cdots \sum_{i_n=i_{n-1}}^{m} c_{i_1,\cdots,i_M} z_{i_1} \cdots z_{i_M} \tag{3.6}$$

This coupled polynomial corresponds to the left side of Fig. 3.1. In this chapter, this will be replaced by the decoupled representation, shown on the right side of the same figure. As mentioned before, the MISO polynomial will be replaced by a transformation matrix, $\mathbf{V}$, and a bank of SISO polynomial branches, $g_i(.)$s. Then the outputs of all the branches will be

summed together to build the output signal of the MISO system, y. The estimated output of the decoupled model can be expressed as:

$$\hat{y}(\mathbf{z}) = c_0 + \sum_{i=1}^{r} g_i({\mathbf{v}_i}^T \mathbf{z}) \tag{3.7}$$

where the summation over i adds the outputs of the r branches together, as illustrated in Fig. 3.1. Therefore, $\mathbf{v}_i$ is the ith column of the transformation matrix, $\mathbf{V}$, and g_i is the univariate polynomial in the ith branch, given by:

$$g_i(x_i) = \sum_{j=1}^{M} c_{j,i} x_i^j \tag{3.8}$$

where $x_i = {\mathbf{v}_i}^T \mathbf{z}$ is the input to the polynomial, as shown in (3.7), in this representation, r, denotes the number of SISO branches in the decoupled structure and M, represents the polynomial degree. Let

$$\mathbf{V} = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \cdots & \mathbf{v_r} \end{bmatrix} \tag{3.9}$$

be the transformation matrix, where $\mathbf{v_i} \in \Re^{m \times 1}$, and

$$\mathbf{c} = \begin{bmatrix} c_0 & c_{1,1} & \cdots & c_{M,1} & c_{1,2} & \cdots & c_{M,r} \end{bmatrix}^T \tag{3.10}$$

be a vector containing all the polynomial coefficients, where $\mathbf{c} \in \Re^{(rM+1) \times 1}$.

3.1.1 Estimation of Decoupled MISO Polynomials

The MISO polynomial will be decoupled using an optimization based approach. Initially r will be treated as known and the methods for determining that will be discussed later.

Given N data points, the goal is to find the estimates of $\mathbf{V}$ and $\mathbf{c}$, that minimize the norm of the error between the output, $\mathbf{y}$, and that of the decoupled output, $\hat{y}(\mathbf{V}, \mathbf{c})$.

Cost Function

The cost function is defined to be the norm of the error between the real output and the estimated output:

$$V_N(\mathbf{V}, \mathbf{c}) = ||\mathbf{y} - \hat{\mathbf{y}}(\mathbf{V}, \mathbf{c})||_2^2 \tag{3.11}$$

where $\mathbf{y}$ and $\hat{\mathbf{y}}$ are N element vectors containing the measured output and estimated output, respectively.

The output of the decoupled polynomial can be also expressed as:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{c} \tag{3.12}$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{1}_N & \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_r \end{bmatrix} \tag{3.13}$$

$\mathbf{1}_N$ is an N element column vector of 1s and each $\mathbf{X}_i$ is a Vandermonde matrix based on the vector $\mathbf{x}_i$ whose t'th element is the value of the intermediate signal, $x_i(t)$, itself. The i'th intermediate signal evaluated at measurement point t is defined as:

$$x_i(t) = \mathbf{v}_i^T \mathbf{z}(t) \tag{3.14}$$

Thus the Vandermonde matrix will be set up as:

$$\mathbf{X_i} = \begin{bmatrix} \mathbf{x}_i & \mathbf{x}_i^2 & \dots & \mathbf{x}_i^n \end{bmatrix} \tag{3.15}$$

where the exponents are applied element by element and

$$\mathbf{x}_i = \begin{bmatrix} x_i(1) & x_i(2) & \dots & x_i(N) \end{bmatrix}^T \tag{3.16}$$

From (3.12), $\hat{\mathbf{y}}$ is linear in the polynomial coefficients while the regression matrix, $\mathbf{X}$, depends nonlinearly on the elements of the transformation matrix, $\mathbf{V}$, due to the Vandermonde matrix in (3.15). Thus, the minimization of the cost function can be done using a separable least squares (SLS) optimization [41].

As mentioned in the previous chapter, the key observation behind SLS optimization is that the parameters that appear linearly, in this case the elements of $\mathbf{c}$, can be written in closed-form as functions of the remaining parameters, $\mathbf{V}$. For any given $\mathbf{V}$, the least squares solution for $\mathbf{c}$ is given by:

$$\hat{\mathbf{c}}(\mathbf{V}) = \arg\min_{\mathbf{c}} ||\mathbf{y} - \mathbf{X}\mathbf{c}|| \tag{3.17}$$

which can be solved using ordinary least squares methods. Recall that $\mathbf{X}$ is dependent on $\mathbf{V}$ as discussed above. In Eq. (3.17), $\hat{\mathbf{c}}$ can be found using OLS method as discussed in Section 2.1.2.

At this point the only unknown parameter is $\mathbf{V}$ which affects the output nonlinearly because of the Vandermonde matrix (3.15) used to generate the matrix $\mathbf{X}$. Since $\hat{\mathbf{c}}$ is a function of $\mathbf{V}$ the optimization can be performed with respect to $\mathbf{V}$ only:

$$\hat{\mathbf{V}} = \arg\min_{\mathbf{V}} V_N(\mathbf{V}, \hat{\mathbf{c}}(\mathbf{V})) \tag{3.18}$$

provided that the dependence in (3.17) is taking into account.

The optimization problem in (3.18) is a nonlinear optimization and can be solved using a quasi-Newton algorithm such as a Gauss-Newton or Levenberg-Marquart algorithm [14]. In either case, one needs to compute the Jacobian of the model output. This can be done using the Variable Projection Algorithm [41], which computes the first order approximation to the Jacobian of the separated problem as:

$$\mathbf{J}_s = \mathbf{J}_v - \mathbf{J}_c(\mathbf{J}_c^T\mathbf{J}_c)^{-1}\mathbf{J}_c^T\mathbf{J}_v \tag{3.19}$$

where $\mathbf{J}_v$ and $\mathbf{J}_c$ are the Jacobians of the original, unseparated problem with respect to $\mathbf{V}$ and $\mathbf{c}$ respectively.

The Jacobian of $\hat{\mathbf{y}}$ with respect to polynomial coefficients is given by:

$$\mathbf{J}_c(k, M*(i-1)+j) = \frac{\partial \hat{\mathbf{y}}(\mathbf{z}(t))}{\partial c_{i,j}} = (\boldsymbol{v}_i^T \mathbf{z}(t))^j \tag{3.20}$$

Similarly, the Jacobian with respect to the elements of mixing vectors are:

$$\mathbf{J}_v(k, m*(i-1)+j) = \frac{\partial \hat{\mathbf{y}}(\mathbf{z}(t))}{\partial \boldsymbol{v}_i(j)} = g_i'({\boldsymbol{v}_i}^T \mathbf{z}(t)) z(t,j) \tag{3.21}$$

where $g_i'(x)$ is the derivative, of g_i with respect to its argument and $z(t,j)$ is the jth element of the vector $\mathbf{z}(t)$. For the purpose of this work, the Levenberg-Marquardt approach is used to find the optimal value of $\hat{V}$ in (3.18).

3.2 Extension to Polynomial NARX Models

In the case of a polynomial NARX model, which is the model that I am considering in this thesis, the input vector, $\mathbf{z}$, in (3.7) will depend on time, $\mathbf{z}(t)$, and will contain past inputs and past outputs to the system.

$$\mathbf{z}(t) = \left[\; y(t-1), \cdots, y(t-n_y), u(t-n_k), u(t-n_k-1), \ldots, u(t-n_k-n_u) \; \right]$$

where n_u and n_y are the number of past input and output samples used in the model and n_k is the input delay which is always greater than or equal to zero.

Stacking all the measurements of inputs and outputs at N different points together will

result in the input matrix $\mathbf{U}$ and the output vector $\mathbf{y}$:

$$\mathbf{U} = \begin{bmatrix} \mathbf{z}(1) & \mathbf{z}(2) & \dots & \mathbf{z}(N) \end{bmatrix}$$
$$\mathbf{y} = \begin{bmatrix} y(1) & y(2) & \dots & y(N) \end{bmatrix}^T$$

Having all the polynomial inputs and outputs at different measurement points, $\mathbf{U}$ and $\mathbf{y}$, the optimization in (3.11) can be solved.

3.2.1 Practical Details

Model structure selection, i.e, finding an appropriate number of branches, r, polynomial degree, M, and number of past inputs and outputs, m is done by minimizing the prediction errors in a separate testing data batch. Different models with various parameters will be scanned to find the model structure that results in the smallest prediction error in the testing data.

Note that the optimization (3.11) is generally non-convex regardless of using the Gauss-Newton algorithm or Levenberg-Marquardt approach. Thus, the optimization must be solved iteratively. In this case, the result may be sensitive to the initial solution. There is no guarantee that the optimization will reach the global optimum but by providing an appropriate initial guess at the solution, the likelihood of reaching the optimal solution is increased.

Different techniques for initialization based on tensor factorization for decoupled polynomial NARX model are proposed in Chapter 4. In the current Chapter, an ensemble of random initial solutions will be applied for each optimization.

As mentioned before, this optimization is an iterative procedure, so one or more stopping criteria are required to stop the optimization. The stopping criteria determine when the optimization result is good enough, hence continuing the optimization is no longer necessary. For the purpose of this thesis, a few stopping criteria have been considered:

- First, total number of 500 iterations has been considered. Therefore, if other criteria

have not been satisfied, the optimization will not go on for ever and will stop after 500 iterations.

- The second stopping condition is that if the cost function has not been improved more than 0.005% over 10 iteration the optimization process will stop.

- Finally, if the ridge size in the optimization becomes very large (in this case higher than 10000) the optimization will stop. This means the ridge size in the L-M step becomes very large.

If any of these criteria happens, the optimization will stop and the result of the optimization is considered as the "best" solution.

3.3 Benchmark Results

The polynomial NARX decoupling algorithm is applied to 2 benchmark data sets: measurement data from an electronic implementation of a forced Duffing Oscillator, known as the Silver-Box [73], and simulation data from the Bouc-Wen model, a nonlinear differential equation based friction model [74, 75]. These data are available through Nonlinear benchmark website (`www.nonlinearbenchmark.org`).

3.3.1 Silver-Box Benchmark

The Silver-Box has been used in the area of nonlinear systems as a simple but real example to study and validate different topics and procedures since 2004 [76, 77] and about a decade later it was stabilized as a benchmark problem for identification of nonlinear systems [73]. The Silver-Box is an electronic implementation of a forced Duffing oscillator: a nonlinear mechanical resonating system incorporating a moving mass, m, a viscous damping, d, and a nonlinear spring, $k(y)$. As shown in Fig. 3.2, this implementation is a second order closed-loop system with a simple nonlinearity in the feedback path. The electrical circuit represents

the relationship between the force, which is the input voltage *u(t)*, and the displacement, represented by the output voltage *y(t)*, using the differential equation:

$$m\frac{d^2y(t)}{dt} + d\frac{dy(t)}{dt} + k(y(t))y(t) = u(t) \tag{3.22}$$

The nonlinear spring is described by Eq. (3.23) which is a static but position-dependent stiffness:

$$k(y(t)) = a + by^2(t) \tag{3.23}$$

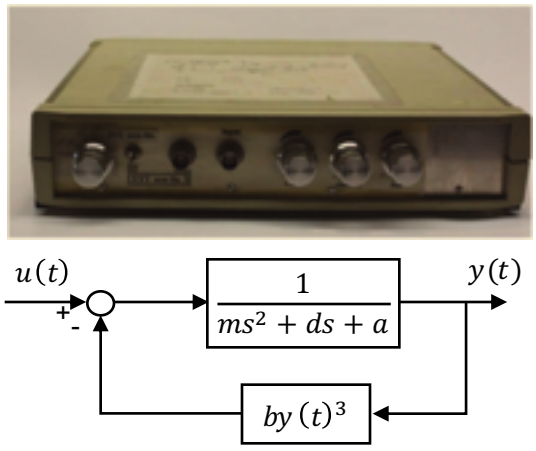

Figure 3.2: Upper: Real Silver-Box, Lower: Schematic Representation of the Silver-Box [20]

The data sets for this benchmark can be downloaded using the links provided at the nonlinear benchmark workshop website (`www.nonlinearbenchmark.org`). There are two sets of data for this benchmark. Fig. 3.4 illustrates the multi-sine data set consisting of two parts: the first part (40000 samples), which is used for validation of the obtained model, is a white Gaussian noise sequence with a slowly increasing amplitude, filtered by a 9th order discrete time Butterworth filter with a 200Hz cut-off frequency. The next part of the reference signal, which is used for identification and testing purposes, consists of 10 realizations (9 realizations used for identification and 1 realization for testing) of a random

phase odd multi-sine. The period of the multi-sine is 8192 samples. The data were sampled at $f_s = 610.35$Hz. Each realization of the random odd multi-sine is followed by 100 zeros (as an indicator to show the start of a new realization). The second sets of data, shown in Fig. 3.5, is a sweep-sine signal with the same sampling rate. The sweep-sine data set is used as another validation data set. Model accuracy is calculated using the RMS model fit:

$$FIT = 100 \times \left(1 - \sqrt{\frac{\sum_{t=1}^{N_t}(y(t) - \hat{y}(t))^2}{\sum_{t=1}^{N_t}(y(t) - y_{avg})^2}}\right) \tag{3.24}$$

It can be noted that the Silver-Box matches the model with moving poles, described in previous chapter perfectly.

In (3.24), $\hat{y}$ can be either the simulated output or the predicted output. The predicted output as shown in the upper part of Fig. 3.3, is the model response at some amount of time in future (in this work, 1-step ahead) using the current and past values of measured inputs and outputs as well as the initial conditions, whereas the simulated output which is the model response when the model has only access to the measured inputs and initial conditions as illustrated in the lower part of Fig. 3.3. So for the prediction output

$$\hat{y}(t) = f(u(t), u(t-1), ...y(t-1), ...)$$

and then for the simulation output

$$\hat{y}(t) = f(u(t), u(t-1), ...\hat{y}(t-1), ...)$$

Therefore, the prediction error defines as the difference between the measured output and the predicted output and the simulation error is the difference between the measured and simulated outputs as well.

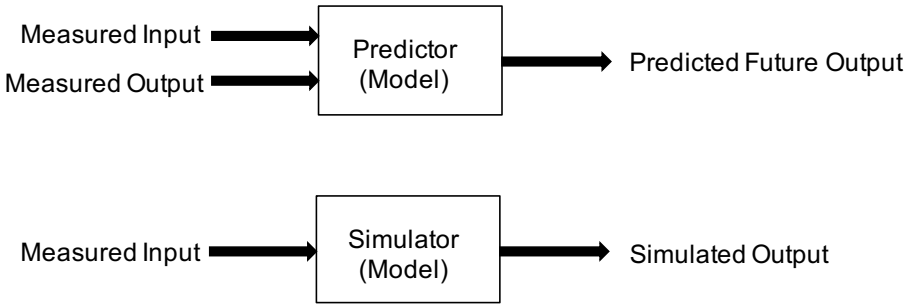

Figure 3.3: Upper: Schematic of a Predictor. Lower: Schematic of a Simulator

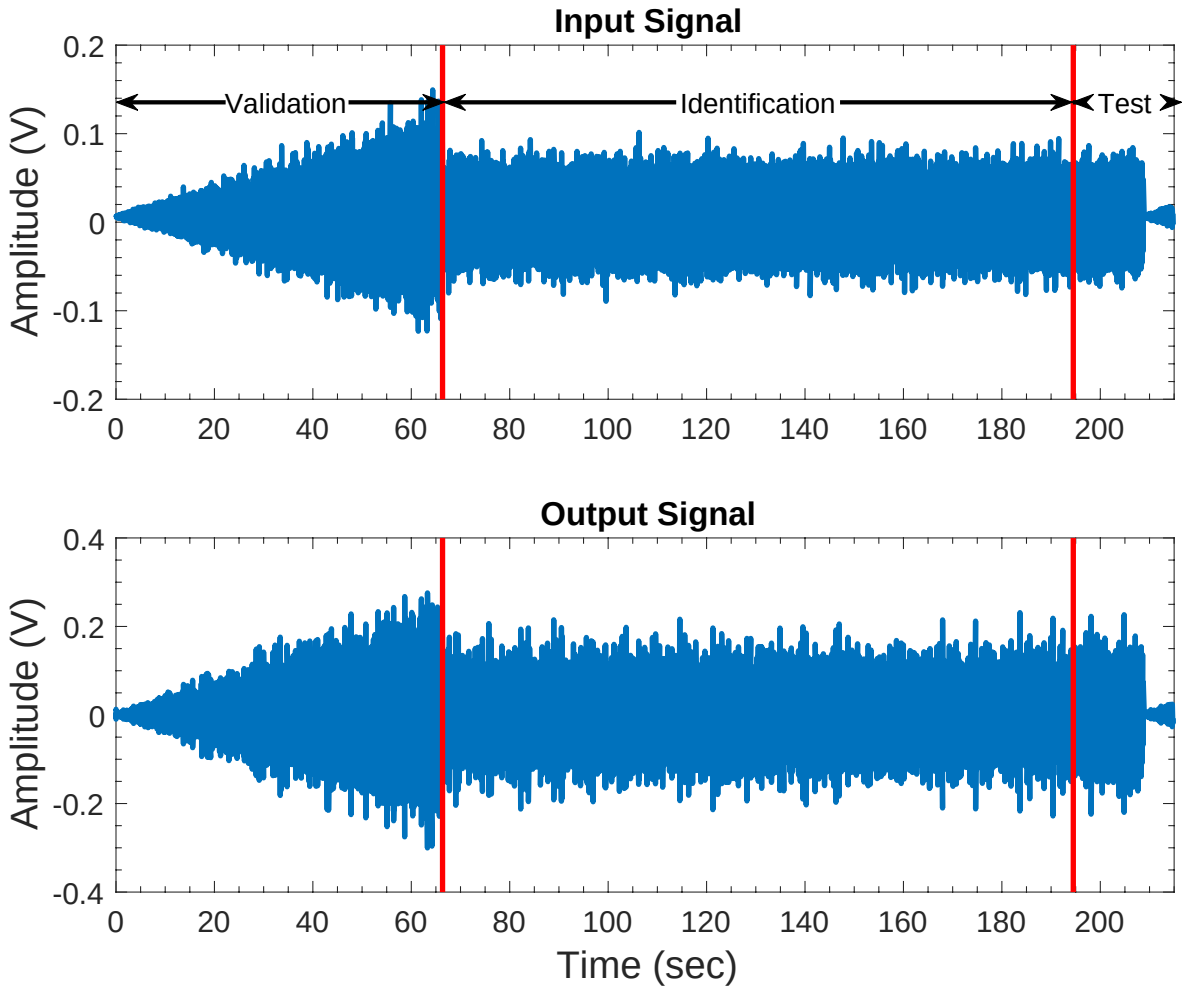

Figure 3.4: Multi-Sine Silver-Box Reference Signal. Upper: Input Signal. Lower: Output Signal

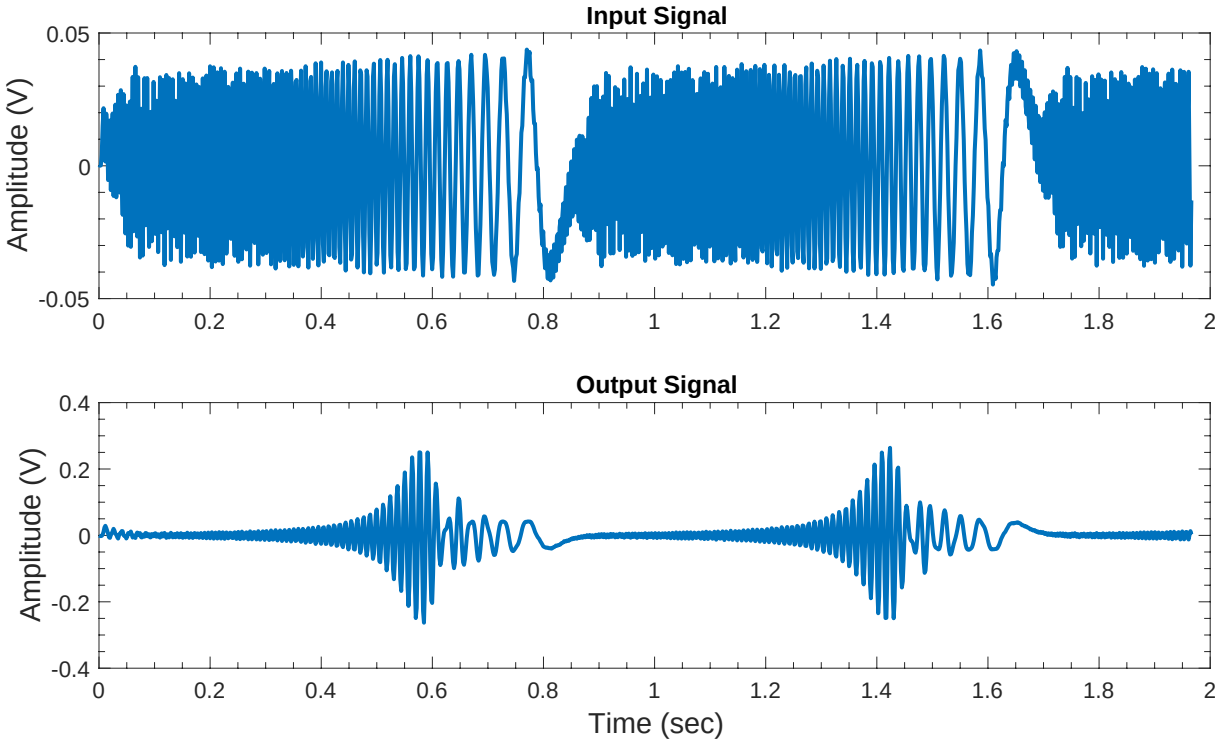

Figure 3.5: Sweep-Sine Silver-Box Validation Signal. Upper: Input Signal. Lower: Output Signal

3.3.2 Application of the Decoupling Algorithm to the Silver-Box Benchmark

In practice, several choices of the lag numbers for the input and output and for the polynomial degree have been scanned, fitting the best such polynomial NARX model to the data using random initialization. Full P-NARX models with $3rd$ degree nonlinearities and between 1 to 5 lagged inputs and outputs and an input delay of 0 to 5 samples were fitted to data. A model with 3 lagged inputs and 3 lagged outputs and no input delay generated the best predictions in the test data and was therefore chosen as the optimal full NARX model. The polynomial NARX decoupling algorithm was then applied to this model structure. Note, however, that the number of branches, r, was also unknown. Therefore, a similar scanning approach was used to choose the optimal number of branches. Table 3.1 shows the accuracy of the decoupling algorithm for different numbers of branches and polynomial degrees. As it

is illustrated in the table, using $3rd$ degree nonlinearities, the prediction accuracy in the test data started to decrease after adding the $4th$ branch. Increasing the polynomial degree also has the same effect and results in decreasing the accuracy of the identified model. Thus, a model with 4 branches and $3rd$ degree nonlinearities with 3 lagged inputs and outputs was selected as the best model.

Number of Branches	Polynomial Degree	Number of Parameters	Accuracy (Fit %)
1	3	10	98.96%
2	3	19	99.78%
3	3	28	99.79%
4	3	37	99.81%
5	3	46	99.78%
4	5	45	99.81%

Table 3.1: Accuracy of Decoupled NARX Model in Testing Data for the Silver-Box System Using 100 Random Starting Points

For the Silver-Box benchmark, (3.3) generates 84 parameters, but the orthogonal forward regression algorithm eliminated 24 of these parameters. Thus, the full P-NARX model actually had 60 parameter as compared to the 37 parameters in the final decoupled polynomial NARX calculated from (3.4). Even for a model as small as the Silver-Box (only 3 past inputs and 3 past outputs and also just a 3rd degree nonlinearity), there is a significant reduction in the number of parameters.

Figure 3.6 shows the simulation and prediction errors for identified model in the triangular validation data and Fig. 3.7 shows the same signals for the sweep-sine validation data set for the model that provides the best testing performance. As Table 3.2 and both figures confirm, the model is more accurate in 1-step ahead prediction than in simulation. Note that the performance in validation of the 1-step ahead prediction was similar to that achieved in the testing data.

	Simulation	1-Step Ahead Prediction
Multi-Sine(Fit% $\pm Std.$)	98.89% $\pm$0.27	99.77% $\pm$0.028
Sweep-Sine(Fit% $\pm Std.$)	97.41% $\pm$0.92	99.65 $\pm$0.057

Table 3.2: Accuracy of Decoupled NARX Model in Validation Data for the Silver-Box System Using Random Initialization (Average of 1000 Random Runs)

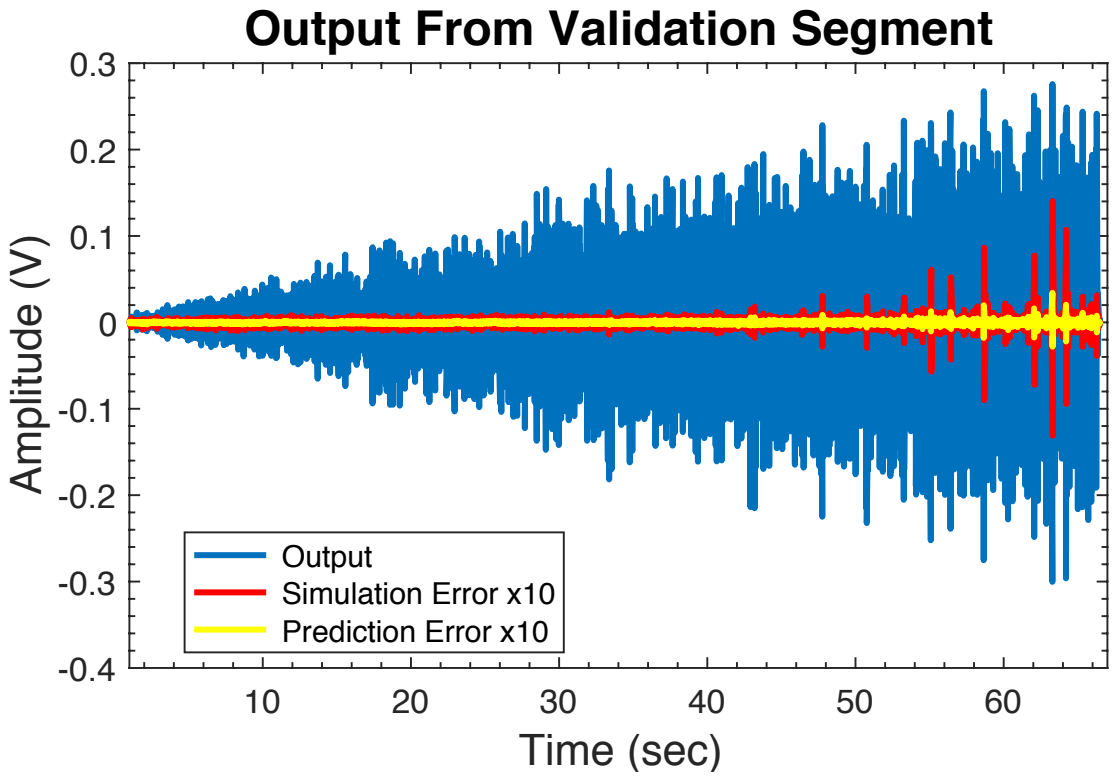

Figure 3.6: Prediction and Simulation Errors on the White Gaussian Noise Sequence Filtered by a 9th Order Discrete Time Butterworth Filter Data Using the Best Random Initialization.

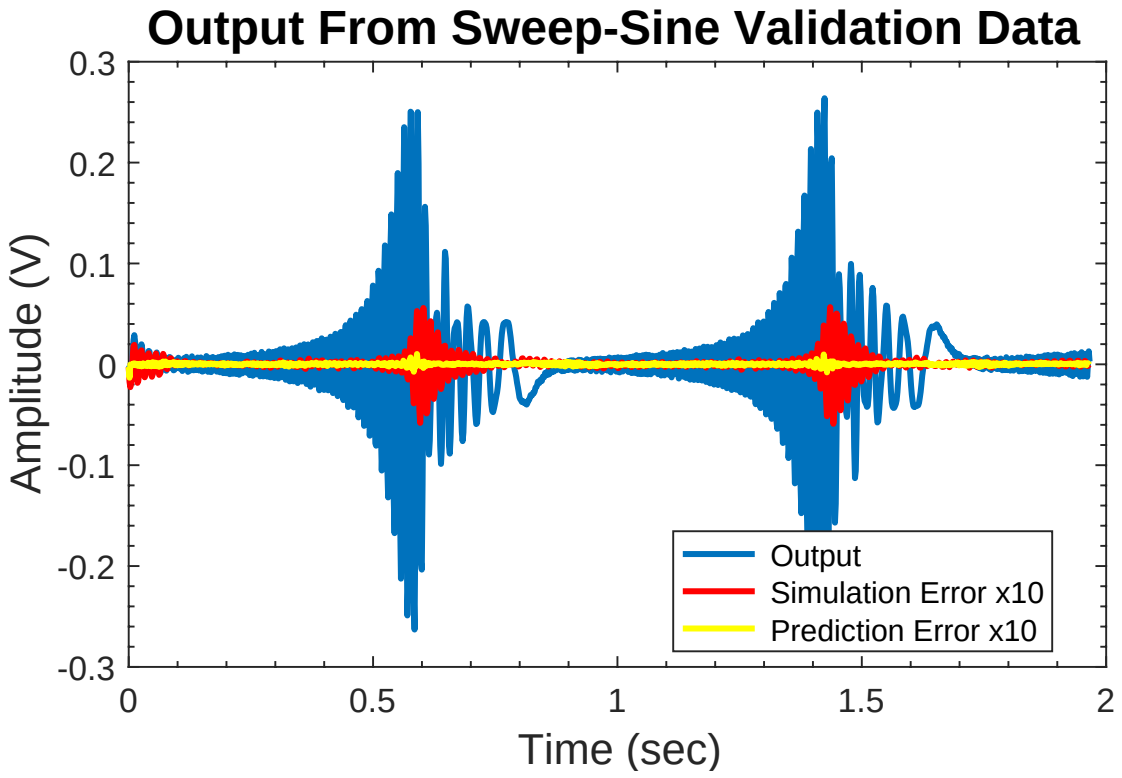

Figure 3.7: Prediction and Simulation Errors on the Sweep-Sine Validation Data Using the Best Random Initialization.

Figures 3.8 and 3.9 show the magnitude frequency responses of the Finite Impulse Response (FIR) filters that are applied to the input and output terms, respectively, in each individual branch in the decoupled model. It is not instructive to consider the filters in these branches as a recursive filters, since all branches receive the system output and not the output of any of the individual linear elements. That is the reason why the input and output terms are presented separately.

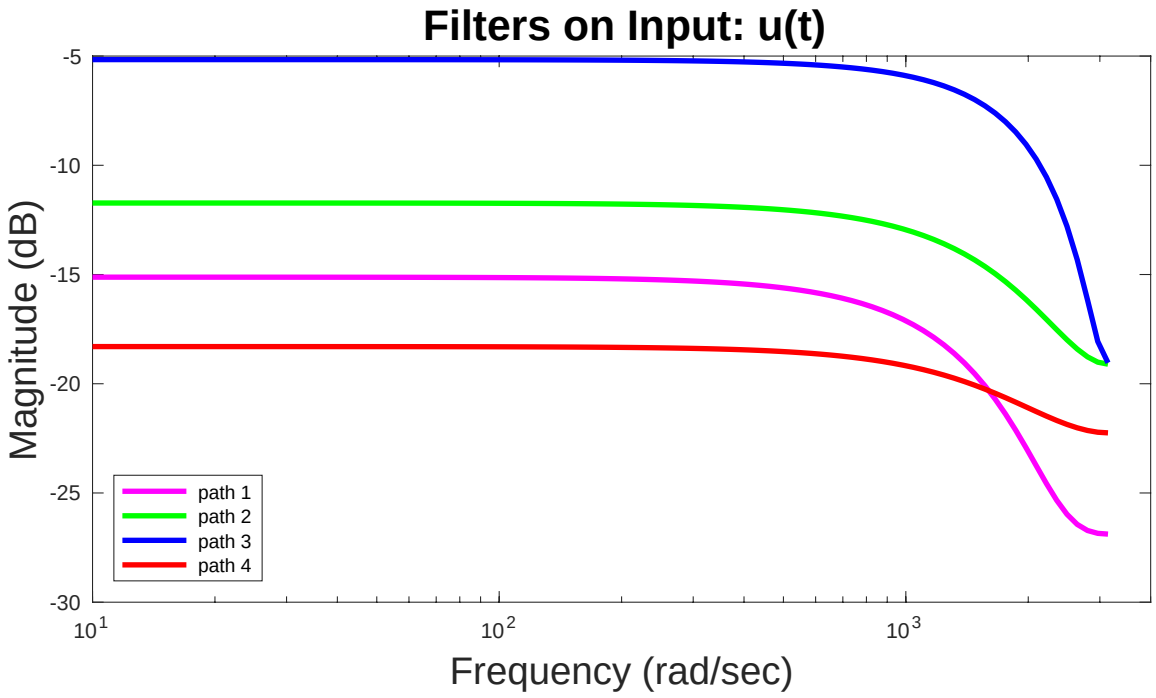

Figure 3.8: Frequency Response Magnitudes for the Input Terms of the Filters in the 4 Branches of the Best Random Initialization Fit%

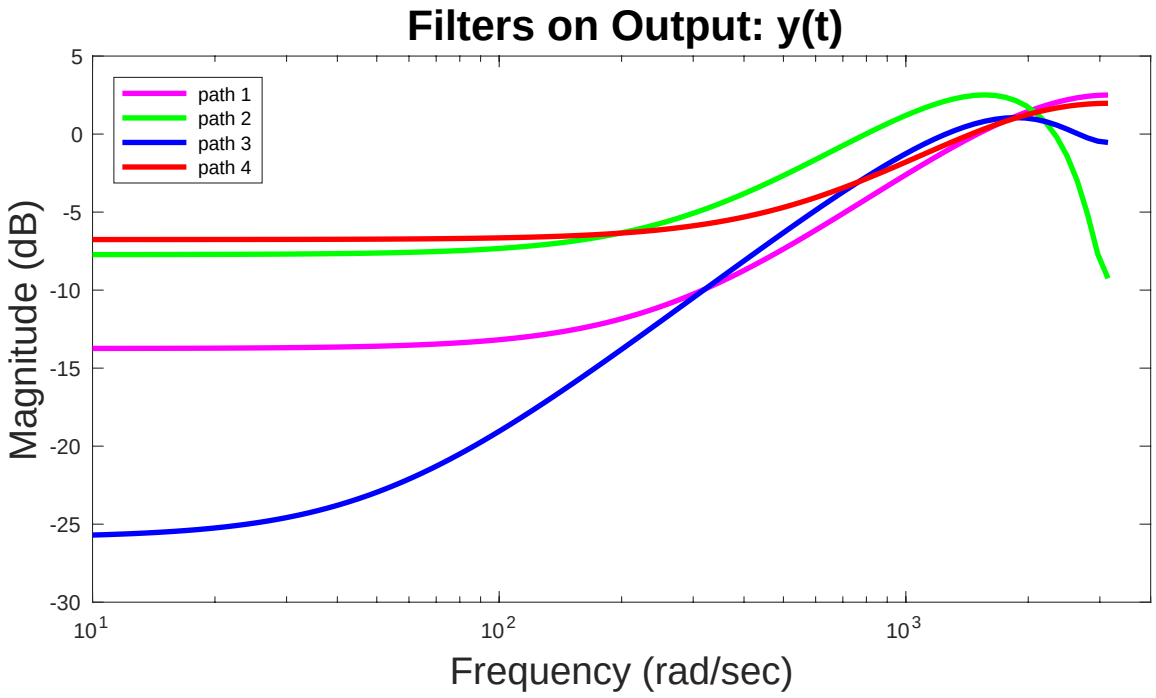

Figure 3.9: Frequency Response Magnitudes for the Output Terms of the Filters in the 4 Branches of the Best Random Initialization Fit%

3.3.3 Bouc-Wen Benchmark

The second data set that is used to test and validate the decoupling of polynomial NARX models is the Bouc-Wen [78, 79, 80, 81] benchmark system. The Bouc-Wen model, models friction in a mechanical system and friction is a form of hysteresis. This benchmark represents hysteresis using a set of coupled, nonlinear differential equations. Hysteresis is a very common phenomenon that exists in diverse fields of science and engineering such as aerodynamics [82], biology [83], ecology [84], psychology [85] and etc. The defining property of a hysteretic system is the persistence of an input-output loop as the input frequency approaches zero [74]. Hysteretic systems are nonlinear systems naturally and do not have fading memories as defined by Boyd and Chua [27].

A detailed description of the Bouc-Wen benchmark system can be found in [74]. This benchmark system consist of a single mass, m_L, spring, k_L, and damper system, c_L, with a hysteretic friction term, $z(.)$. In continuous-time, the system can be expressed by the second order differential equation:

$$m_L\ddot{y}(t) + c_L\dot{y}(t) + k_L y(t) + z(y(t), \dot{y}(t)) = u(t) \tag{3.25}$$

where $y(t)$ is the displacement and represents the output and $u(t)$ represents the external force which is the input. The hysteretic friction term, $z(.)$, obeys this differential equation:

$$\dot{z}(y(t), \dot{y}(t)) = \alpha\dot{y}(t) - \beta(\gamma|\dot{y}(t)||z(t)|^{\nu-1}z(t) + \delta\dot{y}(t)|z(t)|^{\nu}) \tag{3.26}$$

Using the Bouc-Wen parameters, α, β, γ, δ and ν, the smoothness of the hysteresis loop can be tuned and shaped. Table 3.3 lists the physical parameter values for this benchmark.

Parameter	m_L	c_L	k_L	α	β	γ	δ	nu
Value (SI unit)	2	10	5×10^4	5×10^4	1×10^3	0.8	-1.1	1

Table 3.3: Physical Parameters of the Bouc-Wen System

By integrating (3.25) and (3.26) and using the Newmark method at a sampling rate of 15000Hz, simulation data for the Bouc-Wen system are generated. Note that the data are then low-pass filtered and down sampled to a final sampling rate of 750Hz. Finally, noise was added to the output such that the SNR was about 40 dB. Figure 3.10 shows the input and output signals used as identification data for the Bouc-Wen benchmark. Two such data-sets were generated: one was used for parameter estimation, while the second was used for structure selection. These data were generated using the default multi-sine input excitation which has a duration of $N = 8192$ but, as suggested in [50], the RMS force was increased from 50N to 55N, hence, the identification data would cover a larger range than the validation sets. Also, two noise-free validation data sets are provided by the benchmark: a multi-sine input and a swept-sine input. The performance of the model is presented using the RMS error in Eq. (3.27) as recommended by the benchmark:

$$e_{RMS} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_{model}(t) - y(t))^2} \tag{3.27}$$

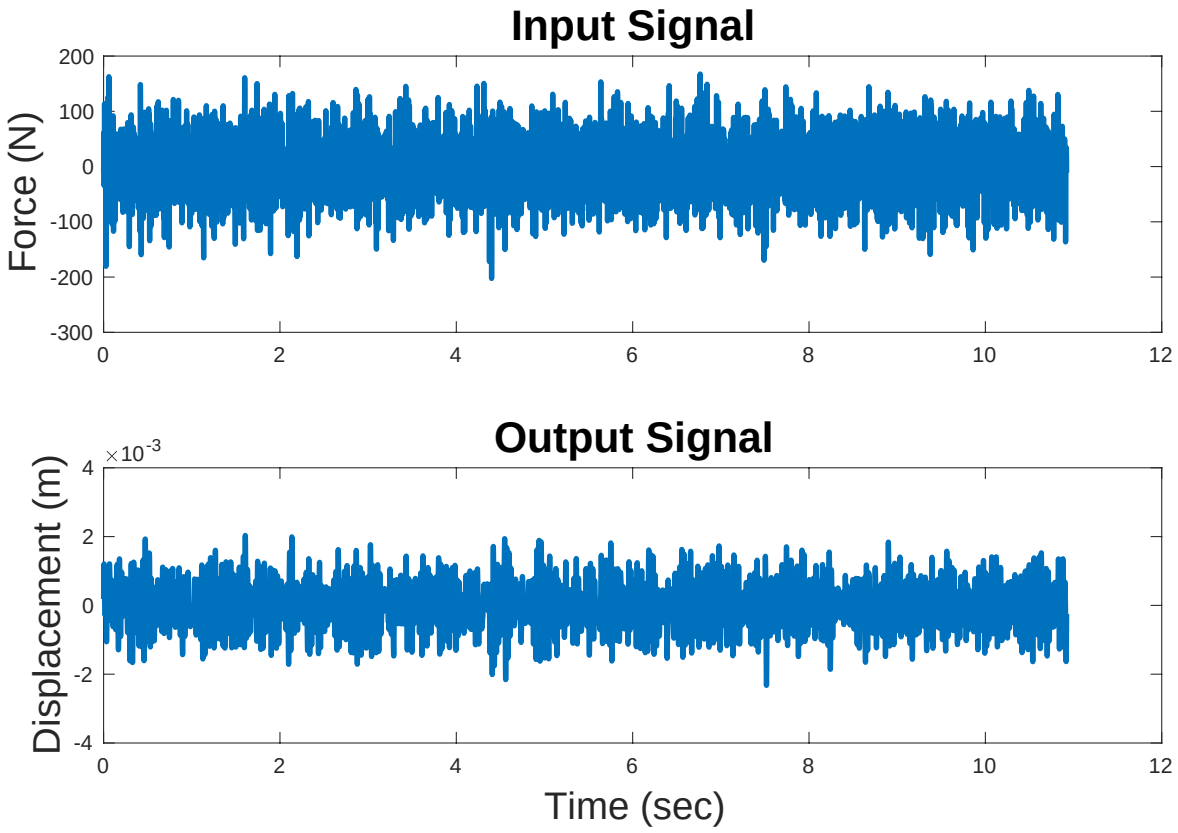

Figure 3.10: Multi-sine Identification Data from the Bouc-Wen System. Upper: Input Signal. Lower: Output Signal

The swept-sine validation data set is shown in Fig. 3.11. The amplitude of the input is set to 40N. Also the frequency bands from 20 to 50Hz is covered at a sweep rate of 10Hz/min.

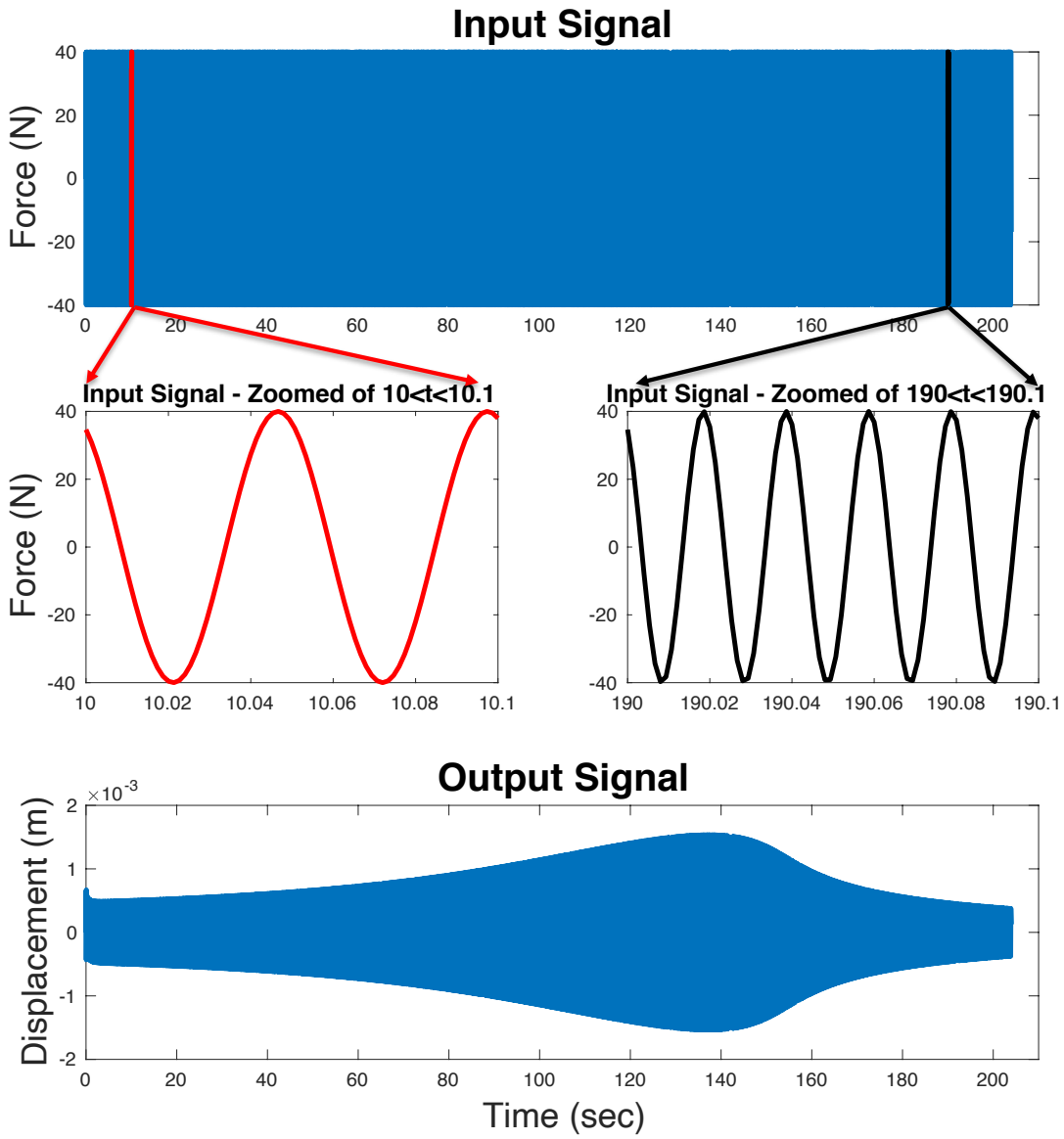

Figure 3.11: Swept-Sine Validation Data from the Bouc-Wen System. Upper: Input Signal. Lower: Output Signal. 2 Middle Figures Show the Input Signal for Time Intervals of $10 < t < 10.1$ and $190 < t < 190.1$ for Better Visualization of the Input Signal

One of the challenges with this system is that the nonlinearty is applied to an internal state $z(.)$ which can not be measured directly, whereas the model that we are using in this work, the polynomial NARX model, is an input-output model. So in order to model the system's nonlinear structure, the internal state has to be reconstructed from just the input and output measurements.

Pruned NARX Model

To fit a polynomial NARX model to the identification data, an orthogonal least squares pruning algorithm was used. This model included $3rd$ degree nonlinearities with all possible cross terms; assuming a system with 3 inputs, z_1, z_2, z_3, and 3rd degree nonlinearity, all possible terms would include 1st, 2nd and 3rd order terms:

$$\underbrace{\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}}_{1st\ order\ terms} \quad \underbrace{\begin{bmatrix} z_1z_1 & z_2z_2 \\ z_1z_2 & z_2z_3 \\ z_1z_3 & z_3z_3 \end{bmatrix}}_{2nd\ order\ terms} \quad \underbrace{\begin{bmatrix} z_1z_1z_1 & z_1z_2z_2 & z_2z_1z_1 & z_2z_2z_2 & z_3z_1z_1 & z_3z_2z_2 \\ z_1z_1z_2 & z_1z_2z_3 & z_2z_1z_2 & z_2z_2z_3 & z_3z_1z_2 & z_3z_2z_3 \\ z_1z_1z_3 & z_1z_3z_3 & z_2z_1z_3 & z_2z_3z_3 & z_3z_1z_3 & z_3z_3z_3 \end{bmatrix}}_{3rd\ order\ terms}$$

In order to find the best model structure for the provided data set, different models with various number of inputs and outputs were scanned. The number of delayed input and output terms was scanned from 0 to 16 of each type, while the input delay was scanned from 0 to 2 samples. The model that achieved the lowest prediction error on the testing data was selected as the best model structure. This model contained 15 lagged input terms and output terms with no input delay. In terms of the number of parameters, if we were interested in a full polynomial NARX model, this model would include 5456 coefficients based on (3.3) but the pruned model only contained 638 coefficients.

3.3.4 Application of the Decoupling Algorithm to the Bouc-Wen Benchmark

Once we had found the polynomial NARX model that best fitted the data, in this case a model with 15 past input and output terms, we used that structure to obtain the best decoupled model. For the purpose of the Bouc-Wen model, the number of branches was scanned from 1 to 6 with the polynomial nonlinearity in each branch of degree up to 11. As Table 3.4 summarizes the results, the model with 5 branches and a 9th degree nonlinearity resulted in the most accurate 1-step ahead predictions in the testing data.

Number of Branches	Polynomial Degree	Number of Parameters	Accuracy (Fit %)
4	3	133	98.30%
5	3	166	98.46%
6	3	199	98.33%
5	5	176	98.49%
5	7	186	98.54%
5	9	196	98.55%
5	11	206	98.50%

Table 3.4: Average Accuracy of Decoupled NARX Model in Testing Data for the Bouc-Wen System Using 100 Random Starting Points

Figure 3.13 shows the simulation and prediction error o the swept-sine validation data set for the model that provides the best testing performance. Table 3.5 compares the accuracy of the full P-NARX model and the decoupled P-NARX model as well as the number of parameters. As illustrated in the Table, the decoupled P-NARX model performs better, likely because it includes much higher degree nonlinearities while still having far fewer parameters than the full P-NARX model. Both models used the same number of past inputs and outputs.

Model	Number of Parametersuc.	Multi-Sine Simulation Accuracy	Multi-Sine Prediction Accuracy
Full P-NARX	638	Unstable	98.52%
Decouple P-NARX, 5branch	196	91.46%	98.89%

Table 3.5: Comparison of the Full P-NARX Model and the Decoupled P-NARX Model- Best Random Initialization

	Simulation	1-Step Ahead Prediction
Multi-Sine(Fit%$\pm Std.$)	91.86% $\pm$0.29	97.13% $\pm$11.49
Sweep-Sine(Fit% $\pm Std.$)	97.08% $\pm$0.45	99.42% $\pm$0.037

Table 3.6: Average Accuracy of Decoupled NARX Models of the Bouc-Wen System Using Random Initialization

As illustrated in upper-right corner of Table 3.6, the standard deviation for 1-step ahead prediction in multi-sine validation data is very high. Examining the individual trials, it is evident that the large standard deviation is due to 1 very bad model out of the ensemble of 100 runs. The prediction and simulation outputs for this single model, applied to the multi-sine validation signal, are plotted in Fig. 3.12, which shows that the poor performance, in prediction only, is due to an extremely large transient, but the final cost function value for this trial is 2.0842×10^{-4} which is relatively small. This behavior indicates that random initialization can not be reliable at all times. Since the model includes 15 past inputs and outputs, for the first few points there is not enough information for the model to result in a good prediction, therefore a very bad transient might happen. Table 3.7 shows the average accuracy for all 100 models after removing the first 15 points of the response, hence eliminating the start-up transients in the 1-step ahead predictions. The swept-sine results are unchanged, but the fit percent for the multi-sines are increased in both simulation and 1-step ahead prediction and the standard deviation of the prediction is greatly reduced.

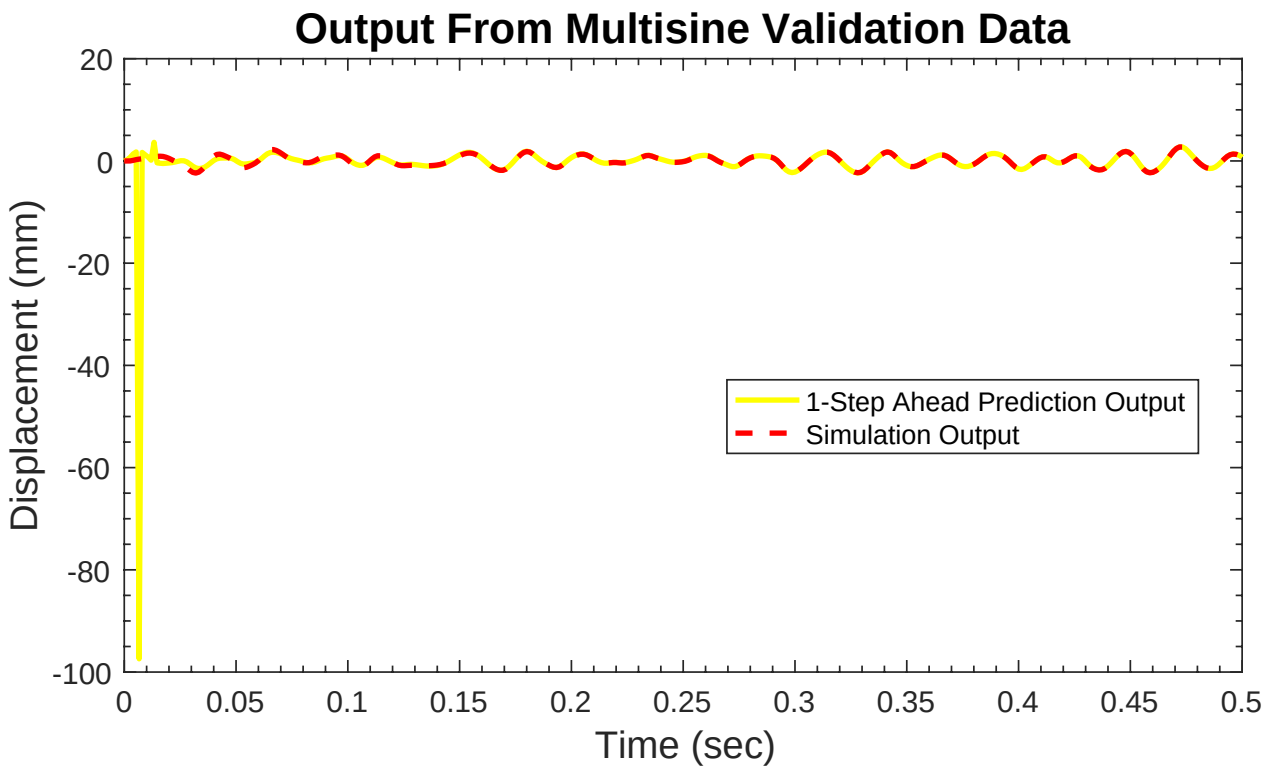

Figure 3.12: 1-Step Ahead Prediction and Simulation Outputs on the Multi-Sine Validation Data Using the Worst Randomly Initialized Model

	Simulation	1-Step Ahead Prediction
Multi-Sine(Fit%$\pm Std.$)	93.45% $\pm$0.34	99.37% $\pm$0.041
Sweep-Sine(Fit% $\pm Std.$)	97.08% $\pm$0.45	99.42% $\pm$0.037

Table 3.7: Average Accuracy of Decoupled NARX Models in the Validation Data for the Bouc-Wen System Using Random Initialization After Removing the first 15 Transient Points

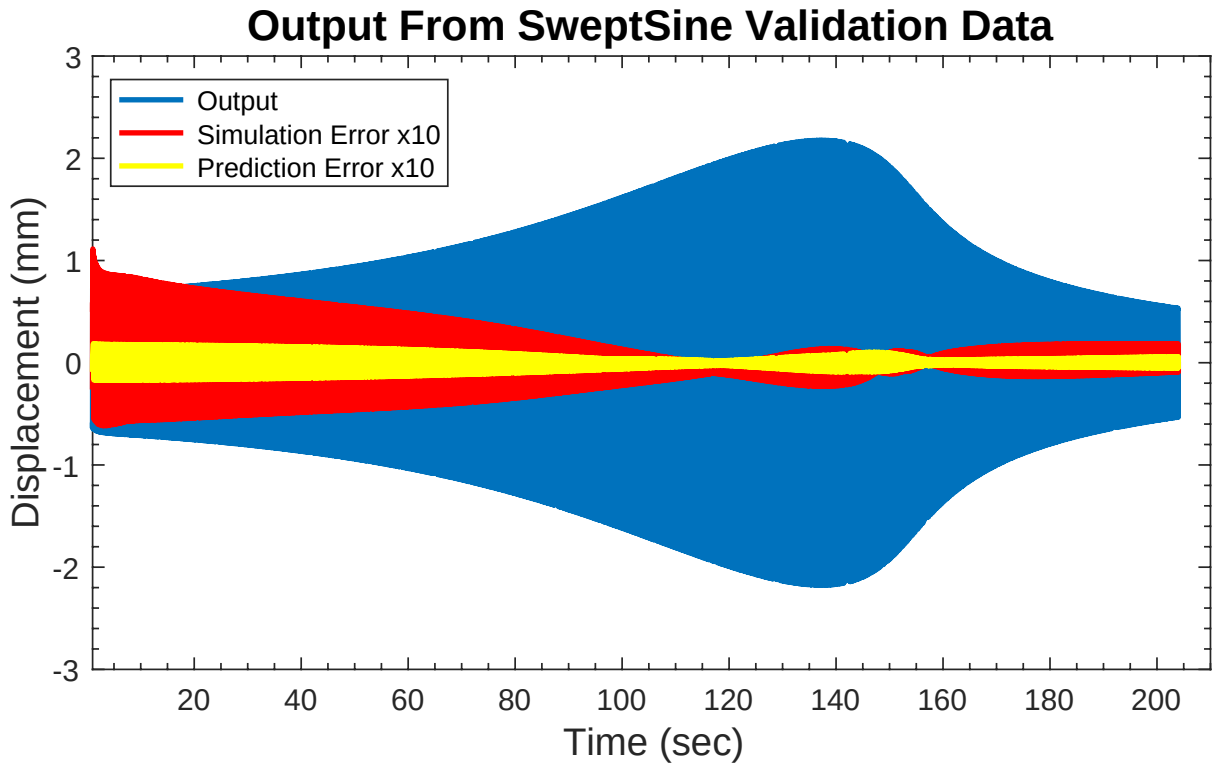

Figure 3.13: Prediction and Simulation Errors on the Sweep-Sine Validation Data Using Best Random Initialization

Figure 3.14 and 3.15 show the frequency response magnitude for the input and output terms in each of the branches.

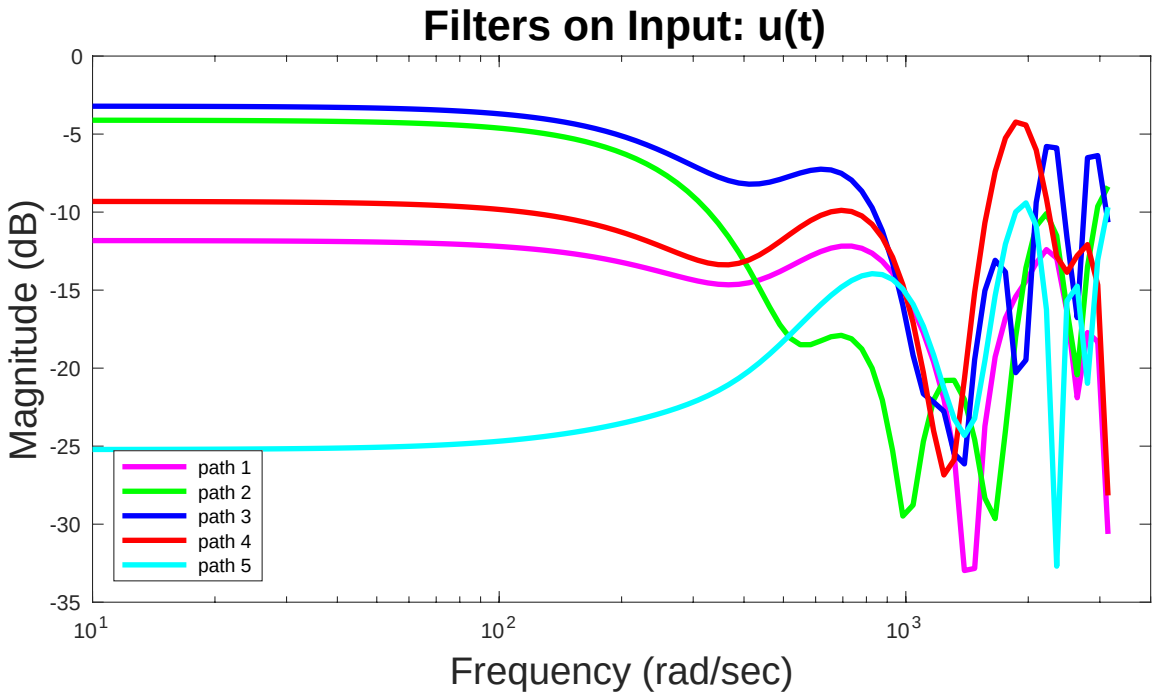

Figure 3.14: Frequency Response Magnitudes for the Input Terms of the Filters in the 5 Branches Using Random Initialization

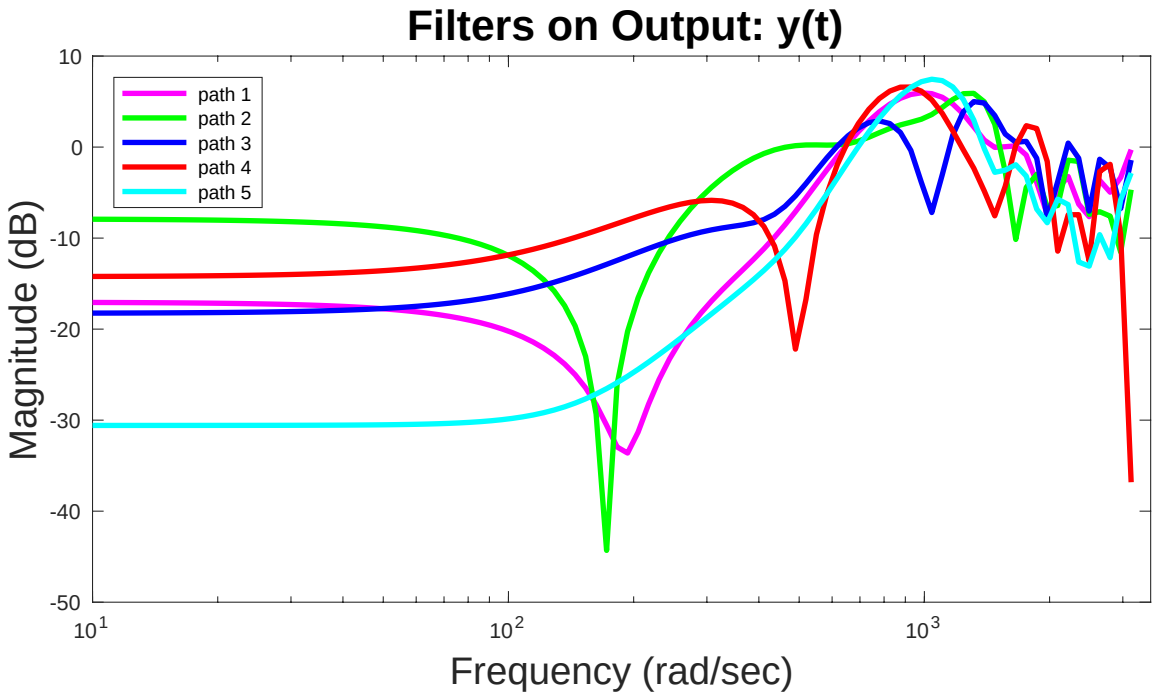

Figure 3.15: Frequency Response Magnitudes for the Output Terms of the Filters in the 5 Branches Using Random Initialization

Chapter 4

Optimization Initialization Techniques for Polynomial NARX Model Decoupling [1]

As mentioned in the previous chapter, the optimization problem (3.18), involved in the decoupling algorithm, is an iterative optimization over a non-convex error surface. Since the decoupling algorithm uses separable least squares, therefore, the iterative optimization is only performed over the entries in the mixing matrix, and the polynomial coefficients are treated as closed-form functions of the mixing matrix. As such a good initial estimate of the optimization variables, $\mathbf{V}$, is required. A good initial guess in an iterative optimization, not only will speed up the iteration progress considerably, but also increases the probability of finding the global solution.

In this chapter different methods for initialization of the optimization used to decouple a polynomial NARX model are proposed. Algorithm 1 summarizes the identification approach for a decoupled polynomial NARX (DP-NARX) model from input-output data. The second

[1] Portions of this chapter have been accepted for publication in the Journal of Mechanical Systems and Signal Processing [86] and presented (remotely due to the COVID-19 pandemic) at the 21st IFAC World Congress, IFAC 2020, Berlin, Germany [87].

step of the following algorithm is the focus of this chapter.

Algorithm 1 Decoupling Polynomial NARX Model from Input-Output Data

Assume:

- N points of input/output data are given
- The model structure is selected as mentioned in Section 3.2.1: number of input and output lags, input delays, number of branches and polynomial order

Then:

1. Fit a polynomial NARX model to the provided data using standard approaches such as an orthogonal forward regression algorithm [3]

2. Find an initial value of $\mathbf{V}$ using one of the approaches described in this chapter

3. Use the initial estimate of $\mathbf{V}$ from Step 2 and refine it using the separable least squares based optimization. This optimization is done with respect to the model output. Note that the polynomial coefficients, $\mathbf{c}$, will be estimated during each step of the optimization using ordinary least squares as discussed in Chapter 3

4.1 Factoring of the Jaccobian

Initially Dreesen et al. [51] proposed a method for decoupling MIMO polynomials. His technique uses the Canonical Polyadic Decomposition (CPD) to decompose a tensor constructed by stacking the Jacobian matrices of the original MIMO polynomial evaluated at several operating points. The Jacobian matrices contain the partial derivatives of each output with respect to each input evaluated at several measurement points. Fig. 3.1 and Fig. 4.1 show the decoupling of both MISO and MIMO polynomials, respectively. The main difference is that in case of multiple-output polynomials, a second transformation matrix, W, will appear that combines the outputs of the univariate polynomial branches to form the various outputs of the polynomial, whereas in the single-output case this becomes a simple summation.

Fig. 4.2 shows the function $f(z)$ and the coloured squares show local linear approximations, i.e. the Jacobian of $f(z)$ from Fig. 4.1 evaluated at different measurement points. Each of these Jacobians is a matrix that corresponds to the colored blocks in Fig. 4.3. Stacking these matrices together, will produced a 3-way tensor which was then factored using the CPD [52]. As the structure on the right hand side of Fig. 4.3 shows, the factors resulting from the decomposition include two transformation matrices, W and V and the matrix H which contains the first degree information of the model.

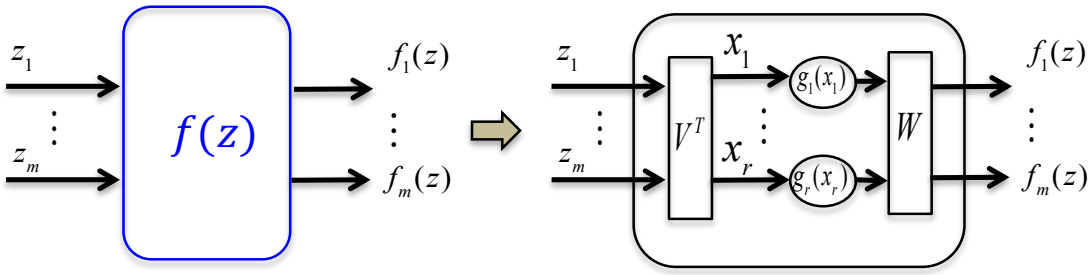

Figure 4.1: Decoupling a MIMO Nonlinearity into SISO Polynomial Branches and Two Transformation Matrices

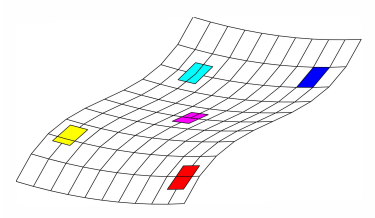

Figure 4.2: Surface of a Jacobian

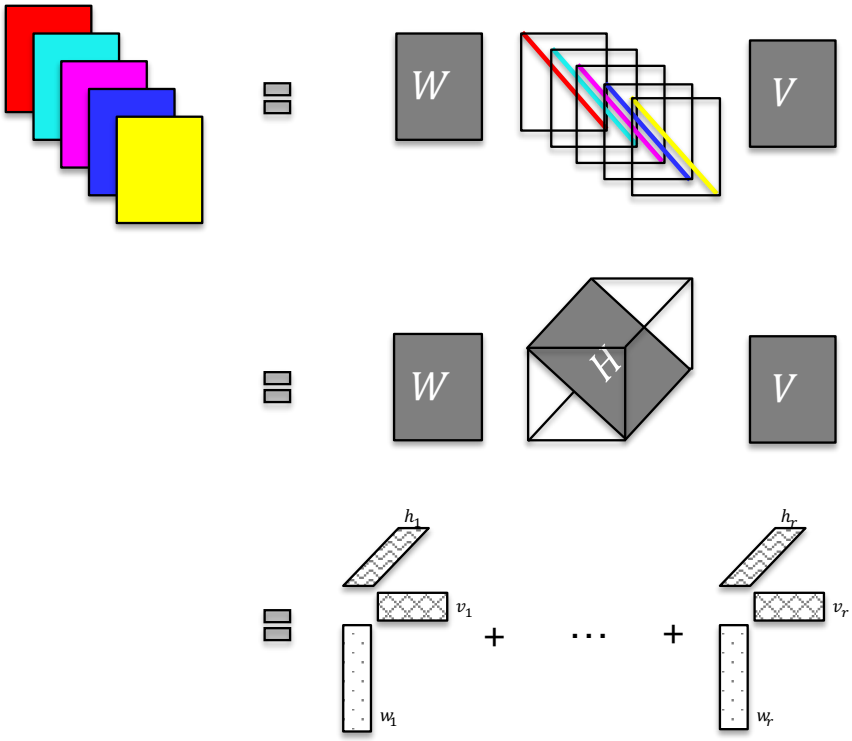

Figure 4.3: Decoupling a Tensor Using Canonical Polyadic Decomposition

Kruskal in [88] proved the uniqueness of the factors of triple product decompositions of 3rd and higher-order tensors, which is not the case for 2-way tensors, known as matrices. Matrix factorization is not unique due to the possibility of using a similarity transform. For any invertible transformation matrix, there is a different representation of the system, thus in the case of a decomposition, the decomposed factors are not unique. Note that matrix factorizations impose additional structure in order to guarantee uniqueness, for example in the QR decomposition, the R factor will be upper triangular and the Q factor will be orthonormal. Uniqueness of the CPD is an important property that allows to recover the individual summands from a tensor. In [88] a series of six uniqueness theorems have been proved and assured that the factorization of a 3-way tensor is always unique.

For the purpose of this work, where the system is single-output, if one follows the approach used in [51] and computes the Jacobian, the result is a vector (4.1). Evaluating (4.1) at several measurement points and stacking the results produces a matrix (4.2). Note that, unlike a 3rd

or higher order tensor, matrix factorizations without imposing additional structure cannot be unique. Hence the reliable performance of the Alternating Least Squares (ALS) algorithm used to compute the CPD is lost. Thus, for a single-output polynomial, written in decoupled form:

$$\mathbf{J}_y = \frac{\partial \mathbf{y}(\mathbf{z})}{\partial \mathbf{z}} = \sum_{i=1}^{r} g_i'(\boldsymbol{v}_i{}^T \mathbf{z}) \boldsymbol{v}_i{}^T \tag{4.1}$$

$$\mathbf{J} = \mathbf{V}\mathbf{W}^T \tag{4.2}$$

where $\mathbf{W}$ contains the derivatives of the SISO polynomials with respect to their inputs, evaluated at all measurement points. Since the Jacobian is a matrix, rather than a 3-way tensor, the powerful uniqueness results for tensor factorization no longer apply, since for any inevitable transformation matrix (of appropriate size), one can replace $\mathbf{V}\mathbf{W}^T$ with $\mathbf{V}\mathbf{M}^{-1}\mathbf{M}\mathbf{W}^T$(so that the factors become $\mathbf{V}\mathbf{M}^{-1}$ and $\mathbf{M}\mathbf{W}^T$). Thus there are an infinite number of possible factorizations. The transformation $\mathbf{M}$ will (in general) destroy any structure imposed on a particular matrix factorization. Given this lack of uniqueness, factoring the Jacobian does not produce a suitable initialization.

4.2 Hessian Decomposition

In order to overcome the non-uniqueness problem with matrix factorizations, in this section, the use of second order information of the system is proposed. The second order information, namely the Hessian matrix, will be used to generate a tensor which can then be uniquely factorized to reveal the decoupled structure using the CPD.

In the case of multi-input single-output systems, evaluating the Hessian of the output of the decoupled model as in (3.7) with respect to the inputs (the elements of $\mathbf{z}$) at the

measurement points, $\mathbf{z}(k)$:

$$\left.\frac{\partial^2 \hat{\mathbf{y}}(\mathbf{z})}{\partial \mathbf{z}^2}\right|_{\mathbf{z}=\mathbf{z}(k)} = \sum_{i=1}^{r} g_i''(\boldsymbol{v}_i{}^T \mathbf{z}(k)) \boldsymbol{v}_i \boldsymbol{v}_i{}^T \tag{4.3}$$

If the Hessian is evaluated at N different measurement points, the results may be stacked into a 3-way tensor as:

$$\boldsymbol{\mathcal{H}}(i,j,k) = \sum_{n=1}^{r} g_n''(\boldsymbol{v}_n{}^T \mathbf{z}(k)) \boldsymbol{v}_n(i) \boldsymbol{v}_n(j) \tag{4.4}$$

This can be written more compactly using the Kronecker product:

$$\boldsymbol{\mathcal{H}} = \sum_{n=1}^{r} \mathbf{w}_n \otimes \boldsymbol{v}_n \otimes \boldsymbol{v}_n \tag{4.5}$$

where the vector $\mathbf{w_n} \in \Re^N$ can be defined for $n = 1, \cdots, r$, to contain the second derivative of the polynomial in branch n with respect to its input, evaluated at each of the N measurement points:

$$\mathbf{w}_n = \begin{bmatrix} g_n''(\boldsymbol{v}_n{}^T \mathbf{z}(1)) & \dots & g_n''(\boldsymbol{v}_n{}^T \mathbf{z}(N)) \end{bmatrix}^T \tag{4.6}$$

The decomposition in (4.5) is the CPD which is illustrated in Fig. 4.4, the same decomposition used by Dreesen et al. [51] to decouple MIMO polynomials and can be computed using standard tools such as those in Tensorlab [89].

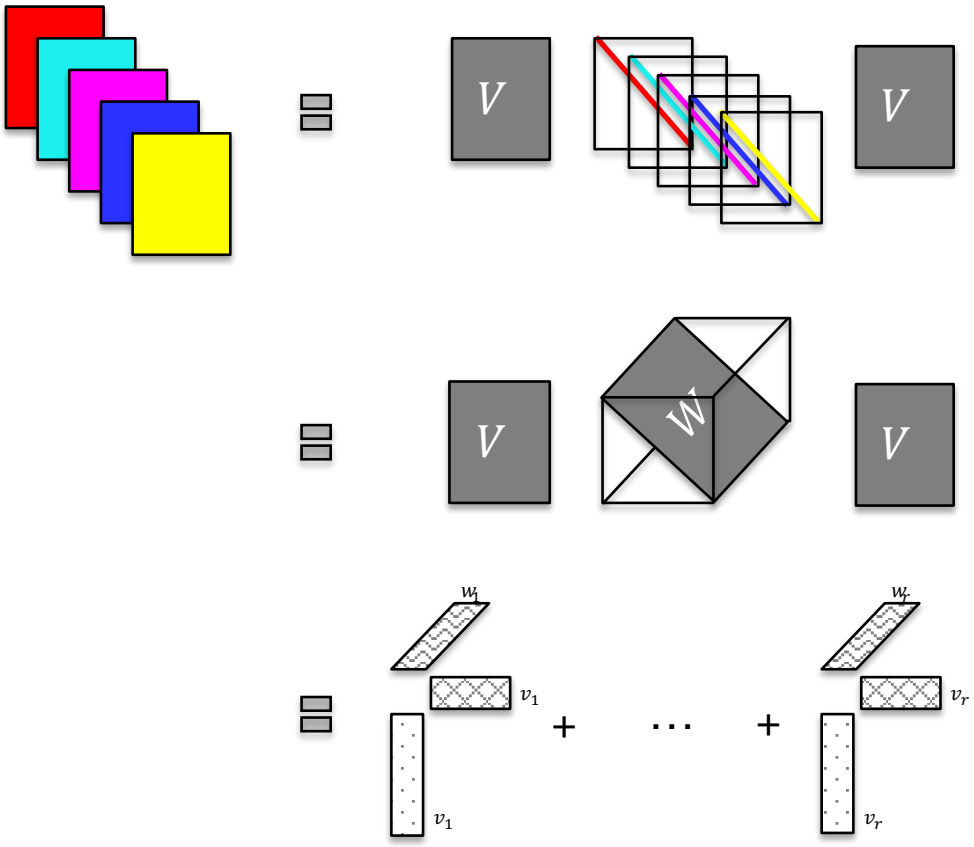

Figure 4.4: Decoupling a Tensor Build from the Hessian of a MISO System Using Canonical Polyadic Decomposition

---

Algorithm 2 Decoupling Polynomial NARX Model from Input-Output Data Initialized Using Hessian Decomposition

---

1. Fit a polynomial NARX model to the provided data using standard approaches such as an orthogonal forward regression algorithm [3]

2. Evaluate the Hessian of the coupled model output, then stack the Hessian matrices to form a 3-way tensor

3. Compute the CPD of the Hessian tensor and find the CPD factors (4.5)

4. Use the **V** from CPD as an initial estimate of the mixing matrix in the decoupled model. Refine the initial estimate using SLS based optimization (3.18)

4.2.1 Silver-Box Benchmark Results

In Chapter 3, it was shown that using the decoupling algorithm not only reduces the number of parameters in the identified model, but also increases the accuracy of the model. On the other hand, as mentioned before, the optimization problem might be sensitive to the initial solution. In this section, I will examine the effectiveness of the initialization for the optimization problem proposed in the previous section for the two benchmark problems discussed in Chapter 3.

Fig. 4.5 compares the convergence of the cost function in (3.11) for two different initial solutions. The magenta line shows the average cost for 1000 random initializations as well as the minimum and maximum cost after each iteration in cyan and yellow respectively. The blue line shows the cost function convergence when the initial solution is chosen using the factorization of the Hessian of the output as discussed in Section 4.2. As it is illustrated in the figure, choosing an appropriate initial solution is critical for the optimization problem. As it is obvious from Fig. 4.5, using the CPD of the Hessian for initialization not only results in faster convergence, but also the cost is much smaller than most of the models generated using random initializations. I have to mention that only about 20% of the randomly initializations resulted in models whose final cost was within 7.9% of that achieved using the model that was initialized using the Hessian decomposition.

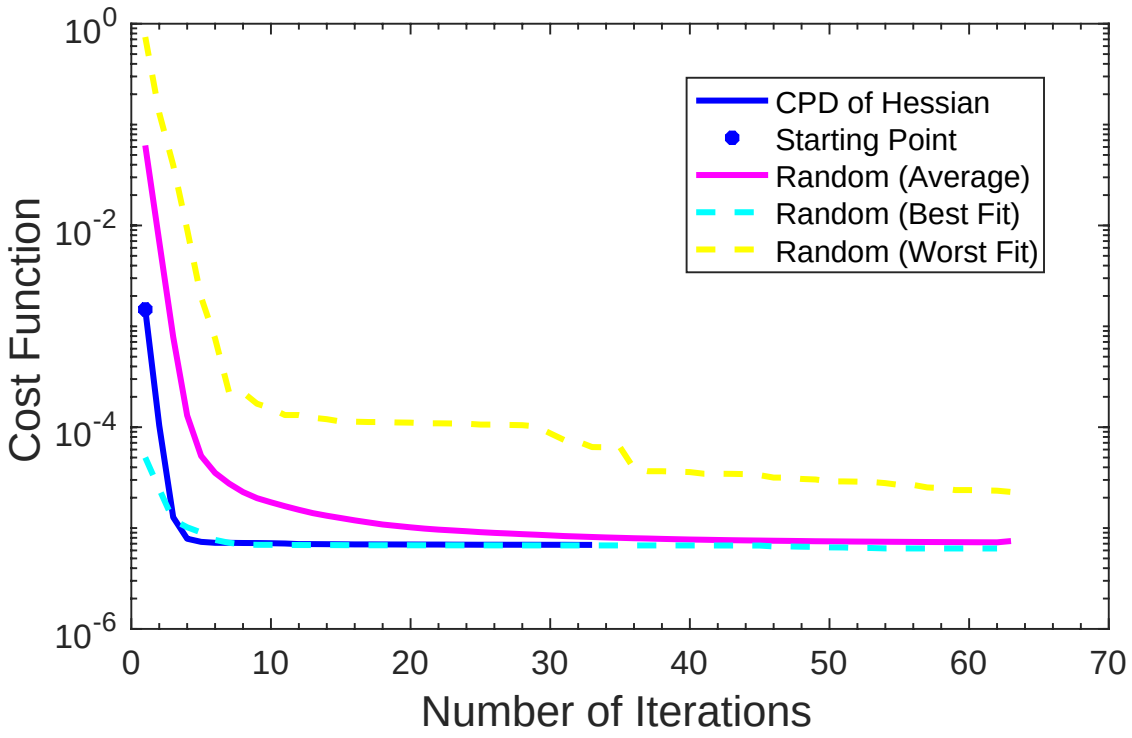

Figure 4.5: Cost Function During Optimization of Decoupled Polynomial NARX Model of the Silver-Box Benchmark, Comparing the Average Cost of 1000 Random Initializations and the Cost after Using the Hessian Based Initialization

The columns of the CPD factor matrix, $\mathbf{W}$, are expected to contain the second derivatives of the SISO polynomials evaluated at the points $v^T z$. Fig. 4.6 shows the CPD factors of the decoupled model. These factors were obtained using the algorithm as described in the previous section. As the figure demonstrates, these components do not exactly represent the second derivatives of any sort of smooth functions (like polynomials), as clouds of points arise. In order to get the univariate polynomials in the decoupled model, $g_1, \cdots, g_r$, these clouds of points should be interpolated and then integrated twice. Note that in the complete algorithm, this is not necessary, as the polynomial coefficients are fitted using an ordinary least squares regression.

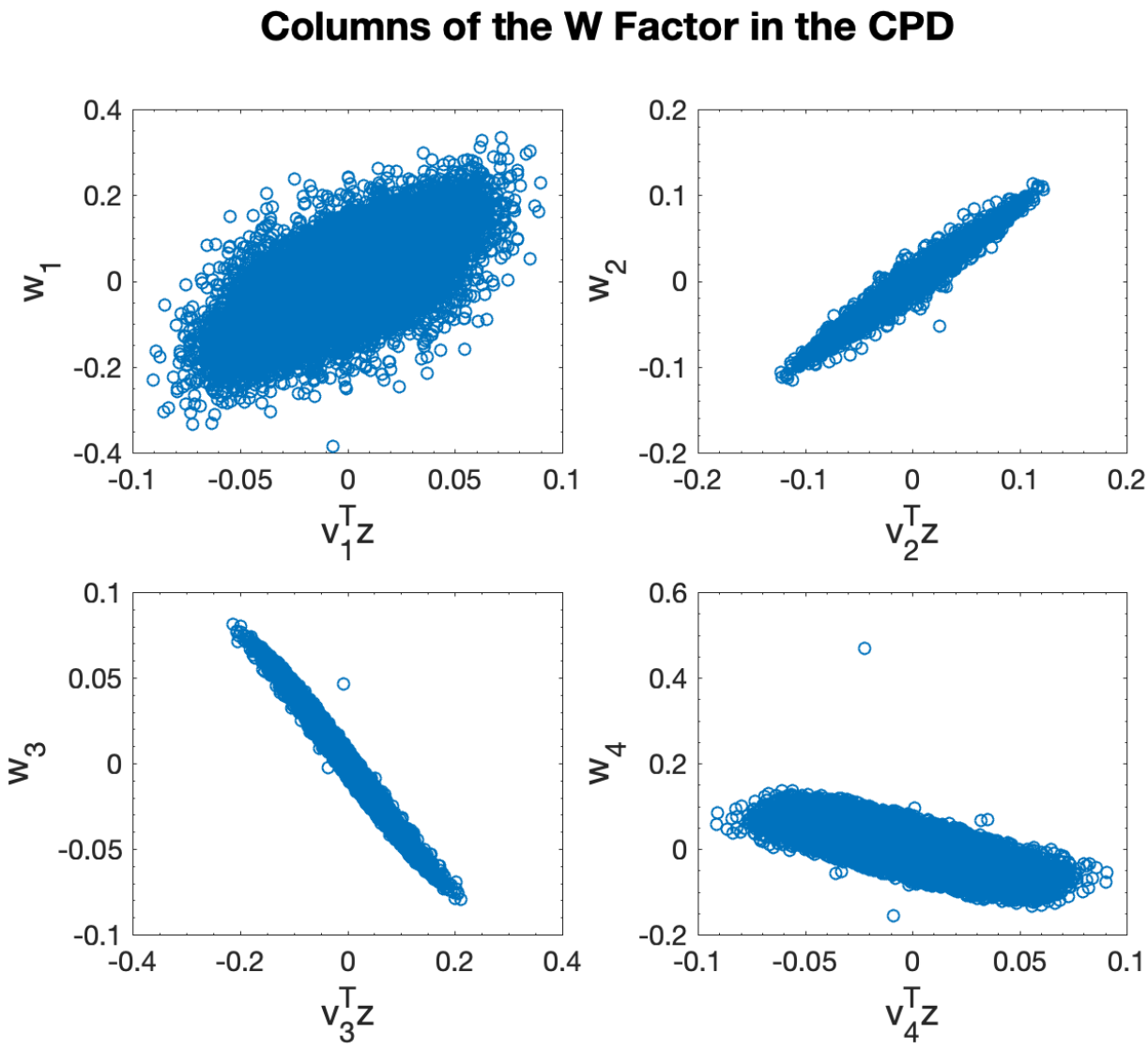

Figure 4.6: Graphics of $\mathbf{w}$ Vectors Containing the Estimates of the Mapping Functions Component Obtained From the CPD of the Hessian Using the Silver-Box Data Using Hessian Decomposition Initialization

One of the major difficulties with factoring the Hessian in (4.5) is that the number of variables in $\mathbf{W}$ increases in proportion with the number of measurement points N since in (4.6) $\mathbf{w}_n \in \Re^{N\times 1}$. Thus, unlike the $\mathbf{V}$ factor, whose dimensions are independent of the number of measurements, increasing N will not reduce the sensitivity of the $\mathbf{W}$ estimate to noise. One solution to this problem, will be discussed in Section 4.3.

Figure 4.7 shows the components of the individual branches of the decoupled model.

There are 4 branches and each branch contains a 3^{rd} degree nonlinearity. Fig. 4.7 shows the nonlinearity of each branch. The inputs and outputs of nonlinearites have been normalized.

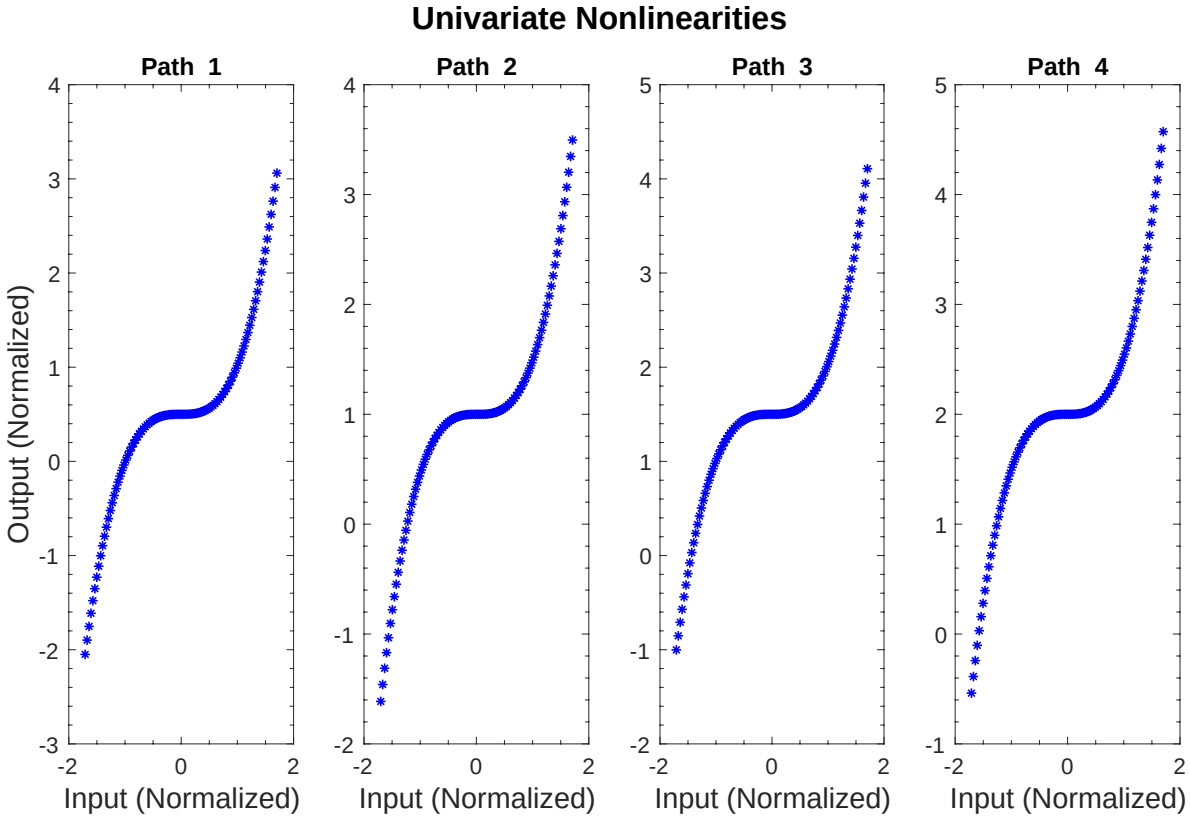

Figure 4.7: Nonlinearities of the All 4 Branches for the Silver-Box Benchmark. The Linear Terms Are Removed to Emphasize the Nonlinear Aspects of the Functions Using Hessian Decomposition Initialization

Note that the filter in each branch receives the output of the entire system, thus, it is not appropriate to combine the input and output terms into a single transfer function. As a result, the linear element in each branch is treated as two finite impulse response filters: one applied to the input signal, while the other is applied to the overall system output ($y(t)$). Figures 4.8 and 4.9 show the magnitude frequency responses of these filters. The output filters are all high-pass in nature.

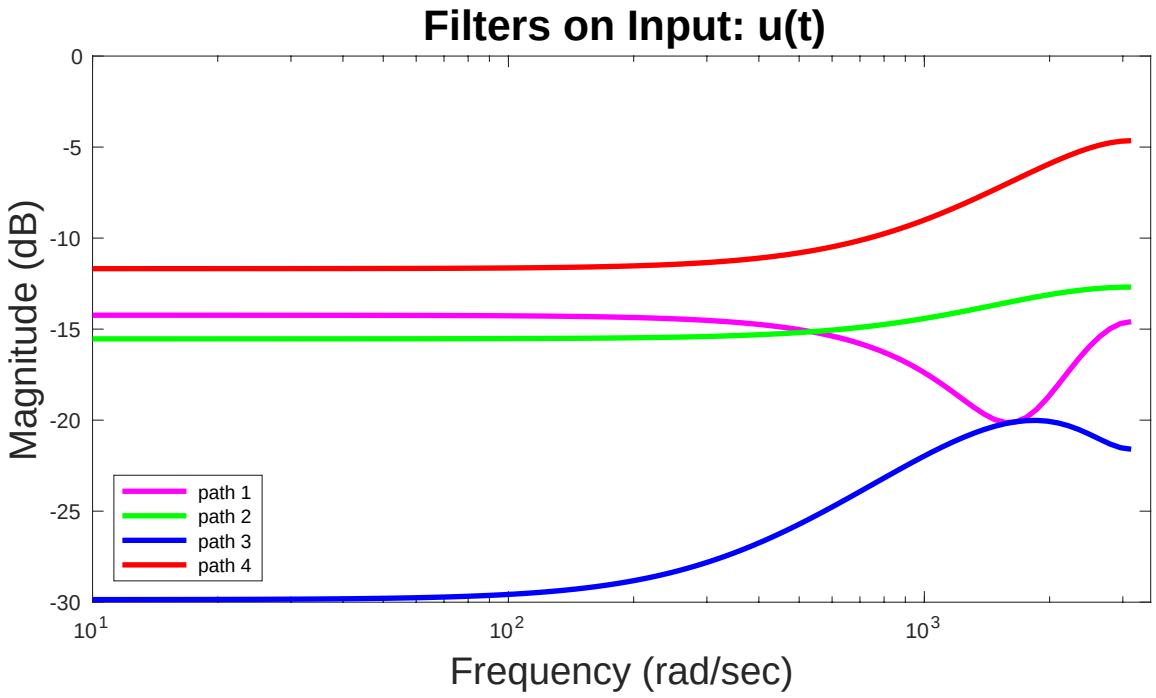

Figure 4.8: Frequency Response Magnitudes for the Input Terms of the Filters in the 4 Branches Using Hessian Decomposition Initialization

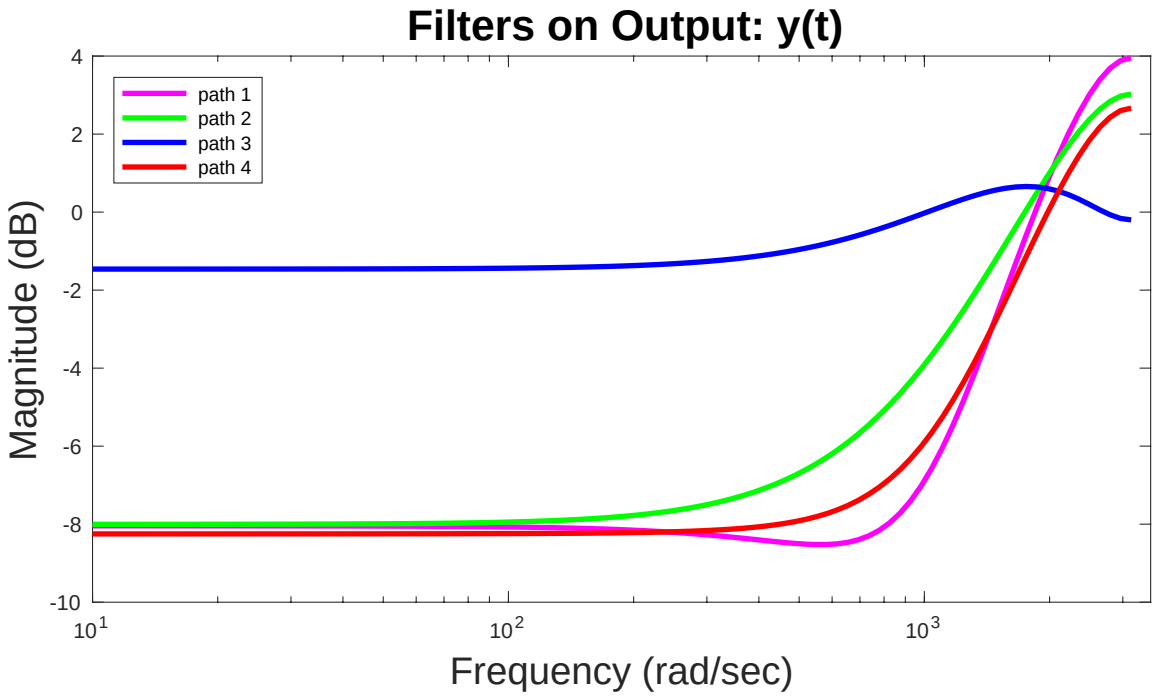

Figure 4.9: Frequency Response Magnitudes for the Output Terms of the Filters in the 4 Branches Using Hessian Decomposition Initialization

Figures 4.10 and 4.11 and Table 4.1 show the accuracy of identified model in the triangular and sweep-sine validation data. Both 1-step ahead prediction error and simulation error are shown. In the prediction error, shown in yellow, the model has access to past inputs and outputs whereas in the simulation error, shown in red, the model only has access to past inputs.

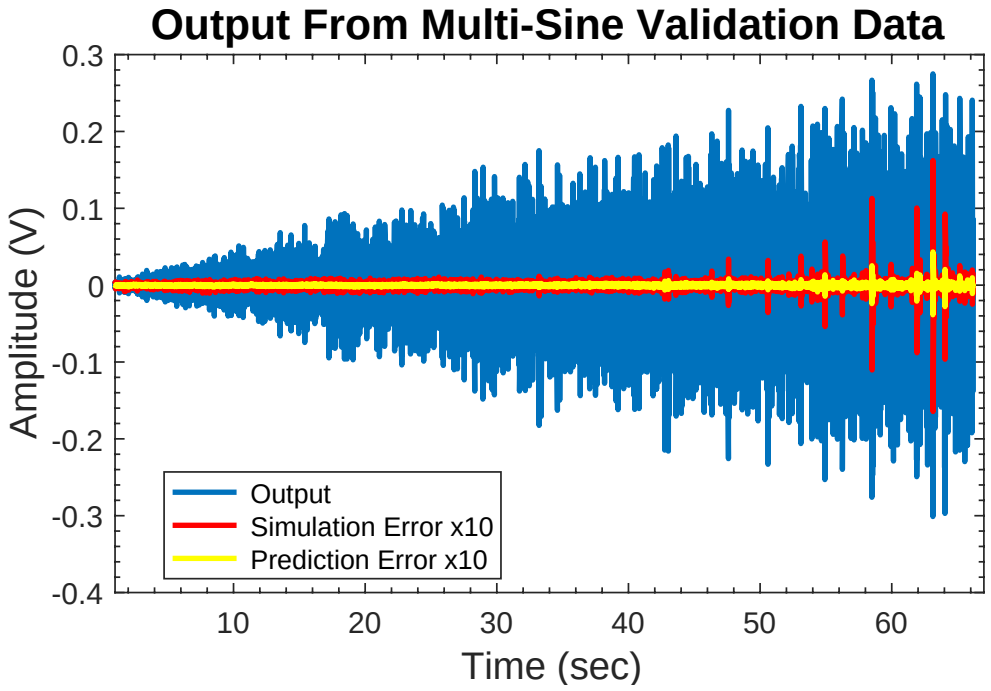

Figure 4.10: Prediction and Simulation Errors on the Validation Data for the Silver-Box Using Hessian Decomposition Initialization. Errors Are Scaled Up 10x for Better Visibility

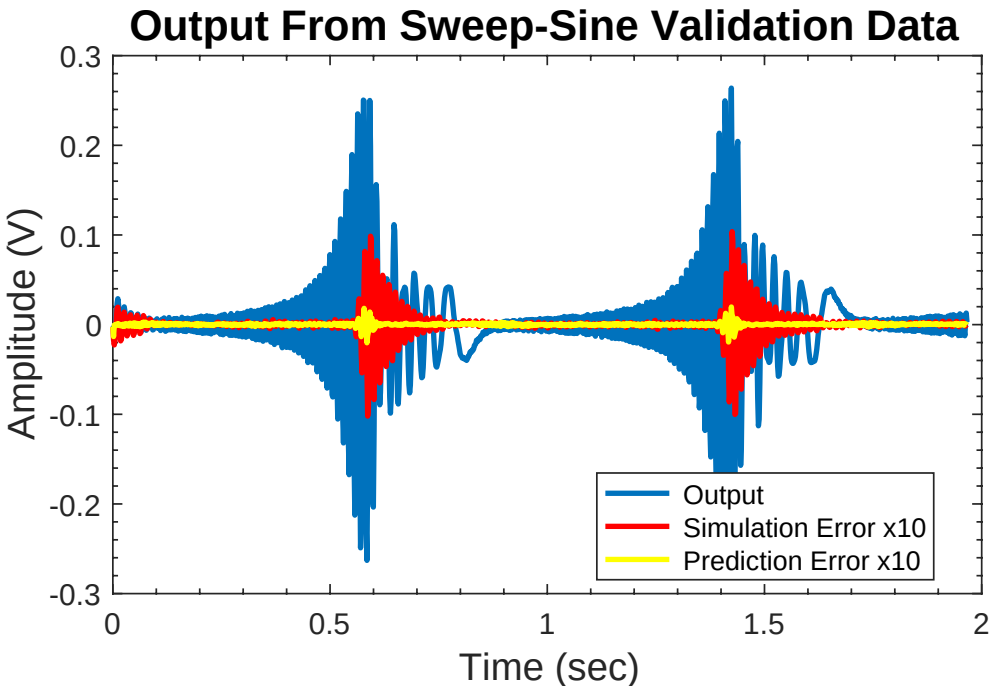

Figure 4.11: Prediction and Simulation Errors on the Sweep-Sine Validation Data for the Silver-Box Using Hessian Decomposition Initialization. Errors Are Scaled Up 10x for Better Visibility

	Simulation	1-Step Ahead Prediction
Multi-Sine(Fit%)	98.85 %	99.78 %
Sweep-Sine(Fit%)	96.98 %	99.60 %

Table 4.1: Accuracy of Decoupled NARX Model in Validation Data for the Silver-Box System Using Hessian Decomposition Initialization

4.2.2 Bouc-Wen Benchmark Results

Cost function convergence during the optimization for the Bouc-Wen benchmark is shown in Fig. 4.12 using random initializations and factorization of the Hessian initialization. The magenta line is the average cost function of 100 different runs starting with a various initial point each time.

As with the Silver-Box benchmark, it is obvious from Fig. 4.12 that starting from a more appropriate initial solution, will help the optimization to generate the optimal solution faster

and more accurately. It takes almost half of the iterations to find the optimal solution in case of CPD of the Hessian as the starting point, compare to using a random initial solution. For this model about 40% of the random initializations outperformed the CPD of the Hessian initialization but I have to also mention that the model accuracy can be as bad as 20%.

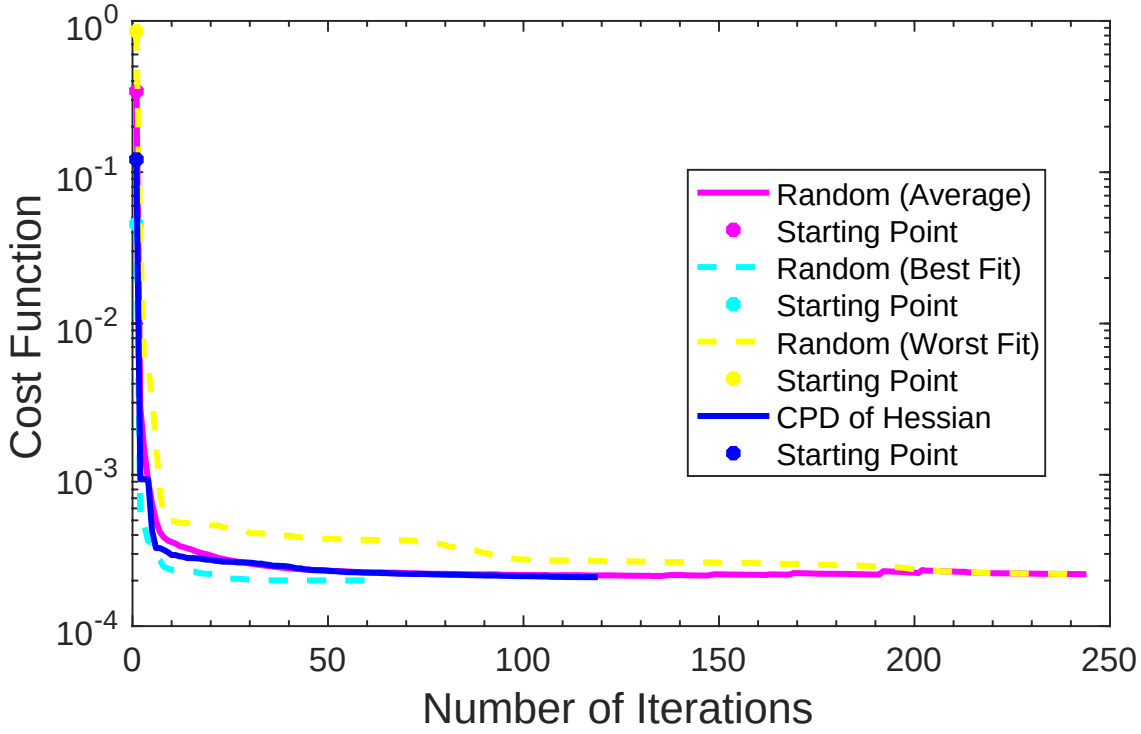

Figure 4.12: Cost Function During Optimization of Decoupled Polynomial NARX Model of the Bouc-Wen Benchmark, Comparing the Average Cost of 100 Random Initialization and the Cost of CPD of Hessian Initialization

Figure 4.13 shows the components of the columns $\mathbf{W}$ evaluated at $v^T z$, each of the subplots corresponds to one of the branches of the decoupled model. The CPD factors appear as clouds of points which hardly represents the second derivatives of the decoupled functions. Since the clouds are almost circular here, it is almost impossible to interpolate anything out of the points, whereas in the Silver-Box data which was a simpler model, the clouds were more likely very noisy lines. Therefore, as the system gets more complex, extracting information from these clouds is less possible.

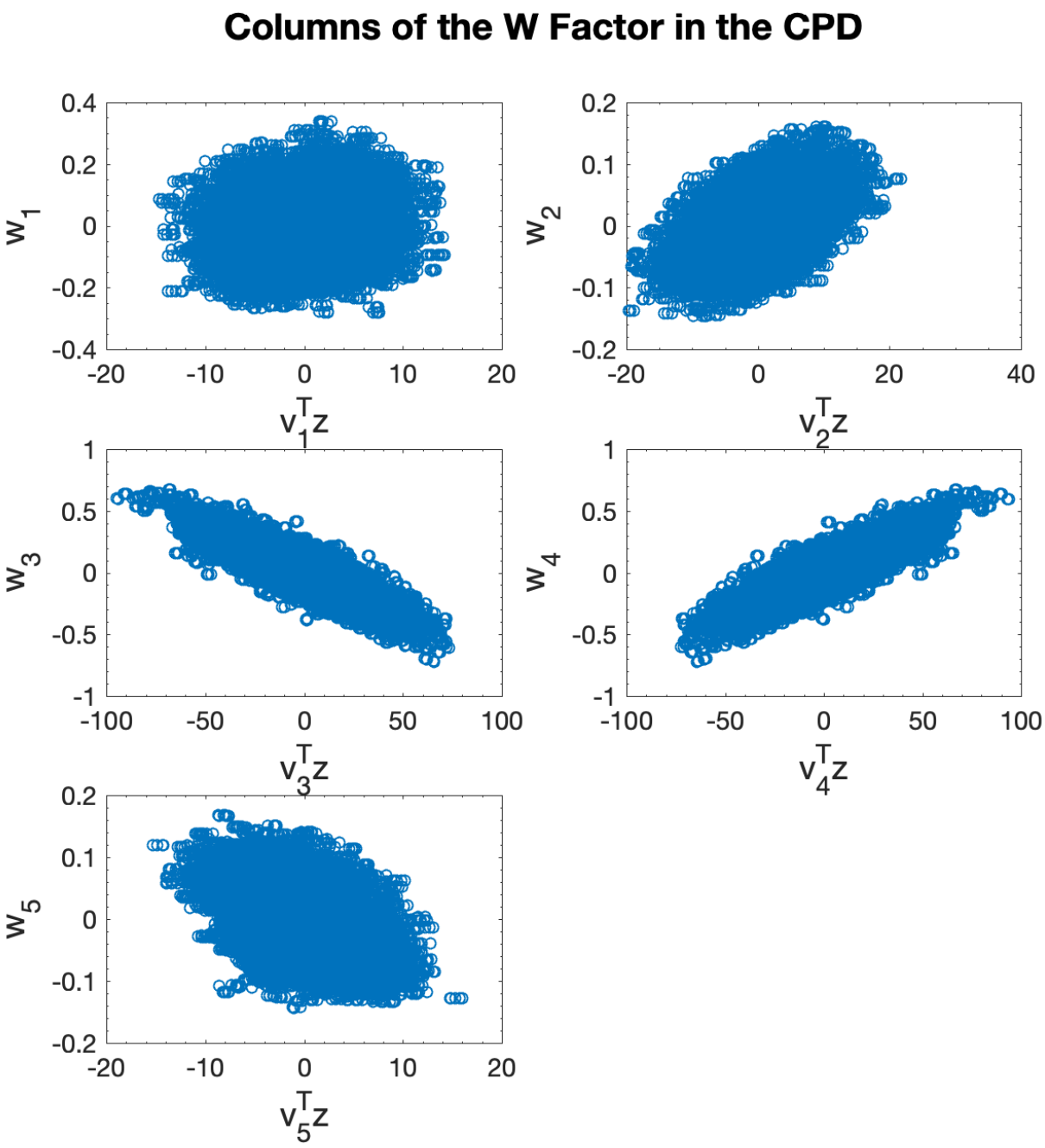

Figure 4.13: Graphics of **w** Vectors Containing the Estimates of the Mapping Functions Component Obtained from the CPD of the Hessian Using the Bouc-Wen Data Using Hessian Decomposition Initialization

Univariate nonlinearities in each branch of a 5 branch model for the Bouc-Wen benchmark are shown in Fig. 4.14. The inputs and outputs of the nonlinearities are normalized. A dead-

zone comportment is visible as all of them are flat around 0.

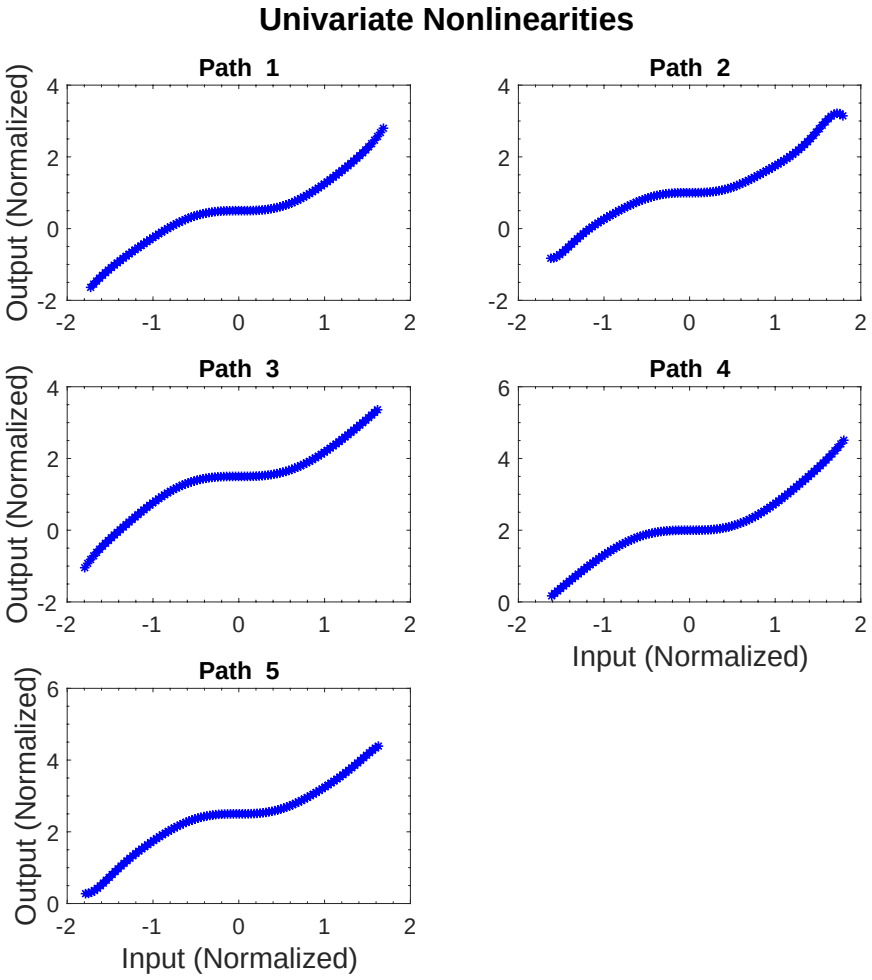

Figure 4.14: Nonlinearities of the All 5 Branches for the Bouc-Wen Benchmark. The Linear Terms Are Removed to Emphasize the Nonlinear Aspects of the Functions Using Hessian Decomposition Initialization

Figure 4.15 displays the magnitude of the frequency responses of the input and output terms in the 5 filters. The input and output filters show almost the same behavior in all of the branches, the input filters have a concavity around 800rad/sec and output filters show convexity around 1000rad/sec.

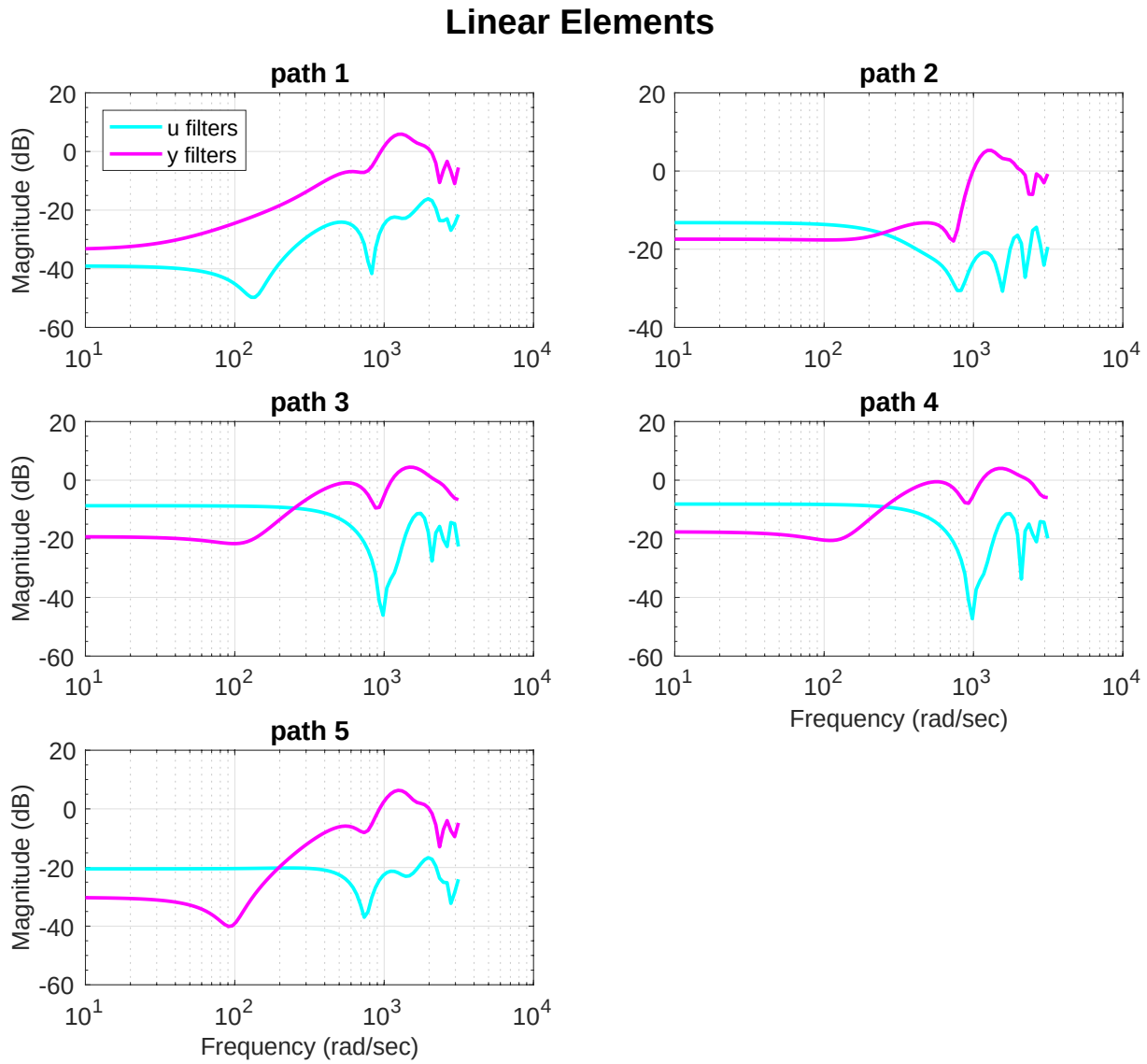

Figure 4.15: Frequency Response Magnitudes for the Input (Cyan) and Output (Magenta) Terms of the Filters in the 5 Branches for the Bouc-Wen Data

Figure 4.16 shows the simulation and prediction error for swept-sine validation data set on top of the validation output data. Table 4.2 includes the accuracy of the identified model using both validation data sets, the multi-sine and swept-sine. Comparing to the model achieved by random initialization, when we use more appropriate starting point the final model is more accurate. The error signals in Fig. 4.16 are scaled up 10 times for better visualization since the error were really small and the predicted/simulated output matches the measured output accurately.

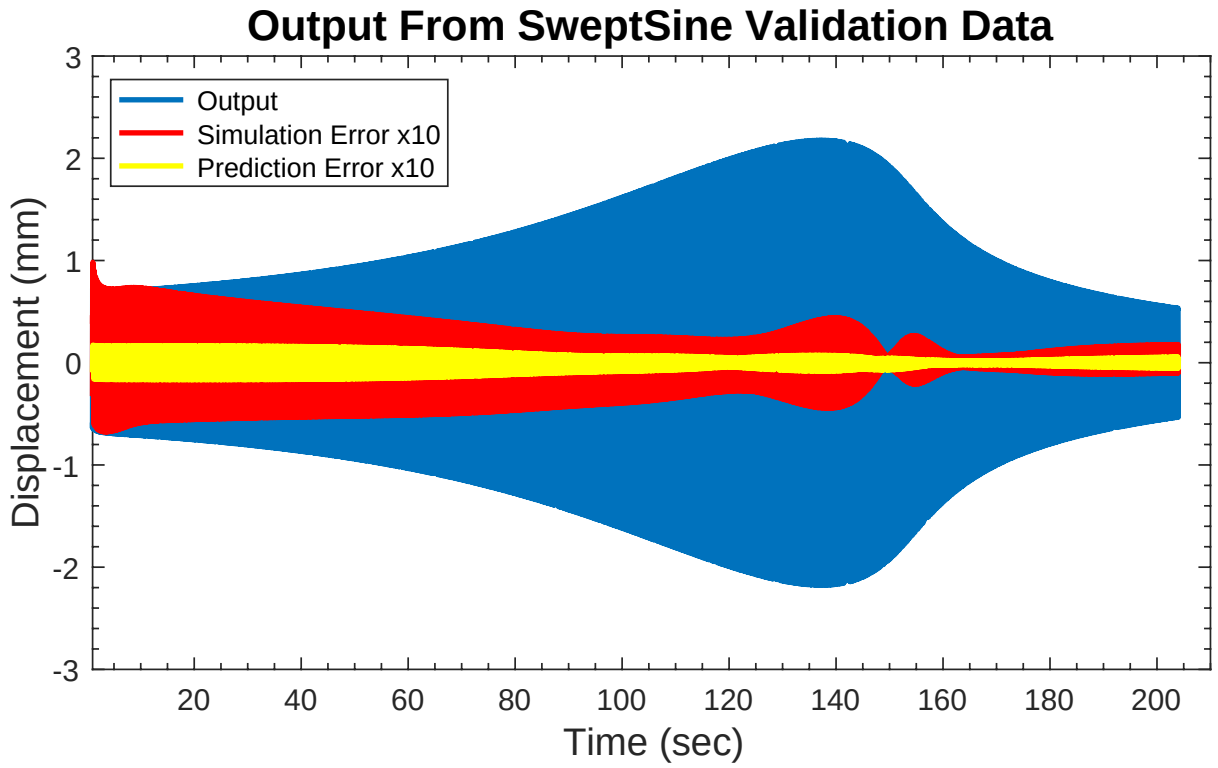

Figure 4.16: Prediction and Simulation Errors on the Sweep-Sine Validation Data for the Bouc-Wen Using Hessian Decomposition Initialization. Errors Are Scaled Up 10x for Better Visibility

	Simulation	1-Step Ahead Prediction
Multi-Sine(Fit%)	91.94%	98.48 %
Sweep-Sine(Fit%)	97.17 %	99.43 %

Table 4.2: Accuracy of Decoupled NARX Model in Validation Data for the Bouc-Wen System Using Hessian Decomposition Initialization

4.3 Imposing Polynomial Structure on the Hessian

As shown in Fig. 4.6 and Fig. 4.13, when no structure is added during the CPD of a noisy tensor, the decomposed factors are noisy. In particular, clouds of points will appear when plotting the columns of $\mathbf{w}$ against $\boldsymbol{v}^T\mathbf{z}$. Specifically, these clouds of points in the factor $\mathbf{W}$ can hardly represent the second derivatives of the decoupled functions. Note that similar

problems occur in the CPD of the Jacobian tensor used to decouple MIMO polynomials [90].

Instead of solving directly for the elements of $\mathbf{W}$, I will solve for the coefficients of the polynomials g_n'' in Eq.(4.4). By adding this polynomial constraint during the CPD, the output factor, $\mathbf{W}$, now represents directly the second derivatives of the functions $g_1, \cdots, g_r$. In order to impose the polynomial structure I will update the coefficients of the second derivatives of $g_1, \cdots, g_r$ instead of directly updating the entries in $\mathbf{W}$.

An optimization problem will be solved that minimizes the mean squared error between the Hessian evaluated from the NARX model and its approximation computed from the CPD factor,$\mathbf{V}$, and the polynomial coefficients,$\mathbf{C}$. Since a quasi-Newton optimization will be used to update the polynomial coefficients, I will calculate the Jacobian of the Hessian, the partial derivatives of the Hessian with respect to the mixing matrix elements, $\mathbf{V}$, and the polynomial coefficients, $\mathbf{C}$, as shown in (4.7) and (4.10):

$$\begin{aligned}\frac{\partial \mathbf{H}(i,j,k)}{\partial \mathbf{V}(m,l)} &= \frac{\partial}{\partial \mathbf{V}(m,l)} \sum_{n=1}^{r} \mathbf{V}(i,n)\mathbf{V}(j,n)\mathbf{W}(k,n) \\ &= \frac{\partial}{\partial \mathbf{V}(m,l)} \big(\mathbf{V}(i,l)\mathbf{V}(j,l)\mathbf{W}(k,l)\big) \\ &= \frac{\partial \mathbf{V}(i,l)}{\partial \mathbf{V}(m,l)} \mathbf{V}(j,l)\mathbf{W}(k,l) \\ &+ \mathbf{V}(i,l)\frac{\partial \mathbf{V}(j,l)}{\partial \mathbf{V}(m,l)} \mathbf{W}(k,l) \\ &+ \mathbf{V}(i,l)\mathbf{V}(j,l)\frac{\partial \mathbf{W}(k,l)}{\partial \mathbf{V}(m,l)}\end{aligned} \tag{4.7}$$

where $i, j, m = 1, \cdots, (n_a + n_b)$, $k = 1, \cdots, N$ and $l = 1, \cdots, r$. Simplifying Eq. (4.7):

$$\begin{aligned}\frac{\partial \mathbf{H}(i,j,k)}{\partial \mathbf{V}(m,l)} &= \delta_{im}\mathbf{V}(j,l)\mathbf{W}(k,l) \\ &+ \delta_{jm}\mathbf{V}(i,l)\mathbf{W}(k,l) \\ &+ \mathbf{V}(i,l)\mathbf{V}(j,l)\frac{\partial \mathbf{W}(k,l)}{\partial \mathbf{V}(k,l)}\end{aligned} \tag{4.8}$$

where δ is a the Kronecker delta function and

$$\mathbf{W}(k,l) = \frac{\partial^2 g_i(x_i)}{\partial x_i^2} = \sum_{j=1}^{M} j(j-1)c_{j,i}x_i^{(j-2)} \tag{4.9}$$

and

$$\begin{aligned}\frac{\partial \mathbf{H}(i,j,k)}{\partial \mathbf{C}(t,l)} &= \frac{\partial}{\partial \mathbf{C}(t,l)} \sum_{n=1}^{r} \mathbf{V}(i,n)\mathbf{V}(j,n)\mathbf{W}(k,n) \\ &= \mathbf{V}(i,l)\mathbf{V}(j,l)\frac{\partial \mathbf{W}(k,l)}{\partial \mathbf{C}(t,l)}\end{aligned} \tag{4.10}$$

Note that storage requirements for the Jacobian of the Hessian are large – the Hessian contains Nm^2 elements, and the model contains $r(m+M-2)$ parameters. Thus the Jacobian will have $Nrm^2(m+M-2)$ elements. The size of this matrix scales approximately with m^3, the cube of the number of inputs in the vector $\mathbf{z}$.

In addition to the storage requirement, one must also consider the computational complexity, usually defined in terms of the number of floating point operations (flops) required to perform a given computation. If the structured CPD is implemented using a Levenberg-Marquardt optimization, the dominant computational cost in each iteration will be associated with solving a linear least squares problem involving the Jacobian (2.42) to compute the parameter update. Since the Jacobian is Nm^2 by $r(m+M-2)$, the approximate cost of the least squares solution will be on the order of $Nm^2(r(m+M-2))^2$ flops [91]. Thus, the computational cost (per iteration) depends on the fourth power of the number of input values.

The norm of the error between the Hessian and its decoupled approximation will be minimized using Eqs. (4.5) and (4.9). The tensor values are nonlinear in the mixing matrix ($\mathbf{V}$) elements but linear in the polynomial's coefficients ($\mathbf{C}$). Therefore, I will treat $\mathbf{C}$ as a function of $\mathbf{V}$ and optimize over the mixing matrix. This optimization decomposes the tensor, but with the addition of the polynomial constraint.

Algorithm 3 Decoupling Polynomial NARX Model from Input-Output Data Initialized by Decomposing the Hessian using a Polynomial Constraint

1. Fit a polynomial NARX model to the provided data using standard approaches such as an orthogonal forward regression algorithm [3]

2. Compute the Hessian with respect to the inputs and outputs using (4.3)

3. Compute the CPD of the unconstrained Hessian using standard tools [89]

4. Impose the polynomial structure to the Hessian described in Section 4.3

5. Use the $\mathbf{V}$ from CPD as an initial estimate of the mixing matrix in the decoupled model. Refine the initial estimate using SLS based optimization (3.18)

4.3.1 Silver-Box Benchmark Results

In Section 4.2.1, the result of the optimization for the Silver-Box benchmark using an appropriate initial solution, i.e. decomposition factors of the Hessian, was showed and compared to the case of random initialization. The results indicated that random initialization will not guarantee the best solution. However, in the model identified in Fig. 4.6, the $\mathbf{W}$ factors were very noisy, and clearly did not represent any sort of polynomial of their input. Therefore, in this section, I imposed the polynomial structure on the identified model.

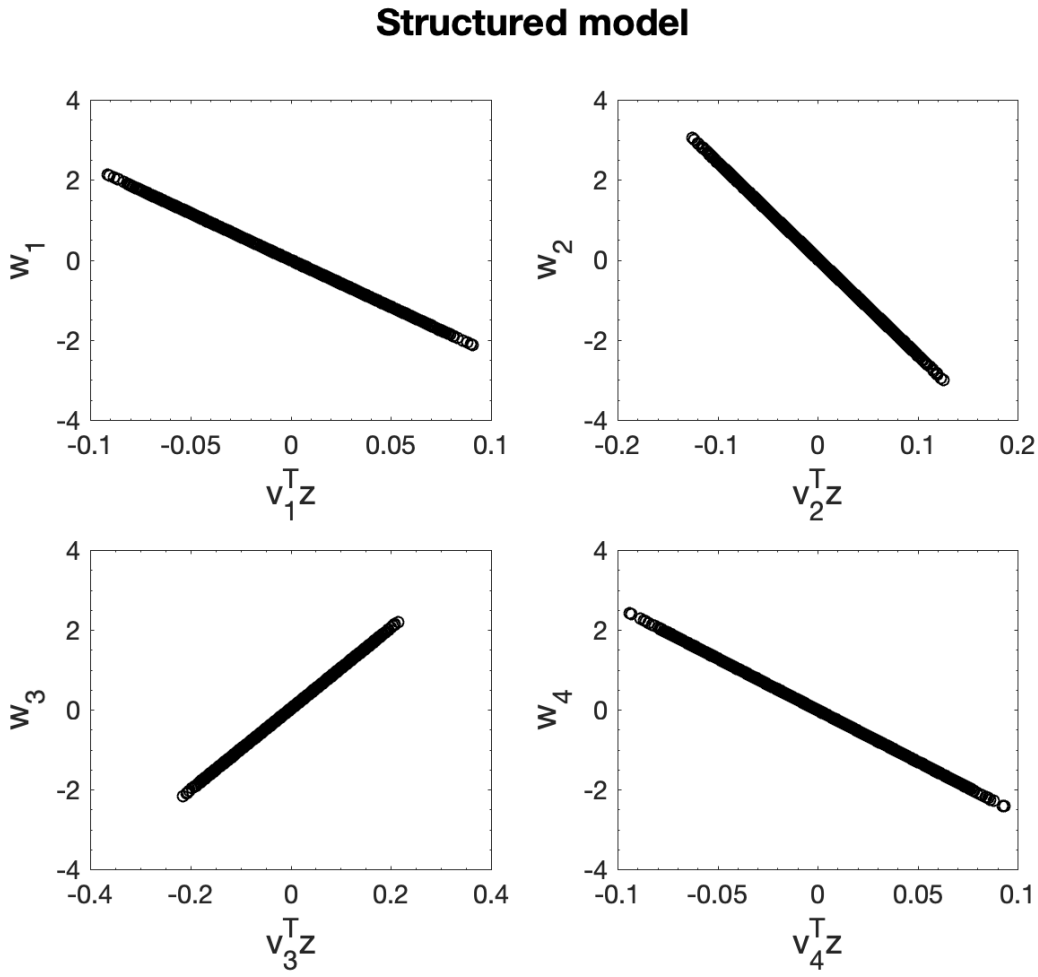

Figure 4.17: Graphics of **w** Vectors Containing the Estimates of the Mapping Functions Component Obtained From Polynomial Structure Enforced on the CPD of the Hessian Using Same Data as in Figure 4.6

In Figure 4.17 the polynomial structure is imposed on the Hessian. The plots for Fig. 4.17 are generated from the same experiments as in Fig. 4.6.

Here, I will directly update the coefficients of the second derivative of the univariate polynomials instead of their evaluations in $v^T z$. Because of the additional polynomial constraints, the interpolation step can be skipped and these points only need to be integrated twice to retrieve the univariate functions coefficients.

Figure 4.18 compares the cost function between the case of structured and unstructured CPD of the Hessian. Even though there is not much difference between the two cost functions,

the final cost in case of structured CPD of the Hessian is slightly smaller (6.806×10^{-6} vs. 6.809×10^{-6}). With such a simple model like the Silver-Box, the effectiveness of introducing the polynomial constraint is not obvious. But later in this chapter, its efficacy will be shown on the more complex Bouc-Wen model.

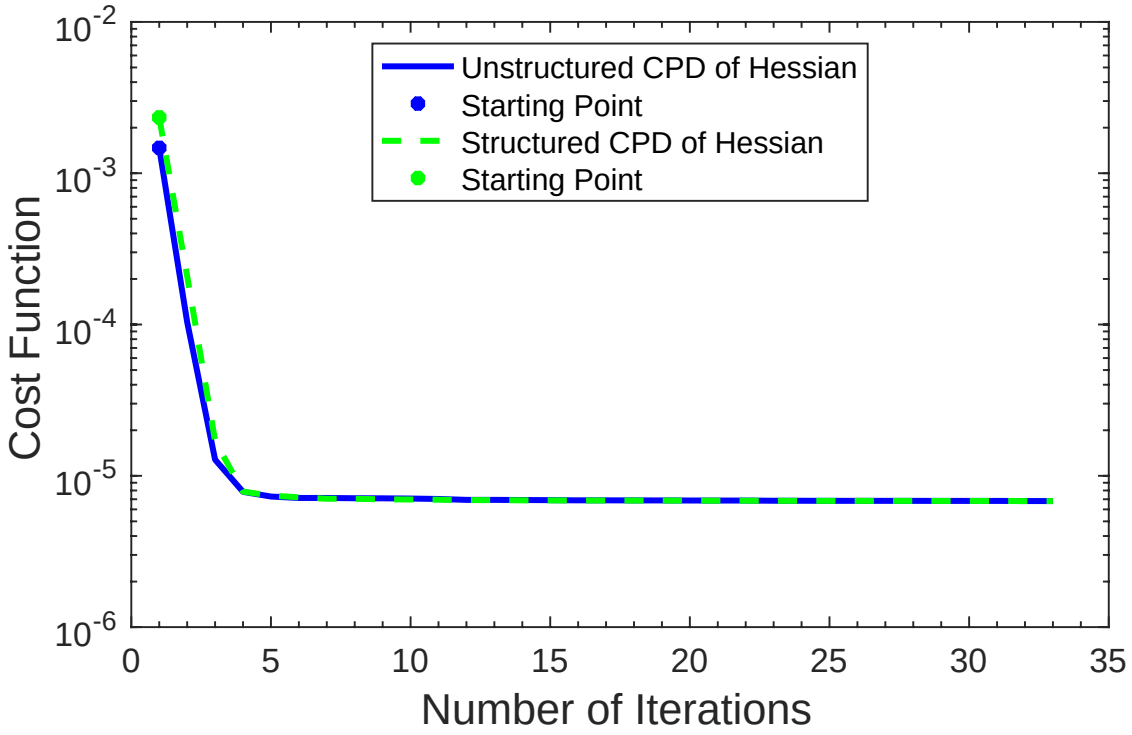

Figure 4.18: Cost Function During Optimization of Decoupled Polynomial NARX Model of the Silver-Box Benchmark, Comparing the Cost Function When the Optimization Is Initialized Using Unstructured CPD against Polynomial Structure Imposed on the Initial Solution

Figures 4.19 and 4.20 demonstrate the simulation and prediction errors for both sets of validation data, triangular multi-sine and sweep-sine. In both figures the errors are scaled up 10 times for better visualization. In both cases, the prediction error is smaller than the simulation error, and it makes sense since the optimization is trying to minimize the prediction error.

Table 4.3 summarize the fit percentage of the case of simulation and 1-step ahead prediction for both validation data sets. Comparing to Table 4.1, there is a small gain in simulation accuracy when the polynomial structure is enforced but there is no change in the

1-step ahead prediction accuracy.

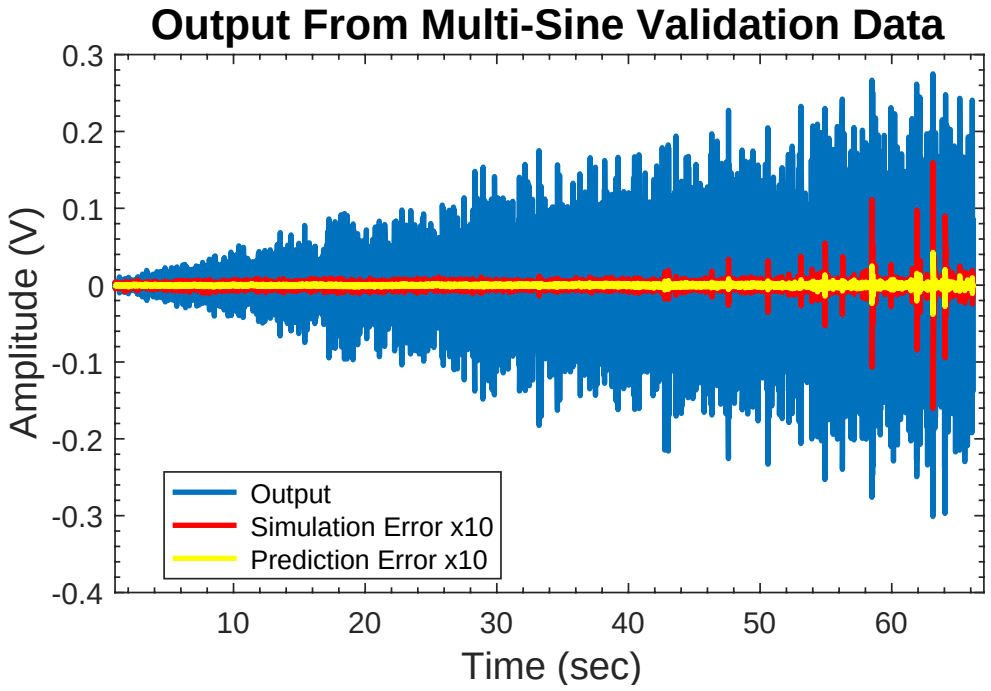

Figure 4.19: Prediction and Simulation Errors on the Validation Data for the Silver-Box Using Hessian Decomposition Initialization When Polynomial Structure is Forced. Errors Are Scaled Up 10x for Better Visibility

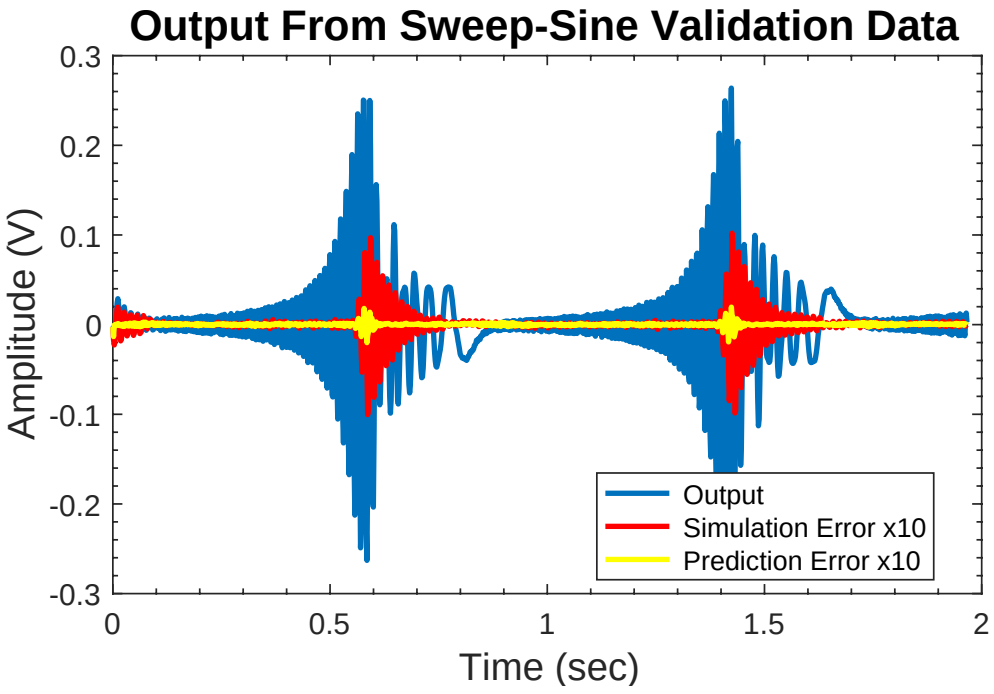

Figure 4.20: Prediction and Simulation Errors on the Sweep-Sine Validation Data for the Silver-Box Using Hessian Decomposition Initialization When Polynomial Structure is Forced. Errors Are Scaled Up 10x for Better Visibility

	Simulation	1-Step Ahead Prediction
Multi-Sine(Fit%)	98.88 %	99.78 %
Sweep-Sine(Fit%)	97.04 %	99.60 %

Table 4.3: Accuracy of Decoupled NARX Model in Validation Data for the Silver-Box System Using Hessian Decomposition Initialization When Polynomial Structure is Forced

4.3.2 Bouc-Wen Benchmark Results

For the Bouc-Wen benchmark, as mentioned before, one must compute the Jacobian of the Hessian of the initial, coupled, model. For the model that obtained the best fit percentage (in the testing data), a model with 15 past inputs and 15 past outputs, this computation is prohibitively expensive. Hence, I tried to find a smaller model that still represents the Bouc-Wen model with acceptable accuracy. As mentioned in the previous chapter, a wide range of model structures were scanned at the beginning in order to find an appropriate model

structure for the study. I examined all the fit values in order to pick a smaller model that still provides an accurate enough model. The original model, 15 past inputs and outputs, delivers 98.53% model accuracy using the full polynomial NARX model. Hence I was looking for the smallest model that provides an accuracy about at least 98%. The Simplest model providing the desired accuracy contained 4 past inputs, 6 past outputs and 0 input delay for the full polynomial NARX model with 98.08% model accuracy in testing data.

In order to achieve the desired accuracy in the decoupled model with the simpler structure, I had to increase the number of branches. The final model picked for this study includes, 4 past inputs, 6 past outputs and 0 input delay, with 10 branches and 8 degree polynomial nonlinearity

Table 4.4 shows the number of elements in the Jacobian for both models, the original model used in Chapter 3 and the reduced model in this section, indicating a 10× reduction in the required storage. As noted previously, the computational cost of each Levenberg-Marquardt iteration will be approximately proportional to Nrm^4. Thus, the 3-fold reduction in m, and the doubling of r, as noted in Table 4.4, would result in an approximate 40-fold reduction in the cost of each iteration.

As a result, it was infeasible to factor the Hessian tensors of those larger structures. Thus, a smaller model with acceptable performance was chosen.

	M	r	m	# of Elements in Jacobian	Required Memory
Original Model	9	5	30	6.8198×10^9	50.8 GB
Reduced Model	8	10	10	6.5536×10^8	4.9 GB

Table 4.4: The Number of Elements in the Jacobian and the Required Memory for Two Different Models

Therefore, in order to be able to compare the different initialization techniques, I will also generate a Bouc-Wen model using 4 past inputs and 6 past outputs, the same model

that is used in Section 4.3, using the other initialization techniques described in Section 4.2.

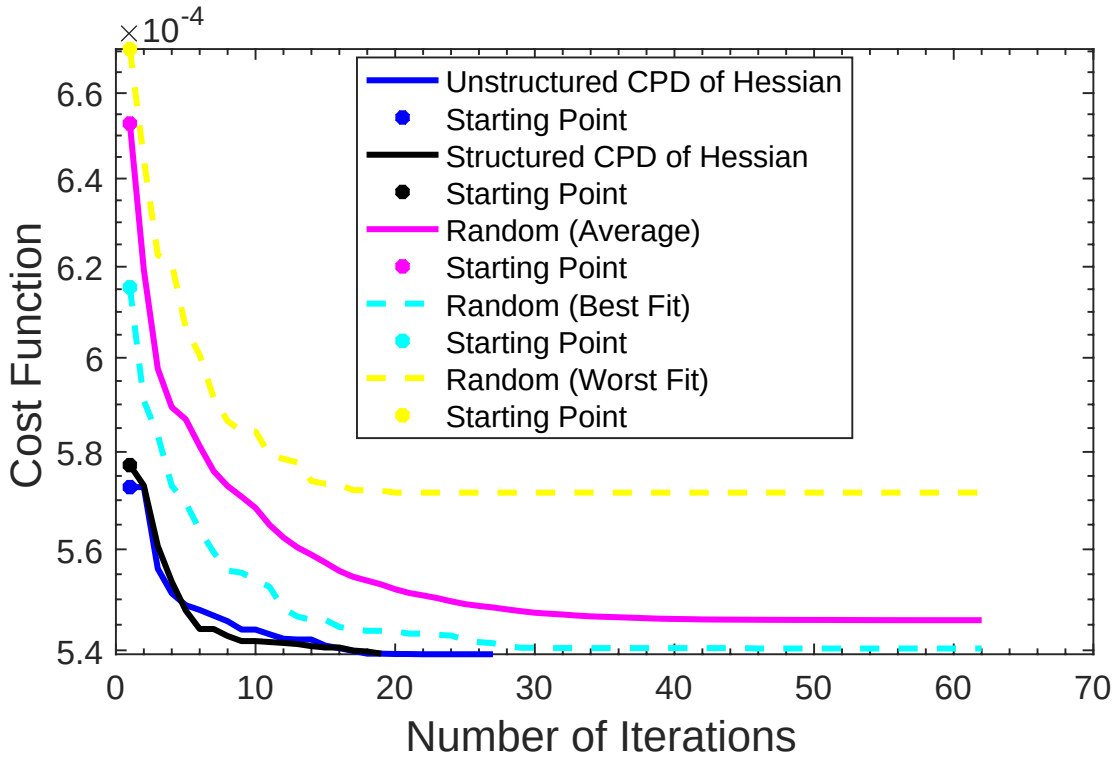

Figure 4.21: Cost Function During Optimization of Decoupled Polynomial NARX Model of the Bouc-Wen Benchmark

Figure 4.21 shows the convergence of the cost function for all 3 different initializations, random, unstructured CPD of the Hessian and the structured CPD of the Hessian. It is obvious that both techniques with CPD factorization outperform the random initialization, they converge faster and cheaper even from the best case of the random initialization.

For better comparison between cost functions of both CPD factorization cases, look at Fig. 4.22. It is illustrated that when the polynomial structure is enforced on the Hessian, the optimization reaches its optimum faster.

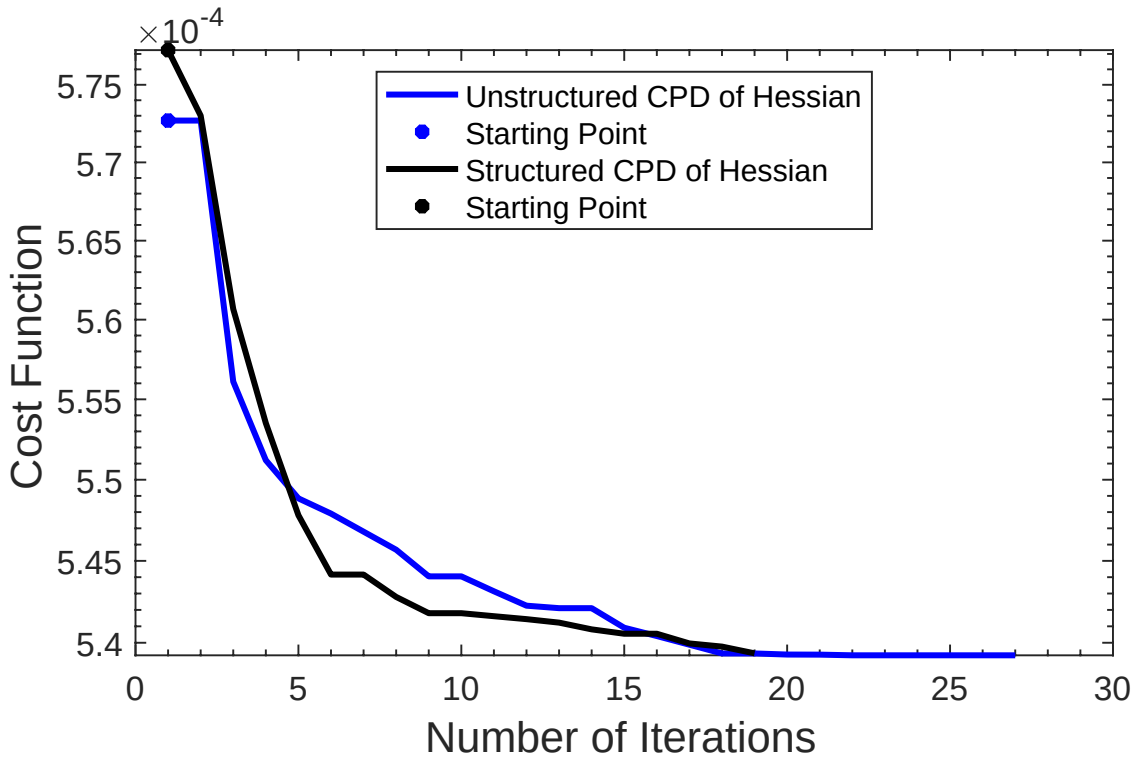

Figure 4.22: Cost Function During Optimization of Decoupled Polynomial NARX Model of the Bouc-Wen Benchmark for 3 Different Initialization Method for Better Visualization

Figure 4.23 shows the component of the **W** when the polynomial structure is imposed as well as with out any structure forcing to the Hessian. Both blue and black figures are using the same sets of data. Vividly, when the polynomial structure is imposed, the columns of **W** represent the second derivative information of the univariate polynomials.

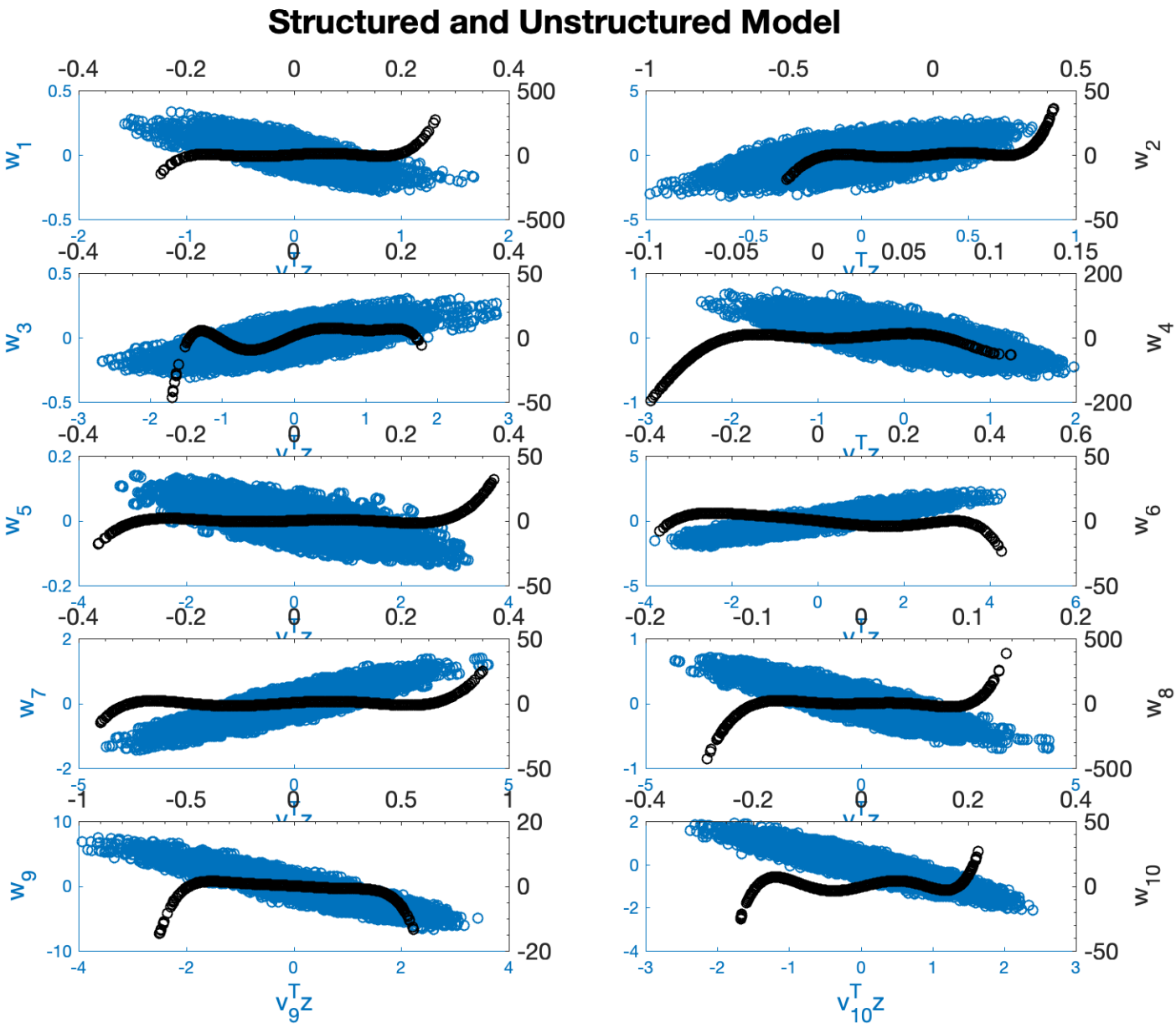

Figure 4.23: Visualization of the Vectors **w** Which Contain the Estimates of the Mapping Functions Obtained From the CPD of the Hessian Using the Bouc-Wen Data (Blue: Unstructured, Blak: Structured)

Figure 4.24 indicates the simulation and 1-step ahead prediction error on swept-sine validation data set. The errors are a bit larger here, and that is due to simpler model structure.

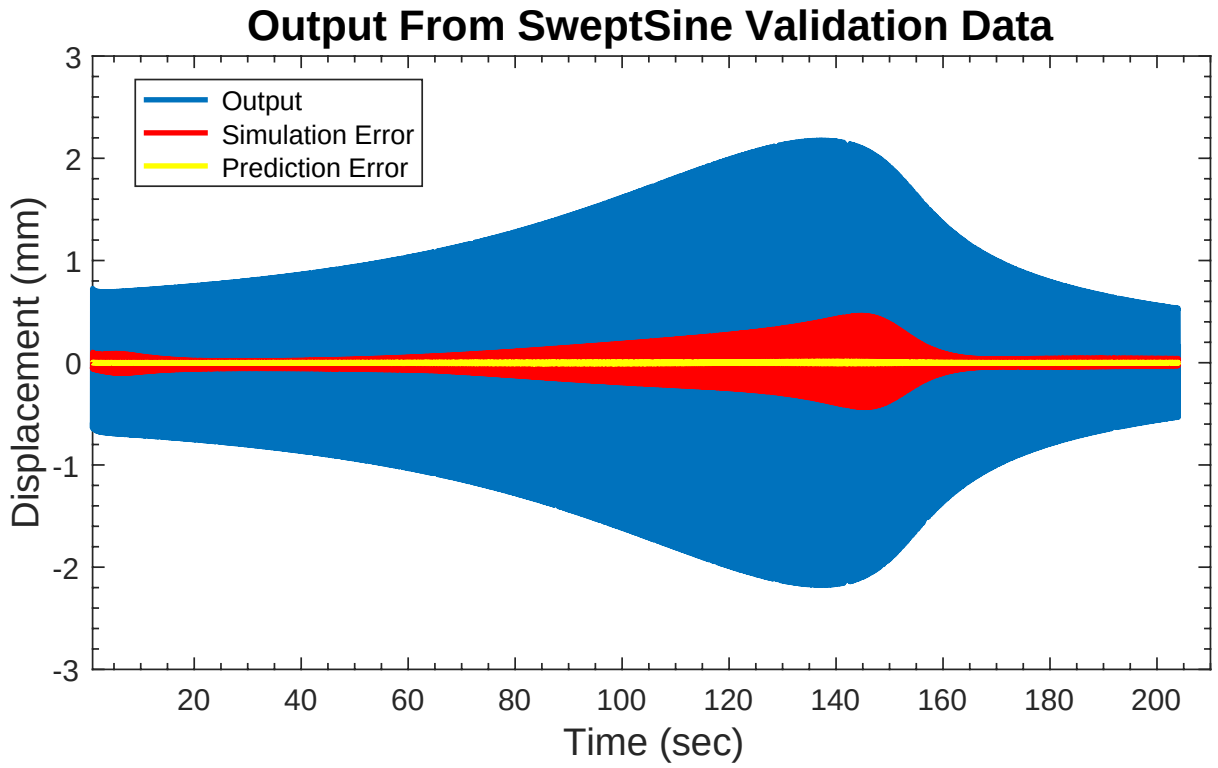

Figure 4.24: Prediction and Simulation Errors on the Sweep-Sine Validation Data for the Bouc-Wen Using Hessian Decomposition Initialization When Polynomial Structure Is Forced

Table 4.5 includes the results obtained from the reduced size model for a fair comparison. It is interesting that the structured CPD gets worse results than the unstructured CPD across the board, and also that there is little difference between the average random model and the unstructured CPD, except for the multi-sine in simulation, where the random models tended to be unstable. This suggests that perhaps it isn't worth expending the effort required to impose the polynomial structure to get better accuracy, but you have to keep in mind that it will provide you the polynomial coefficients directly. It would be up to the user to decide if these information is needed for the application for the price of a little bit less accurate model or not. Or maybe the modeling errors in the reduced order model are too significant that cause no improvement in the accuracy when imposing the polynomial structure.

		Simulation	1-Step Ahead Prediction
Struc.	Multi-Sine(Fit%)	81.86%	87.30%
	Sweep-Sine(Fit%)	86.33%	99.58%
UnStruc.	Multi-Sine(Fit%)	81.94%	98.92%
	Sweep-Sine(Fit%)	86.44%	99.60%
Rand(Avg).	Multi-Sine(Fit%)	Unstable	95.90%
	Sweep-Sine(Fit%)	86.76%	99.59%

Table 4.5: Accuracy of Decoupled NARX Model in Validation Data for the Bouc-Wen System Using All 3 Different Initializations for the Model with 4 Past Inputs and 6 Past Outputs

4.4 Polynomial Tensor Decomposition

In this section another approach to initialize the estimate of the variables in the mixing matrix, $\mathbf{V}$, used in the decoupling algorithm, is proposed. Note that the polynomial coefficients, $\mathbf{c}$, are treated as closed-form functions of $\mathbf{V}$, so they do not need to be initialized. This initialization is based on the Waring decomposition. The Waring problem has been extensively discussed in the literature [92, 93]. In its classical form, the Waring problem for polynomials indicates that a general homogeneous polynomial of n variables and degree kd can be represented as the sum of terms that are each a sum of degree d terms raised to the power k [94]. Such representations are often called Waring decompositions. The relationships between the Waring decomposition and various tensor decompositions have become an important research area [52, 92, 95, 96].

Let $y(\mathbf{z})$ be the output of a MISO polynomial, this function includes m generic inputs, $z_i, i = 1..m$, and that in a P-NARX model these can be either past outputs, or current or past inputs.

$$y(\mathbf{z}) = y_0 + \sum_{i_1=1}^{m} \alpha_{i_1} z_{i_1} + \sum_{i_1=1}^{m} \sum_{i_2=i_1}^{m} \alpha_{i_1,i_2} z_{i_1} z_{i_2} + \sum_{i_1=1}^{m} \cdots \sum_{i_n=i_{n-1}}^{m} \alpha_{i_1,\cdots,i_n} z_{i_1} \cdots z_{i_n} \tag{4.11}$$

For the purpose of illustration, let $y(\mathbf{z})$ be a polynomial function with 3 inputs z_1, z_2 and z_3 of $3rd$ degree. The first order (linear) term in (4.11) is given by:

$$f_1(\mathbf{z}) = \sum_{i_1=1}^{3} \alpha_{i_1} z_{i_1} = \mathbf{t}_1^T \mathbf{z} \tag{4.12}$$

where $\mathbf{t}_1$ includes the first order information in a $1st$ order tensor (vector):

$$\mathbf{t}_1 = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{4.13}$$

For the second degree term:

$$f_2(\mathbf{z}) = \sum_{i_1=1}^{3} \sum_{i_2=i_1}^{3} \alpha_{i_1,i_2} z_{i_1} z_{i_2} = \mathbf{z}^T T_2 \mathbf{z} \tag{4.14}$$

where T_2 collects the second order information in a $2nd$ order tensor (matrix):

$$T_2 = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} \end{bmatrix} \tag{4.15}$$

(4.14) can also be written in forms of tensor product as $f_2(z) = T_2 \times_1 \mathbf{z} \times_2 \mathbf{z}$. Then I will compute the SVD of T_2 as it is shown in Fig. 4.25. Let $T_2 = VSV^T$ be the SVD of T_2, then (4.14) can be written as

$$f_2(\mathbf{z}) = (V^T \mathbf{z})^T S V^T \mathbf{z}$$

Note that the terms $V^T \mathbf{z}$ are weighted sums of the inputs $\mathbf{z}$, and that the diagonal S matrix turns this into a sum of squares of sums form.

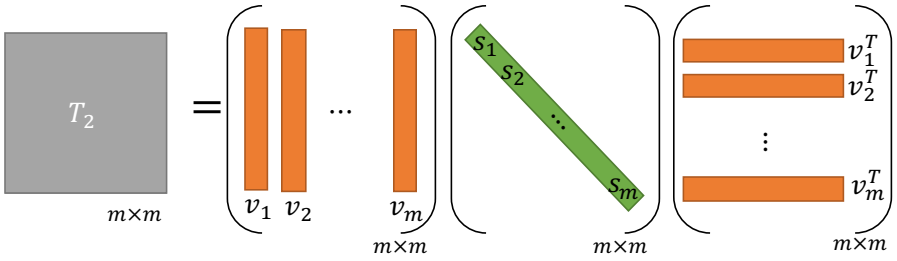

Figure 4.25: Schematic Representation of Singular Value Decomposition

Finally, for the third degree term:

$$f_3(\mathbf{z}) = \sum_{i_1=1}^{3} \sum_{i_2=i_1}^{3} \sum_{i_3=i_2}^{3} \alpha_{i_1,i_2,i_3} z_{i_1} z_{i_2} z_{i_3}$$
$$= \boldsymbol{\mathcal{T}}_3 \times_1 \mathbf{z} \times_2 \mathbf{z} \times_3 \mathbf{z} \tag{4.16}$$

$\boldsymbol{\mathcal{T}}_3$ stacks up the third order information in a 3rd order tensor.

$$\boldsymbol{\mathcal{T}}_3 = \begin{array}{|ccc|} \hline \alpha_{1,1,3} & \alpha_{1,2,3} & \alpha_{1,3,3} \\ \alpha_{2,1,3} & \alpha_{2,2,3} & \alpha_{2,3,3} \\ \alpha_{3,1,3} & \alpha_{3,2,3} & \alpha_{3,3,3} \\ \hline \end{array} \quad \begin{array}{|ccc|} \hline \alpha_{1,1,2} & \alpha_{1,2,2} & \alpha_{1,3,2} \\ \alpha_{2,1,2} & \alpha_{2,2,2} & \alpha_{2,3,2} \\ \alpha_{3,1,2} & \alpha_{3,2,2} & \alpha_{3,3,2} \\ \hline \end{array} \quad \begin{array}{|ccc|} \hline \alpha_{1,1,1} & \alpha_{1,2,1} & \alpha_{1,3,1} \\ \alpha_{2,1,1} & \alpha_{2,2,1} & \alpha_{2,3,1} \\ \alpha_{3,1,1} & \alpha_{3,2,1} & \alpha_{3,3,1} \\ \hline \end{array} \tag{4.17}$$

After decomposing the tensor using CPD, it results in the tensor product of 3 copies of $V^T\mathbf{z}$ where V is the CPD factor.

$$\boldsymbol{\mathcal{T}}_3 = \sum_{i=1}^{r} \mathbf{v}_i \otimes \mathbf{v}_i \otimes \mathbf{v}_i \tag{4.18}$$

where $\mathbf{v}_i$'s are columns of the mixing matrix. Note that this could be expanded to higher order polynomial degrees. In general the dimension of t_1 is $m \times 1$, T_2 is $m \times m$ and $\boldsymbol{\mathcal{T}}_3$ is $m \times m \times m$.

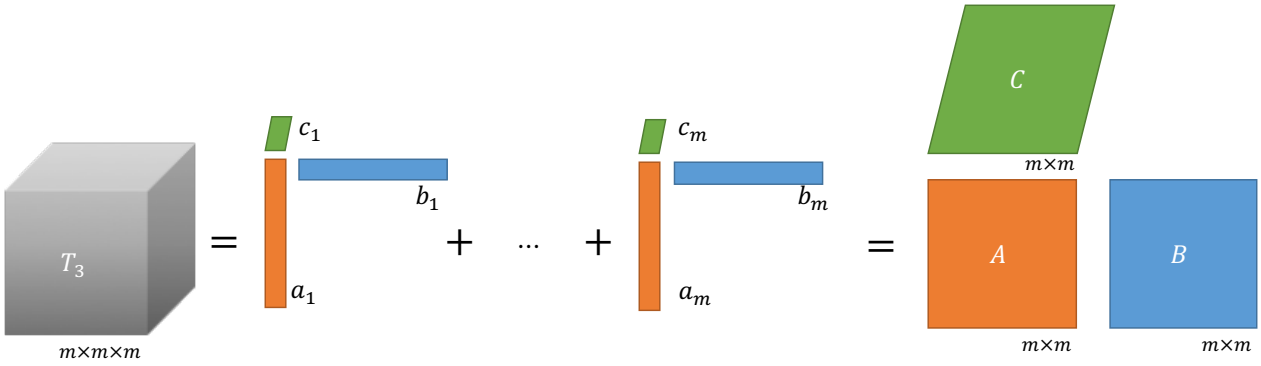

Figure 4.26: Schematic Representation of Canonical Polyadic Decomposition

The polynomial information will be used to find an initial solution for the optimization problem in (3.18). It is possible to use all first, second and third order information of the polynomial by combining the first order vector, singular vectors from the singular value decomposition (SVD) of the second order matrix (Fig 4.25) and CPD factors from the canonical polyadic decomposition (CPD) of the third order tensor (Fig 4.26). Note that since T_2 and $\boldsymbol{\mathcal{T}}_3$ are symmetric, the decomposition factors are equal.

The other approach is to only use the third order information in $\boldsymbol{\mathcal{T}}_3$. This way I will get rid of redundant information from $\mathbf{t}_1$ and T_2 at the cost of potentially losing information that does not appear in the third-degree terms. I will use the canonical polyadic decomposition approach as implemented in Tensorlab [89] to decompose the tensor, $\boldsymbol{\mathcal{T}}_3$, and use its factors as an initial solution to the optimization problem.

Comparing this method to previous proposed initial solutions in Sections 4.2 and 4.3, which involves computing the Hessian of the polynomial NARX model and evaluating it at all measurement points, this method is much less computationally expensive. Assuming that you have N data points, and m past inputs/outputs, the Hessian tensor will be $m \times m \times N$. On the other hand, the proposed method requires the CPD of a $m \times m \times m$ tensor. The

quality of the optimization will be compared in Section 4.5.

Algorithm 4 Decoupling Polynomial NARX Model from Input-Output Data Initialized by Decomposing the Polynomial Coefficients Information

1. Fit a polynomial NARX model to the provided data using standard approaches such as an orthogonal forward regression algorithm [3]

2. Collect the coefficients from the 3rd order terms in the polynomial and form a symmetric 3-way tensor

3. Compute the CPD of the coefficients tensor using standard tools [89]

4. Use the $\mathbf{V}$ from CPD as an initial estimate of the mixing matrix in the decoupled model. Refine the initial estimate using SLS based optimization (3.18)

4.4.1 Silver-Box Benchmark Results

According to Fig. 4.27 and Table 4.6, the optimization takes a few more steps to converge when using polynomial NARX coefficients for initialization, as compared to the Hessian decomposition. Also, it generates a model that is a little bit less accurate. But on the other hand, it is computationally cheaper compared to the other techniques.

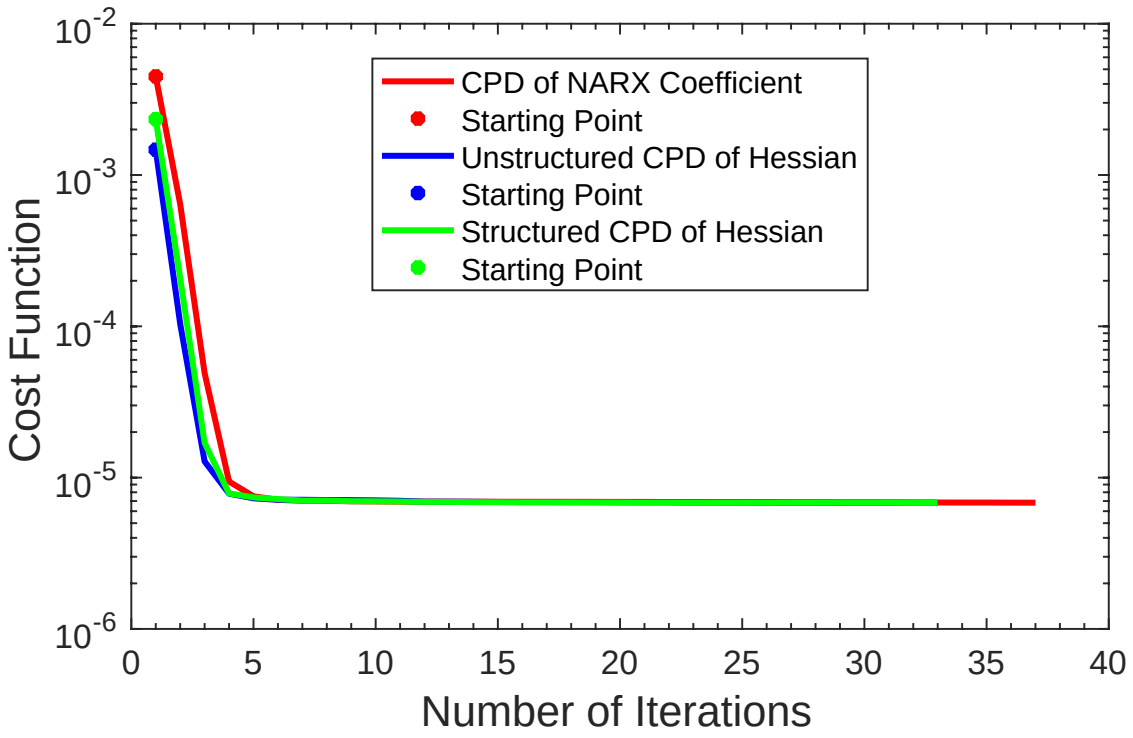

Figure 4.27: Cost Function During Optimization of Decoupled Polynomial NARX Model of the Silver-Box Benchmark for 3 Different Initialization Method for Better Visualization

	Simulation	1-Step Ahead Prediction
Multi-Sine(Fit%)	98.62 %	99.77 %
Sweep-Sine(Fit%)	96.46 %	99.57 %

Table 4.6: Accuracy of Decoupled NARX Model in Validation Data for the Silver-Box System Using NARX coefficients Tensor Decomposition Initialization

Figures 4.28 and 4.29 show the errors on validation data sets. As these figures indicate the prediction errors are very small.

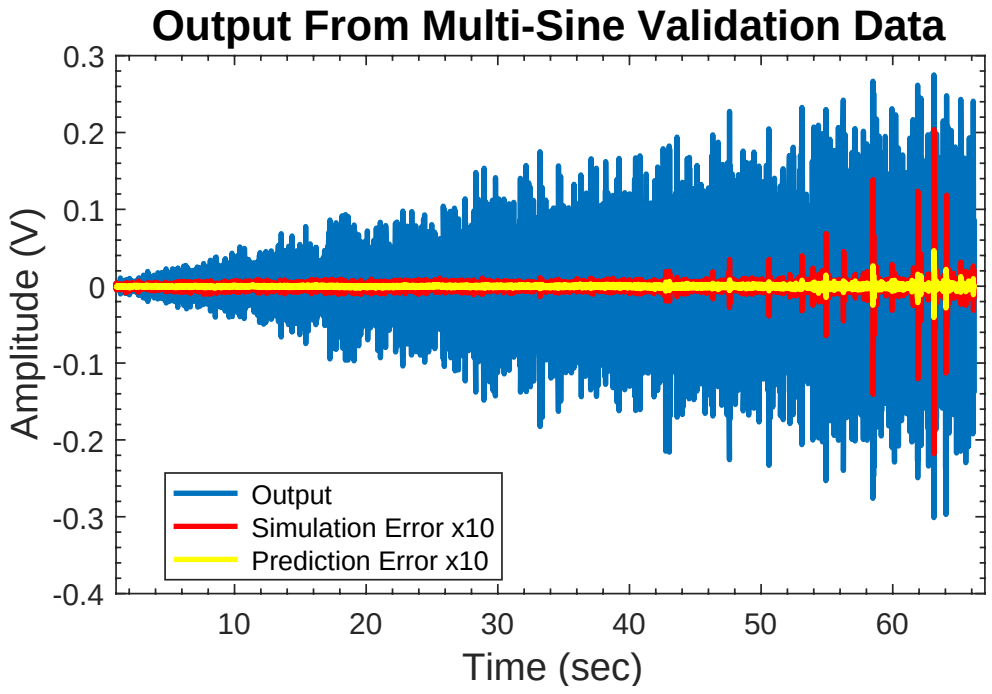

Figure 4.28: Prediction and Simulation Errors on the Validation Data for the Silver-Box Using NARX Model Coefficient Information for Initialization. Errors Are Scaled Up 10x for Better Visibility

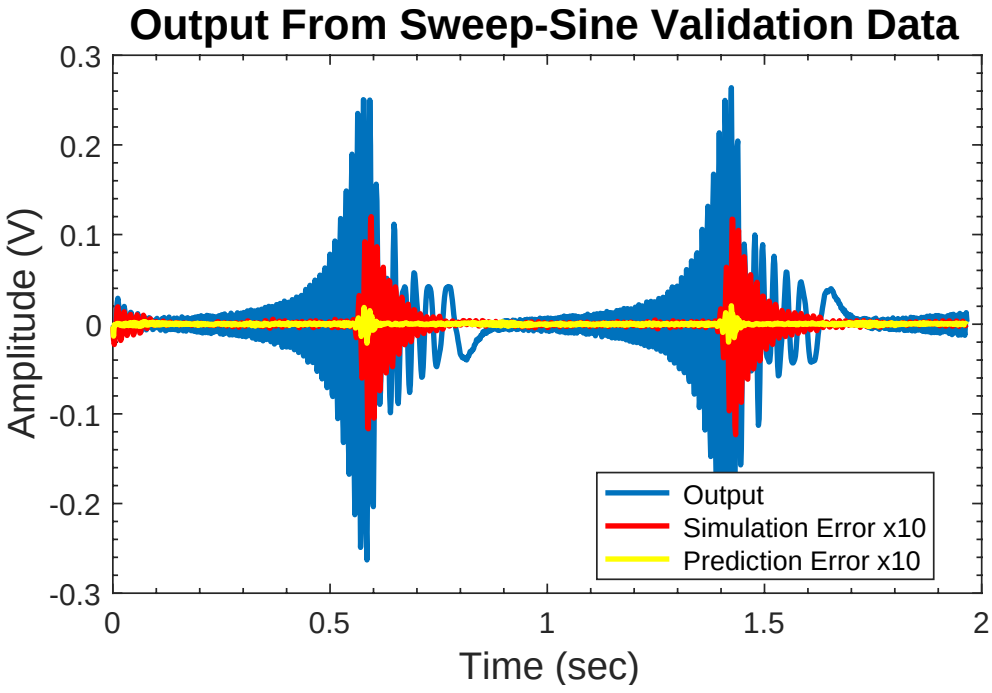

Figure 4.29: Prediction and Simulation Errors on the Sweep-Sine Validation Data for the Silver-Box Using NARX Model Coefficient Information for Initialization. Errors Are Scaled Up 10x for Better Visibility

4.4.2 Bouc-Wen Benchmark Results

In this section, the results for the optimization problem and model accuracy in case of initialization using NARX coefficients tensor decomposition is shown. Figure 4.30 shows the cost function convergence during the optimization for the original model with 15 past inputs and outputs. It is clear that the cost function takes more iterations to converge for when using NARX coefficients information for initialization, and comparing Table 4.7 and 4.2 shows a little bit less accuracy, but I have to mention that computational complexity is reduced in this case very impressively. For example, for this problem instead of decomposing a $30 \times 30 \times 40960$ tensor, I will decompose a $30 \times 30 \times 30$ tensor. In addition to the reduced storage requirements, the ALS algorithm used to compute the CPD of the tensor will involve a sequence of smaller, and hence much cheaper, least squares solutions.

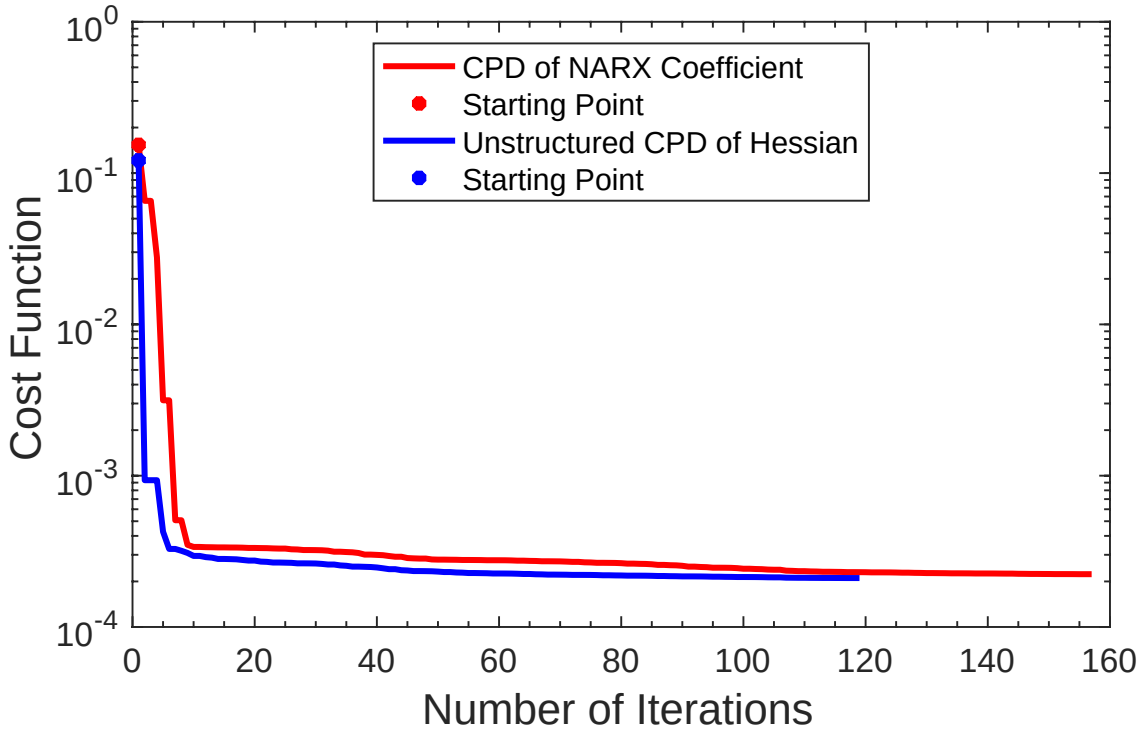

Figure 4.30: Cost Function During Optimization of Decoupled Polynomial NARX Model of the Bouc-Wen Benchmark for 2 Different Initialization Method for the Original Model with 15 Past Inputs and Outputs

	Simulation	1-Step Ahead Prediction
Multi-Sine(Fit%)	91.53 %	98.82 %
Sweep-Sine(Fit%)	97.51 %	99.51 %

Table 4.7: Accuracy of Decoupled NARX Model in Validation Data for the Bouc-Wen System Using NARX Coefficients Tensor Decomposition Initialization

As Fig. 4.31 indicates the prediction error is very small on the validation data set. Note that both of the errors shown in this figure are scaled up 10 times for better visualization.

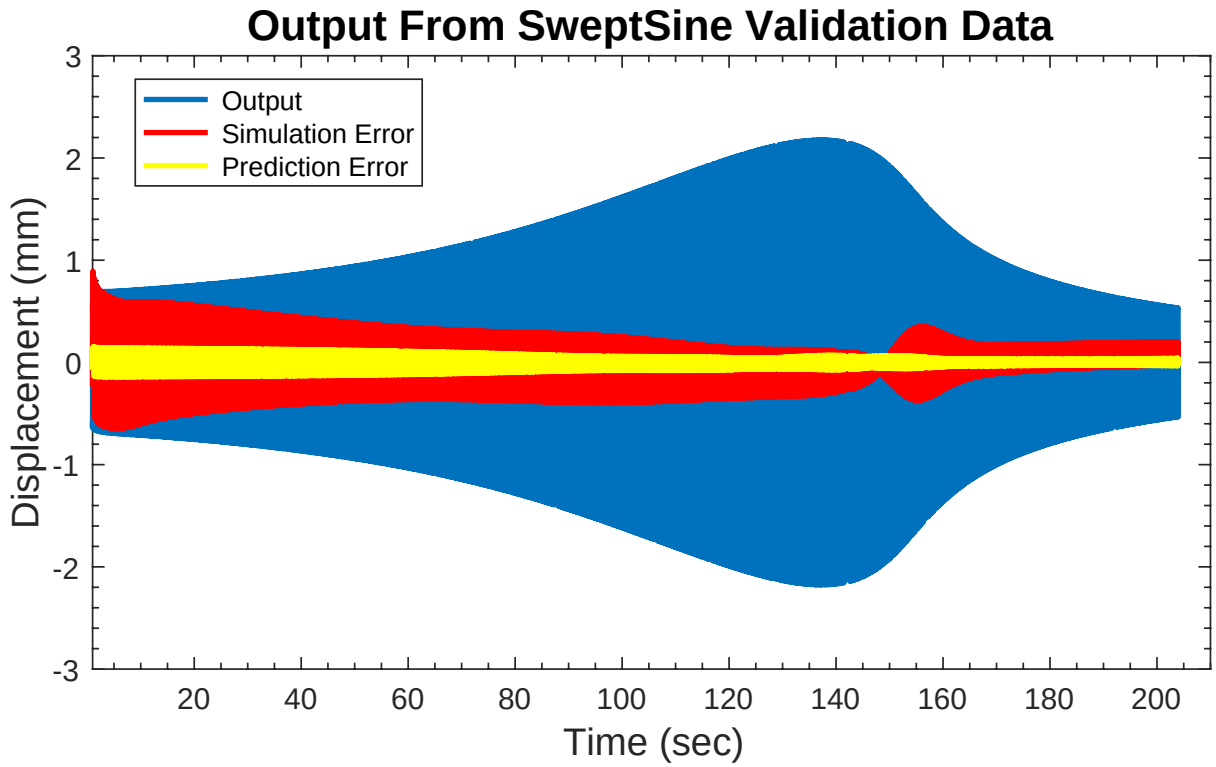

Figure 4.31: Prediction and Simulation Errors on the Sweep-Sine Validation Data for the Bouc-Wen Using NARX Model Coefficient Information for Initialization. Errors Are Scaled Up 10x for Better Visibility

4.5 Benchmark Results Comparison

In this section I will summarize and compare the results from all 3 different proposed initializations techniques.

4.5.1 Silver-Box Benchmark

Figure 4.32 includes the cost function convergence for all the initialization techniques and Table 4.8 summarizes the final cost and number of iterations for different approaches. It is very clear that initialization plays an important role in the non-convex optimization problem. Random initialization is not efficient and even sometimes generates unstable models, it is therefore unreliable.

Using any of the initialization approaches described in this chapter is beneficial and effective. It is a user choice to figure out what is more important for the application and find the balance between the accuracy and the computational complexity in every single case.

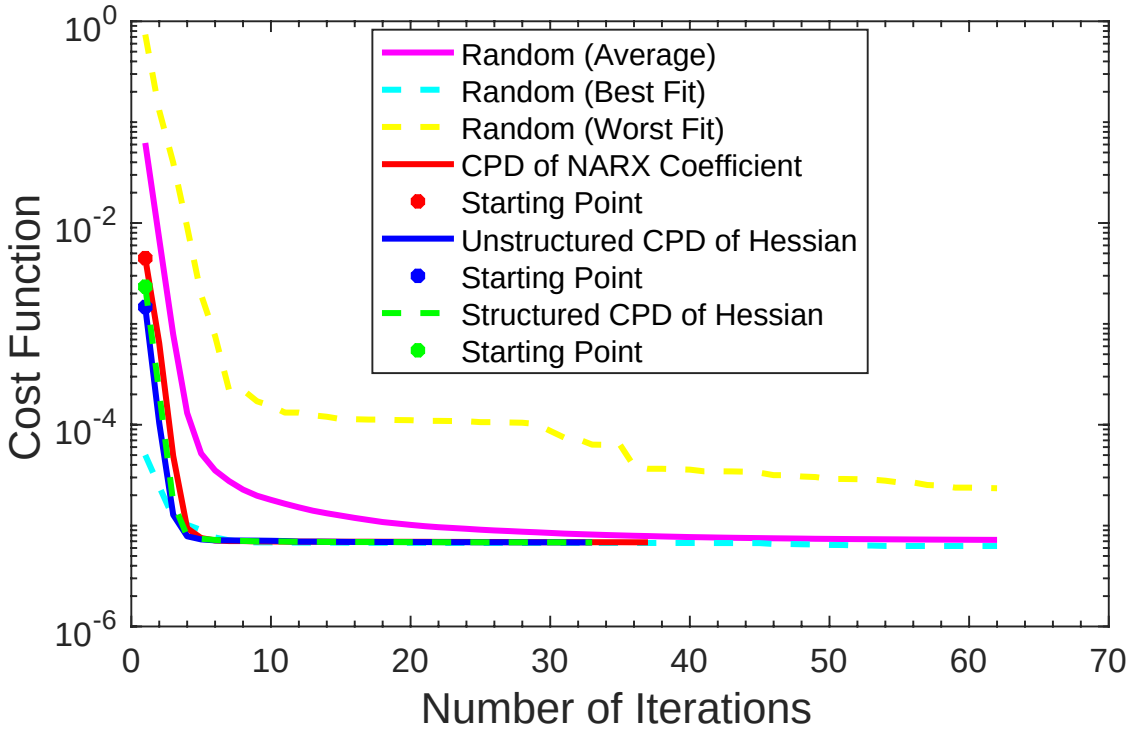

Figure 4.32: Cost Function During Optimization of Decoupled Polynomial NARX Model of the Silver-Box Benchmark

Initialization Approach	Final Cost	Number of Iterations
Random (Worst Fit)	7.015×10^{-6}	230
Random (Average)	6.882×10^{-6}	230
Random (Best Fit)	6.696×10^{-6}	173
CPD of NARX Coefficients	6.83×10^{-6}	37
Unstructured CPD of Hessian	6.809×10^{-6}	33
Structured CPD of Hessian	6.806×10^{-6}	33

Table 4.8: Comparison of the Cost and the Number of Iterations for different Initialization Techniques for the Silver-Box Model

Table 4.9 indicates how close the initial and final values of the mixing matrix, $\mathbf{V}$, are to each other, for all the intializations described in this chapter. As the numbers approach zero, the similarity between the 2 compared matrices increases. The results are computed using the `cpderr(X,Y)` in the Tensorlab [89] toolbox and they are the relative error in Frobenius norm between the two matrices. These are computed after sorting the columns to maximize the similarity.

		Initial Value				Final Value			
		Random	Hessian	Hessian w/struc.	NARX Coeff.	Random	Hessian	Hessian w/struc.	NARX Coeff.
Initial Value	Random	-	0.9060	0.9060	0.8994	0.7760	0.8603	0.8603	0.9016
	Hessian	0.9528	-	0.0000	0.0417	0.9712	0.3134	0.3134	0.1533
	Hessian w/struc.	0.8782	0.0000	-	0.0436	0.7951	0.2415	0.2415	0.1597
	NARX Coeff.	0.8709	0.0391	0.0391	-	0.9660	0.2069	0.2069	0.1310
Final Value	Random	0.7664	0.7951	0.7951	0.7938	-	0.7572	0.7572	0.7921
	Hessian	0.8294	0.2415	0.2415	0.1930	0.7572	-	0.0000	0.2188
	Hessian w/struc.	0.8294	0.2415	0.2415	0.1930	0.7572	0.0000	-	0.2188
	NARX Coeff.	0.8672	0.1597	0.1597	0.1479	0.7921	0.2188	0.2188	-

Table 4.9: Comparison of the Initial and Final Mixing Matrices, $\mathbf{V}$, for Different Initialization Technique Using Normalized Mean Square Error

4.5.2 Bouc-Wen Benchmark

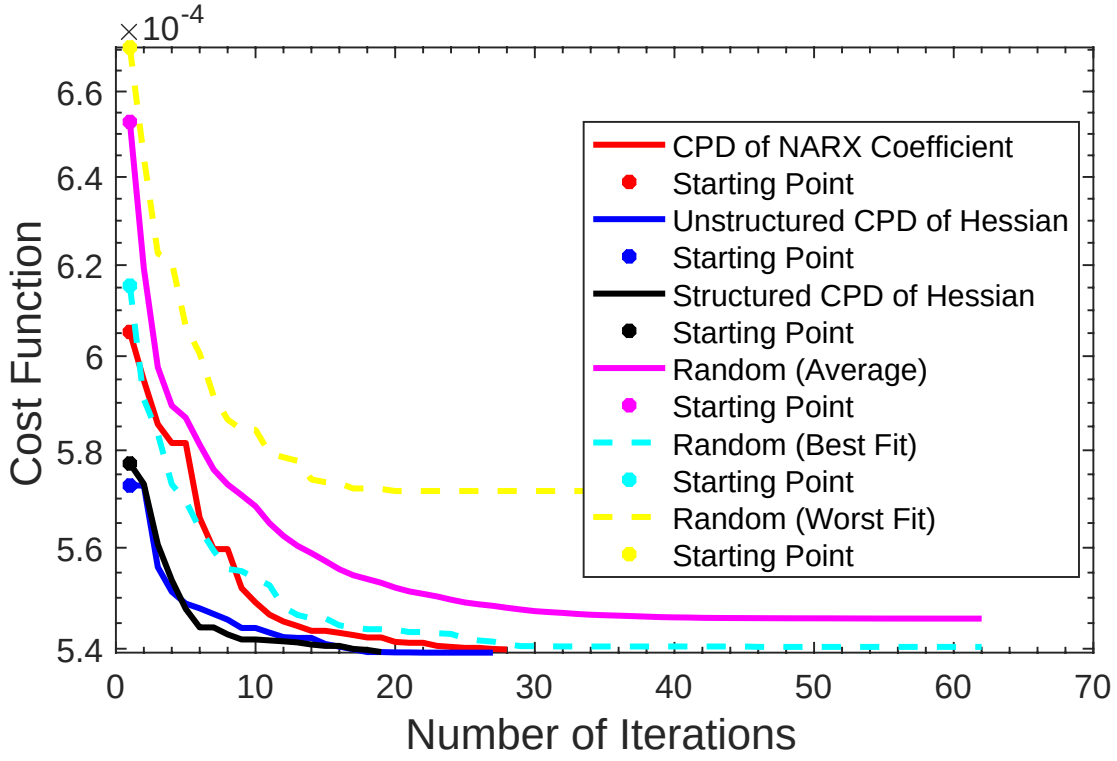

Figure 4.33: Cost Function During Optimization of Decoupled Polynomial NARX Model of the Bouc-Wen Benchmark for the Model with 4 Past Inputs and 6 Past Outputs

Initialization Approach	Final Cost	Number of Iterations
Random (Worst Fit)	5.716×10^{-4}	62
Random (Average)	5.459×10^{-4}	62
Random (Best Fit)	5.442×10^{-4}	62
CPD of NARX Coefficients	5.399×10^{-4}	28
Unstructured CPD of Hessian	5.392×10^{-4}	27
Structured CPD of Hessian	5.392×10^{-4}	19

Table 4.10: Comparison of the Cost and the Number of Iterations for different Initialization Techniques for the Bouc-Wen Model for the Model with 4 Past Inputs and 6 Past Outputs

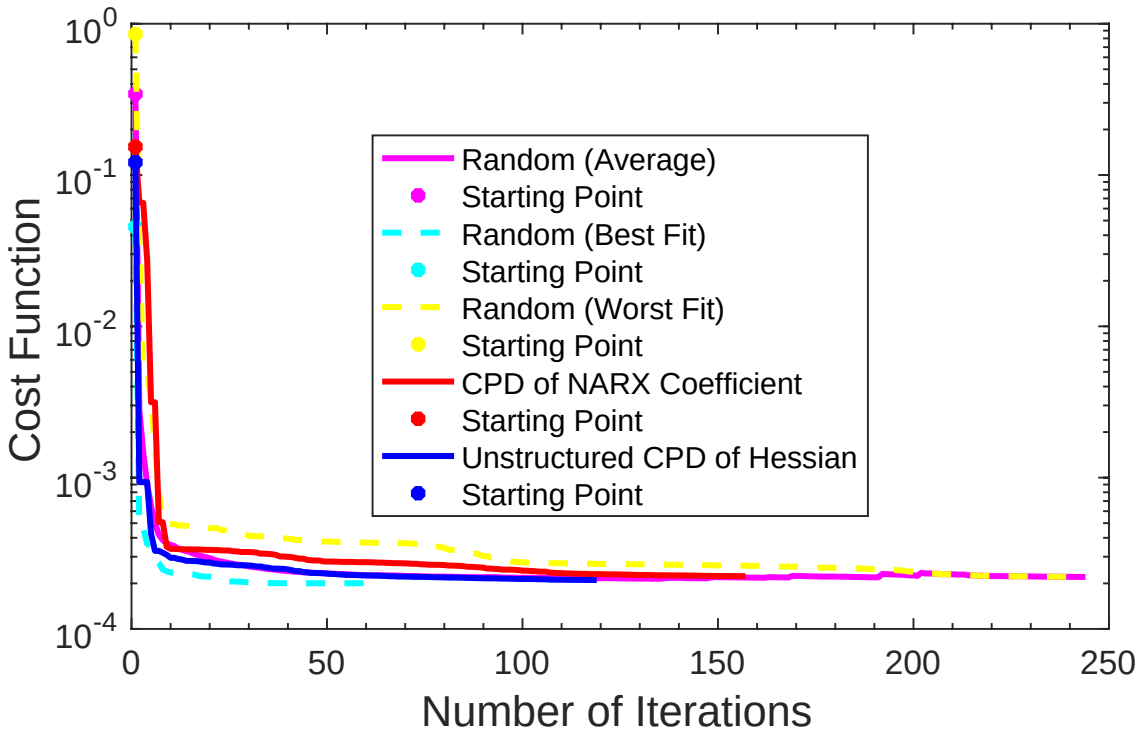

Figure 4.34: Cost Function During Optimization of Decoupled Polynomial NARX Model of the Bouc-Wen Benchmark for the Model with 15 Past Inputs and Outputs

Initialization Approach	Final Cost	Number of Iterations
Random (Average)	2.205×10^{-4}	244
CPD of NARX Coefficients	2.223×10^{-4}	157
Unstructured CPD of Hessian	2.106×10^{-4}	119
Structured CPD of Hessian	-	-

Table 4.11: Comparison of the Cost and the Number of Iterations for different Initialization Techniques for the Bouc-Wen Model for the Model with 15 Past Inputs and Outputs

To summarize this chapter, the unstructured Hessian and the coefficient CPD methods are both promising, and their choice will depend on the specific application. it seems that the polynomial constrained Hessian is very expensive to compute, and doesn't improve the results, but provides the univariate polynomials in each branch directly. Also there is the problem that the constrained CPD is itself a non-convex optimization, which is being used to initialized a non-convex optimization.

Chapter 5

Simulation Error Minimization Approach for Decoupled Polynomial NARX Models [1]

In the previous chapters, models were identified in a Prediction Error Minimization (PEM) framework. The model computes a 1-step ahead prediction of the current output sample, using measurements of current and past inputs, and past output signals. The optimization then minimizes a cost function, generally the sum of squared errors, based on the errors in this 1-step ahead predictions. This approach is commonly used to fit recursive parametric models, including the NARX and NARMAX models of nonlinear systems. This approach is shown in Fig. 5.1.

In the benchmark results, reported in Chapters 3 and 4, model accuracy was evaluated both in terms of their 1-step ahead predictions, and using the outputs generated in pure simulation mode, where the model does not have access to any past output measurements, as shown in Fig. 5.2. PEM models may not perform well in simulation – for example, at high sampling rates, the previous output may provide a very good prediction of the current

[1] The contents of this chapter have been presented at the 2019 American Control Conference, ACC2019, Philadelphia, PA, USA [97].

output, but such a model may well fail in simulation since it will not have access to the output samples that it relies on. The results tables in Chapters 3 and 4, where all models performed significantly worse in simulation than they did in prediction, confirms that.

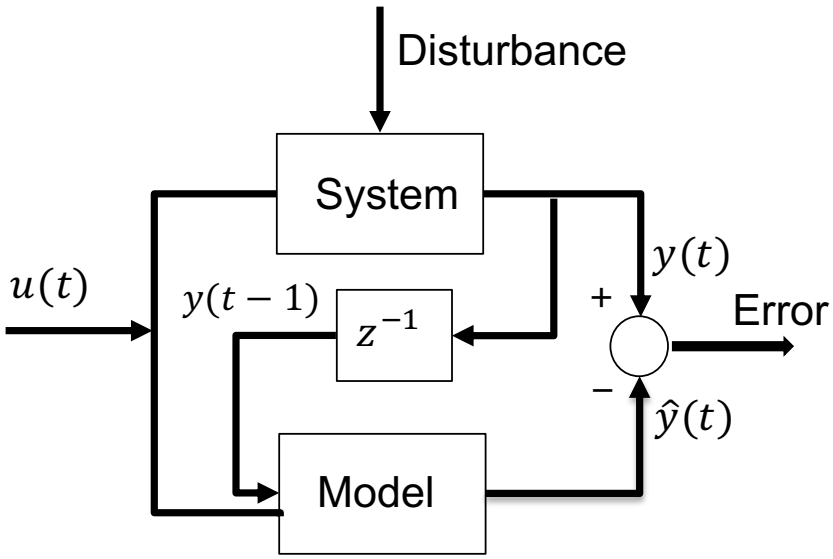

Figure 5.1: Prediction Error Minimization Schematic. Note that the Model Has Access to the Output of the System (After a 1-Step Delay)

An alternate approach to PEM is Simulation Error Minimization (SEM), where an optimization is used to minimize a cost function based on the model's simulation error. Depending on the application, it may be more appropriate to use models that are fit by minimizing the simulation error rather than 1-step ahead prediction error, however, it is a more complicated approach. In the simulation error minimization approach the model output is computed using past input measurements and simulated model outputs, as illustrated in Fig. 5.2. In this chapter, I will develop a method based on Simulation Error Minimization to identify decoupled polynomial NARX models.

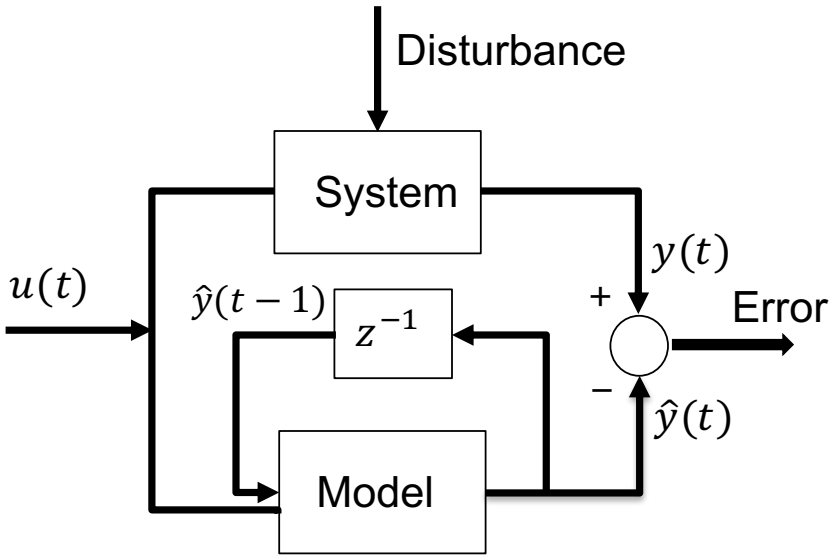

Figure 5.2: Simulation Error Minimization Schematic. Here the Model Does Not Have Access to Any Past Outputs from the System

5.1 Simulation Error Minimization

Consider the output of the DP-NARX model as described in Chapter 3, the output was represented by the following equations:

$$y(t) = y_0 + \sum_{k=1}^{r} g_k(x_k(t)) + e(t) \tag{5.1}$$

$$g_k(x) = \sum_{l=1}^{n} \mathbf{C}(l,k) x_k^l \tag{5.2}$$

$$x_k(t) = \sum_{j=1}^{n_y} \mathbf{V}(j,k) y(t-j) + \sum_{i=0}^{n_u} \mathbf{V}(i+n_y+1,k) u(t-i) \tag{5.3}$$

where $e(t)$, the equation error, is a sequence of IID zero mean random variables.

The model's 1-step ahead prediction can be computed as follows. Let $\mathbf{z}(t)$ be a vector

containing the past input and output variables:

$$\mathbf{z}(t) = \begin{bmatrix} y(t-1) & \dots & y(t-n_y) & u(t) & u(t-1) & \dots & u(t-n_u) \end{bmatrix}^T \tag{5.4}$$

and let $\mathbf{x}(t)$ be a vector containing the intermediate signals in each of the branches:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) & x_2(t) & \dots & x_r(t) \end{bmatrix}^T \tag{5.5}$$

$$= \boldsymbol{V}^T \mathbf{z}(t) \tag{5.6}$$

The 1-step ahead prediction is then computed as:

$$\hat{y}(t) = y_0 + \sum_{k=1}^{r} g_k(x_k(t)) \tag{5.7}$$

In order to identify the decoupled polynomial NARX model using PEM, the optimization problem in (5.8) should be solved.

$$V_N(\mathbf{V}, \mathbf{c}) = ||\mathbf{y} - \hat{\mathbf{y}}(\mathbf{V}, \mathbf{c})||_2^2 \tag{5.8}$$

In SEM, the vector $\mathbf{z}(t)$ is constructed as follows

$$\mathbf{z}(t) = \begin{bmatrix} \hat{y}(t-1) & \dots & \hat{y}(t-n_y) & u(t) & u(t-1) & \dots & u(t-n_u) \end{bmatrix}^T \tag{5.9}$$

using measurements of the present and past input values, but feeding back past model (simulated) outputs, to be used as the past output terms. The rest of the computations (Equations (5.6), (5.7) and (5.8)) remain unchanged, at least in form.

The differences between the two cost function for PEM and SEM are two-fold: 1) in SEM the model output $\hat{y}$ depends on the past inputs and previous values of the model output and 2) because of 1, the output is a nonlinear function of the polynomial coefficients, hence SLS cannot be used.

As mentioned in previous chapters, this optimization is a nonlinear optimization. Therefore, computation of the Jacobian of the model output is necessary regardless of the choice of optimization algorithm.

In PEM vector $z(t)$ in (5.4), is a vector containing past inputs and past outputs. Hence, it is a fixed vector build from the measured data. In this case, computing the Jacobian is straightforward as discussed in Section 3.1.1. On the other hand, in SEM, the vector $z(t)$ in (5.9) is not a fixed vector anymore since the past outputs must be generated by the model, and now depends on the parameters. In such a case, computing the Jacobian of the output is more complicated.

In [48], SEM was used to identify polynomial nonlinear state-space models. Paduart showed that the partial derivatives of the simulation error with respect to model parameters could be computed using recursive nonlinear filtering operations. A similar approach can be applied to the decoupled polynomial NARX model.

5.1.1 Jacobian Computations

The nonlinear optimization problem is solved using the Levenberg–Marquardt algorithm [98, 99]. It requires the computation of the Jacobian of the model output with respect to the model parameters. Here I will show the calculation of the partial derivatives of the model output with respect to the constant output term, y_0, the polynomial coefficients, $\mathbf{C}$, and finally the elements of mixing matrix, $\mathbf{V}$.

Consider the partial derivative of the simulation output (5.1), with respect to the constant term y_0:

$$\frac{\partial \hat{y}(t)}{\partial y_0} = 1 + \sum_{k=1}^{r} g_k'(x_k(t)) \frac{\partial x_k(t)}{\partial y_0} \tag{5.10}$$

where,

$$\frac{\partial x_k(t)}{\partial y_0} = \boldsymbol{V}(:,k)^T \frac{\partial \mathbf{z}(t)}{\partial y_0} \tag{5.11}$$

$$\frac{\partial \mathbf{z}(t)}{\partial y_0} = \left[\begin{array}{cccccc} \frac{\partial \hat{y}(t-1)}{\partial y_0} & \cdots & \frac{\partial \hat{y}(t-n_y)}{\partial y_0} & 0 & \cdots & 0 \end{array} \right]^T \tag{5.12}$$

To simplify the notation, let $\alpha(t) = \partial \hat{y}(t)/\partial y_0$, and expand (5.10):

$$\alpha(t) = 1 + \sum_{k=1}^{r} g_k'(x_k(t)) \left(\sum_{j=1}^{n_y} \mathbf{V}(j,k) \right) \alpha(t-j) \tag{5.13}$$

$$= 1 + \sum_{j=1}^{n_y} \left(\sum_{k=1}^{r} g_k'(x_k(t)) \boldsymbol{V}(j,k) \right) \alpha(t-j) \tag{5.14}$$

Thus, the partial derivative may be computed via a time-varying recursive filtering operation.

Similarly, consider the partial derivatives of the model output with respect to the polynomial coefficients:

$$\frac{\partial \hat{y}(t)}{\partial \boldsymbol{C}(a,b)} = x_b^a(t) + \sum_{k=1}^{r} g_k'(x_k(t)) \frac{\partial x_k(t)}{\partial \boldsymbol{C}(a,b)} \tag{5.15}$$

Clearly, this may also be computed using a time-varying recursive filter. As above let $\beta(t) = \partial \hat{y}(t)/\partial \mathbf{C}(a,b)$, then we have:

$$\beta(t) = x_b^a(t) + \sum_{j=1}^{n_y} \left(\sum_{k=1}^{r} g_k'(x_k(t)) \boldsymbol{V}(j,k) \right) \beta(t-j) \tag{5.16}$$

Finally, use the chain rule to compute the partial derivatives with respect to the elements of the mixing matrix $\boldsymbol{V}$.

$$\frac{\partial \hat{y}(t)}{\partial \boldsymbol{V}(a,b)} = \sum_{k=1}^{r} g_k'(x_k(t)) \frac{\partial x_k(t)}{\partial \boldsymbol{V}(a,b)} \tag{5.17}$$

Consider the term $\partial x_k(t)/\partial \boldsymbol{V}(a,b)$. Recall from (5.3) that $x_k(t) = \boldsymbol{V}(:,k)^T \mathbf{z}(t)$. Thus, provided $b \neq k$, we need only consider the dependence of the output elements in $\mathbf{z}(t)$ on $\boldsymbol{V}(a,b)$. Hence

$$\frac{\partial x_k(t)}{\partial \boldsymbol{V}(a,b)} = \sum_{j=1}^{n_y} \boldsymbol{V}(j,k) \frac{\partial \hat{y}(t-j)}{\partial \boldsymbol{V}(a,b)} \qquad b \neq k$$

If $b = k$, then

$$\frac{\partial x_k(t)}{\partial \boldsymbol{V}(a,b)} = \mathbf{z}_a(t) + \sum_{j=1}^{n_y} \boldsymbol{V}(j,k) \frac{\partial \hat{y}(t-j)}{\partial \boldsymbol{V}(a,b)}$$

where $\mathbf{z}_a(t)$ is the ath element of the vector $\mathbf{z}(t)$. Therefore, the partial derivatives of the model output with respect to the elements of $\boldsymbol{V}$ are given by the following time-varying recursive filtering operation, where $\gamma(t) = \partial \hat{y}(t)/\partial \mathbf{V}(a,b)$:

$$\begin{aligned} \gamma(t) &= g_b'(x_b(t))\mathbf{z}_a(t) \\ &\quad + \sum_{j=1}^{n_y} \left(\sum_{k=1}^{r} g_k'(x_k(t)) \boldsymbol{V}(j,k) \right) \gamma(t-j) \end{aligned} \tag{5.18}$$

Now that we have access to the Jacobians, we can solve the optimization problem in Eq. (3.18) using an iterative quasi-Newton nonlinear least-squares optimization. Note that the derivatives with respect to the elements of $\mathbf{C}$ depend on the past outputs, and hence on the values of $\mathbf{C}$. Thus, the simulation output is not linear in the $\mathbf{C}$ parameters, so SLS cannot be used in this case.

5.1.2 Algorithm Summary

The following algorithm identifies a decoupled P-NARX model from input/output data using simulation error minimization.

Algorithm 5 Identification of a Decoupled Polynomial NARX Model Using Simulation Error Minimization

1. Fit a decoupled polynomial NARX model to the data, using prediction error minimization as described in Chapters 3 and 4

2. Use the PEM model as the initial simulation model

3. Use an iterative nonlinear least squares algorithm to minimize the simulation error. At each iteration, the columns of the Jacobian are computed using Equations (5.14), (5.16) and (5.18)

5.2 Benchmark Results

The simulation error minimization algorithm described in Section 5.1 is used to fit a decoupled polynomial NARX model to data from the two benchmark problems introduced in Section 3.3.

5.2.1 Silver-Box Benchmark Results

The selected model for the Silver-Box is the same model used in previous chapters. It has 3 lagged inputs, 3 lagged outputs and no input delay. The decoupled polynomial NARX model has 4 branches with 3rd degree polynomial nonlinearities.

Figure 5.3 shows the simulation errors in the triangular validation data for the models identified using SEM (in red) and PEM (in yellow) while Table 5.1 shows both the prediction and simulation accuracy for both models. As both the figure and the table confirm, the SEM model is more accurate in simulation than the PEM model. Similarly, the PEM model outperformed the SEM model in the prediction task. In all cases, these results come from the validation data, which were not used at any stage of the identification process. Since both the PEM and SEM models outperformed the other in their specific tasks (prediction vs. simulation), the results are as expected.

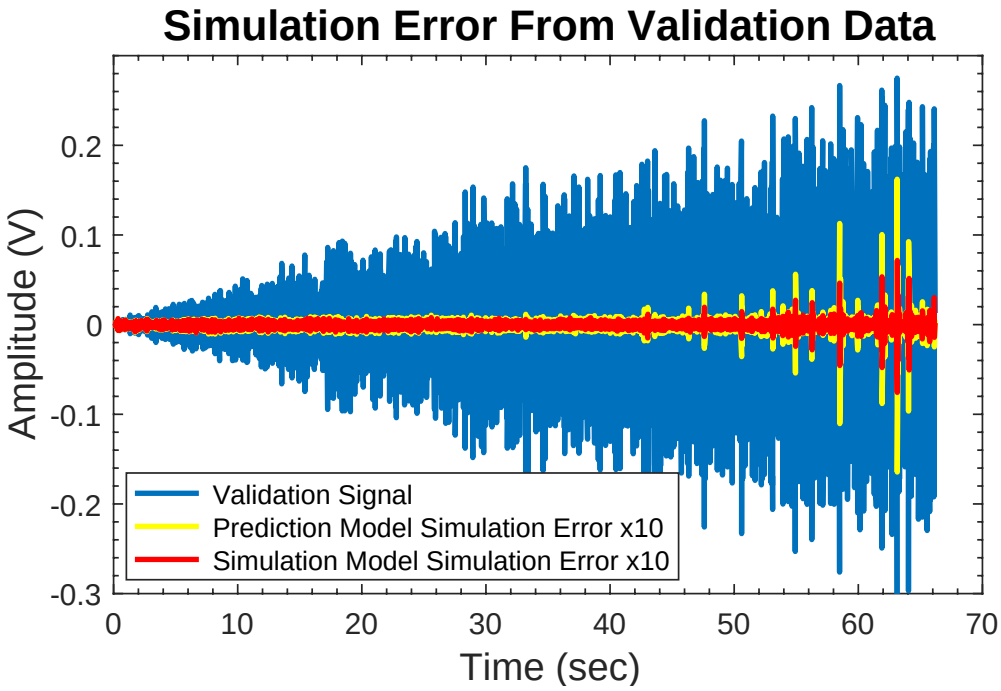

Figure 5.3: Simulation Errors on the Validation Data for the Silver-Box Using Prediction and Simulation Model. Errors Are Scaled Up 10x for Better Visibility

	Identification Method	
	Prediction Error Minimization	Simulation Error Minimization
Simulation Error(Fit%)	98.85%	99.33%
Prediction Error(Fit%)	99.78%	99.68%

Table 5.1: Validation Results for the Models Fitted Using PEM and SEM for the Silver-Box Benchmark

The expectation is true for the sweep-sine validation signal as well, as Fig. 5.4 and Table 5.2 confirm that the simulation error model outperforms the prediction error model in the simulation task, but the PEM model was (slightly) better at 1-step ahead prediction.

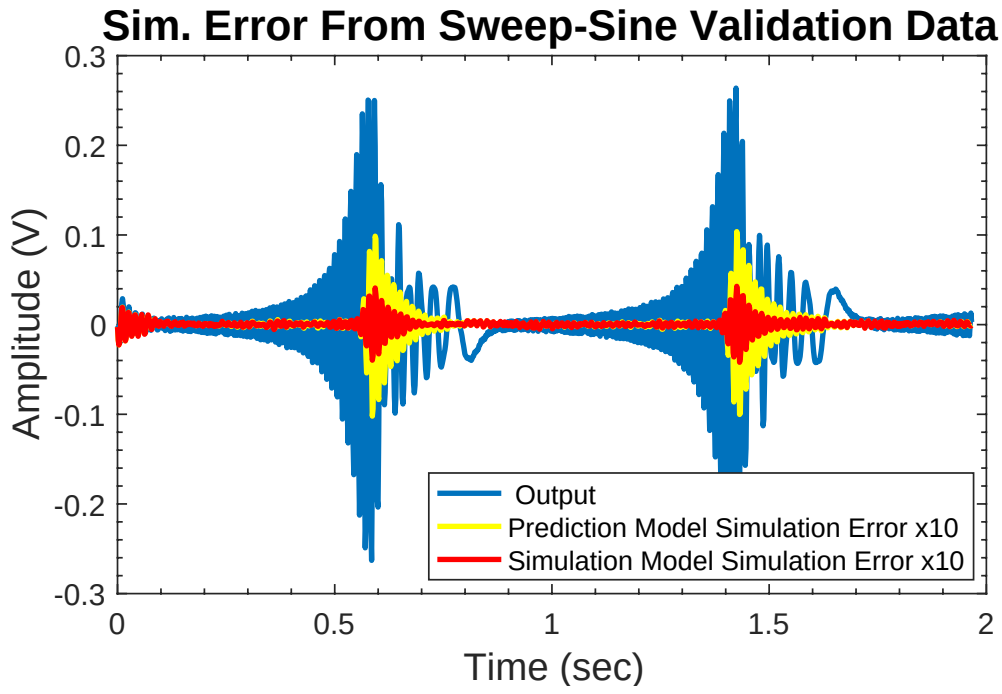

Figure 5.4: Simulation Errors on the Sweep-Sine Validation Data for the Silver-Box Using Prediction and Simulation Model. Errors Are Scaled Up 10x for Better Visibility

	Identification Method	
	Prediction Error Minimization	Simulation Error Minimization
Simulation Error(Fit%)	96.98%	98.70%
Prediction Error(Fit%)	99.60%	99.58%

Table 5.2: Sweep-Sine Validation Results for the Models Fitted Using PEM and SEM for the Silver-Box Benchmark

The results obtained from the Silver-Box benchmark, show that using simulation error minimization is a promising approach for developing models intended for simulation applications. For example, in model-based predictive control, where the model must repeatedly simulate the system out to the prediction horizon, without the benefit of any output measurements. In that case, it is more suitable to use SEM to find the model parameters rather than using PEM.

5.2.2 Bouc-Wen Benchmark Results

Previously, I had identified decoupled NARX models using prediction error minimization for the Bouc-Wen benchmark. The best such model had 5 branches each of which filtered 15 past inputs and 15 past outputs and transformed the results with a 9th order polynomial nonlinearity. This model was used as the starting point for the simulation error minimization optimization.

The expectation is that the SEM model will outperform the PEM in simulation task, as was the case with the Silver-Box benchmark. According to Table 5.3 this is not true for the multi-sine validation signal for the Bouc-Wen benchmark and the model trained using SEM performed worse than the PEM model in both prediction and simulation tests. The reason for the poor simulation performance can be seen in Fig. 5.5. The SEM trained model produced a large spike in the simulation error for the multi-sine validation signal followed by a transient decay, at about $t = 4.5$ seconds. This large error dominates the RMS results.

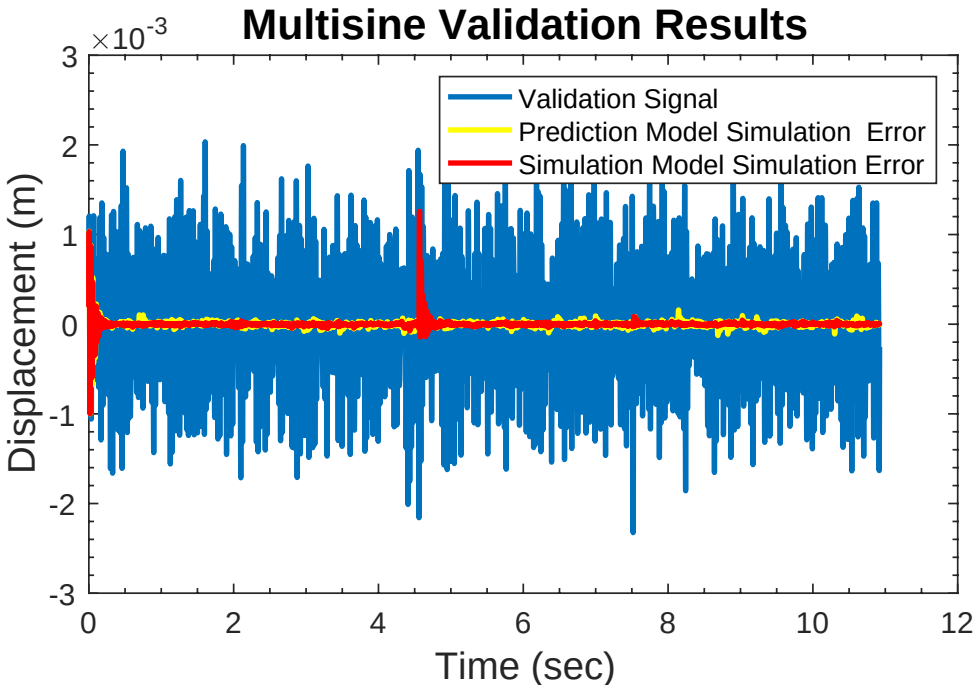

Figure 5.5: Simulation Errors on the Multi-Sine Validation Data for the Bouc-Wen Using Prediction and Simulation Model

	Identification Method	
	Prediction Error Minimization	Simulation Error Minimization
Simulation Error(Fit%)	91.96%	90.60%
Prediction Error(Fit%)	98.54%	81.11%

Table 5.3: Multi-Sine Validation Results for the Models Fitted Using PEM and SEM for the Bouc-Wen Benchmark

In investigating the poor performance on the multi-sine signal, I examined the $x_k(t)$, the inputs to the nonlinearities in all of the branches. As shown in Fig. 5.6, two of the branches (path 1 and 4) remained inside the limits excited by the identification data, show as the dashed red lines in the figure, at all times, while the other three branches (path 2, 3 and 5 at around $t = 4.5$ seconds) exceed the identification limits. Fig. 5.7 shows the simulation error signal (topmost panel) as well as the inputs to the nonlinearities in branches 2, 3 and 5 (lower 3 panels). These three intermediate signals exceed the range of the identification data at about $t = 4.5$ seconds. To be more exact, in branch 2 and 5 this issue happens at $t = 4.56$ seconds and in the 3rd branch, this last violation happens 13 samples after the first two, at $t = 4.577$ seconds.

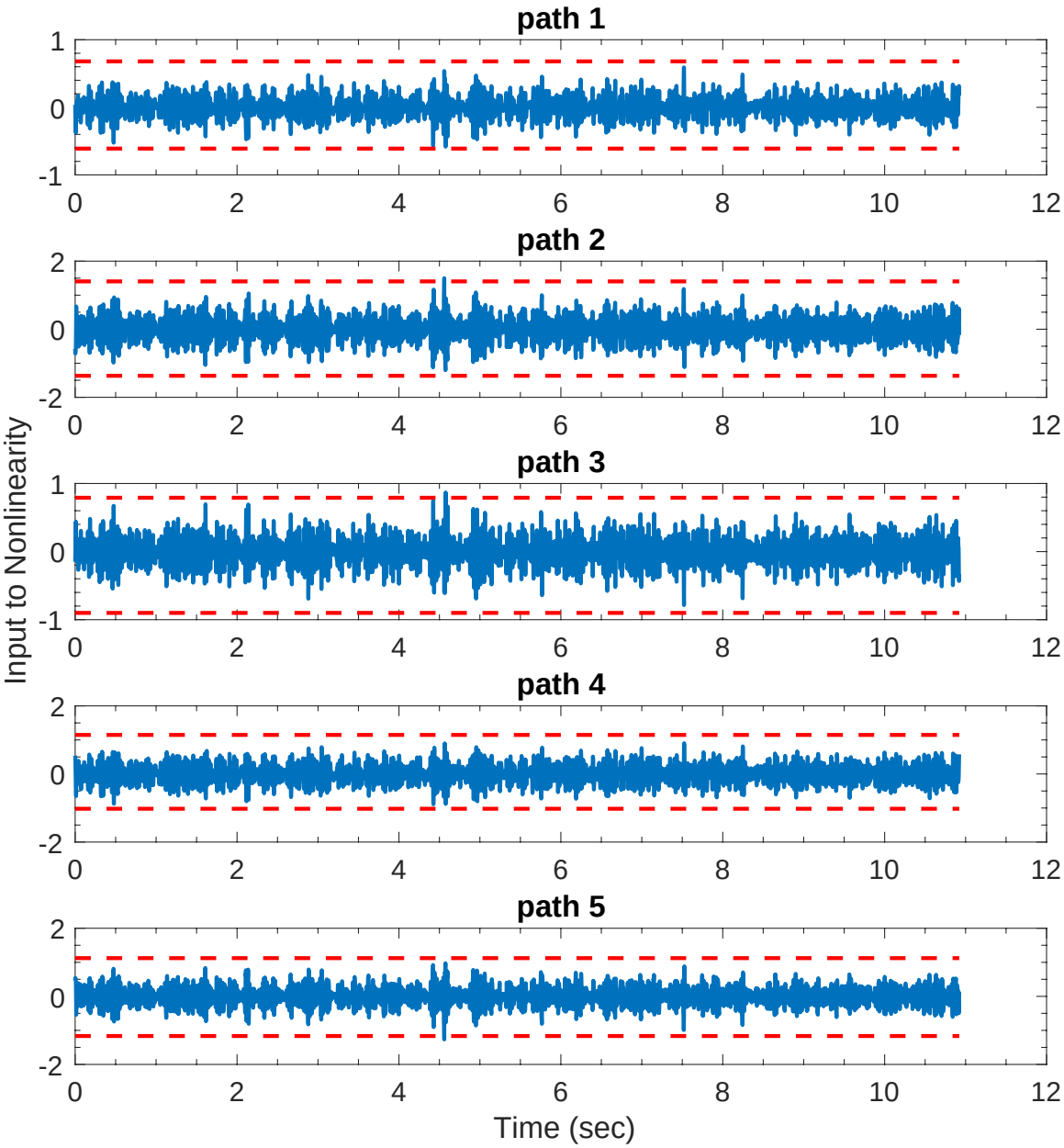

Figure 5.6: Validation Signals of All 5 Branches for the Bouc-Wen Using Simulation Error Model. The Dashed Red Lines Represent the Limits Excited by the Identification Data and the Blue Lines Are the Validation Signals

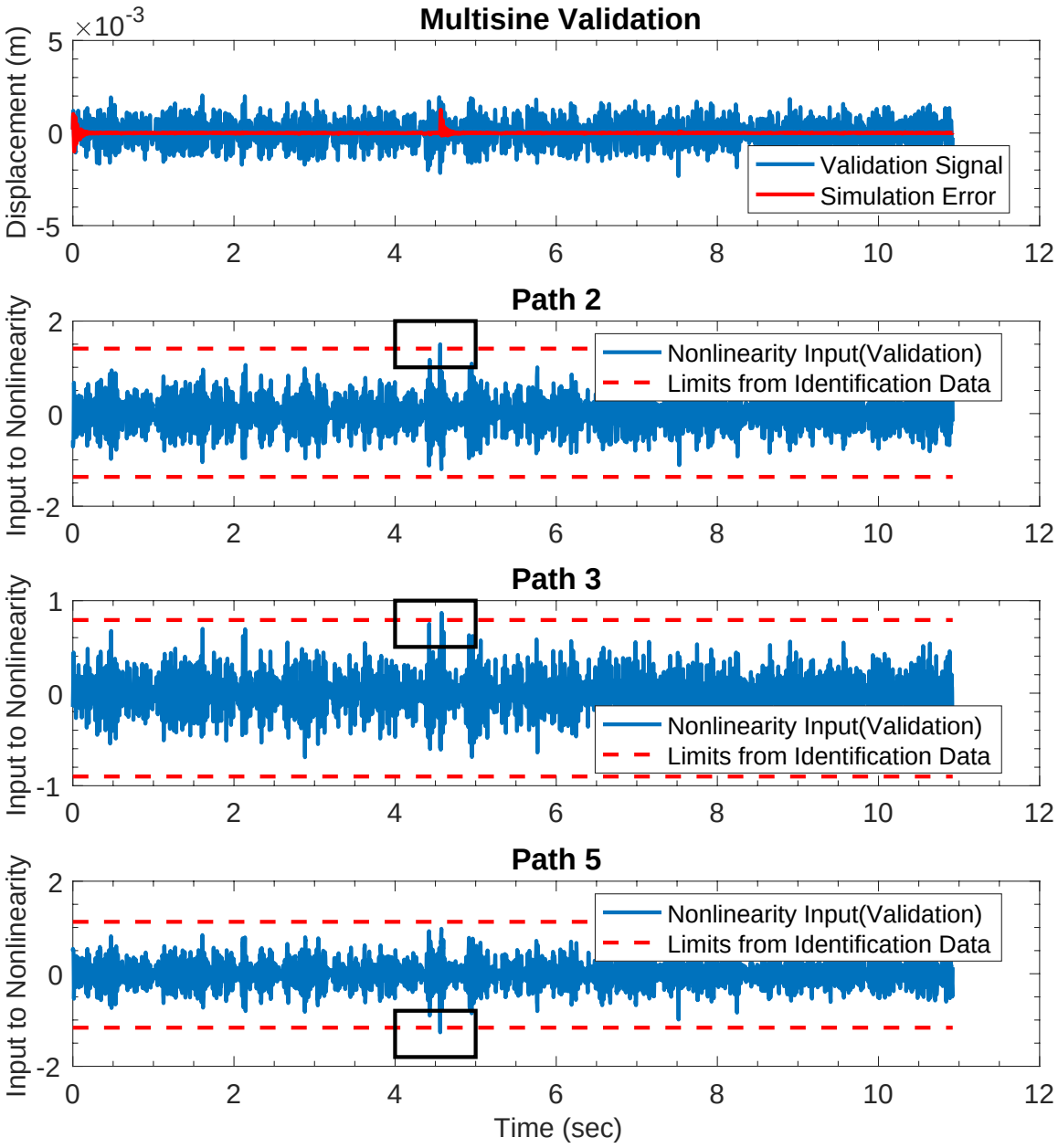

Figure 5.7: Validation Signals of 2nd, 3rd and 5th Branch for the Bouc-Wen Using Simulation Error Model Where Input to the Nonlinearity Exceeds Identification Limits

A closer look at the point where path 2, 3 and path 5 behave unexpectedly, as shown in Fig. 5.8, shows that exactly when the input to nonlinearity exceeds the limits, the error on the validation data has a huge spike followed by a transient decay. The spike in the error signal at $t = 4.56$ seconds corresponds to the violation of the nonlinearity inputs in paths 2 and 5 which are larger compared to the later violation in path 3. There is, perhaps, a

small change in the error signal at this point, but the error is still dominated by the decaying transient that started with the first violations at $t = 4.56$ seconds.

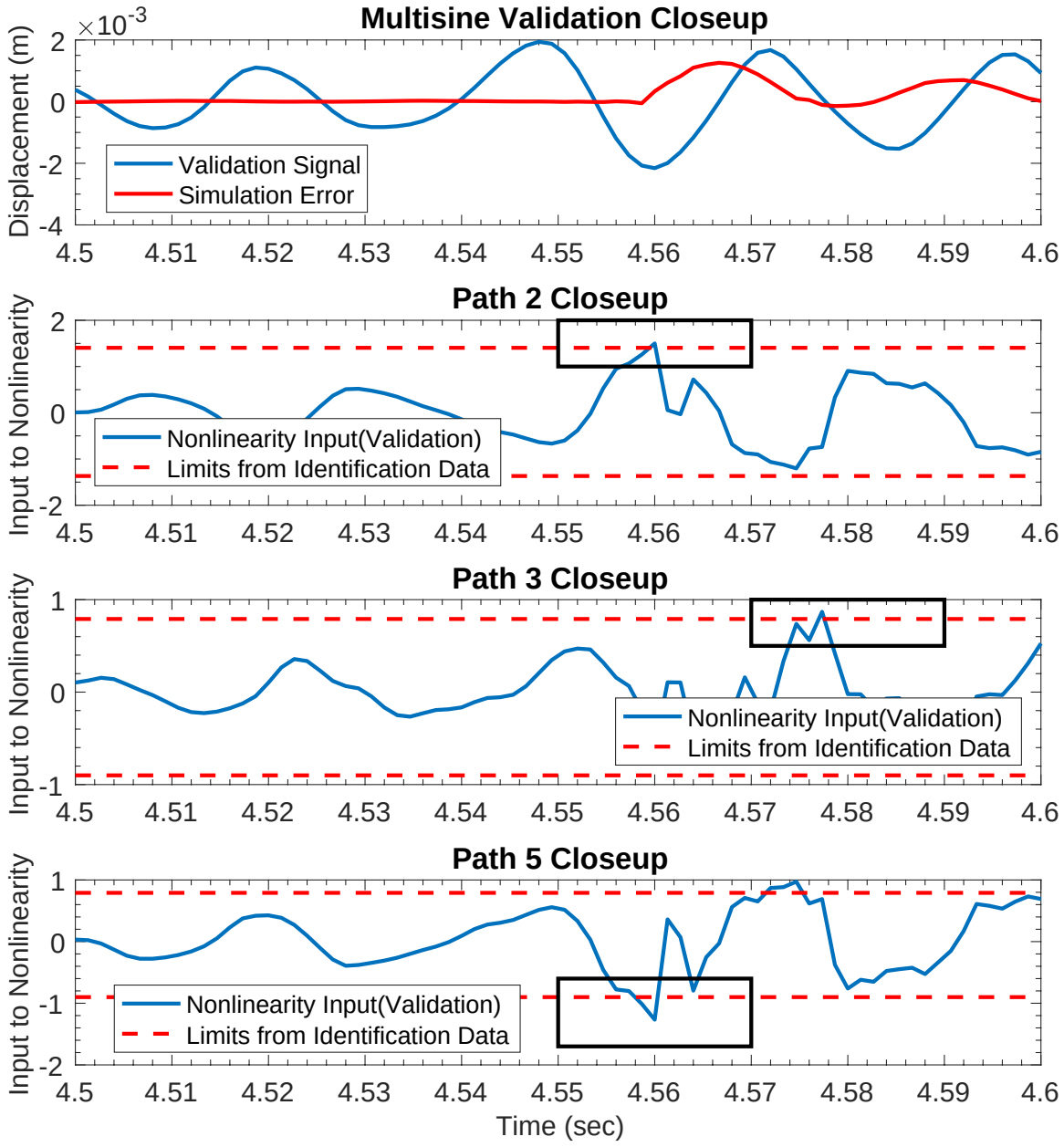

Figure 5.8: Zoomed Validation Signals of 2nd and 3rd Branch for the Bouc-Wen Using Simulation Error Model Where Input to the Nonlinearity Exceeds Identification Limits

Figure 5.9 shows the reason that is causing the poor performance of the model in the specific paths. Based on this figure, an extrapolation happened in the nonlinearity of these three paths. In the figure, the red points in the black box are the extrapolated points. I

believe that the reason for higher impact of changes in path 2 and 5 on the error signal relates to the shape of the nonlinearities, for path 2 and 5 the slope is very steep, whereas in path 3, the extrapolated nonlinearity is very shallow and curves back down. Since there is no information available on how the actual nonlinearity is supposed to behave at these points, therefore it is hard to quantify the resulting error.

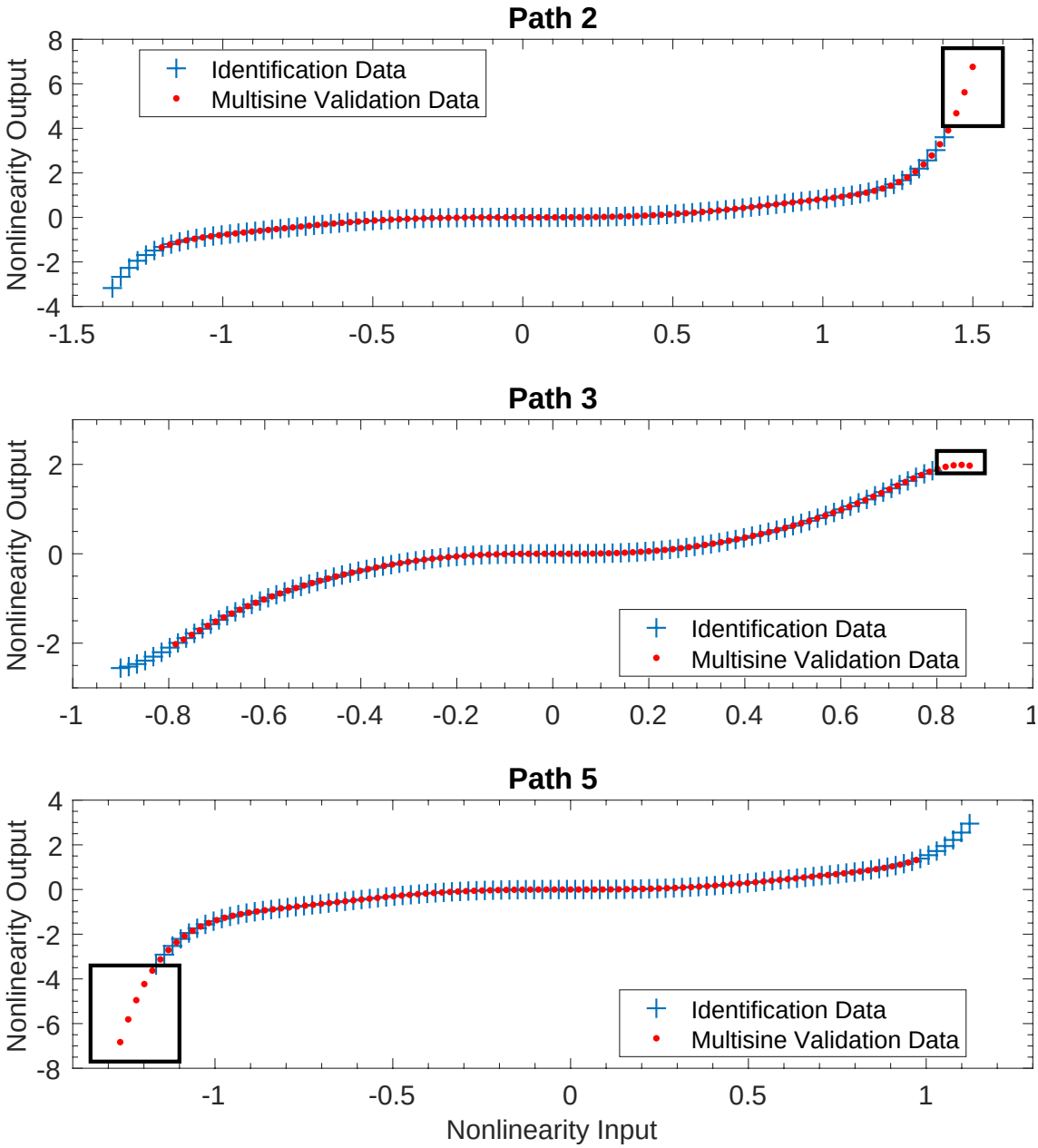

Figure 5.9: Nonlinearity of 2nd and 3rd Branch for the Bouc-Wen Using Simulation Error Model Where Shows Extrapolation in Validation Data Set

It is clear that a small excursion in the input produced a very large change in the output signal. Indeed, if the error is computed for the last 6 seconds of data, so that the large extrapolation error is removed, the SEM model considerably outperforms the PEM model and achieves 98.42% accuracy.

As a solution to prevent the extrapolation in the branches, as shown in the upper part of Fig. 5.11, I forced the inputs to the nonlinearity to stay inside the bounds on each signal that were encountered during the identification data as illustrated in Fig. 5.10. Thus, at any point in time where the input was exceeding the bounds, I forced it to keep the maximum or minimum possible value instead, which is the identification limit. However, not knowing the true system behavior, it is not possible to compute the resulting error introduced to the model.

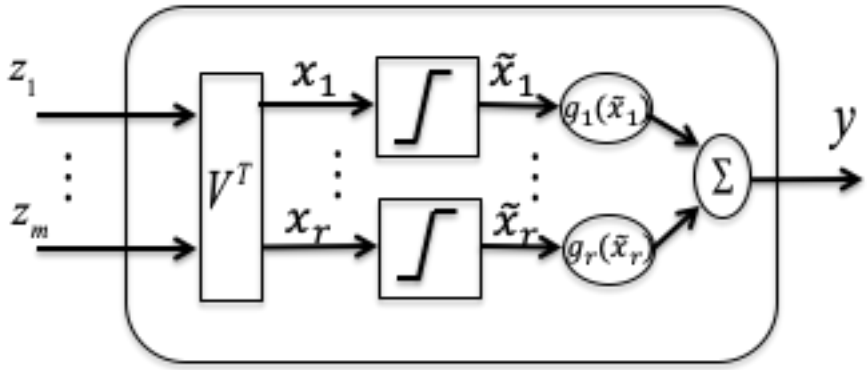

Figure 5.10: Decoupled MISO Polynomial into SISO Polynomial Branches with Saturation and a Transformation Matrix

Figure 5.11 shows what was expected to happen when the clamping of the inputs was applied after the simulation. When the clamping was applied online, instead of producing a smaller transient in the simulation errors it resulted in a limit cycle. Thus, this approach does not appear to be a suitable fix at this point and finding a stable method to address these extrapolation errors remains a topic for future research.

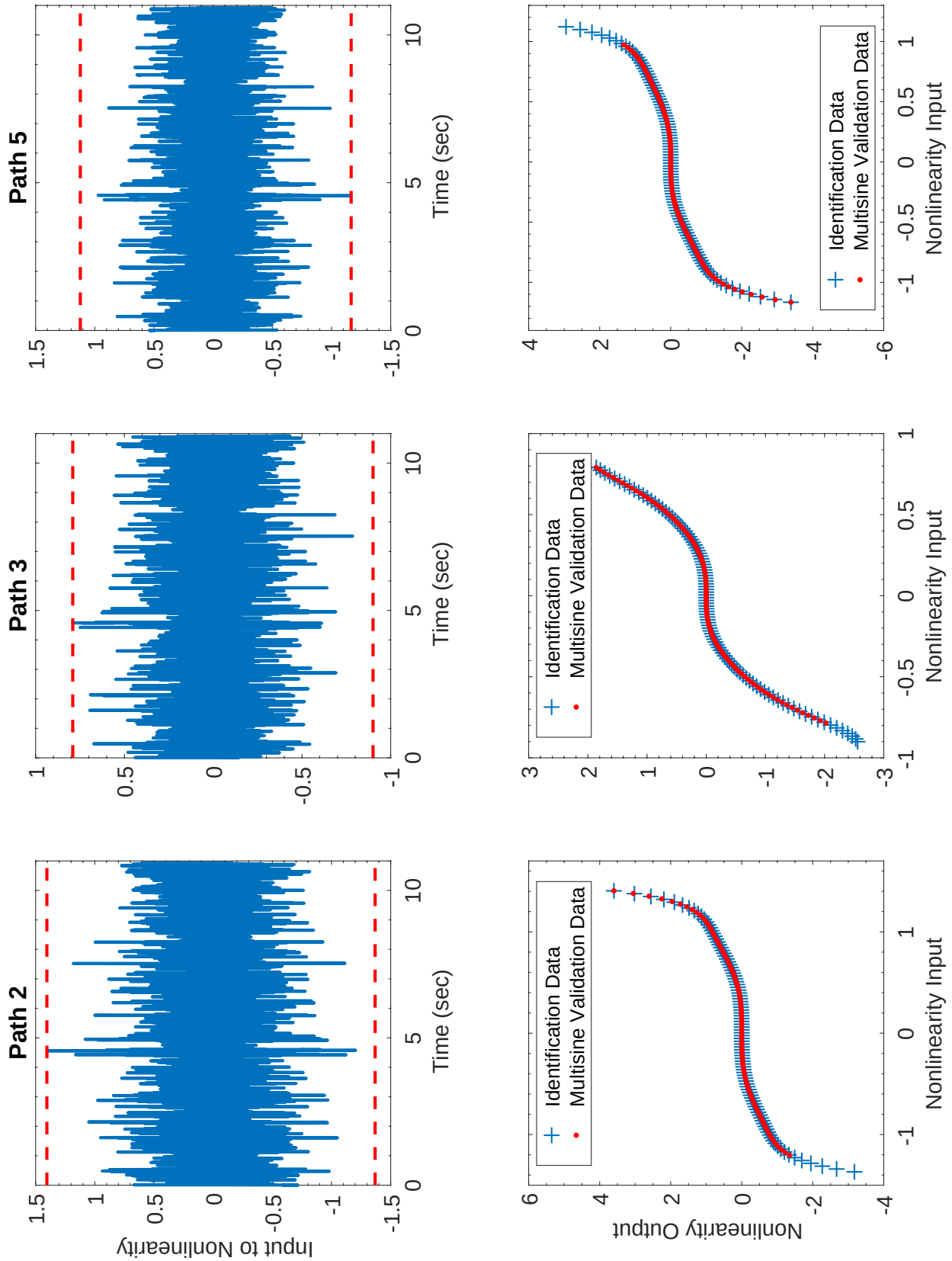

Figure 5.11: Upper: Input to the Nonlinearities and Lower: Nonlinearity of 2nd, 3rd and 5th Branch for the Bouc-Wen Using Simulation Error Model When the Saturation Was Applied to the Data in Figures 5.6 and 5.9 (Note the Saturation was Applied Offline, After the Simulation)

The swept-sine validation results are also puzzling, the simulation error minimization model not only outperformed the prediction error model in the simulation task but also in the prediction task as well, as indicated in Table 5.4.

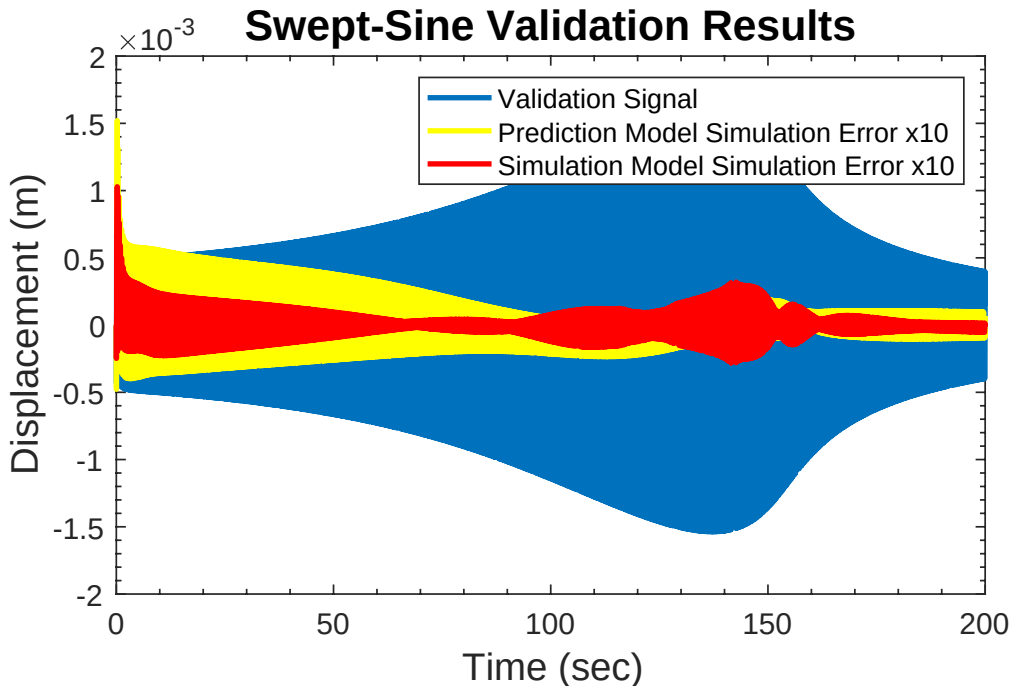

Figure 5.12: Simulation Errors on the Sweep-Sine Validation Data for the Bouc-Wen Using Prediction and Simulation Model. Errors Are Scaled Up 10x for Better Visibility

As it is shown in Fig. 5.12, the simulation error in the PEM model starts with a much bigger transient than the SEM model, but according to Fig. 5.13, around 140 seconds, when the resonance is happening the error is smaller in the PEM model. Since the fit persent for both models are pretty similar, so the PEM model makes up for the poor transient with better performance during the resonance, even if the overall result is still a little worse than SEM.

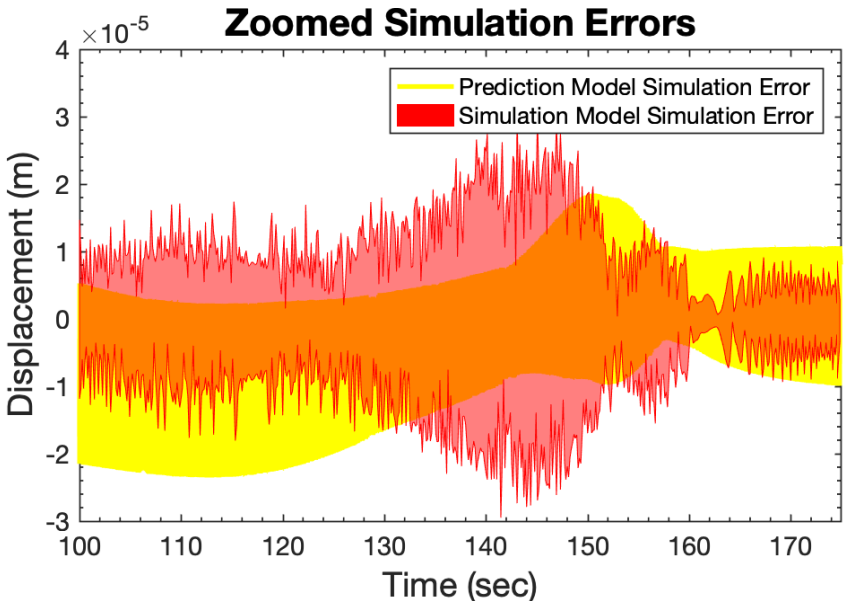

Figure 5.13: Zoomed Simulation Errors on the Sweep-Sine Validation Data for the Bouc-Wen Using Prediction and Simulation Model for $100 < t < 175$ Seconds

	Identification Method	
	Prediction Error Minimization	Simulation Error Minimization
Simulation Error(Fit%)	97.48%	98.71%
Prediction Error(Fit%)	99.47%	99.54%

Table 5.4: Swept-Sine Validation Results for the Models Fitted Using PEM and SEM for the Bouc-Wen Benchmark

According to Fig. 5.14, which shows the 1-step ahead prediction errors of the two models, the error of the PEM model has its peak at the beginning and decreases afterwards. But for the SEM model errors comparable to the initial transient, but lasting for several seconds, happen at about 140 seconds where the resonance happens. During the resonance, as illustrated in Fig. 5.15, the SEM model performs worse than the PEM model. Keep in mind that these errors have been magnified 10 times for better visualization.

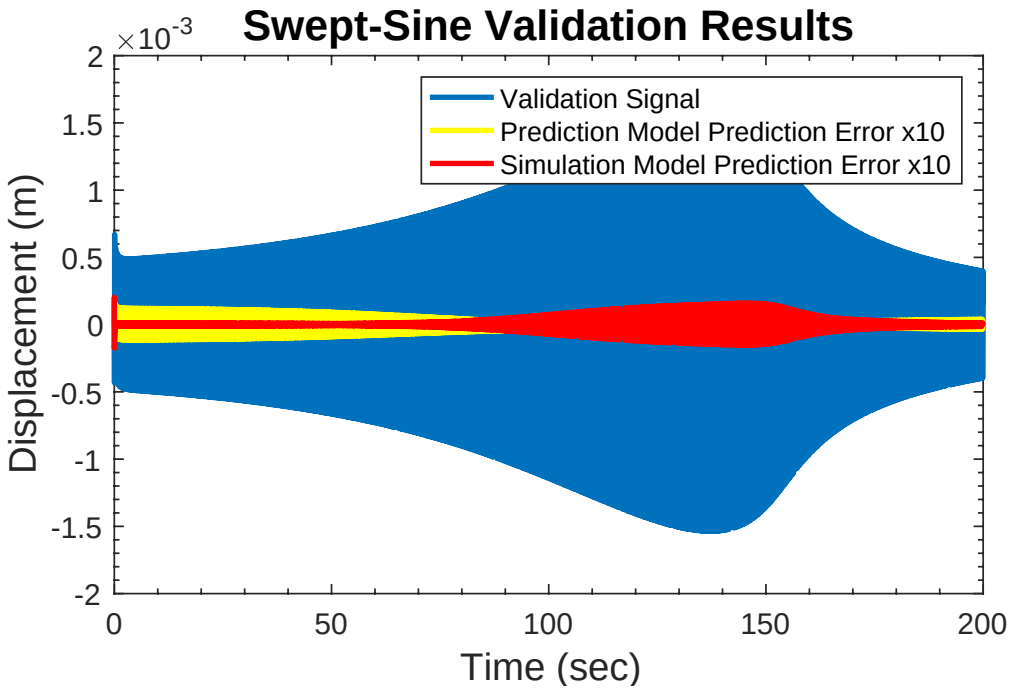

Figure 5.14: Prediction Errors on the Sweep-Sine Validation Data for the Bouc-Wen Using Prediction and Simulation Model. Errors Are Scaled Up 10x for Better Visibility

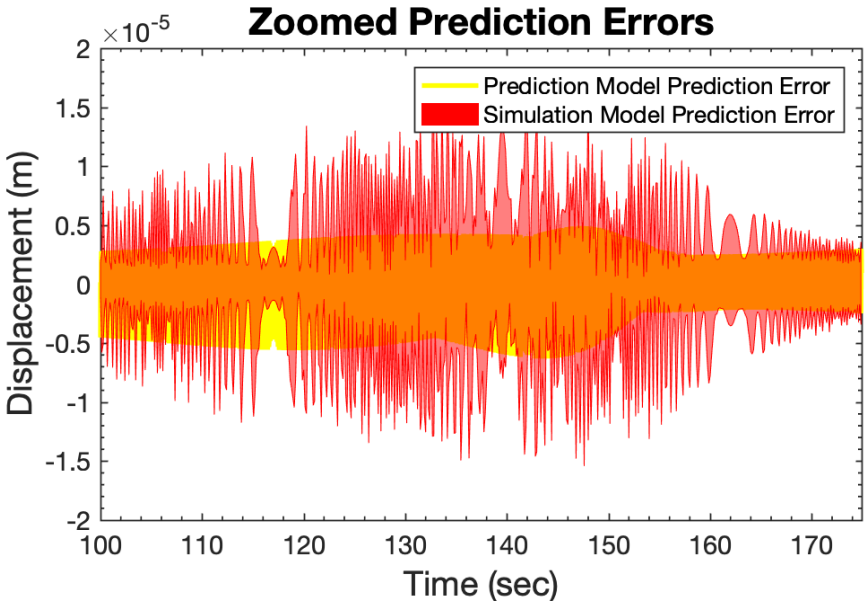

Figure 5.15: Zoomed Prediction Errors on the Sweep-Sine Validation Data for the Bouc-Wen Using Prediction and Simulation Model for $100 < t < 175$ Seconds

The results from the Silver-Box and the Bouc-Wen benchmarks are promising in regards to simulation error minimization model, but also demonstrate problems with model stability

and sensitivity. On the other hand, since the proposed decoupled model's nonlinearities are all single-input single-output polynomials, it is easy to detect the points where extrapolation is happening, which was one of the flaws of the coupled model. Extrapolation detection can be done on-line, therefore, the user will realize that a better identification set is needed and the model is not good enough for this data. Thus the output cannot be trusted in the occurrence of extrapolation. One of the main benefits of the decoupled polynomial NARX model is that it is easy to figure out what is happening inside the model, which is not the case in the coupled NARX model, which is a black-box model.

The most likely application of a simulation model would be in nonlinear model-based predictive control, which is an advanced method that is used to control a process while satisfying a set of constraints. The control input is chosen by repeatedly solving an optimization involving the tracking errors of the simulated plant output from the current time out to a so-called prediction horizon. In the case of these applications it is possible to add constraints – for example, no extrapolation on any path – to the optimization, so that the control input is always chosen such that there is no extrapolation. So, detecting extrapolation will be sufficient in these applications.

Also, the next step in this research path, could be finding a solution in order to overcome the sensitivity of the decoupled model to errors resulting from these extrapolations.

Chapter 6

Conclusion, Summary and Future Works

Since lots of systems around us, either natural or human-made, are behaving nonlinearly, it is important to have a simple nonlinear model to represent these systems. Recall the research question:

Can one propose a simpler model for dynamic nonlinear systems that is still very accurate? In particular, can one develop an identification method that is able to handle non-Volterra systems, whose linearizations contain moving poles, but without suffering from the curse of dimensionality, while it is capable of providing some insight into the model?

The answer is YES, as shown in Chapter 3, 4 and 5, the decoupled polynomial NARX model has the following advantages over the conventional polynomial NARX model:

- The NARX model, and hence the decoupled variant proposed here, can handle the class of non-Volterra systems whose linearizations have moving poles.
- It provides a simpler framework than the P-NARX model by reducing the number of parameters significantly. For example, for the Bouc-Wen benchmark which was a complex system, only 196 parameters exist whereas the full P-NARX model that

contains 638 parameters.

- The model simplification happened while the model accuracy is comparable to the full P-NARX model. For the Bouc-Wen benchmark the proposed decoupled model identified a model with 97.08% accuracy while the full P-NARX model has the prediction accuracy of 98.52% in the validation data set.

- unlike the full NARX model, the decoupled model is no longer a black-box model. Hence you can look at the nonlinearity in each branch separately. Therefore, it has the ability to detect extrapolation, and/or the dead-zone behaviour evident in the decoupled Bouc-Wen models.

6.1 Summary

In this research project, a decoupling algorithm for the polynomial NARX models is proposed. The decoupled polynomial NARX model framework replaces the multivariate polynomial with a bank of several single-input single-output univariate polynomials that are related to the inputs using a mixing matrix. Then to form the system‘s output, all of the outputs of these branches are summed together. The proposed model provides a much simpler model than the full NARX model by reducing the number of parameters that need be identified. Since now I am dealing with SISO univariate nonlinearities it is possible to have an intuition into the model which is more challenging with the full NARX model.

The decoupling approach was applied and tested using 2 different nonlinear benchmark problems, The Silver-Box system that is a physical system and provides measurement data and the Bouc-Wen hysteresis benchmark problem which is a more complex system than the Silver-Box but it is a simulated problem. Since hysteresis is a very common nonlinear behavior and also very challenging, it was a good selection to test the proposed approach.

Although the decoupling framework has many advantages, but its disadvantage is that it results in a non-convex optimization. Therefore, the optimization would be sensitive to the

initial solution. In this thesis, 3 different initialization approach for the optimization problem is proposed. And all 3 seem to produce promising results with their own advantages and disadvantages. It would be the user‘s choice to use which initialization technique based on the application, needs and available storage.

Even though using PEM to fit the polynomial NARX models is a very classical approach, it is not always possible to access the system's output, therefore generating 1-step ahead predictions is not always feasible. In this thesis a simulation error minimization approach for the decoupled polynomial NARX model is proposed as well. The SEM technique tries to fit a model using measurements of current and past input values and past model output values with the goal of producing an accurate model that is appropriate for applications requiring simulation instead of 1-step ahead prediction. Using the two benchmark problems, I showed that the simulation error minimization model generates promising results, but also revealed problems with model sensitivity and stability. At the same time, the decoupled model has the potential to show that what is happening inside the model.

6.2 Future Research Directions

Future work includes but is not limited to:

- Using non-polynomial basis function for the NARX model and adapt the decoupling algorithm for that framework. This optimization based approach can be used with any basis expansion model of the nonlinearity (since it will still be linear in the variables), provided that the basis elements are differentiable. For example, spline functions have been routinely used in nonlinear block-structured models such as the Wiener-Hammerstein model and could be used to represent the SISO nonlinearities in the branches.

- Expanding on the sensitivity of the decoupled P-NARX model in simulation mode that is due to the extrapolation in the identified model.

- Applying the decoupling algorithm to the data from real complex experiments. New benchmarks are being added to the website every year such as recently added wrist joint manipulation data set.

- Using the decoupled P-NARX model to generate a nonlinear initial estimate of a PNLSS model instead of using its BLA as an initialization as is the current practice [49, 100]. Since being able to initialize the PNLSS model with a nearly exact solution could be very important for models that are sensitive to instability. PNLSS models are known for their flexibility, ease of manipulation and also the ability to capture the severe nonlinear phenomenon.

- Developing methods to identify DP-NARX models from closed-loop data. The basic problem with the closed-loop data is that whenever the feedback controller is not identically zero, the input and the unmeasurable noise are correlated [101]. On the other hand, many industrial or biological processes are operating under feedback control, therefore it is important to be able to adjust the identification methods to deal with closed-loop experimental data [102].

Bibliography

[1] L. Lennart, "System identification: theory for the user," *PTR Prentice Hall, Upper Saddle River, NJ*, 1999.

[2] R. Pintelon and J. Schoukens, *System identification: a frequency domain approach*. John Wiley & Sons, 2012.

[3] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[4] P. Eykhoff, "System identification," *Wiley, New York, USA*, 1974.

[5] L. Ljung, "Perspectives on system identification," *Annual Reviews in Control*, vol. 34, no. 1, pp. 1–12, 2010.

[6] K. J. Åström and P. Eykhoff, "System identification—a survey," *Automatica*, vol. 7, no. 2, pp. 123–162, 1971.

[7] R. V. Jategaonkar, D. Fischenberg, and W. Gruenhagen, "Aerodynamic modeling and system identification from flight data-recent applications at dlr," *Journal of Aircraft*, vol. 41, no. 4, pp. 681–691, 2004.

[8] K. Soal, Y. Govers, J. Bienert, and A. Bekker, "System identification and tracking using a statistical model and a kalman filter," *Mechanical Systems and Signal Processing*, vol. 133, p. 106127, 2019.

[9] M. P. Vlaar, G. Birpoutsoukis, J. Lataire, M. Schoukens, A. C. Schouten, J. Schoukens, and F. C. van der Helm, "Modeling the nonlinear cortical response in eeg evoked by wrist joint manipulation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 1, pp. 205–215, 2017.

[10] K. Karami and D. T. Westwick, "Identification of stretch reflex dynamics from closed-loop data," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 1397–1402, 2015.

[11] D. T. Westwick and R. E. Kearney, *Identification of nonlinear physiological systems*, vol. 7. John Wiley & Sons, 2003.

[12] R. Mehra, "Optimal inputs for linear system identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 3, pp. 192–200, 1974.

[13] J. Bruls, C. Chou, B. Haverkamp, and M. Verhaegen, "Linear and non-linear system identification using separable least-squares," *European Journal of Control*, vol. 5, no. 1, pp. 116–128, 1999.

[14] L. Ljung, "System identification," in *Signal Analysis and Prediction*, pp. 163–173, Springer, 1998.

[15] T. Soderstrom and P. Stoica, "System identification," *Hemel Hempstead, UK: Prentice-Hall*, 1989.

[16] G. A. Seber and A. J. Lee, *Linear regression analysis*, vol. 329. John Wiley & Sons, 2012.

[17] Wikipedia contributors, "Ordinary least squares — Wikipedia, the free encyclopedia," 2019. [Online; accessed 23-December-2019].

[18] A. S. Goldberger, "Econometric theory. new york: Johnwiley & sons," *Inc. GoldbergerEconometric Theory1964*, 1964.

[19] B. Pasik-Duncan, *Stochastic Theory and Control: Proceedings of a Workshop Held in Lawrence, Kansas*, vol. 280. Springer Science & Business Media, 2002.

[20] J. Schoukens, M. Vaes, and R. Pintelon, "Linear system identification in a nonlinear setting: Nonparametric analysis of the nonlinear distortions and their impact on the best linear approximation," *IEEE Control Systems*, vol. 36, no. 3, pp. 38–69, 2016.

[21] O. Nelles, "Nonlinear system identification," *Springser, Berlin, Germany*, 2001.

[22] J. Schoukens, A. Marconato, R. Pintelon, Y. Rolain, M. Schoukens, K. Tiels, L. Vanbeylen, G. Vandersteen, and A. Van Mulders, "System identification in a real world," in *2014 IEEE 13th International Workshop on Advanced Motion Control (AMC)*, pp. 1–9, IEEE, 2014.

[23] L. Ljung and A. Vicino, "Guest editorial: special issue on system identification," *IEEE Transactions on Automatic Control*, vol. 50, no. 10, pp. 1473–1473, 2005.

[24] J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented roadmap," *IEEE Control Systems magazine (in press)*, 2019.

[25] G. Birpoutsoukis, *Volterra Series Estimation in the Presence of Prior Knowledge*. PhD thesis, VRIJE UNIVERSITEIT BRUSSEL, 2018.

[26] F. Giri and E.-W. Bai, *Block-oriented nonlinear system identification*, vol. 1. Springer, 2010.

[27] S. Boyd and L. Chua, "Fading memory and the problem of approximating nonlinear operators with volterra series," *IEEE Transactions on circuits and systems*, vol. 32, no. 11, pp. 1150–1161, 1985.

[28] J. Amorocho and A. Brandstetter, "Determination of nonlinear functional response functions in rainfall-runoff processes," *Water Resources Research*, vol. 7, no. 5, pp. 1087–1101, 1971.

[29] V. Z. Marmarelis, "Identification of nonlinear biological systems using laguerre expansions of kernels," *Annals of biomedical engineering*, vol. 21, no. 6, pp. 573–589, 1993.

[30] H. M. Abbas and M. M. Bayoumi, "Volterra-system identification using adaptive real-coded genetic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 36, no. 4, pp. 671–684, 2006.

[31] G. Birpoutsoukis and J. Schoukens, "Nonparametric volterra kernel estimation using regularization," in *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, pp. 222–227, IEEE, 2015.

[32] G. Birpoutsoukis, A. Marconato, J. Lataire, and J. Schoukens, "Regularized nonparametric volterra kernel estimation," *Automatica*, vol. 82, pp. 324–327, 2017.

[33] J. G. Stoddard, *Innovative Data-Driven Modeling and Control for Nonlinear Systems using Volterra Series*. PhD thesis, University of Newcastle, 2019.

[34] M. J. Korenberg, "Parallel cascade identification and kernel estimation for nonlinear systems," *Annals of biomedical engineering*, vol. 19, no. 4, pp. 429–455, 1991.

[35] G. Birpoutsoukis, P. Z. Csurcsia, and J. Schoukens, "Efficient multidimensional regularization for volterra series estimation," *Mechanical Systems and Signal Processing*, vol. 104, pp. 896–914, 2018.

[36] I. Hunter and M. Korenberg, "The identification of nonlinear biological systems: Wiener and hammerstein cascade models," *Biological cybernetics*, vol. 55, no. 2-3, pp. 135–144, 1986.

[37] W. J. Rugh, *Nonlinear system theory*. Johns Hopkins University Press Baltimore, 1981.

[38] R. E. Kearney, R. B. Stein, and L. Parameswaran, "Identification of intrinsic and reflex

contributions to human ankle stiffness dynamics," *IEEE Transactions on Biomedical Engineering*, vol. 44, no. 6, pp. 493–504, 1997.

[39] G. Palm, "On representation and approximation of nonlinear systems," *Biological Cybernetics*, vol. 31, no. 2, pp. 119–124, 1978.

[40] G. Palm, "On representation and approximation of nonlinear systems," *Biological Cybernetics*, vol. 34, no. 1, pp. 49–52, 1979.

[41] G. Golub and V. Pereyra, "Separable nonlinear least squares: the variable projection method and its applications," *Inverse problems*, vol. 19, no. 2, p. R1, 2003.

[42] M. Schoukens, *Identification of Parallel Block-Oriented Models starting from the Best Linear Approximation*. PhD thesis, VRIJE UNIVERSITEIT BRUSSEL, 2015.

[43] M. Enqvist, "Linear models of nonlinear systems," 2005.

[44] M. Enqvist and L. Ljung, "Linear approximations of nonlinear fir systems for separable input processes," *Automatica*, vol. 41, no. 3, pp. 459–473, 2005.

[45] J. Schoukens, J. Lataire, R. Pintelon, G. Vandersteen, and T. Dobrowiecki, "Robustness issues of the best linear approximation of a nonlinear system," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 5, pp. 1737–1745, 2009.

[46] J. J. Bussgang, "Crosscorrelation functions of amplitude-distorted gaussian signals," 1952.

[47] T. B. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," *Automatica*, vol. 47, no. 1, pp. 39–49, 2011.

[48] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, and R. Pintelon, "Identification of nonlinear systems using polynomial nonlinear state space models," *Automatica*, vol. 46, no. 4, pp. 647–656, 2010.

[49] A. F. Esfahani, P. Dreesen, K. Tiels, J.-P. Noël, and J. Schoukens, "Polynomial state-space model decoupling for the identification of hysteretic systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 458–463, 2017.

[50] A. F. Esfahani, P. Dreesen, K. Tiels, J.-P. Noël, and J. Schoukens, "Parameter reduction in nonlinear state-space identification of hysteresis," *Mechanical Systems and Signal Processing*, vol. 104, pp. 884–895, 2018.

[51] P. Dreesen, M. Ishteva, and J. Schoukens, "Decoupling multivariate polynomials using first-order information and tensor decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 2, pp. 864–879, 2015.

[52] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[53] R. A. Harshman *et al.*, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[54] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[55] P. Dreesen, M. Ishteva, and J. Schoukens, "Recovering wiener-hammerstein nonlinear state-space models using linear algebra," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 951–956, 2015.

[56] J. Stoev, J. Ertveldt, T. Oomen, and J. Schoukens, "Tensor methods for mimo decoupling and control design using frequency response functions," *Mechatronics*, vol. 45, pp. 71–81, 2017.

[57] I. Leontaritis and S. A. Billings, "Input-output parametric models for non-linear sys-

tems part i: deterministic non-linear systems," *International journal of control*, vol. 41, no. 2, pp. 303–328, 1985.

[58] J. M. Caswell, "A nonlinear autoregressive approach to statistical prediction of disturbance storm time geomagnetic fluctuations using solar data," *Journal of Signal and Information Processing*, vol. 5, no. 02, p. 42, 2014.

[59] R. W. Chan, J. K. Yuen, E. W. Lee, and M. Arashpour, "Application of nonlinear-autoregressive-exogenous model to predict the hysteretic behaviour of passive control systems," *Engineering Structures*, vol. 85, pp. 1–10, 2015.

[60] L. Ruiz, M. Cuéllar, M. Calvo-Flores, and M. Jiménez, "An application of non-linear autoregressive neural networks to predict energy consumption in public buildings," *Energies*, vol. 9, no. 9, p. 684, 2016.

[61] Y. Zhao, S. A. Billings, H. Wei, F. He, and P. G. Sarrigiannis, "A new narx-based granger linear and nonlinear casual influence detection method with applications to eeg data," *Journal of neuroscience methods*, vol. 212, no. 1, pp. 79–86, 2013.

[62] G. Alcan, M. Unel, V. Aran, M. Yilmaz, C. Gurel, and K. Koprubasi, "Predicting nox emissions in diesel engines via sigmoid narx models using a new experiment design for combustion identification," *Measurement*, vol. 137, pp. 71–81, 2019.

[63] A. Shmilovici and J. Aguilar-Martin, "Adaptive matching pursuit of narx models with spline basis functions," *International journal of systems science*, vol. 30, no. 8, pp. 879–888, 1999.

[64] J. P. Jones and S. Billings, "Recursive algorithm for computing the frequency response of a class of non-linear difference equation models," *International Journal of Control*, vol. 50, no. 5, pp. 1925–1940, 1989.

[65] L. Piroddi and W. Spinelli, "An identification algorithm for polynomial narx models based on simulation error minimization," *International Journal of Control*, vol. 76, no. 17, pp. 1767–1781, 2003.

[66] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky, "Nonlinear black-box modeling in system identification: a unified overview," *Automatica*, vol. 31, no. 12, pp. 1691–1724, 1995.

[67] S. Billings and K. Tsang, "Spectral analysis for non-linear systems, part ii: Interpretation of non-linear frequency response functions," *Mechanical Systems and Signal Processing*, vol. 3, no. 4, pp. 341–359, 1989.

[68] S. Billings and J. Peyton Jones, "Mapping non-linear integro-differential equations into the frequency domain," *International Journal of Control*, vol. 52, no. 4, pp. 863–879, 1990.

[69] D. T. Westwick, G. Hollander, K. Karami, and J. Schoukens, "Using decoupling methods to reduce polynomial narx models," *IFAC-PapersOnLine*, vol. 51, no. 15, pp. 796–801, 2018.

[70] A. Weinstein *et al.*, "R. courant and d. hilbert, methods of mathematical physics," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 578–579, 1954.

[71] X. J. Jing, Z. Q. Lang, and S. A. Billings, "New bound characteristics of narx model in the frequency domain," *International Journal of Control*, vol. 80, no. 1, pp. 140–149, 2007.

[72] X. J. Jing, Z. Q. Lang, and S. A. Billings, "Mapping from parametric characteristics to generalized frequency response functions of non-linear systems," *International Journal of Control*, vol. 81, no. 7, pp. 1071–1088, 2008.

[73] T. Wigren and J. Schoukens, "Three free data sets for development and benchmarking in nonlinear system identification," in *Proc. 2013 Eur. Control Conf.(ECC2013)*, pp. 17–19, 2013.

[74] J. Noël and M. Schoukens, "Hysteretic benchmark with a dynamic nonlinearity," in *Workshop on nonlinear system identification benchmarks*, pp. 7–14, 2016.

[75] M. Schoukens and J. P. Noël, "Three benchmarks addressing open challenges in nonlinear system identification," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 446–451, 2017.

[76] L. Sragner, J. Schoukens, and G. Horváth, "Modelling of a slightly nonlinear system: A neural network approach," *IFAC Proceedings Volumes*, vol. 37, no. 13, pp. 387–392, 2004.

[77] V. Verdult, "Identification of local linear state-space models: the silver-box case study," *IFAC Proceedings Volumes*, vol. 37, no. 13, pp. 393–398, 2004.

[78] R. Bouc, "Forced vibrations of mechanical systems with hysteresis," in *Proc. of the Fourth Conference on Nonlinear Oscillations, Prague, 1967*, 1967.

[79] Y. Wen *et al.*, "Method for random vibration of hysteretic systems," *Journal of the engineering mechanics division*, vol. 102, no. 2, pp. 249–263, 1976.

[80] M. Ismail, F. Ikhouane, and J. Rodellar, "The hysteresis bouc-wen model, a survey," *Archives of Computational Methods in Engineering*, vol. 16, no. 2, pp. 161–188, 2009.

[81] F. Ikhouane and J. Rodellar, *Systems with hysteresis: analysis, identification and control using the Bouc-Wen model*. John Wiley & Sons, 2007.

[82] T. J. Mueller, "The influence of laminar separation and transition on low reynolds number airfoil hysteresis," *Journal of Aircraft*, vol. 22, no. 9, pp. 763–770, 1985.

[83] D. Angeli, J. E. Ferrell, and E. D. Sontag, "Detection of multistability, bifurcations, and hysteresis in a large class of biological positive-feedback systems," *Proceedings of the National Academy of Sciences*, vol. 101, no. 7, pp. 1822–1827, 2004.

[84] B. E. Beisner, D. T. Haydon, and K. Cuddington, "Alternative stable states in ecology," *Frontiers in Ecology and the Environment*, vol. 1, no. 7, pp. 376–382, 2003.

[85] V. S. Ramachandran and S. M. Anstis, "Perceptual organization in multistable apparent motion," *Perception*, vol. 14, no. 2, pp. 135–143, 1985.

[86] K. Karami, D. Westwick, and J. Schoukens, "Applying polynomial decoupling methods to the polynomial narx model." *Mechanical Systems and Signal Processing*, Accepted, 2020.

[87] K. Karami and D. Westwick, "Initialization approach for decoupling polynomial narx model using tensor decomposition." *21st IFAC World Congress*, Accepted, 2020.

[88] J. B. Kruskal, "Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear algebra and its applications*, vol. 18, no. 2, pp. 95–138, 1977.

[89] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab user guide," *Available on: http://www. tensorlab. net*, 2016.

[90] G. Hollander, *Multivariate Polynomial Decoupling in Nonlinear System Identification*. PhD thesis, Vrije Universiteit Brussel, 2017.

[91] G. H. Golub and C. F. Van Loan, "Matrix computations, 3rd edition," *Johns Hopkins University Press, Baltimore, MD*, 1996.

[92] P. Comon and B. Mourrain, "Decomposition of quantics in sums of powers of linear forms," *Signal Processing*, vol. 53, no. 2-3, pp. 93–107, 1996.

[93] L. Oeding and G. Ottaviani, "Eigenvectors of tensors and algorithms for waring decomposition," *Journal of Symbolic Computation*, vol. 54, pp. 9–35, 2013.

[94] R. Fröberg, G. Ottaviani, and B. Shapiro, "On the waring problem for polynomial rings," *Proceedings of the National Academy of Sciences*, vol. 109, no. 15, pp. 5600–5602, 2012.

[95] J. Brachat, P. Comon, B. Mourrain, and E. Tsigaridas, "Symmetric tensor decomposition," *Linear Algebra and its Applications*, vol. 433, no. 11-12, pp. 1851–1872, 2010.

[96] M. Schoukens and Y. Rolain, "Cross-term elimination in parallel wiener systems using a linear input transformation," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 3, pp. 845–847, 2012.

[97] K. Karami, D. Westwick, and J. Schoukens, "Identification of decoupled polynomial narx model using simulation error minimization," in *2019 American Control Conference (ACC)*, pp. 4362–4367, IEEE, 2019.

[98] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[99] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[100] J. Decuyper, P. Dreesen, J. Schoukens, M. C. Runacres, and K. Tiels, "Decoupling multivariate polynomials for nonlinear state-space models," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 745–750, 2019.

[101] J. C. Forman, S. Bashash, J. L. Stein, and H. K. Fathy, "Reduction of an electrochemistry-based Li-ion battery model via quasi-linearization and Padé approximation," *Journal of The Electrochemical Society*, vol. 158, no. 2, pp. A93–A101, 2011.

[102] P. Van den Hof, "Closed-loop issues in system identification," *Annual reviews in control*, vol. 22, pp. 173–186, 1998.