CONTROL-RELEVANT SYSTEM IDENTIFICATION USING NONLINEAR VOLTERRA AND VOLTERRA-LAGUERRE MODELS

by

Abhishek S. Soni

B.S. Chemical Engineering, University Institute of Chemical Technology, Mumbai, India, 2000

M.S. Chemical Engineering, University of Pittsburgh, 2002

Submitted to the Graduate Faculty of
the School of Engineering in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2006

UNIVERSITY OF PITTSBURGH

SCHOOL OF ENGINEERING

This dissertation was presented

by

Abhishek S. Soni

It was defended on

March 24th 2006

and approved by

Prof. R. S. Parker, Assistant Professor, Department of Chemical Engineering

Prof. J. J. McCarthy, Associate Professor, Department of Chemical Engineering

Prof. J. K. Johnson, Department of Chemical Engineering

Prof. M. A. Simaan, Department of Electrical Engineering

Dissertation Director: Prof. R. S. Parker, Assistant Professor, Department of Chemical Engineering

CONTROL-RELEVANT SYSTEM IDENTIFICATION USING NONLINEAR VOLTERRA AND VOLTERRA-LAGUERRE MODELS

Abhishek S. Soni, PhD

University of Pittsburgh, 2006

One of the key impediments to the wide-spread use of nonlinear control in industry is the availability of suitable nonlinear models. Empirical models, which are obtained from only the process input-output data, present a convenient alternative to the more involved fundamental models. An important advantage of the empirical models is that their structure can be chosen so as to facilitate the controller design problem. Many of the widely used empirical model structures are linear, and in some cases this basic model formulation may not be able to adequately capture the nonlinear process dynamics. One of the commonly used nonlinear dynamic empirical model structures is the Volterra model, and this work develops a systematic approach to the identification of third-order Volterra and Volterra-Laguerre models from process input-output data.

First, plant-friendly input sequences are designed that exploit the Volterra model structure and use the prediction error variance (PEV) expression as a metric of model fidelity. Second, explicit estimator equations are derived for the linear, nonlinear diagonal, and higher-order sub-diagonal kernels using the tailored input sequences. Improvements in the sequence design are also presented which lead to a significant reduction in the amount of data required for identification. Finally, the third-order off-diagonal kernels are estimated using a cross-correlation approach. As an application of this technique, an isothermal polymerization reactor case study is considered.

In order to overcome the noise sensitivity and highly parameterized nature of Volterra models, they are projected onto an orthonormal Laguerre basis. Two important variables

that need to be selected for the projection are the Laguerre pole and the number of Laguerre filters. The Akaike Information Criterion (AIC) is used as a criterion to determine projected model quality. AIC includes contributions from both model size and model quality, with the latter characterized by the sum-squared error between the Volterra and the Volterra-Laguerre model outputs. Reduced Volterra-Laguerre models were also identified, and the control-relevance of identified Volterra-Laguerre models was evaluated in closed-loop using the model predictive control framework. Thus, this work presents a complete treatment of the problem of identifying nonlinear control-relevant Volterra and Volterra-Laguerre models from input-output data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

With ever increasing energy prices and the need to remain competitive in a global marketplace in the face of strict environmental regulations, the chemical industry needs to operate in an efficient manner with respect to both raw material and energy usage. This goal can be achieved by either improved process technology or plant retrofitting, by the implementation of advanced control technologies for regulation and control. Compared to new process equipment, which is both capital and time intensive, using an advanced control system in existing plants offers a cost-effective and more immediate alternative. In addition, the combination of new process technology and advanced process control can lead to safe, energy-efficient and economical plants [1]. As an example, the reduction in emissions and fuel consumption in automobiles has been brought about by a combination of improved process technology (catalytic converters) and advanced process control (microcomputer based engine control) [2]. One of the basic tenets of model-based control is that theoretically achievable controller performance is directly related to model quality [3]. Thus a good process model is a key requirement for any successful controller. In fact, it has been reported that in some cases, process model development accounts for about 75% of the time required to implement an advanced control strategy [4], thereby highlighting the critical nature that model development plays in advanced controller design.

The nature of the chemical industry itself, *i.e.,* economics of operation and the unit operations themselves, imposes additional requirements on process models. The need for improved product quality while maintaining a safe and economical operation requires that plants be operated over a broad range about the nominal operating point. This leads to plant operation close to constraints and excites nonlinearities in the system behavior. As an example, high purity distillation columns exhibit asymmetric output changes (overhead

product quality) to symmetric input changes (reflux ratio) [5]. In addition, reacting systems often display nonlinearities arising due to the reaction mechanisms or due to the non-isothermal nature of the rate constants. Thus nonlinearity is an integral part of chemical process operation and must be accounted for while developing process models. There are various approaches for developing nonlinear models. Models could be derived based on the process physics using ordinary and/or partial differential equations, or by using process input-output data and a suitable empirical model structure. Another approach is to develop mixed models which are a combination of the aforementioned approaches.

Fundamental models, also called first-principles models are based on the underlying physics of the system. These models are developed by applying mass and energy balances over the components or states and may also include a description of the fluid flow and transport processes that occur in the system. A number of fundamental models have been developed to model processes ranging from a single cell [6, 7, 8], to a complex two phase pulp digester [9]. Fundamental models offer several potential benefits. Since they include details about the physics of the system, they can better represent the nonlinear behavior and system dynamics; this allows the model to be used with confidence, beyond the operating range in which the model was constructed. Another benefit of utilizing the first-principles approach is that the states are generally physical variables that can be measured, such as temperature or concentrations of the system components. However, these models are time consuming to develop and they often have a high state dimension [10, 11]. The involved mathematical description may result in a model with a large number of equations and many parameters that need to be estimated. In many cases, all of these parameters cannot be obtained experimentally. In some cases, regression analysis is used or adjustable parameters are employed to adequately represent the system behavior. On account of the complexity included in these models they often take person-years to develop.

In many cases the actual system may be complex and the underlying phenomena are not well enough understood to develop a first-principles model. Empirical models are useful in this scenario. In empirical modeling, a model structure is first selected and the model identification problem involves determining the model parameters that best fit the input-output data. Among the widely used empirical models are the auto-regressive with exogenous

inputs (ARX) model [12], Artificial Neural Networks (ANN) [13],[14], and polynomial models such as the Volterra-Laguerre models [15],[16],[17]. The parameters describing the model are then identified by a regression analysis of the process data [18]. As an example of the empirical modeling framework, Parker *et al.* utilized a second order Volterra series model to capture the dynamic behavior exhibited by continuous cultures of *Klebsiella pneumoniae* [15]. Although practical insight into the problem may be lost, as physical variables of the system such as temperature and concentration may not be explicitly represented in the empirical model, the time requirement to obtain these models is often significantly reduced [19].

Mixed models are developed, as the name suggests, by combining the fundamental and empirical models, thus encompassing the benefits of both. As an example, Henson used a fundamental model to capture the key process dynamics and then appended an empirical model to capture the modeling error between the fundamental model and the actual plant [20].

The quality and level of complexity included in the process model is intricately related with the subsequent controller design and implementation. In the commonly used model predictive control (MPC) algorithm, the control input implemented in the plant is the solution to an optimization problem that minimizes the deviation of the output from a given reference. Generally, there are time constraints within which the optimization problem must be solved so that the input can be implemented in the plant. Complexities in the process model may cause problems in the solution of the optimization problem, such as infeasibilities and lack of algorithm convergence in real-time. For process control applications it may not be necessary for a model to describe the complete set of dynamic process behavior. Such a model that is built with an emphasis on controller design is called a control-relevant model. As an example, Skogestad and Morari consider a linear two time constant model for a distillation column where the two time constants that capture the dynamic behavior are motivated by the internal and external flows in the column. The main focus of their work was to develop a model, that included the factors that were most important for feedback control, rather than an accurate physical model for feed-back control [21]. Similarly, Balasubramhanya and Doyle III [22] used traveling wave phenomena to develop a low-order nonlinear model

that captured the nonlinear dynamic behavior of a high purity distillation column for use in model-based control strategies. Thus, the main idea behind control-relevant modeling is that in order to facilitate the subsequent controller design problem, it is advisable to employ the least complex model that adequately captures the relevant process dynamics [19].

The level of complexity included in fundamental models may preclude them from being used in control design. In contrast, the simplicity of empirical models offer a lot of advantages. They are comparatively easier and less time-consuming to develop as compared to fundamental models. They can be control-relevant, as the model structure in empirical models can be chosen to facilitate subsequent controller design. As a testament to their appeal, a vast majority of industrial model predictive controllers (MPC) employ empirical dynamical models [23]. Thus, the focus of this work is the development of a specific class of empirical nonlinear dynamic models from input-output data. Before moving on to specifics, the next section offers a broad overview of empirical model identification.

1.1 EMPIRICAL SYSTEM IDENTIFICATION

A schematic of empirical model identification is given in Figure 1. The main goal is to develop an empirical model for the unknown physical process. In its most basic form, the problem of system identification involves determining the parameters, θ, of the empirical model such that the predicted response from the model $\hat{y}$ closely approximates the response of the physical process y, for the same inputs u. In a historical context, the term "system identification" was first defined by Lotfi Zadeh: "Identification is the determination, on the basis of input and output, of a system within a specified class of systems, to which the system under test is equivalent" [24]. With this definition in mind, system identification in practice involves the following steps [25]:

1. Selection of a model structure
2. Given a model structure, design of the input sequence, $u(k)$
3. Given $u(k)$, generation of the system response $y(k)$
4. From the input-output dataset, estimation of the model parameters

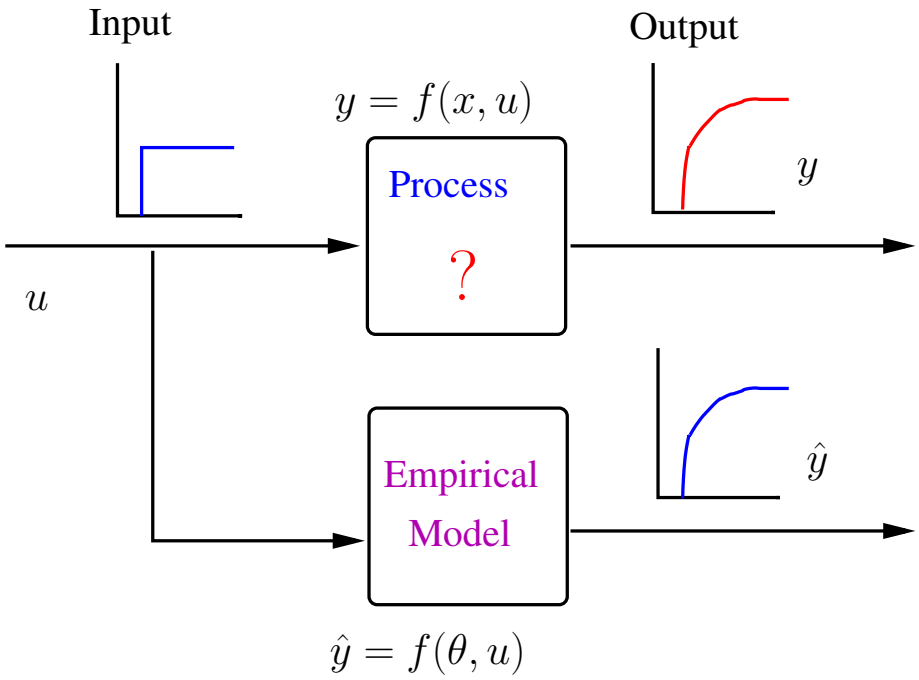

Figure 1: Schematic of the system identification problem

5. Assessment of identified model quality based on the estimated model parameters
6. Iteration and model refinement as necessary

1.1.1 Steps in Empirical Model Identification

A schematic of the various steps in empirical model identification is shown in Figure 2, and these are explained in the subsequent sections.

1.1.1.1 Model Structure Selection One of the most difficult steps in constructing an empirical model is the selection of a suitable model structure. The selection of the model structure could be carried out using two distinct philosophies. In the first, the exploratory philosophy, structural process data such as the time constant, magnitude and sign of the gain, and steady-state input-output map are determined. This information can be obtained by subjecting the process to specific screening inputs such as sinusoids, steps of various magnitude, *etc.* [26]. This available process data is then used to select a model structure that exhibits a qualitative behavior similar to that exhibited by the actual process. The other philosophy is the confirmatory philosophy whereby a model structure is selected and the process data is used to accept or reject its appropriateness [26]. Closely related to these ideas is the concept of parametric, non-parametric, or semi-parametric modeling [27]. In parametric modeling, the structure of the model is fixed and the identification problem is reduced to determining the model parameters that best-fit the process input-output dataset. In non-parametric modeling, a model structure itself, *i.e.* the functional form of the structure, is sought from from the available data [18]. Finally, in semi-parametric modeling, only part of the model structure is completely specified. In general, selection of the model structure is greatly facilitated by an adequate knowledge of *a priori* process information.

1.1.1.2 Input Sequence Design The input used to generate the $u - y$ dataset is user-defined, and it can be tailored to facilitate estimation. In addition, since the input is implemented in the plant, it should satisfy certain characteristics. First, the input signal

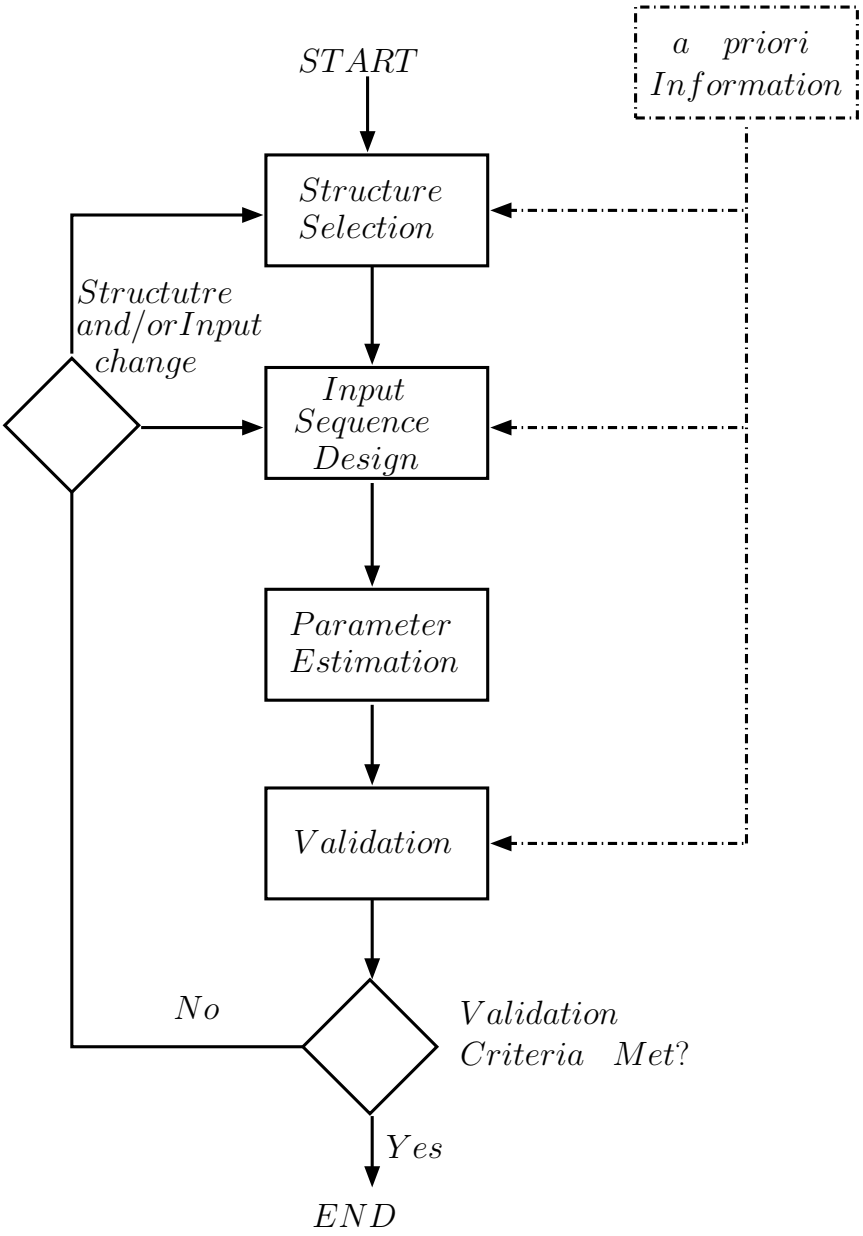

Figure 2: Schematic of the steps involved in system identification

must have sufficient energy to excite the full range of nonlinear process dynamics. For linear systems, the random binary signal or the deterministic pseudo random binary signal (PRBS) have been commonly used. However, these signals cannot excite certain nonlinearities (*eg.,* even-ordered symmetric behavior), so that more input levels in the sequence are necessary. As a special case, Nowak and van Veen have shown that identification a Volterra model of order N requires the input sequence to have at least $N + 1$ levels [28]. Second, the input should be plant-friendly. Since this input signal is implemented by an actuator in the plant, such as a control valve, the input sequence should not have frequent transitions which cause actuator wear. In addition, the magnitude of the input sequence must not be large so that valve saturation is avoided. At the same time the magnitude of the input should be high enough to ensure that the output response lies outside the noise-band for process operation. In addition, a sequence with a length as short as possible is desirable so that system identification does not interfere with normal plant operation [29].

However, these practical requirements are often in conflict with theoretical identification results. It has been shown that the coefficient error variance can be reduced by using a sequence as long as possible. In fact, asymptotic convergence is achieved as the sequence length approaches infinity [18]. In addition, the variance also decreases when the input spectrum, *i.e.,* input magnitude, is as high as possible so that the operation is under a high input signal to noise ratio [18]. An example of such a sequence is the Gaussian White Noise (GWN) sequence, which is completely random and uncorrelated with itself, *i.e.,* this sequence has the maximum possible input energy.

Various metrics have been proposed to characterize the plant-friendliness of input sequences [29, 30, 31]. Braun *et al.* [29] employed multi-sine signals in the frequency domain with a goal of minimizing the crest-factor of the input signal. The crest-factor is a ratio of the $\mathbf{L}_\infty$ norm to the $\mathbf{L}_2$ norm of the input signal. A low crest factor signifies that most of the input values are distributed around its extreme ends. Minimization of the crest factor improves the signal to noise ratio of the output, thereby making it plant-friendly. The metric used to account for plant-friendliness in this work is the friendliness factor, F_f, given as [30]:

$$F_f = 100\left(1 - \frac{n_t}{N_L - 1}\right) \tag{1.1}$$

Here, F_f is the friendliness factor, N_L is the total sequence length, and n_t is the total number of level transitions. Any sequence that remains at the same level is 100% plant friendly (since there are no transitions). However, it would not provide any meaningful data for identifying the system dynamics or nonlinearities. A sequence where n_t equals $N_L - 1$, *e.g.,* a sequence that changes at every time-step, is 0% plant friendly as the frequent transitions would cause actuator wear.

1.1.1.3 Parameter Estimation Both the selection of the model structure and the design of the input sequence can affect the complexity of the parameter estimation problem. As an example, Hammerstein models, which consist of a memoryless nonlinear function followed by a linear dynamic element, are nonlinear in the parameters and require an iterative procedure, the Narendra-Gallman algorithm, to estimate the model parameters [26, 32]. On the other hand, Wiener models reverse the order of the two blocks, *i.e.,* the linear dynamic element precedes the memoryless nonlinearity, so that the estimation problem is comparatively easier and non-iterative in nature. In general, parameter estimation for models that are linear in the parameters, such as the Finite Impulse Response (FIR) model, is easier, and the parameters can be obtained by a simple least squares regression. However, for models that are a nonlinear function of the parameters, such as the ARMAX, Box-Jenkins, and Output Error models, a slightly more involved nonlinear iterative solution to the parameter estimation problem is required [18, 26].

1.1.1.4 Model Validation Once a model is obtained that best fits the input-output dataset, its validity needs to be evaluated. Validation of the identified model is carried out by assessing the performance of the model for a dataset other than that used for identification, and such a validation exercise is termed as cross-validation. However, a simple comparison of the outputs from the actual process and the identified model may not be enough to ascertain the validity or invalidity of the model. Consequently, the model's response to an impulse or step input is used to compare with *a priori* process information such as the time constant, or the gain sign and magnitude. When the identified model yields estimates that deviate significantly from known process behavior, the existing model needs to be

revised, *i.e.,* a new model structure, or a more informative input-output dataset should be used. Residual analysis is another metric that can be used to assess the performance of the identified model. When the auto-correlation of the residual time series, or the cross-correlation of the residual time series with the input does not show a significant interaction effect, *i.e.,* when these correlations lie within 95% confidence limits, the model can be selected as an adequate representation of the actual process [18]. Finally, another important metric for model validation is model stability, *i.e.,* the model of a stable process must be bounded input-bounded output (BIBO) stable over the process operating range [33].

1.1.2 Linear Model Identification

With the wide array of available choices and associated constraints in the selection of model structures, in input sequence design, and in validation criteria, empirical system identification is a non-trivial challenge. However, the model identification problem is somewhat simplified for linear models. Linear models are very easy to develop because the model identification step often requires specification of relatively few parameters. As an example, for FIR models, only the number of lags, or past input terms, need to be identified in order to completely specify the model. As it relates to input sequence design, any binary level sequence is applicable for linear analysis. A popular sequence used for linear identification is the pseudo random binary sequence (PRBS) which is deterministic in nature, and its properties are well understood [18]. Also, since linear models are linear with respect to parameters, the parameter estimation problem simplifies to a least-squares problem. Other non-parametric approaches have also been used to identify linear models [33]. Prominent examples are correlation analysis, transient analysis, and frequency based analysis. Unlike the correlation and transient analysis, which obtain the impulse response of the system, frequency analysis identifies the entire frequency response or transfer function of the system under study. This technique involves the discrete Fourier transform (DFT) of the input and output, and the advantage of this approach is that it allows one to concentrate on a frequency range of interest. The other advantage of using linear models is that the common linear model structures, *eg.,* FIR, ARMA, or linear state-space, are either fully or nearly equivalent.

This is an important result as it implies that preliminary model structure selection is not as important for linear models. As an example, the identified ARX model could be transformed to a linear state-space model for use in optimal or model predictive control [26].

Besides ease of identification, another reason for the wide-spread use of linear models is their suitability for control. In the most general form, when a linear model is used for controller design using the MPC philosophy, the controller objective function is quadratic. This quadratic optimization problem can be easily transformed into a quadratic program, for which a globally optimal solution exists and convergence is guaranteed. However, increasing operational, environmental, and economic constraints, have placed additional challenges on controller performance, so that nonlinear control becomes necessary. In such a scenario, the simplified linear model structures may not be able to capture the full range of process dynamic behavior and consequently may be unsuitable for nonlinear control.

1.1.3 Nonlinear Model Identification

In a stark contrast to linear models, the identification problem is considerably more complex for nonlinear systems. To begin with, nonlinear models exhibit a diverse range of nonlinear behavior, and unlike linear models, nonlinear model structures are generally not equivalent. Because of this diversity, selection of an appropriate model structure becomes critical. In this context, Pearson classified processes using a degree of nonlinearity, *i.e.* mild, intermediate, or strongly nonlinear, and the qualitative nature of process nonlinear behavior. Mildly nonlinear processes included those that exhibited asymmetric response to symmetric changes in input, and showed input multiplicities *i.e.,* the same output could be generated by different input magnitudes. Intermediate nonlinear processes showed an output multiplicative behavior, whereas strongly nonlinear processes showed chaotic dynamics. Based on these behaviors, Pearson presented guidelines for selecting one of the following nonlinear model structures: the block oriented model structure, bilinear, recursive neural network, and finally the Lur'e model structure [26]. Thus, *a priori* process information plays a critical role in the selection of a nonlinear model structure.

Selection of an appropriate input is also considerably more challenging for nonlinear models than their linear counterparts. As an example, the PRBS sequence that is widely used for linear model identification is inadequate for identification of a broad class of the block-oriented models [25]. The input should also possess enough energy to exercise the full range of process nonlinear behavior, and this persistence of excitation condition, well established for linear systems, does not have a well-defined nonlinear equivalent. Finally, as discussed in Section 1.1.1.2, a good sequence for identification may not be plant-friendly, so that input sequence design for nonlinear models is non-trivial.

This complexity in identifying a nonlinear model also extends to the controller design problem. Unlike the linear case, the optimization problem in MPC is now nonlinear, and requires the solution of a nonlinear programming problem. This solution is an iterative process and global optimality cannot be guaranteed.

However, the difficulty in identifying nonlinear models can be alleviated by using intuition gleaned from linear model identification. The parameter estimation problem can be simplified for a sub-class of the block oriented nonlinear models. The structure of the nonlinear model framework can be exploited to design tailored input sequences [30, 34]. In addition, the model structure can also be represented in a compact state-space form to facilitate its use in subsequent controller design problems [15, 17, 35, 36].

One model structure that has been successfully used to develop empirical nonlinear models is the block-oriented model structure [26, 37, 38, 39]. These models consist of a static nonlinearity $\mathcal{G}(\cdot)$ and a linear dynamic element $\mathcal{L}(t)$. The order of these blocks defines the model class: Hammerstein models are composed of a static nonlinearity followed by a linear dynamic element, whereas Wiener models have the order of these blocks reversed (Figure 3). Both the Hammerstein and Weiner models are actually special cases of the sandwich model which consists of a linear dynamic element, followed by a static nonlinearity, and then another linear dynamic element. Using $\mathcal{L}(t)$, and by selecting a specific class of $\mathcal{G}(\cdot)$, the system identification problem can be simplified.

One such simplification is the Volterra model, a subclass of the Wiener model where the static nonlinearity is of a polynomial type and the linear dynamic element is of the Finite Impulse Response (FIR) type [26]. Although the Volterra model itself is nonlinear, it is

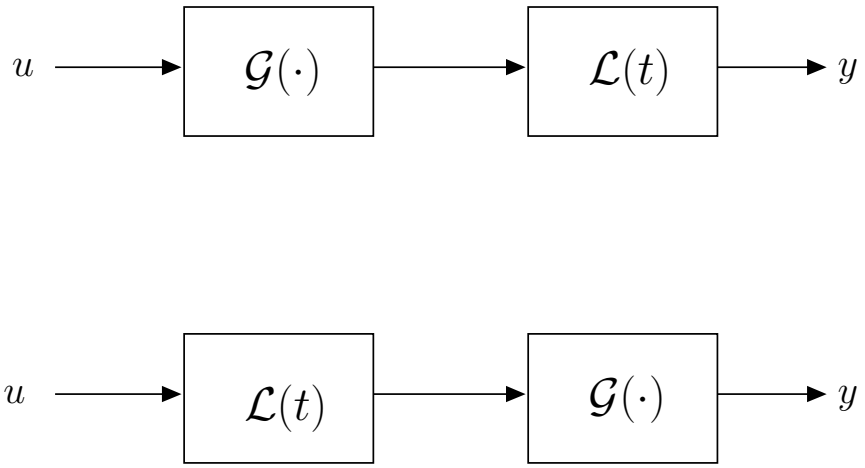

Figure 3: Schematic of block oriented model structures. Top: Hammerstein, Bottom: Wiener

still linear with respect to the model parameters, and this makes the parameter estimation problem comparatively easier. Volterra model identification is the focus of the present work.

1.1.4 Volterra Models

The general form of the discrete-time Volterra model is as follows:

$$\hat{y}(k) = h_0 + \sum_{i=1}^{N} \sum_{j_1=1}^{M} \cdots \sum_{j_N=1}^{M} h_i(j_1, .., j_N) u(k - j_1) \cdots u(k - j_N) \tag{1.2}$$

In this equation h_0 represents the bias term, N is the order of the Volterra model, and M is the model memory, which is the window of past inputs beyond which an input does not have a significant effect on the current or future outputs. The kernels of the Volterra model are given by $h_i(j_1, .., j_N)$ whereas $u(k-j_i)$ represents the past inputs, over a horizon $1 \leq j \leq M$. An essential requirement for the use of Volterra models is that the process must display fading memory, *i.e.,* past inputs must have a decreasing effect on the current output. The kernels of the Volterra series have the following properties [40]:

$$h_i(j_1, .., j_i) = 0 \quad \text{for} \quad j_i \leq 0 \quad i = 1, 2, 3, \cdots, N \tag{1.3}$$

$$\lim_{j_i \to \infty} h_i(j_1, .., j_i) = 0 \qquad i = 1, 2, 3, \cdots, N \tag{1.4}$$

$$h_i(j_1, .., j_N) \quad \text{are or can be made symmetrical} \tag{1.5}$$

The first two properties state that the output depends only on the past inputs and that the response becomes zero as values j_i increase. This is the primary reason for the imposition of the fading memory requirement on processes to be modeled. The last property states that that the kernels are symmetric, or can be made symmetrical, by a cyclic change of the arguments for the asymmetric kernel. In addition, the Volterra model structure also exhibits the following properties:

- For $N > 1$ Volterra models can capture asymmetric output responses to symmetric input changes. Such asymmetric behavior is exhibited by polymerization reactors and distillation columns, thereby rendering the Volterra model an effective structure for modeling chemical engineering systems.

- For an open-loop stable process, an empirical model based on the Volterra series is always stable. This stability is guaranteed in the presence of unmeasured disturbances and even when the input violates allowable bounds [41]. This stability feature makes the Volterra model structure very attractive in nonlinear model-based control systems.

- For $N = 1$ the Volterra model reduces to the popular and widely used linear FIR model so that the Volterra model with $N > 1$ can be thought of as a nonlinear extension of the FIR model.

1.1.4.1 Background Literature on Volterra Models Various approaches have been developed to identify Volterra kernels from process input-output data. One of the earliest was the cross-correlation method [5, 42, 43, 44, 45, 46, 47]. For a detailed description of the cross-correlation technique, the interested reader is referred to [42, 45, 46]. Cross-correlation is essentially a statistical technique that uses the input auto-correlation and input-output cross-correlation data to identify the Volterra kernels. An $N + 1$-level input sequence (at a minimum) is required to identify a Volterra model of order N by cross-correlation [28].

One of the earliest applications of the cross-correlation technique was reported by Lee and Schetzen [43]. Their approach required that the input sequence be zero-mean, white and Gaussian distributed. However, a white sequence has little practical significance as it would cause actuator wear and it may not be practical to implement in a plant setting due to the continuum of levels employed in a Gaussian white noise (GWN) sequence. As an additional requirement, the sequence was to be as long as possible to yield accurate parameter estimates. This is also a cause for concern for highly parameterized and nonlinear systems as the amount of data required to identify the parameters increases as well.

Marmarelis and Marmarelis showed that Constant-switching-pace Symmetric Random Sequences (CSRS) could be used for Volterra model identification instead of GWN signals [44]. The advantage of these signals is the fact that they are finite-level and their probability distributions can be varied to achieve the desired moment properties. The commonly known random binary sequence (RBS) is a CSRS.

Koh and Powers [46] identified a second-order Volterra model whereby the linear and quadratic kernels were identified so as to minimize the prediction error variance σ_p^2 given as:

$$\sigma_p^2 = E\{[\hat{y}(k) - y(k)]^2\} \tag{1.6}$$

In this equation $E\{\cdot\}$ is the expectation operator. Building on the work of Koh and Powers, Pearson *et al.* [48] also designed plant-friendly input sequences that did not require the input sequence to be white. In their identification algorithm for second-order Volterra models, Pearson *et al.* calculate the bias term y_0 by setting $E\{\hat{y}(k)\} = E\{y(k)\}$ *i.e.* $\hat{y}(k)$ is an unbiased estimator of $y(k)$. The linear and quadratic kernels were similarly calculated so as to minimize the PEV expression given in (1.6). In addition, they showed that the only requirements on the input were as follows:

$$\bar{u} = 0 \tag{1.7}$$

$$E\{u(k-i)u(k-m)u(k-n)\} = 0 \quad \forall \ \ i, m, n \tag{1.8}$$

This implies that the input must must have symmetric density and that the expected value of the auto-tri-correlation of the input sequence must be zero. They also identified pruned Volterra models, *i.e.,* models in which certain contributions were set to zero. The advantage of pruned models was that it reduced the number of coefficients to be estimated and hence the data required for estimation. For example, a diagonally pruned second-order kernel is one where all coefficients not on the main diagonal are zero ($h_2(i,j) = h_2(j,i) = 0$). They also analyzed the effect of the sequence of the input distribution (symmetric or Gaussian) on the identification of different combinations of pruned second-order Volterra models.

Parker *et al.* [30] developed a novel plant-friendly tailored-sequence design approach for the identification of the linear and diagonal kernels of a second-order Volterra model. This tailored input sequence was symmetric, satisfied the requirements placed in (1.7), and was significantly shorter than the sequence used in the cross-correlation technique. The identification of the kernels was also based on a PEV approach and their results showed a considerable improvement over the cross-correlation technique.

Since the Volterra model is linear in parameters, Korenberg estimated the kernel values directly by solving the least-squares problem using a QR factorization technique [49]. Zhang *et al.* utilized the Korenberg algorithm to study the effects of model nonlinearity and model memory on the quality of the Volterra kernel estimates for a nonlinear model of the lung [50]. They developed an algorithm for the automatic and simultaneous identification of the kernel and memory length from available data. They found that an incorrect memory led to substantial distortion in the kernel estimates and neglecting higher-order kernel contributions led to a significant bias in the estimates of the lower-order kernels.

Ling and Rivera presented a two-step algorithm in which a nonlinear auto-regressive model with exogenous inputs (ARX) was first estimated from the input-output data and then a Volterra model was generated from this ARX model [51]. Their reason for using this approach was to reduce the number of parameters required to identify the Volterra model since the ARX model required a lesser number of parameters to be estimated.

Dodd *et al.* solved the Volterra identification problem by creating a Hilbert space of functions corresponding to the Volterra series [52]. This Hilbert space was a reduced kernel Hilbert space (RKHS), in which a feature space was created. The feature space consists of all possible polynomials in the input up to and including the Volterra model order. The Volterra kernels were then expressed in terms of this feature space in the RKHS. Thus a mapping was obtained from the Volterra kernels in the $\mathbf{R}^N$ space to a higher dimensional Hilbert space. The advantage of this approach is that functions in the Hilbert space can be approximated using a finite series of point observations. The Volterra kernel was then easily identified in the Hilbert space. However, this technique is applicable only for problems with relatively low model memory. For physical systems where the model memory is generally high, the dimension of the feature space can be prohibitively high.

Parker and Tummala used the group method of data handling approach to identify a second-order Volterra model using neural networks [53]. They used an artificial neural network employing a combination of linear and quadratic layers based on polynomial activation functions. A set of input-output data was used to train the network by computing the necessary neural network coefficients, and then an optimum combination of the coefficient set was used to determine the Volterra model parameters. While training the network, all

possible combinations of the inputs were realized and tested for suitability. This is a problem for systems with large M, as the polynomial activation functions may take a long time to converge. Furthermore, the authors also reported convergence problems when the inputs were not properly selected. There have been other approaches to identify Volterra models using neural networks [54, 55], however they are also restricted to low memory and low order problems, and could potentially have convergence difficulties due to the polynomial activation functions.

Németh et al. identified the Volterra kernel transfer function, *i.e.,* they carried out a Fourier transform of the input and outputs to obtain the Volterra kernel [56]. The authors used a multisine input signal, which was characterized by its maximum frequency and the number of frequency components. Their analysis was restricted to second-order Volterra models and they approximated the second-order kernel space using interpolation functions. B-splines were used as the interpolation functions, and the authors obtained good results for the quadratic case. However, extension to higher orders is difficult because of the complexities in the selection of, and number of, interpolating splines. In addition, interpolation means that the kernels are not exact but only approximations of the exact Volterra kernel, and the accuracy is a strong function of the number of splines selected.

1.1.5 Laguerre Models

Another related class of models that have been used in the identification of nonlinear systems, are those based on the Laguerre function. The similarity with the Volterra model arises because the polynomial output nonlinearity is a feature of the Laguerre function as well. In addition, the Laguerre functions are orthonormal, with the generating basis function given as:

$$\phi_j(i) = \sqrt{1-\alpha^2}\left\{\sum_{k=0}^{j-1}(-\alpha)^k\binom{j-1}{k}\binom{i+k-1}{k-1}\alpha^{i-j+k}U(i-j+k)\right\} \tag{1.9}$$

In this equation, α represents the Laguerre pole and U represents the unit step function.

The Laguerre pole controls the exponential rate of decay of the Laguerre functions, and it takes values such that, $0 \leq \alpha < 1$. The Laguerre orthonormal basis describes a complete set in the real domain and can describe a class of functions (which are discussed in Chapter 3), to within arbitrary accuracy. Thus, in order to describe the dynamic behavior of a nonlinear function exactly, the Laguerre model can be given as:

$$\hat{y}_l = \sum_{j=1}^{\infty} a_j \phi_j \tag{1.10}$$

In this equation, $\hat{y}_l$ represents the Laguerre output, ϕ_j represents the j^{th} Laguerre filter, whereas a_j represent the coefficients that describe the output as a linear combination of the Laguerre filters. In practice, only a finite number of filters are necessary so that the upper limit on the summation term is give by L, the number of Laguerre filters used. Thus, identification of a system using a Laguerre model involves determination of the Laguerre pole and selection of a suitable number of Laguerre filters.

1.1.5.1 Background Literature on Laguerre Models Because of the fact that only two parameters, α and L need to be specified, the Laguerre models have been used extensively [30, 34, 57, 58, 59, 60, 61, 62, 63]. Other benefits of the Laguerre models include a good approximation to systems with time delay, tolerance to unmodeled dynamics, and a reduced sensitivity to the estimated parameters [59]. Furthermore, since the Laguerre series is orthonormal, the lower order coefficients in a Laguerre model remain practically unchanged when the model order is increased. Thus, model order can be changed on-line with minimum transient behavior. However, for models such as the ARMAX models, a change in the model order requires a change in all of the model coefficients.

For linear systems, Laguerre models offer a great substitute for FIR and ARX models. In situations where the process dynamics are fast and rapid sampling is required, the FIR models require a lot of parameters. The model parameterization can be reduced by using Laguerre functions with α selected to be the same as the dominant time constant of the system [58].

One difficulty in the identification of Laguerre models is that the output from the Laguerre model is a nonlinear function of α. On account of this nonlinearity, tailored sequence design for direct identification of Laguerre models is difficult. An alternative to direct Laguerre identification, Volterra-Laguerre (VL) models are often used. VL models involve a projection of the Volterra kernels onto the orthonormal Laguerre space. Lee first proposed this idea [43], and Watanabe and Stark identified the VL models using a least squares algorithm and non-white input sequences [57]. Marmarelis further extended these results by demonstrating the effect of α in capturing Laguerre filter exponential decay to zero.

Zheng and Zafiriou presented necessary conditions that nonlinear Volterra kernels must satisfy in order to be projected onto the Laguerre basis [41, 64]. In addition, the authors also presented an algorithm based on orthogonal regression to retain dominant terms in the VL model. As an application of these results, Seretis and Zafiriou utilized a Volterra-Laguerre model to identify a continuous stirred tank reactor system undergoing a reversible exothermic first-order reaction [65]. Using a Laguerre model with eight basis functions the authors were able to capture the process behavior represented by a fourth-order Volterra model. While Zheng and Zafiriou present very good theoretical results related to projection of Volterra kernels onto the Laguerre space, they do not address the selection of α or L.

Marmarelis used a Laguerre model to characterize renal blood flow dynamics [60]. Significant third-order dynamics were observed, and they were characterized by a Laguerre model with eight basis functions. Mitsis *et al.* used neural networks with polynomial activation functions for the hidden layers identify VL models [63]. The uniqueness in this work was that they used two α values to capture both the fast, and slow system dynamics. They applied this technique to study the cerebral auto-regulation in humans and they reveal this behavior to be significantly nonlinear.

1.2 MODEL PREDICTIVE CONTROL

Model based control is currently one of the most widely used controller design philosophies in process control, of which model predictive control is the most common. In a recent survey, Qin and Badgwell report that MPC has found applications in diverse areas such as chemicals, food processing, aerospace, pulp and paper, and the automotive industries [66]. A number of comprehensive reviews on MPC have appeared in the literature [20, 67, 68, 69, 70].

The basic philosophy in model based control is shown in Figure 4. In this figure, $\mathcal{P}$ represents the process, and it is desired that the measured process output y follow the reference trajectory r, in the presence of external disturbances d. The mathematical model that describes the process is $\mathcal{M}$, and it has an output $\hat{y}$. Both r and y are provided to the controller $\mathcal{C}$, which utilizes $\mathcal{M}$ to calculate the manipulated variable (u) changes required to attain the desired output. The manipulated variable is the solution to an on-line optimization problem which aims to minimize the difference between the predicted process response and the desired response, over a future horizon. As discussed before, the selection of the model $\mathcal{M}$ plays a significant role in the complexity of the controller design problem and in the context of this work, the model is either a Volterra, or Volterra-Laguerre, structure.

1.3 THESIS OVERVIEW

The goal of this dissertation is the development of nonlinear Volterra and Volterra-Laguerre models from input-output data. This is accomplished as follows: Chapter 2 details the Volterra model identification problem focusing on third-order systems. Input sequences are designed, and estimation algorithms are developed to identify the kernels of the Volterra model. Improvements in the sequence design and refinements in the estimation algorithm are also presented. An isothermal methyl-methacrylate polymerization reactor serves as the simulation case study. Chapter 3 details the development of the Volterra-Laguerre model. The problem of optimally determining the Laguerre pole and the number of Laguerre filters, is addressed. In addition, the performance of reduced Volterra-Laguerre models is

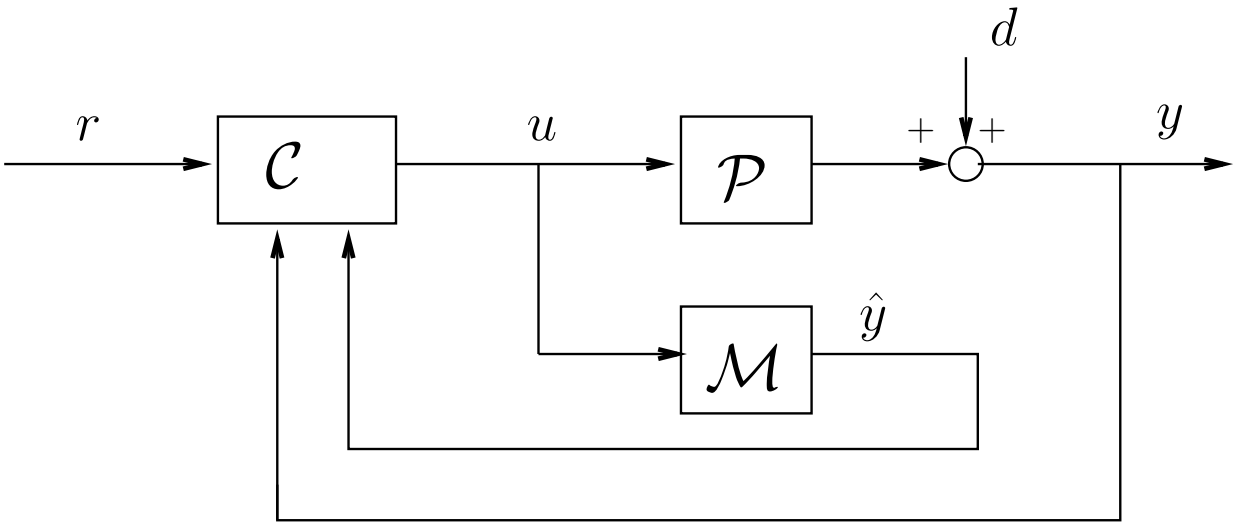

Figure 4: Schematic showing the interface between modeling and control

evaluated. In Chapter 4, MPC controllers based on both the full and reduced Volterra-Laguerre models are designed to evaluate the performance of the identified models in the closed-loop. Finally, Chapter 5 concludes the dissertation with a brief summary of the results and presents recommendations for future work.

2.0 VOLTERRA MODEL IDENTIFICATION

One of the key impediments to the wide-spread use of nonlinear control in industry is the availability of suitable nonlinear models [67]. Nonlinear models can be broadly classified into two main categories: fundamental models and empirical models. Fundamental models are derived from the physics of the underlying system. These models are generally formulated using differential equations and often have a high state dimension [8, 10, 11, 71, 72]. In many cases the model parameters are difficult to estimate, and it is often not possible to obtain full state measurement. This complexity in the system model manifests directly in the controller design and implementation, making model-based controller design a non-trivial problem. In such a scenario, empirical or black-box models provide a viable alternative to fundamental models.

In empirical modeling, a model structure is first selected, and the model identification problem involves determining the model parameters that best fit the input-output data [18]. Since only the input-output data is used to develop these models, the identification is comparatively easier, and the time required for model development is reduced [26]. As discussed in Section 1.1.4, one of the commonly used empirical nonlinear dynamic model structures is the Volterra model [17, 25, 30, 50, 60, 73]. The Volterra model offers several beneficial features. It is a nonlinear extension of the popular FIR model. In spite of being nonlinear, the Volterra model is still linear with respect to the parameters, and this makes the parameter estimation problem easier. The only condition for using a Volterra model is that the process should exhibit a fading memory, and this behavior is exhibited by many chemical engineering systems such as polymerization reactors, stirred tank reactors, catalytic crackers, and bioreactors [15, 17, 25, 73].

In previous studies on Volterra models, the primary area of emphasis has been on the identification of second-order models. However, these models display a multiplicity behavior, *i.e.,* they undergo a change in the sign of the gain. This can be cause for closed-loop instability in processes that do not show inherent second-order behavior [74]. Furthermore, for the polymerization reactor case-study considered in this work, the second-order Volterra model has limited ability to capture the qualitative steady-state behavior. Over a large operating range, the second-order Volterra model is qualitatively incorrect because of the change in the sign of the gain at the upper end of the input range. It also fails to capture the increase in output at the lower end of the input range, a behavior which is more adequately represented by a third-order Volterra model. The use of a third-order model leads to an increase in the number of parameters that need to be estimated. The performance of cross-correlation approaches for Volterra kernel estimation are adequate (as shown in the Results section), but improvements in data requirements and parameter estimate quality are possible. This chapter presents the development of tailored-sequences that use the PEV expression as a metric for model fidelity and exploit the Volterra model kernel structure to identify third-order Volterra models. These sequences are plant-friendly and shorter than the corresponding sequences for cross-correlation. In addition, explicit estimator equations are derived for the identification of the linear, nonlinear diagonal, and third-order sub-diagonal kernels of the third-order Volterra model.

2.1 POLYMERIZATION REACTOR CASE STUDY

The case study used in this work is the Congalidis, Richards, and Ray polymerization reactor [75]. This reaction system involves the isothermal free radical polymerization of methyl-methacrylate using azo-bis-isobutyronitrile as an initiator and toluene as a solvent; a schematic is shown in Figure 5. The output (y) is the number average molecular weight (NAMW) of the polymer and is a measure of the quality of the final product. The NAMW is controlled by manipulating the inlet initiator flow rate which serves as the input (u).

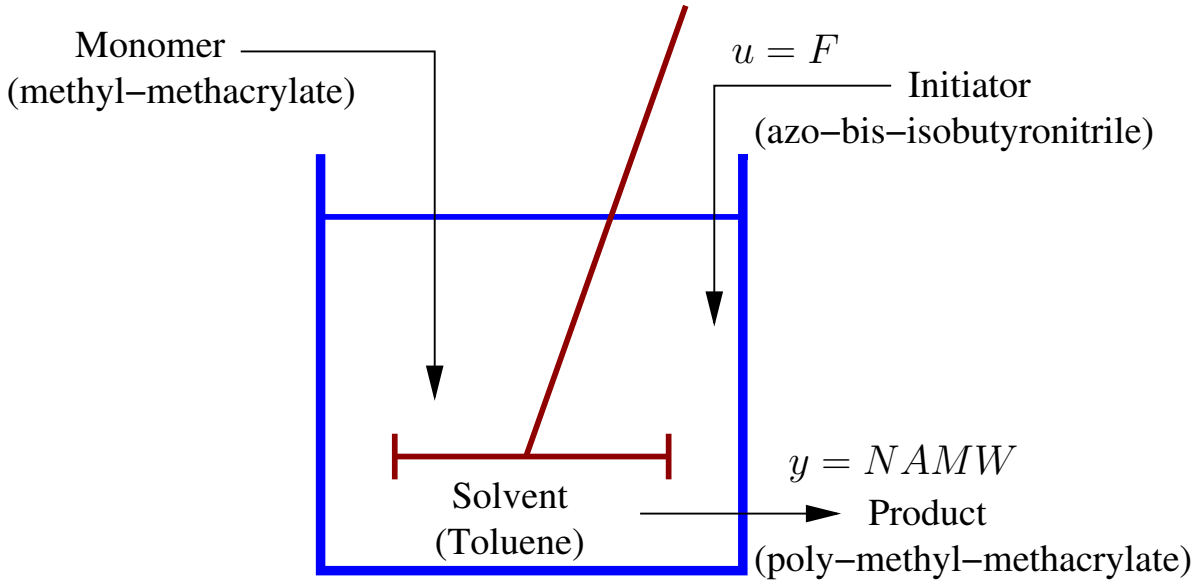

Figure 5: Schematic of the polymerization reactor case study

Table 1: Nominal operating point

Variable	Nominal Value	Units
x_{10}	5.12796	$kmol/m^3$
x_{20}	0.4791035	$kmol/m^3$
x_{30}	0.00623093	$kmol/m^3$
x_{40}	87.308673	kg/m^3
u_0	0.0605	m^3/h
y_0	14012	$kg/kmol$

The mathematical model of this system is [75]:

$$\begin{aligned} \dot{x}_1 &= 60 - 10x_1 - 2.45684x_1\sqrt{x_2} \\ \dot{x}_2 &= 80u - 10.1022x_2 \\ \dot{x}_3 &= 0.0024121x_1\sqrt{x_2} + 0.112184x_2 - 10x_3 \\ \dot{x}_4 &= 245.979x_1\sqrt{x_2} - 10x_4 \\ y &= \frac{x_4}{x_3} \end{aligned} \tag{2.1}$$

The state variables for this system are the monomer concentration (x_1), the initiator concentration (x_2), and the zeroth (x_3) and first moments (x_4) of the polymer molecular weight distribution. Under the restriction of a symmetric input sequence, the nominal operating point was selected as the mid-point of the flow rate end-points [34]. The nominal values are given in Table 1. This represents the fundamental model of the system composed of coupled nonlinear differential equations.

2.1.1 Calculation of Analytical Volterra Kernels

In order to test the effectiveness of the proposed algorithms, and have a fair comparison, a Volterra kernel representation of the fundamental model is needed. Carlemann linearization, or bi-linearization, of the system in 2.1 was performed. The nonlinear reactor system was first written in scaled deviation form such that:

$$\begin{aligned} v &= \frac{u - u_0}{0.001} \\ \tilde{y} &= \frac{y - y_0}{100} \end{aligned} \tag{2.2}$$

The scaling factor for u was chosen such that a Gaussian input sequence with a variance of one does not excite the system in a regime outside the linear region. Also, the scaling factor for y was chosen to ensure that the input and output have similar magnitudes. The scaling factors employed were the same as those used in [30, 34]. The Taylor series expansion of the resulting system (truncated to the third order) is given as:

$$\begin{aligned} \dot{q_1} &= -11.7q_1 - 0.85q_2 + 0.2125q_2^2 - 0.10625q_2^3 - 0.85q_1q_2 \\ &+ 0.2125q_1q_2^2 \\ \dot{q_2} &= 0.1669v - 10.1022q_2 \\ \dot{q_3} &= 1.3740q_1 + 9.31295q_2 - 10q_3 + 0.687q_1q_2 - 0.17175q_2^2 \\ &- 0.17175q_1q_2^2 + 0.085875q_2^3 \\ \dot{q_4} &= 10q_1 + 5q_2 - 10q_4 + 5q_1q_2 - 1.25q_2^2 - 1.25q_1q_2^2 + 0.625q_2^3 \\ \tilde{y} &= 140.12(q_4 - q_4q_3 + q_4q_3^2 - q_3 + q_3^2 - q_3^3) \end{aligned} \tag{2.3}$$

In this system, q_i represent the state variables, v represents the scaled input and $\tilde{y}$ the scaled output, all of which are in deviation form. The complete derivation is shown in Appendix C. Using this system of equations, Carlemann linearization of the system was carried out [40]. The resulting system in state-space form is given as,

$$\begin{aligned} \dot{\tilde{q}} &= A_c\tilde{q} + N_c\tilde{q}v + B_cv \\ \tilde{y}_c &= C_c\tilde{q} \end{aligned} \tag{2.4}$$

For the polymerization reactor case study, these matrices are reported in Appendix D. In equation (2.4),

$$\tilde{q} = [\ \tilde{q}_1\ \tilde{q}_2\ \tilde{q}_3]^T$$

and,

$$\begin{aligned}
\tilde{q_1} &= [\ q_1\ q_2\ q_3\ q_4\ ] \\
\tilde{q_2} &= [\ q_1^2\ q_1q_2\ q_1q_3\ q_1q_4\ q_2^2\ q_2q_3\ q_2q_4\ q_3^2\ q_3q_4\ q_4^2\ ] \\
\tilde{q_3} &= [\ q_1^3\ q_1^2q_2\ q_1^2q_3\ q_1^2q_4\ q_1q_2^2\ q_1q_2q_3\ q_1q_4^2\ q_2^3\ q_2^2q_3\ q_2^2q_4\ q_2q_3^2\ q_2q_3q_4\ q_2q_4^2\ q_3^3\ q_3^2q_4\ q_4^3\ q_3q_4^2\ ]
\end{aligned}$$

The plot of the nonlinear system response to step-changes of magnitude $v = \pm 30$ is shown in Figure 6. The output responses from the third-order Carlemann linearized model to the same inputs are also shown. Also, Figure 7 shows the Integral Square Error (ISE) profiles between the nonlinear and the identified Volterra models in response to steps of integer magnitude in $[-56, 56]$ over 1.2 hr. It is observed that the third-order Carlemann linearized model closely approximates the nonlinear reactor system. However, due to the approximation introduced in the bi-linearization, the Carlemann linearized model output deviates from the nonlinear model output at the input extremes (Figure 7).

The bilinear state-space system (2.4) was then discretized using the fourth-order Runge-Kutta algorithm yielding:

$$\begin{aligned}
\tilde{q}(k+1) &= \hat{A}_0\tilde{q}(k) + \hat{A}_1\tilde{q}(k)v(k) + \hat{A}_2q(k)v^2(k) \\
&\quad +\hat{B}_1v(k) + \hat{B}_2v^2(k) + \hat{B}_3v^3(k) \qquad (2.5) \\
\tilde{y}_{disc}(k) &= C_c\tilde{q}(k)
\end{aligned}$$

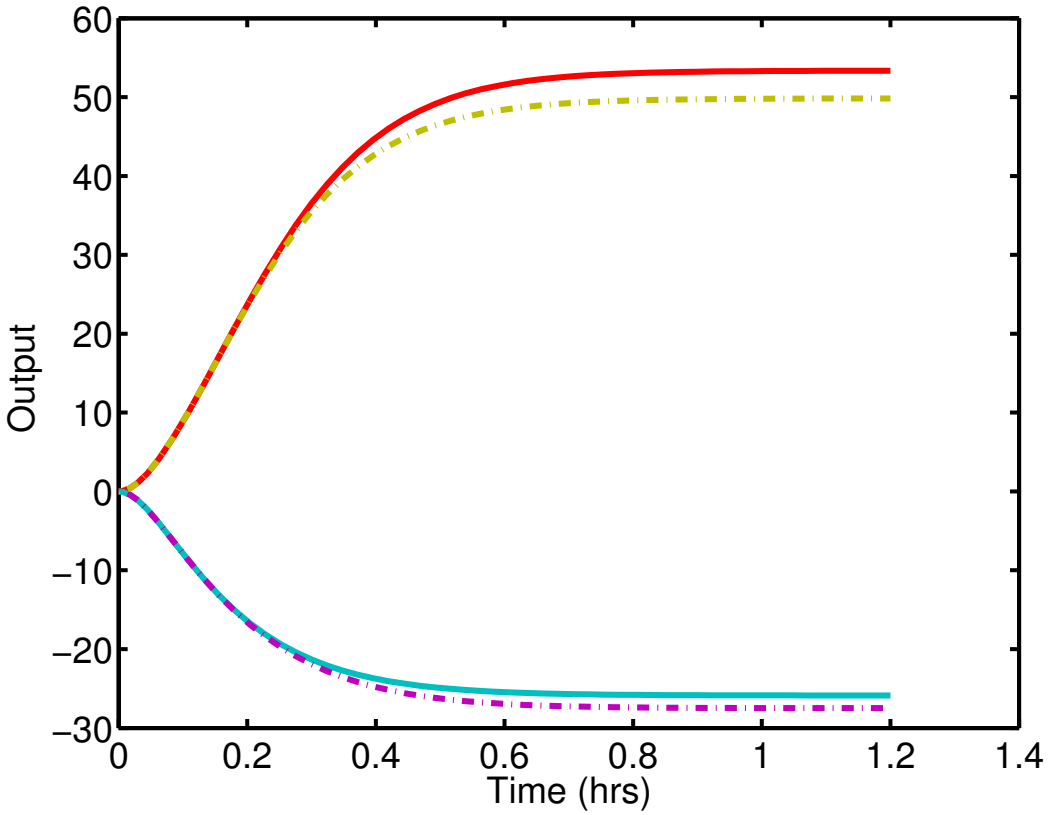

Figure 6: Plot of the polymerizer response to a step-change of ± 30 in v. Nonlinear model (—) and the Carlemann linearized model truncated to third-order terms $(- \cdot -)$

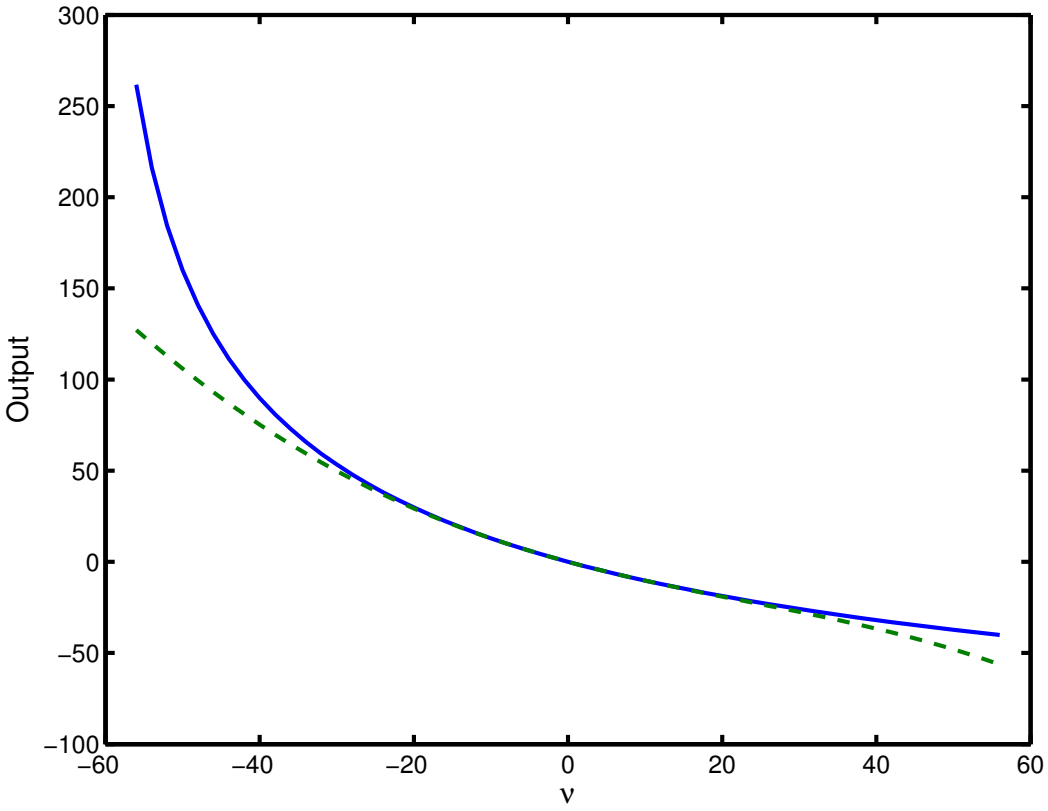

Figure 7: Steady-state step response ISE profiles between the nonlinear (—) and the Carlemann linearized model truncated to third-order terms ($- \cdot -$)

The discrete time matrices in (2.5) are related to their continuous time counterparts via the following relations:

$$
\begin{aligned}
\tilde{A}_0 &= I + \frac{\Delta t}{6}\{6A_c + 3\Delta t A_c^2 + \Delta t^2 A_c^3 + \frac{1}{4}\Delta t^3 A_c^4\} \\
\tilde{A}_1 &= \frac{\Delta t}{6}\{6N_c + 3\Delta t(A_c N_c + N_c A_c) + \Delta t^2(A_c^2 N_c + A_c N_c A_c + N_c A_c^2) \\
&\quad + \frac{1}{4}\Delta t^3(A_c^3 N_c + A_c^2 N_c A_c + A_c N_c A_c^2 + N_c A_c^3)\} \\
\tilde{A}_2 &= \frac{\Delta t}{6}\{3\Delta t N_c^2 + \Delta t^2(A_c N_c^2 + N_c A_c N_c + N_c^2 A_c) \\
&\quad + \frac{1}{4}\Delta t^3(A_c^2 N_c^2 + A_c N_c A_c N_c + A_c N_c^2 A_c + N_c A_c^2 N_c + N_c A_c N_c A_c + N_c^2 A_c^2)\} \\
\tilde{B}_1 &= \frac{\Delta t}{6}\{6B_c + 3\Delta t A_c B_c + \Delta t^2(A_c^2 B_c) + \frac{1}{4}\Delta t^3(A_c^3 B_c)\} \\
\tilde{B}_2 &= \frac{\Delta t}{6}\{3\Delta t N_C B_c + \Delta t^2(A_c N_C B_c + N_c A_c B_c) + \frac{1}{4}\Delta t^3(A_c^2 N_c B_c + A_c N_c A_c B_c + N_c A_c^2 B_c)\} \\
\tilde{B}_3 &= \frac{\Delta t}{6}\{\Delta t^2 N_C^2 B_c + \frac{1}{4}\Delta t^3(A_c N_c^2 B_c + N_c A_c N_c B_c + N_c^2 A_c B_c)\}
\end{aligned}
$$

In the equation for $\tilde{A}_0$, I is the identity matrix. Based on the analysis in [34, 30, 25], a sampling time (Δt) of 0.06 hr and model-memory $M = 20$ were selected. From the discretized equations, the Volterra kernels for the system were determined as follows [40]:

$$
\begin{aligned}
h_1(i) &= C\tilde{A}_0^{i-1}\tilde{B}_1 \;\; \forall\, i \in [1, M],\; j = l = 0 \\
h_2(i,i) &= C\tilde{A}_0^{j-1}\tilde{B}_2 \;\; \forall\, i \in [1, M],\; j \in [1, i],\; i = j,\; l = 0 \\
h_2(i,j) &= C\tilde{A}_0^{j-1}\tilde{A}_1\tilde{A}_0^{i-1}\tilde{B}_1 \;\; \forall\, i \in [1, M],\; j \in [1, i],\; i \neq j,\; l = 0 \\
h_3(i,i,i) &= C\tilde{A}_0^{i-1}\tilde{B}_3 \;\; \forall\, i \in [1, M],\; j \in [1, i],\; l \in [1, j],\; i = j = l \\
h_3(i,j,l) &= C\tilde{A}_0^{l-1}\tilde{A}_1\tilde{A}_0^{j-1}\tilde{B}_2 \;\; \forall\, i \in [1, M],\; j \in [1, i],\; l \in [1, j],\; j = l \\
h_3(i,j,l) &= C\tilde{A}_0^{l-1}\tilde{A}_2\tilde{A}_0^{i-1}\tilde{B}_1 \;\; \forall\, i \in [1, M],\; j \in [1, i],\; l \in [1, j],\; i = l \\
h_3(i,j,l) &= C\tilde{A}_0^{l-1}\tilde{A}_1\tilde{A}_0^{j-1}\tilde{A}_1\tilde{A}_0^{i-1}\tilde{B}_1 \;\forall\, i \in [1, M],\; j \in [1, i],\; l \in [1, j],\; i \neq j \neq l
\end{aligned}
$$

The kernels thus obtained are symmetric, or triangular, kernels and they can be converted to full kernels using (2.9). A third-order Volterra model is thus obtained, which is used to test the effectiveness of the identification algorithms vis-a-vis the real system.

2.2 VOLTERRA MODEL IDENTIFICATION USING TAILORED SEQUENCES

2.2.1 Third-order Volterra Model Decomposition

Without loss of generality, the third-order Volterra model can be written as [30, 76]:

$$\hat{y}(k) = h_0 + L(k) + D_2(k) + D_3(k) + S_3(k) + O_2(k) + O_3(k) \tag{2.6}$$

In this equation L, D, S, and O represent the linear, diagonal, sub-diagonal, and off-diagonal terms, respectively. A schematic of the Volterra model kernels is given in Figure 8, and mathematically these kernels are given as:

$$\begin{aligned}
L(k) &= \sum_{i=1}^{M} h_1(i)u(k-i) \\
D_2(k) &= \sum_{i=1}^{M} h_2(i,i)u^2(k-i) \\
D_3(k) &= \sum_{i=1}^{M} h_3(i,i,i)u^3(k-i) \\
S_3(k) &= 3\sum_{i=1}^{M}\sum_{j\neq i}^{M} h_3(i,i,j)u^2(k-i)u(k-j) \\
O_2(k) &= \sum_{i=1}^{M}\sum_{j\neq i}^{M} h_2(i,j)u(k-i)u(k-j) \\
O_3(k) &= \sum_{i=1}^{M}\sum_{j\neq i}^{M}\sum_{l\neq j\neq i}^{M} h_3(i,j,l)u(k-i)u(k-j)u(k-l)
\end{aligned} \tag{2.7}$$

The linear term, $L(k)$, is an FIR model and can be visualized as a column vector. The first order kernel $h_1(i)$, is comprised of the impulse response coefficients. The second order-kernel, $h_2(i,j)$, can be visualized as a matrix, whereas the third-order kernel, $h_3(i,j,l)$, can be thought of as a 3-index tensor (cube). The contributions of the past inputs are given by the terms $u(k-i)$, and the first, second, and third-order kernels include the effects of one, two and three past inputs, respectively. The diagonal and the off-diagonal terms include contributions from both the second- $(D_2(k), O_2(k))$ and third-order $(D_3(k), O_3(k))$

kernels. The diagonal terms are those for which all the indices are the same, whereas for the off-diagonal kernel all the indices are different from one another. Finally, the third-order sub-diagonal term ($S_3(k)$) represents the kernel having one index repeated, so that the sub-diagonal coefficients are just above or just below the main diagonal of the full third-order kernel.

The Volterra model can be assumed symmetric without loss of generality [40]. For example, considering the two-dimensional case the following holds:

$$h_{2_{sym}}(j_1, j_2) = \frac{1}{2}\left[h_{2_{asym}}(j_1, j_2) + h_{2_{asym}}(j_2, j_1)\right] \tag{2.8}$$

Generalizing for n-dimensions, this property can be written as:

$$h_{i_{sym}}(j_1, j_2, \cdots, j_N) = \frac{1}{N!}\left[h_{2_{asym}}(j_1, \cdots j_N) + \cdots + h_{i_{asym}}(j_N, \cdots j_1)\right] \tag{2.9}$$

Symmetry in the third-order term can be visualized with respect to the main diagonal, and this reduces the number of unique parameters that need to be determined. A full third-order Volterra model has $1 + M + M^2 + M^3$ total coefficients, whereas the total number of unique symmetric coefficients is given by:

$$\sum_{i=1}^{3} \binom{M+i-1}{i} = 1 + M + \frac{M(M+1)}{2!} + \frac{M(M+1)(M+2)}{3!}$$

As an example, for $M = 20$ the full kernel has 8421 coefficients, whereas the symmetric kernel has 1771 coefficients.

Another advantage of carrying out the decomposition in (2.7) is that it facilitates development of the identification algorithm as the identification is carried out on the basis of structure and order (L and D, O_2, S, and O_3) rather than based only on the polynomial order, *i.e.*, first, followed by the second, and finally the third-order kernel.

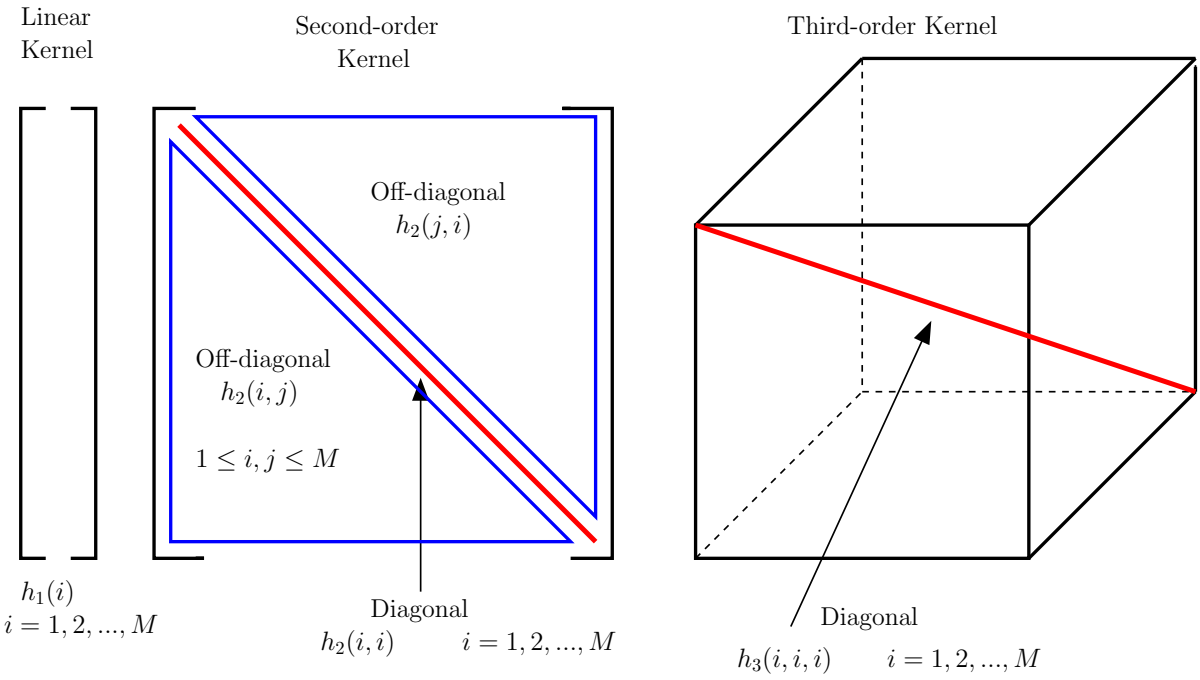

Figure 8: Schematic of the Volterra model kernels

2.2.2 Prediction Error Variance

The PEV expression forms the conceptual basis on which the identification of the third-order Volterra kernels is carried out. It can be represented mathematically as:

$$\sigma_p^2 = E\left\{[\hat{y}(k) - y(k)]^2\right\} \tag{2.10}$$

In this expression, σ_p^2 represents the PEV, $E\{\cdot\}$ the expectation operator, and $y(k)$, and $\hat{y}(k)$ represent the outputs from the process (or fundamental model), and the Volterra model, respectively. For the third-order Volterra model the PEV expression can be written in terms of model coefficient errors as follows [25]:

$$\begin{aligned}\sigma_p^2 \;=\; & \sigma_0^2 + \sigma_u^2 \sum_{i=1}^{M} \delta_1^2(i) + (\kappa+2)\sigma_u^4 \sum_{i=1}^{M} \delta_2^2(i,i) + (m_6 - \frac{m_4^2}{\sigma_u^2}) \sum_{i=1}^{M} \delta_3^2(i,i,i) \\ & + 9(\kappa+2)\sigma_u^6 \sum_{i=1}^{M}\sum_{j\neq i}^{M} \delta_3^2(i,j,j) + 2\sigma_u^4 \sum_{i=1}^{M}\sum_{j\neq i}^{M} \delta_2^2(i,j) + 6\sigma_u^6 \sum_{i=1}^{M}\sum_{j\neq i}^{M}\sum_{l\neq j\neq i}^{M} \delta_3(i,j,l)\end{aligned} \tag{2.11}$$

This equation assumes that a Volterra model is identified from a process which is described by a Volterra model structure. In equation (2.11) σ_0^2 is the bias term, σ_u^2 is the input signal variance, and m_k is the k^{th} moment of the input signal. The coefficient κ represents the kurtosis. The PEV equation is written with respect to the coefficient error variance, where δ denotes the coefficient error between the identified and actual Volterra kernels. In the PEV expression, the terms are as follows: first term - bias; second - linear model error contribution; third and fourth - contributions of the nonlinear diagonal terms; fifth - sub-diagonal term contribution; and sixth and seventh - contributions of the off-diagonal coefficients.

For a given input signal, the kurtosis is mathematically defined as:

$$\kappa = \frac{m_4}{\sigma_u^4} \tag{2.12}$$

Kurtosis is a measure of the peak and tail shape of the input signal distribution relative to a normally distributed input sequence. As it pertains to input sequence design, kurtosis was used as a measure to characterize the situation when the bulk of the input levels are of a certain magnitude, whereas the remainder are of a higher magnitude, thus leading to a sharp distinction in the two input levels. Figure 9 shows an example of such a sequence.

The kurtosis of this sequence is 16.10, and the probability distribution that generated this sequence is given as:

$$u_{\kappa 1}(k) = \begin{cases} 56 & \text{with probability } \frac{3}{100} \\ 5.6 & \text{with probability } \frac{47}{100} \\ -5.6 & \text{with probability } \frac{47}{100} \\ -56 & \text{with probability } \frac{3}{100} \end{cases} \tag{2.13}$$

In contrast, consider a sequence given by the following probability distribution:

$$u_{\kappa 2}(k) = \begin{cases} 56 & \text{with probability } \frac{15}{100} \\ 5.6 & \text{with probability } \frac{35}{100} \\ -5.6 & \text{with probability } \frac{35}{100} \\ -56 & \text{with probability } \frac{15}{100} \end{cases} \tag{2.14}$$

The probabilities of the two levels are not as different from each other (*i.e.,* one level does not occur significantly more often than the other). Consequently, the kurtosis of this sequence (Figure 10) is -0.09. Hence, for signals represented as probability distributions, kurtosis can be used as a surrogate measure of outlier frequency.

2.2.3 Linear and Nonlinear Diagonal Identification

The identification of the bias, linear, and nonlinear diagonal parameters is carried out first in order to take advantage of tailored input sequences that selectively excite these contributions. Furthermore, since the diagonal parameters are involved in the calculation of all of the other kernels, an accurate estimate of the diagonal terms at the beginning ensures better estimates for the subsequent parameters. The general form of the input sequence used for

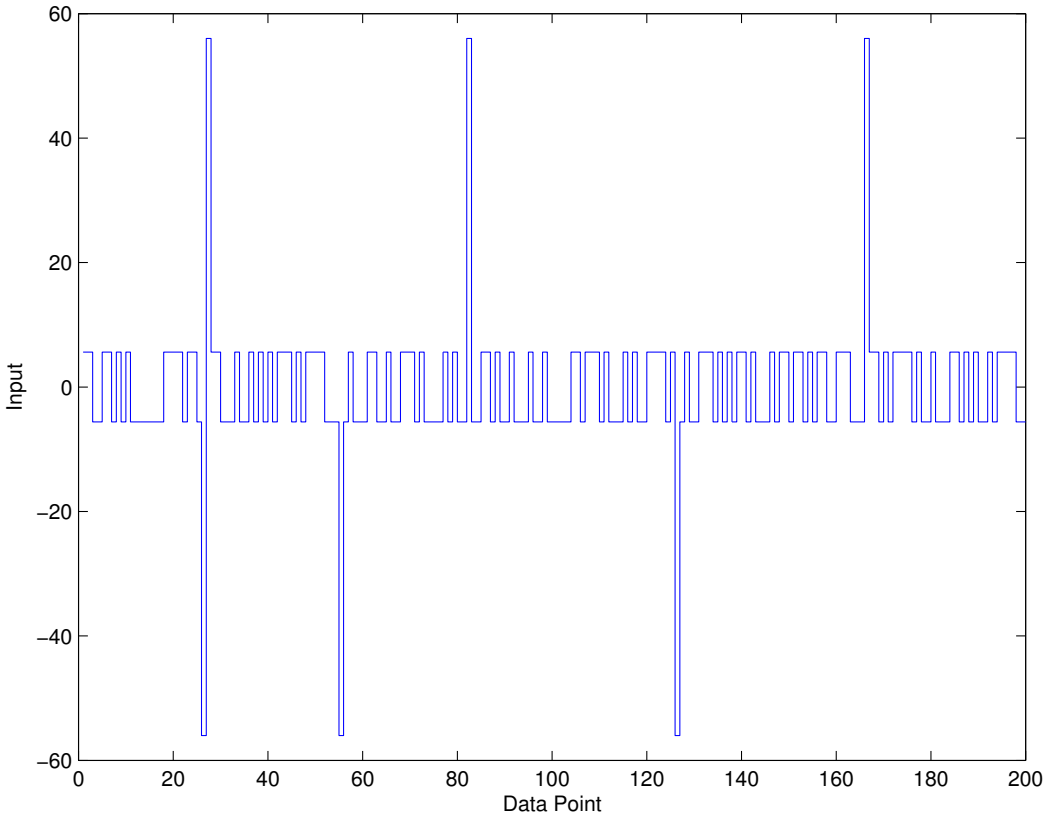

Figure 9: Distribution of input levels in a sequence with high kurtosis

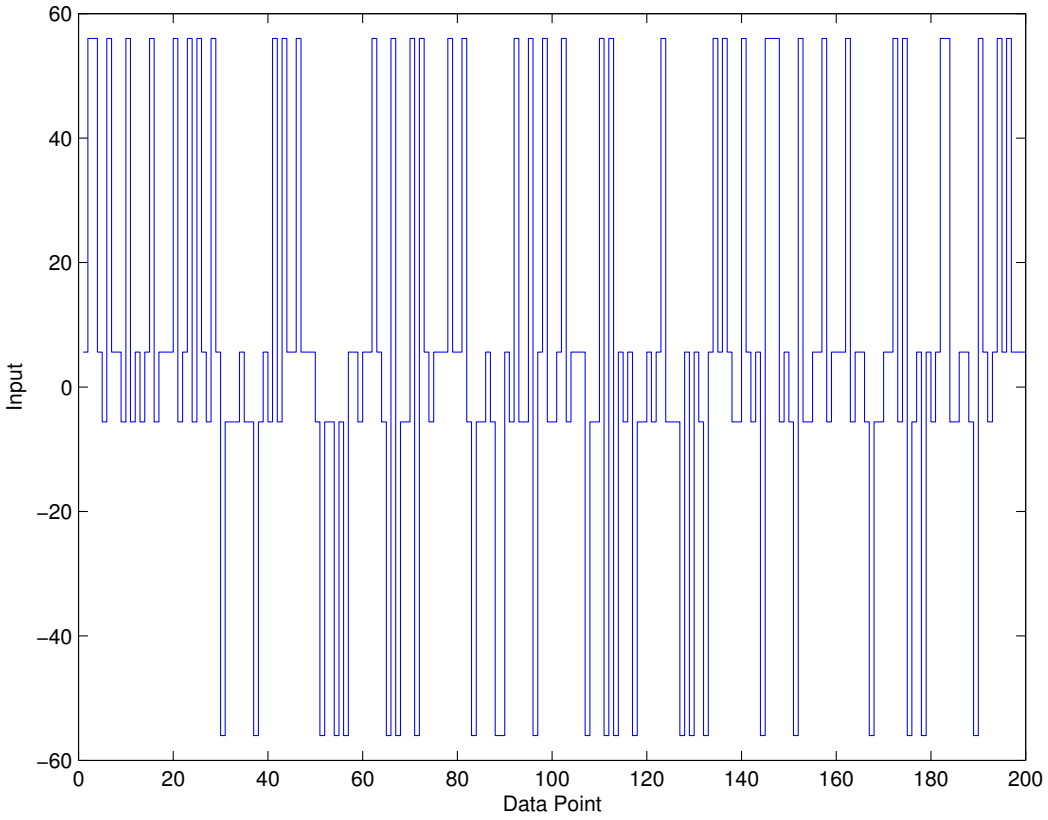

Figure 10: Distribution of input levels in a sequence with low kurtosis

the estimation of the linear and nonlinear diagonal kernels is given as:

$$u(k) = \begin{cases} \gamma_1 & k = 0 \\ 0 & 1 \leq k \leq M \\ \gamma_3 & k = M+1 \\ 0 & M+2 \leq k \leq 2M+1 \\ \gamma_2 & k = 2M+2 \\ 0 & 2M+3 \leq k \leq 3M+2 \\ \gamma_4 & k = 3M+3 \\ 0 & 3M+4 \leq k \leq 4M+3 \end{cases} \tag{2.15}$$

This sequence ensures that the output contributions due to the nonlinear sub-diagonal and off-diagonal terms are zero identically ($u(k-i)u(k-j) = 0 \quad \forall i \neq j \quad (1 \leq i,\ j \ \leq M)$). Applying this input to the third-order Volterra model yields the following output:

$$\hat{y}(k) = \begin{cases} h_0 & k = 0 \\ h_0 + \gamma_1 h_1(k) + \gamma_1^2 h_2(k,k) + \gamma_1^3 h_3(k,k,k) & 1 \leq k \leq M \\ h_0 & k = M+1 \\ h_0 + \gamma_3 h_1(k-M-1) + \gamma_3^2 h_2(k-M-1, k-M-1) & \\ \quad +\gamma_3^3 h_3(k-M-1, k-M-1, k-M-1) & M+2 \leq k \leq 2M+1 \\ h_0 & k = 2M+2 \\ h_0 + \gamma_2 h_1(k-2M-2) + \gamma_2^2 h_2(k-2M-2, k-2M-2) & \\ \quad +\gamma_2^3 h_3(k-2M-2, k-2M-2, k-2M-2) & 2M+3 \leq k \leq 3M+2 \\ 0 & k = 3M+3 \\ h_0 + \gamma_4 h_1(k-3M-3) + \gamma_4^2 h_2(k-3M-3, k-3M-3) & \\ \quad +\gamma_4^3 h_3(k-3M-3, k-3M-3, k-3M-3) & 3M+4 \leq k \leq 4M+3 \end{cases} \tag{2.16}$$

The linear and nonlinear diagonal kernels are derived by minimizing the sum-squared prediction error given as:

$$J_d = \sum_{k=0}^{4M+3} e(k)^2 = \sum_{k=0}^{4M+3} \{y(k) - \hat{y}(k)\}^2$$

In this equation $e(k)$ is a zero-mean prediction error sequence due to measurement errors in the data-set $y(k)$ and any plant-model mismatch, and $\hat{y}(k)$ is the model prediction with the input sequence given in (2.15). Applying the condition

$$\frac{\partial J_d}{\partial h_n} = 0 \qquad n = 0, 1, 2, 3$$

and simultaneously solving the equations for $\frac{\partial J_d}{\partial h_1(k)} = 0$ and $\frac{\partial J_d}{\partial h_3(k,k,k)} = 0$ yields estimators for the linear and the third-order diagonal kernels. This is a departure from [30] where the estimator $h_1(k)$ was calculated independently. Since the current sequence identifies a third-order Volterra model, the estimators for $h_1(k)$ and $h_3(k,k,k)$ are interdependent and must be calculated simultaneously. Thus, for $h_1(k)$, $\frac{\partial J_d}{\partial h_1(k)} = 0$ yields:

$$\begin{aligned}
&(\gamma_1 + \gamma_3 + \gamma_2 + \gamma_4)h_0 + (\gamma_1^2 + \gamma_3^2 + \gamma_2^2 + \gamma_4^2)h_1(k) \\
&+(\gamma_1^3 + \gamma_3^3 + \gamma_2^3 + \gamma_4^3)h_2(k,k) + (\gamma_1^2 + \gamma_3^2 + \gamma_2^2 + \gamma_4^2)h_3(k,k,k) \\
&= \gamma_1 y(k) + \gamma_3 y(k+M+1) + \gamma_2 y(k+2M+2) + \gamma_4 y(k+3M+3) \quad \forall\ k = 1, \cdots, M
\end{aligned} \tag{2.17}$$

Similarly, $\frac{\partial J_d}{\partial h_3(k,k,k)} = 0$ yields:

$$\begin{aligned}
&(\gamma_1^3 + \gamma_3^3 + \gamma_2^3 + \gamma_4^3)h_0 + (\gamma_1^4 + \gamma_3^4 + \gamma_2^4 + \gamma_4^4)h_1(k) \\
&+(\gamma_1^5 + \gamma_3^5 + \gamma_2^5 + \gamma_4^5)h_2(k,k) + (\gamma_1^6 + \gamma_3^6 + \gamma_2^6 + \gamma_4^6)h_3(k,k,k) \\
&= \gamma_1^3 y(k) + \gamma_3^3 y(k+M+1) + \gamma_2^3 y(k+2M+2) + \gamma_4^3 y(k+3M+3) \quad \forall\ k = 1, \cdots, M
\end{aligned} \tag{2.18}$$

Likewise, the even-order estimators are co-dependent and must be derived simultaneously. Solving for $\frac{\partial J_d}{\partial h_0} = 0$ yields:

$$\begin{aligned}
\sum_{k=0}^{4M+3} y(k) \;=\; & 4M + 4h_0 + (\gamma_1 + \gamma_3 + \gamma_2 + \gamma_4)\sum_{k=1}^{M} h_1(k) + (\gamma_1^2 + \gamma_3^2 + \gamma_2^2 + \gamma_4^2)\sum_{k=1}^{M} h_2(k,k) \\
& +(\gamma_1^3 + \gamma_3^3 + \gamma_2^3 + \gamma_4^3)\sum_{k=1}^{M} h_3(k,k,k)
\end{aligned} \tag{2.19}$$

Finally, $\frac{\partial J_d}{\partial h_2(k,k)} = 0$ returns:

$$\begin{aligned}
&(\gamma_1^2 + \gamma_3^2 + \gamma_2^2 + \gamma_4^2)h_0 + (\gamma_1^3 + \gamma_3^3 + \gamma_2^3 + \gamma_4^3)h_1(k) \\
&+(\gamma_1^4 + \gamma_3^4 + \gamma_2^4 + \gamma_4^4)h_2(k,k) + (\gamma_1^5 + \gamma_3^5 + \gamma_2^5 + \gamma_4^5)h_3(k,k,k) \\
&= \gamma_1^2 y(k) + \gamma_3^2 y(k+M+1) + \gamma_2^2 y(k+2M+2) + \gamma_4^2 y(k+3M+3) \quad \forall\ k = 1, \cdots, M
\end{aligned} \tag{2.20}$$

Solving Equations (2.17) through (2.20) simultaneously, one can derive equations for the linear and nonlinear diagonal kernel estimators. However, solving these equations in the general form is algebraically tedious and the solution provides limited insight. As a special case, consider:

$$\begin{aligned} \gamma_3 &= -\gamma_1 \\ \gamma_4 &= -\gamma_2 \\ \gamma_1, \gamma_2 &> 0 \end{aligned} \tag{2.21}$$

With this simplification and using γ_1 and γ_2 to represent the magnitudes of the two unique pulse-heights, the input sequence simplifies to the following:

$$u_d(k) = \begin{cases} \gamma_1 & k = 0 \\ 0 & 1 \leq k \leq M \\ -\gamma_1 & k = M+1 \\ 0 & M+2 \leq k \leq 2M+1 \\ -\gamma_2 & k = 2M+2 \\ 0 & 2M+3 \leq k \leq 3M+2 \\ \gamma_2 & k = 3M+3 \\ 0 & 3M+4 \leq k \leq 4M+3 \end{cases} \tag{2.22}$$

In order to ensure suitable excitation, the pulse-heights were chosen as large as possible. Furthermore, to avoid a singularity in the estimator equations (denominator term in (2.28) and (2.29) derived below), the smaller pulse-height was selected to be 80% of the larger one. This led to $\gamma_1 = 44.8$ and $\gamma_2 = 56$. This magnitude of γ_2 spans the full input range (in scaled deviation variables) for the polymerization reactor example considered in this work. The 20% difference in the pulse-heights also ensures that the output contributions can be distinguished from each other in the presence of noise. The sequence given by Equation (2.22) (Figure 11) has a friendliness factor (Equation 1.1) of 93%.

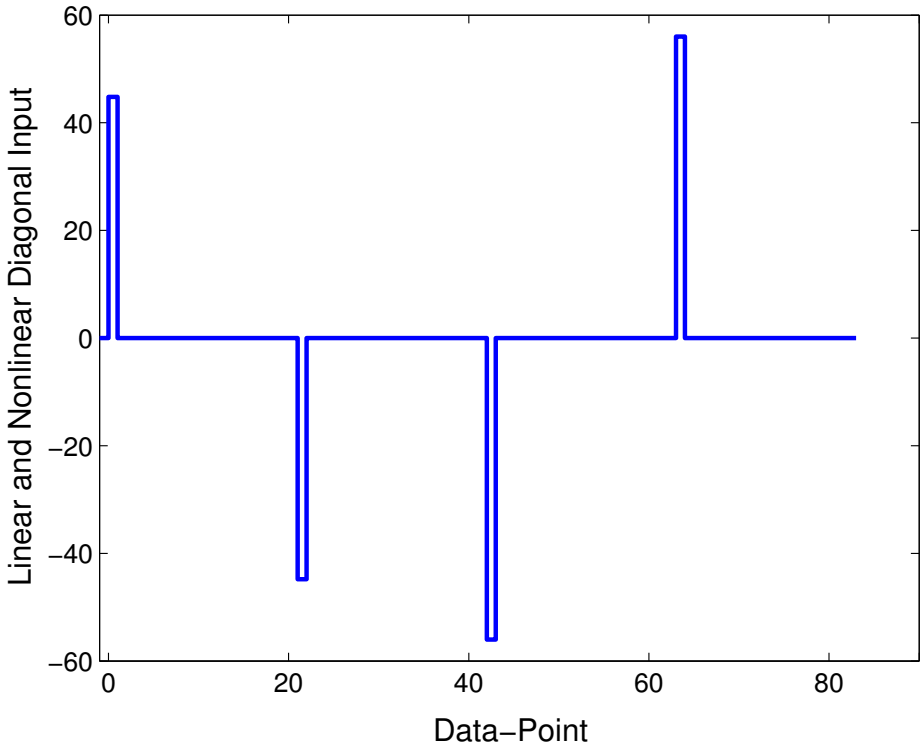

Figure 11: Input sequence used for linear and nonlinear diagonal kernel identification. The levels are $\gamma_1 = 44.8$ and $\gamma_2 = 56$, and $M = 20$.

Applying this input to the third-order Volterra model yields the following output:

$$\hat{y}(k) = \begin{cases} h_0 & k = 0 \\ h_0 + \gamma_1 h_1(k) + \gamma_1^2 h_2(k,k) + \gamma_1^3 h_3(k,k,k) & 1 \le k \le M \\ h_0 & k = M+1 \\ h_0 - \gamma_1 h_1(k-M-1) + \gamma_1^2 h_2(k-M-1,k-M-1) & \\ -\gamma_1^3 h_3(k-M-1,k-M-1,k-M-1) & M+2 \le k \le 2M+1 \\ h_0 & k = 2M+2 \\ h_0 - \gamma_2 h_1(k-2M-2) + \gamma_2^2 h_2(k-2M-2,k-2M-2) & \\ -\gamma_2^3 h_3(k-2M-2,k-2M-2,k-2M-2) & 2M+3 \le k \le 3M+2 \\ 0 & k = 3M+3 \\ h_0 + \gamma_2 h_1(k-3M-3) + \gamma_2^2 h_2(k-3M-3,k-3M-3) & \\ +\gamma_2^3 h_3(k-3M-3,k-3M-3,k-3M-3) & 3M+4 \le k \le 4M+3 \end{cases} \tag{2.23}$$

Applying the condition $\frac{\partial J_d}{\partial h_1(k)} = 0$ yields:

$$\begin{aligned} h_1(k) &= \frac{\gamma_1}{2(\gamma_1^2+\gamma_2^2)}\{y(k) - y(k+M+1)\} \\ &\quad + \frac{\gamma_2}{2(\gamma_1^2+\gamma_2^2)}\{y(k+2M+2) - y(k+3M+3)\} \\ &\quad - \frac{(\gamma_1^4+\gamma_2^4)}{(\gamma_1^2+\gamma_2^2)} h_3(k,k,k) \end{aligned} \tag{2.24}$$

Similarly, $\frac{\partial J_d}{\partial h_3(k,k,k)} = 0$ yields:

$$\begin{aligned} h_1(k) &= \frac{\gamma_1^3}{2(\gamma_1^4+\gamma_2^4)}\{y(k) - y(k+M+1)\} \\ &\quad + \frac{\gamma_2^3}{2(\gamma_1^4+\gamma_2^4)}\{y(k+2M+2) - y(k+3M+3)\} \\ &\quad - \frac{(\gamma_1^6+\gamma_2^6)}{(\gamma_1^4+\gamma_2^4)} h_3(k,k,k) \end{aligned} \tag{2.25}$$

Solving Equations (2.24) and (2.25) simultaneously returns the estimator equations for the linear and the third-order diagonal kernels.

$$
\begin{aligned}
h_1(k) &= \frac{n_1(k)}{d_{odd}} \\
n_1(k) &= \frac{\gamma_1^6+\gamma_2^6}{\gamma_1^4+\gamma_2^4}\Big[\frac{\gamma_1}{2(\gamma_1^2+\gamma_2^2)}\{y(k)-y(k+M+1)\} \\
&\quad +\frac{\gamma_2}{2(\gamma_1^2+\gamma_2^2)}\{y(k+2M+2)-y(k+3M+3)\}\Big] \\
&\quad -\frac{\gamma_1^4+\gamma_2^4}{\gamma_1^2+\gamma_2^2}\Big[\frac{\gamma_1^3}{2(\gamma_1^4+\gamma_2^4)}\{y(k)-y(k+M+1)\} \\
&\quad +\frac{\gamma_2^3}{2(\gamma_1^4+\gamma_2^4)}\{y(k+2M+2)-y(k+3M+3)\}\Big]
\end{aligned}
\tag{2.26}
$$

$$
\begin{aligned}
d_{odd} &= \frac{\gamma_1^6+\gamma_2^6}{\gamma_1^4+\gamma_2^4}-\frac{\gamma_1^4+\gamma_2^4}{\gamma_1^2+\gamma_2^2} \\
h_3(k,k,k) &= \frac{n_3(k)}{d_{odd}(k)} \\
n_3(k) &= \Big[\frac{\gamma_1^3}{2(\gamma_1^4+\gamma_2^4)}-\frac{\gamma_1}{2(\gamma_1^2+\gamma_2^2)}\Big]\{y(k)-y(k+M+1)\} \\
&\quad +\Big[\frac{\gamma_2^3}{2(\gamma_1^4+\gamma_2^4)}-\frac{\gamma_2}{2(\gamma_1^2+\gamma_2^2)}\Big]\{y(k+2M+2) \\
&\quad -y(k+3M+3)\}
\end{aligned}
\tag{2.27}
$$

These equations can be simplified to yield:

$$
\begin{aligned}
h_1(k) = \frac{1}{2\gamma_1\gamma_2(\gamma_2^2-\gamma_1^2)} \quad & \Big[\gamma_2^3\{y(k)-y(k+M+1)\} \\
& -\gamma_1^3\{y(k+2M+2)-y(k+3M+3)\}\Big]
\end{aligned}
\tag{2.28}
$$

$$
\begin{aligned}
h_3(k,k,k) = \frac{1}{2\gamma_1\gamma_2(\gamma_1^2-\gamma_2^2)} \quad & \Big[\gamma_2\{y(k)-y(k+M+1)\} \\
& -\gamma_1^3\{y(k+2M+2)-y(k+3M+3)\}\Big]
\end{aligned}
\tag{2.29}
$$

Similarly, solving for $\frac{\partial J_d}{\partial h_0}=0$ yields:

$$
h_0 = \frac{1}{4M+4}\sum_{k=0}^{4M+3} y(k) - \frac{\gamma_1^2+\gamma_2^2}{2M+2}\sum_{k=1}^{M} h_2(k,k) \tag{2.30}
$$

And finally $\frac{\partial J_d}{\partial h_2} = 0$ returns:

$$\begin{aligned} h_0 &= \frac{\gamma_1^2}{2(\gamma_1^2+\gamma_2^2)}\{y(k)+y(k+M+1)\} \\ &+\frac{\gamma_2^2}{2(\gamma_1^2+\gamma_2^2)}\{y(k+2M+2)+y(k+3M+3)\} \\ &-\frac{\gamma_1^4+\gamma_2^4}{\gamma_1^2+\gamma_2^2}h_2(k,k) \end{aligned} \tag{2.31}$$

Solving Equations (2.30) and (2.31) simultaneously, the bias term and the second-order diagonal kernel estimators are obtained. These are as follows:

$$\begin{aligned} h_0 &= \frac{n_0}{d_0} \\ n_0 &= \frac{\gamma_1^2}{2(\gamma_1^4+\gamma_2^4)}\sum_{k=1}^{M}\{y(k)+y(k+M+1)\} \\ &+\frac{\gamma_2^2}{2(\gamma_1^4+\gamma_2^4)}\sum_{k=1}^{M}\{y(k+2M+2)+y(k+3M+3)\} \\ &-\frac{1}{2(\gamma_1^2+\gamma_2^2)}\sum_{k=0}^{4M+3}y(k) \\ d_0 &= M\frac{\gamma_1^2+\gamma_2^2}{\gamma_1^4+\gamma_2^4}-\frac{2M+2}{\gamma_1^2+\gamma_2^2} \end{aligned} \tag{2.32}$$

$$\begin{aligned} h_2(k,k) &= \frac{\gamma_1^2}{2(\gamma_1^4+\gamma_2^4)}\{y(k)+y(k+M+1)\} \\ &+\frac{\gamma_2^2}{2(\gamma_1^4+\gamma_2^4)}\{y(k+2M+2)+y(k+3M+3)\} \\ &-\frac{\gamma_1^2+\gamma_2^2}{\gamma_1^4+\gamma_2^4}h_0 \end{aligned} \tag{2.33}$$

2.2.4 Sub-diagonal Identification

Once the linear and nonlinear diagonal kernels are identified, an ideal input sequence to identify S_3 would be one that does not excite O_2 and O_3 . If an input sequence is designed such that no more than two points within the model memory, M, have non-zero values, then the third-order off-diagonal terms would be zero identically (according to (2.7)). Furthermore, the ratio of the leading PEV coefficients in (2.11) for the third-order sub-diagonal term to that of the second-order off-diagonal term is:

$$\frac{C_{S_3}}{C_{O_2}} = 4.5(\kappa + 2)\sigma_u^2 \tag{2.34}$$

If the value of this coefficient is large, then the implication is that modeling errors in the third-order sub-diagonal term dominate the second-order off-diagonal term contributions, thereby aiding the identification of S_3. A sequence with a high kurtosis and a high variance could be a candidate sequence to achieve this result. One such CSRS input sequence, which is also plant-friendly ($F_f = 94\%$), is shown in Figure 12.

A short length of the sequence is also shown in order to highlight the signal design. For this sequence, the ratio given by (2.34) is $11,360$. An increase in the coefficient in (2.34) causes the S_3 contribution to dominate, leading to a decrease in error in S_3 according to (2.11). This results in O_2 PEV calculations being weighted comparatively less heavily. Thus, for a value of $11,360$ the O_2 contribution is weighed 10^4 times less than the S_3 contribution, so that the latter dominates. In addition, the O_2 kernel is identified (see Section 2.2.4.1) before the S_3 identification, and the O_2 contributions are removed by pre-whitening. For the same reasons as for the linear and nonlinear diagonal kernels, *i.e.* to ensure sufficient excitation and non-singularity of the estimator equations (denominator term in (2.42) and (2.43) derived below), the pulse-heights were selected as ± 44.8 and ± 56.

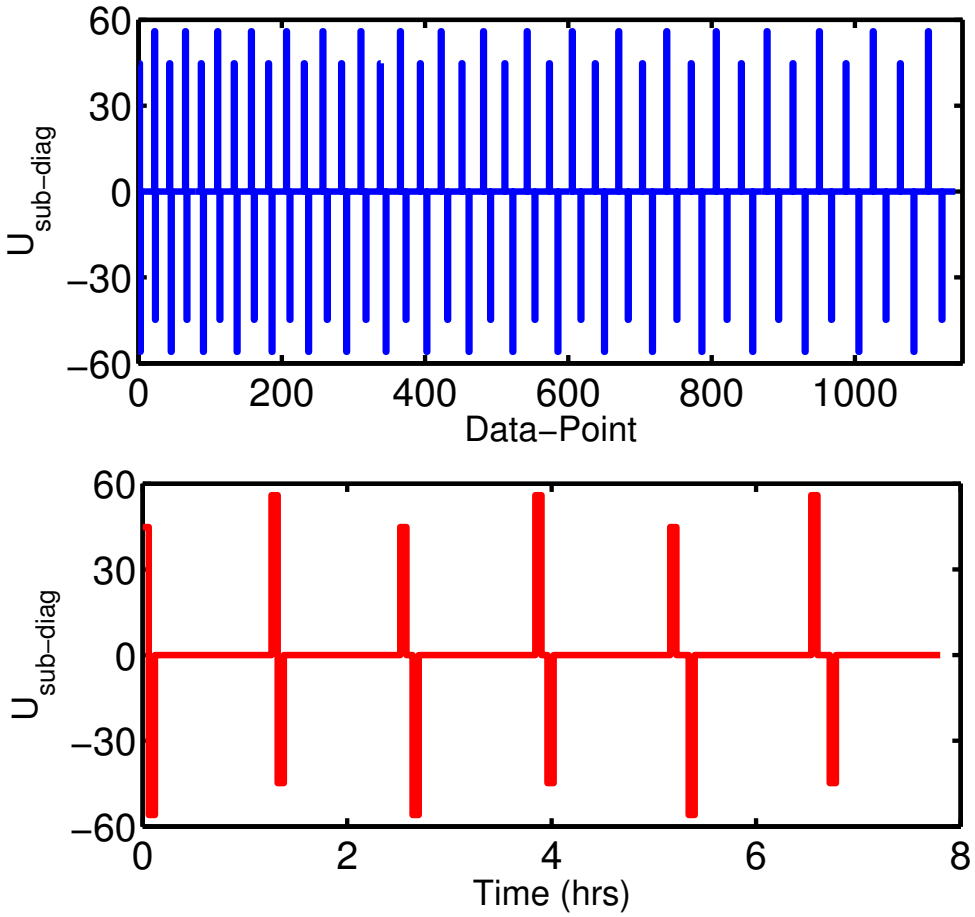

Figure 12: Input sequence used for sub-diagonal identification. The levels used are $\lambda_1 = 44.8$ and $\lambda_2 = -56$, and $M = 20$. Bottom: Zoom of the plot over the first 8 hours to show increasing pulse separation with time.

The first two sub-units of the input sequence used for sub-diagonal identification are given as:

$$\left.\begin{matrix} u_{sd1}(k) \\ \hline u_{sd2}(k) \end{matrix}\right\} = \begin{cases} \lambda_1 & k = 1 \\ \lambda_2 & k = 2 \\ 0 & 3 \leq k \leq M+1 \\ -\lambda_2 & k = M+2 \\ -\lambda_1 & k = M+3 \\ 0 & M+4 \leq k \leq 2M+1 \\ \hline \lambda_1 & k = 2M+2 \\ 0 & k = 2M+3 \\ \lambda_2 & k = 2M+4 \\ 0 & 2M+5 \leq k \leq 3M+3 \\ -\lambda_2 & k = 3M+4 \\ 0 & k = 3M+5 \\ -\lambda_1 & k = 3M+6 \\ 0 & 3M+7 \leq k \leq 4M+5 \end{cases} \tag{2.35}$$

In subsequent sub-units of the input sequence, there is a gap between the two pulses λ_1 and λ_2. This gap length increases by one with increasing sub-units, and it is employed to ensure the tailored excitation of all of the sub-diagonal parameters. The second sub-unit of (2.35) is shown in order to highlight the gap between two successive pulses. Furthermore, the presence of zeros of length M between each successive set of two pulses ensures that the contribution of the third-order off-diagonal term is zero identically.

Removing the bias, linear, nonlinear diagonal, and second-order off-diagonal kernel contributions from the output signal recovers the residual output $z(k)$. The $\hat{z}(k)$ corresponding to the first sub-unit of the input sequence is as follows:

$$\hat{z}_1(k) = \begin{cases} 0 & k = 1 \\ 0 & k = 2 \\ h_3(k-2,k-1,k-1)\lambda_1^2\lambda_2 & 3 \leq k \leq M+1 \\ +h_3(k-2,k-2,k-1)\lambda_1\lambda_2^2 & \\ 0 & k = M+2 \\ 0 & k = M+3 \\ -h_3(k-M-3,k-M-2,k-M-2)\lambda_1\lambda_2^2 & M+4 \leq k \leq 2M+2 \\ -h_3(k-M-3,k-M-3,k-M-2)\lambda_1^2\lambda_2 & \end{cases} \tag{2.36}$$

The estimates for the third-order sub-diagonal coefficients can be obtained by minimizing the following sum-squared prediction error:

$$J_{sd} = \sum_{k=0}^{2M+1} \{z_1(k) - \hat{z}_1(k)\}^2 \tag{2.37}$$

In order to obtain the coefficients for S_3, the following equations need to be satisfied:

$$\frac{\partial J_{sd}}{\partial h_3(k,k+1,k+1)} = 0 \tag{2.38}$$

$$\frac{\partial J_{sd}}{\partial h_3(k,k,k+1)} = 0 \tag{2.39}$$

$$k = 1,\ 2,\ \ldots M-1$$

Applying the conditions in Equations (2.38) and (2.39) yields:

$$\begin{aligned} (\Lambda_1^2 + \Lambda_2^2)h_3(k,k+1,k+1) + & \\ (2\Lambda_1\Lambda_2)h_3(k,k,k+1) &= \Lambda_1 z(k+1) - \Lambda_2 z(k+M+2) \qquad (2.40) \\ (2\Lambda_1\Lambda_2)h_3(k,k+1,k+1) + & \\ (\Lambda_1^2 + \Lambda_2^2)h_3(k,k,k+1) &= \Lambda_2 z(k+1) - \Lambda_1 z(k+M+2) \qquad (2.41) \\ \Lambda_1 &= \lambda_1^2\lambda_2 \\ \Lambda_2 &= \lambda_2^2\lambda_1 \\ k &= 1,\ 2,\ \ldots M-1 \end{aligned}$$

Solving Equations (2.40) and (2.41) simultaneously, the estimators for the third-order sub-diagonal kernel are as calculated as follows:

$$\begin{aligned} h_3(k,k+1,k+1) &= \frac{\Lambda_1}{\Lambda_1^2-\Lambda_2^2}z(k+1)+\frac{\Lambda_1}{\Lambda_1^2-\Lambda_2^2}z(k+M+2) && (2.42)\\ h_3(k,k,k+1) &= \frac{-\Lambda_2}{\Lambda_1^2-\Lambda_2^2}z(k+1)+\frac{-\Lambda_1}{\Lambda_1^2-\Lambda_2^2}z(k+M+2) && (2.43)\\ k &= 1,\ 2,\ \ldots M-1 \end{aligned}$$

This calculation can be generalized for the entire length of the S_3 sequence as follows:

$$\begin{aligned} \begin{bmatrix} z(2+2(q-1)(M+1)+(q-1)^2+p) \\ z(2+2(q-1)(M+1)+(q-1)^2+M+q+p) \end{bmatrix} &= \begin{bmatrix} \Lambda_1 & \Lambda_2 \\ -\Lambda_2 & -\Lambda_1 \end{bmatrix}\begin{bmatrix} h_3(p,p+q,p+q) \\ h_3(p,p,p+q) \end{bmatrix} \\ z_{Sq,p} &= u_S h_{Sq,p} \\ Z_{Sq} &= \left[z_{Sq,1}^T \cdots z_{Sq,M-q}^T\right]^T = U_S H_{Sq} \\ U_S &= \text{block}\{u_S\} \\ H_{Sq} &= \left[h_{Sq,1}^T \cdots h_{Sq,M-q}^T\right]^T \end{aligned} \quad (2.44)$$

In this equation, the sub-script S is used to denote the sub-diagonal kernel, q represents the sub-sequence number and p counts along the length of the q^{th} sub-diagonal.

$$\begin{aligned} \mathcal{Z}_S &= \left[Z_{S1}^T \cdots Z_{S(M-1)}^T\right]^T = \mathcal{U}_S\mathcal{H}_S && (2.45)\\ \mathcal{U}_S &= \text{block}\{U_S\} \\ \mathcal{H}_S &= \left[H_{S1}^T \cdots H_{S(M-1)}^T\right]^T \end{aligned}$$

In this equation $\mathcal{Z}_S$ is a 380×1 vector of output data points. $\mathcal{U}_S$ is a 380×380 block-diagonal matrix, with U_S along the main diagonal. The column vector $\mathcal{H}_S$ represents the sub-diagonal coefficients for the entire sequence. The solution reduces to the relaxed least-squares problem:

$$\mathcal{H}_S = (\mathcal{U}_S^T\mathcal{U}_S)^{-1}\mathcal{U}_S^T\mathcal{Z}_S \quad (2.46)$$

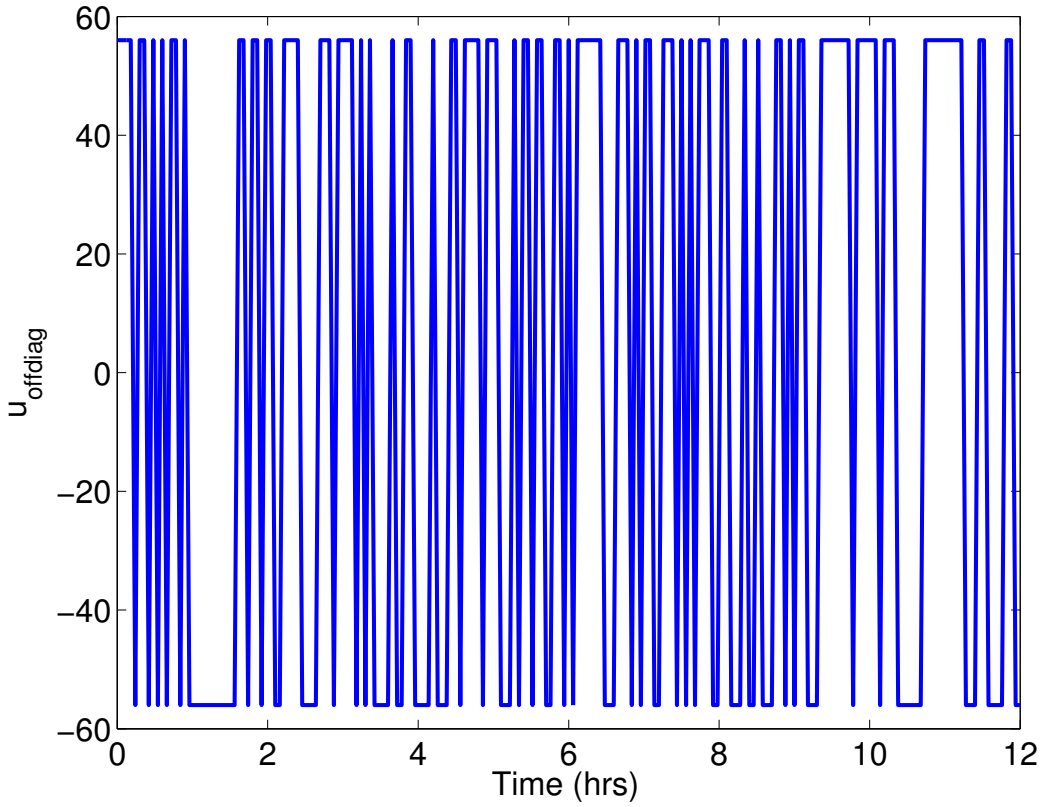

Figure 13: First 12 hours of the sequence used for the off-diagonal identification. The input levels used are ± 56

2.2.4.1 Off-diagonal Identification

Consider the input sequence given in Figure 13. This is a random binary sequence (RBS) which has a theoretical kurtosis of -2. From Equation (2.11), if an RBS with a high input variance is used, the off-diagonal terms dominate. Using this sequence and the cross-correlation technique, the off-diagonal kernels can be estimated [30]. Cross-correlation is essentially a statistical technique which utilizes the correlation between the input and output of a system to identify the Volterra kernels. For a detailed description of this technique, the interested reader is referred to [42, 43, 45].

For the second-order off-diagonal kernel identification, pre-whitening was used to eliminate the contributions of the bias, linear, and second-order diagonal kernel. The $\frac{M(M-1)}{2} = 190$ off-diagonal coefficients were identified using a 1900 point random binary sequence (RBS). Finally, for the third-order off-diagonal kernel, pre-whitening was used to eliminate the contributions of all the kernels previously identified. The $\frac{M(M-1)(M-2)}{6} = 1,140$ third-order off-diagonal coefficients were identified using a $11,400$ point RBS.

2.2.5 Identification using Cross-correlation

As discussed previously, an alternate approach for calculating Volterra kernels is cross-correlation. Since an $N+1$-level input sequence (at a minimum) is required to identify a Volterra model of order N by cross-correlation [28], five-level sequences were used to identify the linear and nonlinear diagonal kernels, and a four-level sequence was used for the sub-diagonal kernels. A very long four-level sequence could also be used, but different sequences for the linear and nonlinear diagonal, and sub-diagonal kernel enable a comparison with [34]. The cross-correlation approach is statistically based, and the sequence length was chosen so that there were 10 input-output data points per identified coefficient, in order to reduce the variance of the coefficient estimates. Since there are $NM = 60$ unique linear and nonlinear diagonal coefficients (for $N = 3$ and $M = 20$) a 600 point five-level sequence was used. The

mathematical structure of this sequence is given as:

$$u_{ccld}(k) = \begin{cases} 56 & \text{with probability } \frac{1}{10} \\ 11.2 & \text{with probability } \frac{2.5}{10} \\ 0 & \text{with probability } \frac{3}{10} \\ -11.2 & \text{with probability } \frac{2.5}{10} \\ -56 & \text{with probability } \frac{1}{10} \end{cases} \tag{2.47}$$

This sequence has a friendliness factor of 24%. Similarly, for the $M^2 - M = 380$ unique sub-diagonal kernel coefficients, a 3800 point random quaternary sequence (RQS) was employed. This sequence had a friendliness factor of 45%, and its mathematical structure is given as:

$$u_{ccsd}(k) = \begin{cases} 56 & \text{with probability } \frac{3}{100} \\ 11.2 & \text{with probability } \frac{47}{100} \\ -11.2 & \text{with probability } \frac{47}{100} \\ -56 & \text{with probability } \frac{3}{100} \end{cases} \tag{2.48}$$

For comparison purposes, the probability distribution of the sequences in (2.47) and (2.48) are the same as in [34]. The second- and third-order off-diagonal kernels were identified as in Section 2.2.4.1.

2.2.6 Tailored Reduced Length Sequences for Kernel Identification

The length of the input sequence used for the sub-diagonal identification in Section 2.2.4 is given as:

$$\begin{aligned} & 2M^2 - 2 + \sum_{\Xi=0}^{M-2} 2\Xi \\ = \; & 3M(M-1) \end{aligned} \tag{2.49}$$

In this equation, Ξ is the gap between any set of two pulses. In many cases, the value of M may be high so that a substantial amount of data is required to estimate the third-order sub-diagonal kernel. On the contrary, short input sequences are plant-friendly as they lead to a reduction in the duration for which the plant operates in identification mode, thereby reducing off-specification product. This motivates the design of a reduced length input-sequence for third-order sub-diagonal kernel identification.

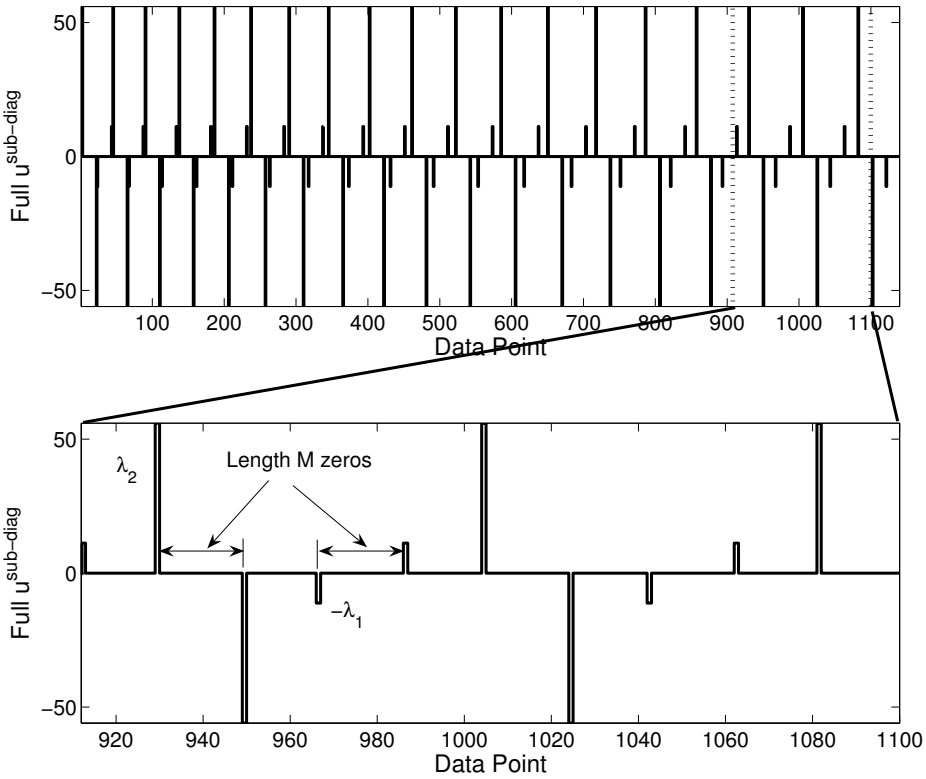

Figure 14: Full length input sequence used for sub-diagonal identification. Bottom: zoom of the plot to show pulses λ_1 and λ_2 and M length zeros. The levels used are $\lambda_1 = 11.2$ and $\lambda_2 = 56$, and $M = 20$

2.2.6.1 Sub-diagonal Identification The main reason for the large number of data-points in the sub-diagonal sequence in Section 2.2.4 was the presence of M zeros after every second pulse. This can be seen in Figure 14 where the top shows the full sequence, whereas the bottom portion shows the sequence between the data-points 900 to 1100. The M-length zeros after every second pulse are also shown. While the M-length zeros effectively eliminated interactions between all but the intended pulses, they add significantly to the sequence length without adding input excitation.

The structural basis of the reduced-length sequence (Figure 15) is similar to the one in Section 2.2.4; the reduced-length sequence also involves two pulses, and the gap between them successively increases. However, the length of each sub-unit is fixed so that the length of zeros after every second pulse actually decreased. Consider the first two sub-units of the reduced-length input sequence:

$$u_{redsd1}(k) = \begin{cases} \beta_1 & k = 1 \\ \beta_2 & k = 2 \\ 0 & 3 \leq k \leq M \\ -\beta_2 & k = M+1 \\ -\beta_1 & k = M+2 \\ 0 & M+3 \leq k \leq 2M \\ \beta_2 & k = 2M+1 \\ \beta_1 & k = 2M+2 \\ 0 & 2M+3 \leq k \leq 3M \\ -\beta_1 & k = 3M+1 \\ -\beta_2 & k = 3M+2 \\ 0 & 3M+3 \leq k \leq 4M \end{cases} \quad u_{redsd2}(k) = \begin{cases} \beta_1 & k = 4M+1 \\ 0 & k = 4M+2 \\ \beta_2 & k = 4M+3 \\ 0 & 4M+4 \leq k \leq 5M \\ -\beta_2 & k = 5M+1 \\ 0 & k = 5M+2 \\ -\beta_1 & k = 5M+3 \\ 0 & 5M+4 \leq k \leq 6M \\ \beta_2 & k = 6M+1 \\ 0 & k = 6M+2 \\ \beta_1 & k = 6M+3 \\ 0 & 6M+4 \leq k \leq 7M \\ -\beta_1 & k = 7M+1 \\ 0 & k = 7M+2 \\ -\beta_2 & k = 7M+3 \\ 0 & 7M+4 \leq k \leq 8M \end{cases} \tag{2.50}$$

The second sub-unit, $u_{redsd2}(k)$ is shown to highlight the gap between the pulses. For the first sub-unit of the input sequence, the input data from points $2M+1$ to $4M$ are the same

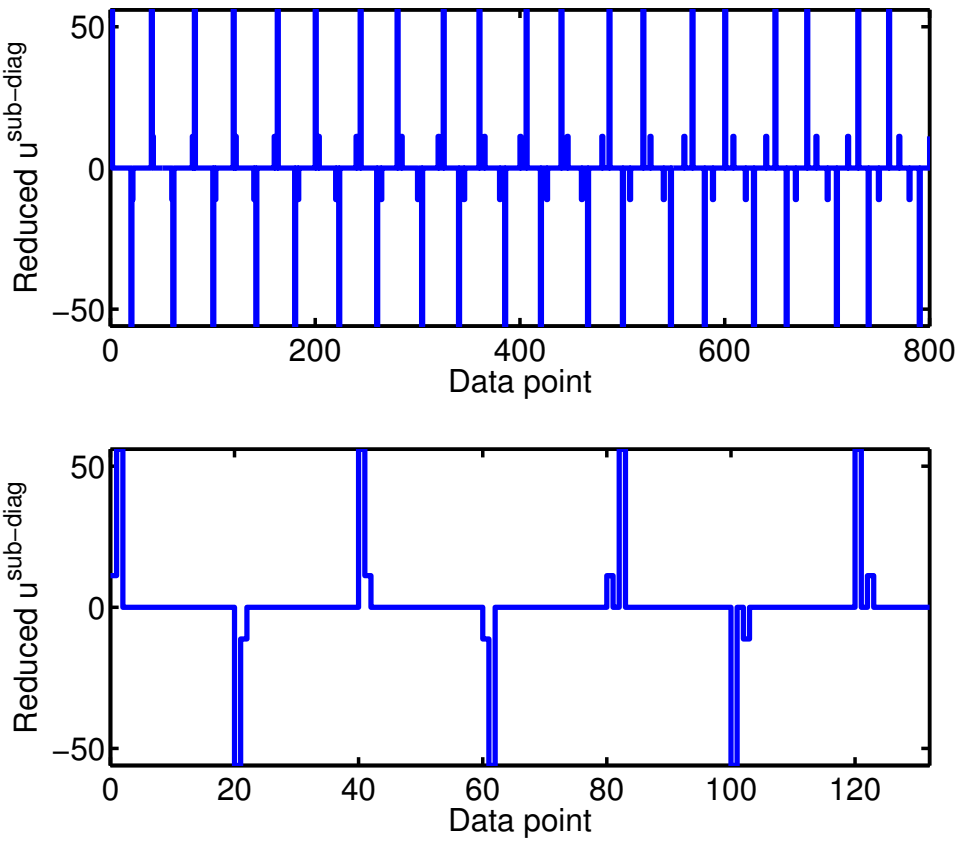

Figure 15: Reduced length input sequence used for sub-diagonal identification. Bottom: zoom of the plot over the first two sub-units to show increasing pulse separation. The levels used are $\beta_1 = 11.2$ and $\beta_2 = 56$.

as that from 1 to $2M$, with the exception that the order of the pulses is interchanged; *i.e.,* β_2 forms the outer, and β_1 the inner, pulses. This was done to avoid singularities in the solution of the estimator equations, *i.e.,* non-singularity of u_a in (2.51). Furthermore, when only the first $2M$ points were used as a sub-unit, the resulting sub-diagonal kernels were poorly estimated owing to the interchanging order of pulses for each sub-unit. The addition of the second $2M$ points averages out the contribution of both pulse orders, *i.e.,* (β_1, β_2) and (β_2, β_1), resulting in smooth estimates for the sub-diagonal kernel. For the next sub-unit there is a gap of one point between the two pulses β_1 and β_2 (or β_2 and β_1). This gap length successively increases by one, until the gap length is of $(\frac{M}{2} - 1)$ points. Since the length of each sub-unit is fixed at $4M$, an increase in the gap-length does not change the sequence length. This results in $\frac{M}{2}$ sub-units, each of length $4M$ so that the total length of this sequence is $2M^2$. This represents a savings of $-2 + \sum_{\Xi=0}^{M-2} 2\Xi$ data-points over the full length sub-diagonal sequence.

To calculate the estimator equations for the reduced-length sequence, consider Figure 16. From the figure, two distinct regions, a and b, can be identified. The region a (length $M-q$) corresponds to the output data-set used to estimate the $h_3(p, p+q, p+q)$ and $h_3(p, p, p+q)$ kernel coefficients for $q = 1, 2, ..., \frac{M}{2}$ and $p = 1, 2, .., M - q$. The region b (length $M - i$) data are used to estimate the $h_3(j, i + j, i + j)$ and $h_3(j, j, i + j)$ kernel coefficients for $i = M - 1, M - 2, ..., \frac{M}{2} + 1$ and $j = 1, 2, ..., M - i$. Note that the first two points of the subsequent sub-unit, *e.g.,* $4M + 1$ and $4M + 2$, are required for estimation of the kernels from a given sub-unit, *e.g.,* sub-unit 1. For each subsequent sub-unit, the length of region a decreases by 1, whereas that of region b increases by 1, until the $\frac{M}{2}$ sub-unit where they have the same length. The kernel structure can be visualized as shown in Figure 17. For the first sub-unit, region a is the hypotenuse and region b is the opposite vertex. For each successive sub-unit, the regions move towards the center, until they finally converge for the $\frac{M}{2}$ sub-unit, *i.e.,* $a = b$.

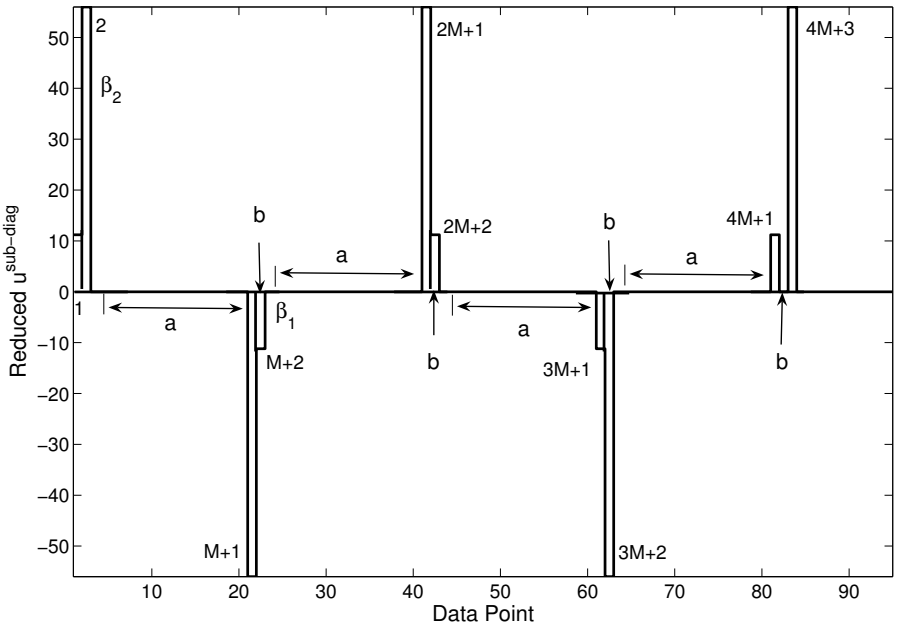

Figure 16: A schematic of the first $4M + 10$ points of the reduced sequence. $\beta_1 = 11.2$ and $\beta_2 = 56$, and $M = 20$.

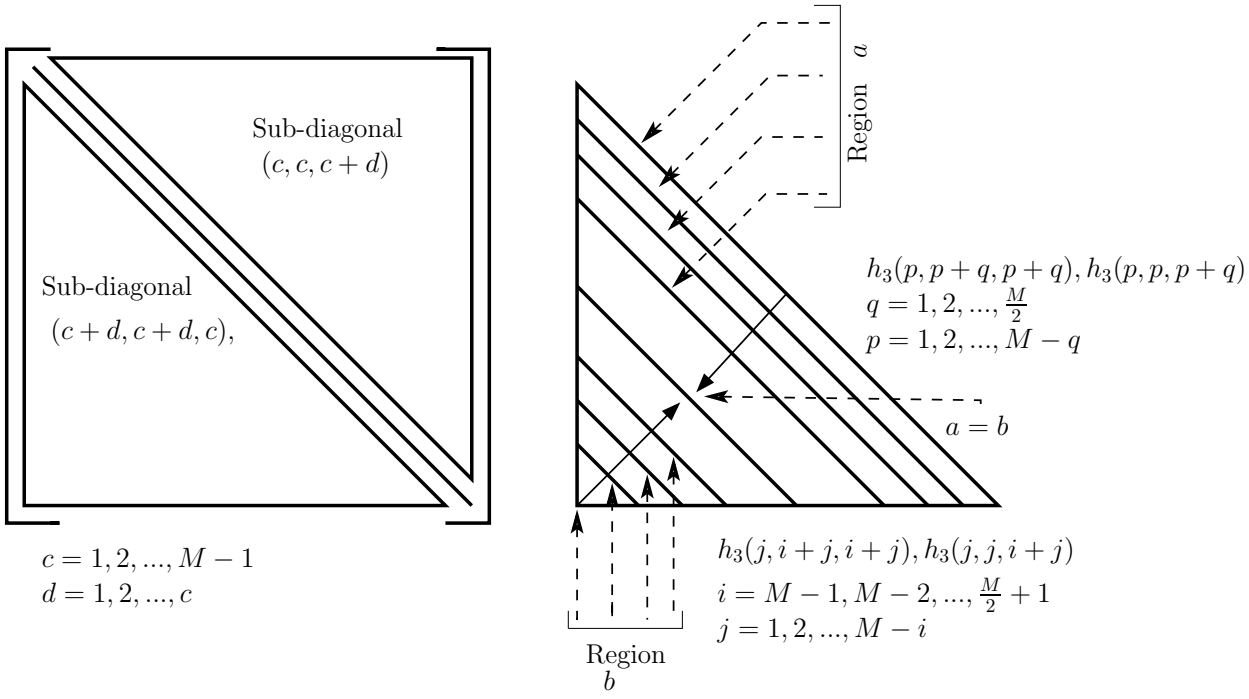

Figure 17: Schematic of the sub-diagonal kernel coefficient calculation based on the reduced-length sub-diagonal sequence.

For the a regions of the sequence:

$$\begin{bmatrix} y(q+1+4M(q-1)+p) \\ y(q+1+4M(q-1)+p+M) \\ y(q+1+4M(q-1)+p+2M) \\ y(q+1+4M(q-1)+p+3M) \end{bmatrix} = \begin{bmatrix} \beta_1^2\beta_2 & \beta_2^2\beta_1 \\ -\beta_2^2\beta_1 & -\beta_1^2\beta_2 \\ \beta_2^2\beta_1 & \beta_1^2\beta_2 \\ -\beta_1^2\beta_2 & -\beta_2^2\beta_1 \end{bmatrix} \begin{bmatrix} h_3(p,p+q,p+q) \\ h_3(p,p,p+q) \end{bmatrix}$$

$$\begin{aligned} y_{q,p} &= u_a h_{q,p} \\ Y_q &= \left[y_{q,1}^T \cdots y_{q,M-q}^T\right]^T = U_a H_{a,q} \\ U_a &= \text{block}\{u_a\} \\ H_{a,q} &= \left[h_{q,1}^T \cdots h_{q,M-q}^T\right]^T \end{aligned} \tag{2.51}$$

In this equation, q represents the sub-sequence number and p counts along the length of the q^{th} sub-diagonal.

$$\begin{aligned} \mathcal{Y}_{redsd} &= \left[Y_1^T \cdots Y_{M-\frac{M}{2}}^T\right]^T = \mathcal{U}_a \mathcal{H}_a \\ \mathcal{U}_a &= \text{block}\{U_a\} \\ \mathcal{H}_a &= \left[H_{a,1}^T \cdots H_{a,M-\frac{M}{2}}^T\right]^T \end{aligned} \tag{2.52}$$

In this equation $\mathcal{Y}_{redsd}$ is a 580×1 column vector of output data points. $\mathcal{U}_a$ is a 580×290 non-square block-diagonal matrix, with U_a along the main diagonal. The 290×1 column vector $\mathcal{H}_a$ represents the sub-diagonal coefficients for the a regions. The solution reduces to the relaxed least-squares problem:

$$\mathcal{H}_a = (\mathcal{U}_a^T \mathcal{U}_a)^{-1} \mathcal{U}_a^T \mathcal{Y}_{redsd} \tag{2.53}$$

The calculation for the kernel contributions corresponding to the b regions for the entire sequence proceeds in a similar manner. The main differences are that the equivalent of q is $i = (M-1) : (\frac{M}{2}+1)$ and u_b (replacing u_a) is given as:

$$u_b = \begin{bmatrix} -\beta_2^3 & \beta_2^3 \\ \beta_1^2\beta_2 & -\beta_2^2\beta_1 \\ -\beta_1^3 & \beta_1^3 \\ \beta_2^2\beta_1 & -\beta_1^2\beta_2 \end{bmatrix}$$

The rest of the calculations proceed in a similar manner as follows:

$$\begin{aligned} w_{i,j} &= u_b h_{i,j} \\ W_i &= \left[w_{i,M-1}^T \cdots w_{i,\frac{M}{2}+1}^T\right]^T = U_b H_{b,i} \\ U_b &= \text{block}\{u_b\} \\ H_{b,i} &= \left[h_{i,M-1}^T \cdots h_{i,\frac{M}{2}+1}^T\right]^T \\ \mathcal{W} &= \left[W_{M-1}^T \cdots W_{\frac{M}{2}+1}^T\right]^T = \mathcal{U}_b \mathcal{H}_b \end{aligned} \tag{2.54}$$

Here, $\mathcal{W}$ (180×1), $\mathcal{U}_b$ (180×90), and $\mathcal{H}_b$ (90×1) are equivalent to $\mathcal{Y}, \mathcal{U}_a,$ and $\mathcal{H}_a$ in region a respectively.

The equations thus derived are equivalent to those obtained by starting from the PEV expression and minimizing the sum-squared prediction error given by:

$$J_{red-sd} = \sum_{k=1}^{2M^2} \{z_{red-sd}(k) - \hat{z}_{red-sd}(k)\}^2 \tag{2.55}$$

In this equation, $\hat{z}_{red-sd}(k)$ is the model output prediction and $z_{red-sd}(k)$ is the residual output from the plant, in response to the reduced-length input sequence after removing the contribution of the bias, linear, nonlinear diagonal kernels, and second-order off-diagonal kernels. The details of the derivation are presented in Appendix B.

2.2.6.2 Simultaneous Identification of O_2 and S_3 Since the sequence designed for identifying the sub-diagonal kernel also excites the second-order off-diagonal kernel, it can also be used to identify the second-order off-diagonal kernel. The advantage of this approach over the cross-correlation method used in Section 2.2.4.1 is that explicit estimator equations can be derived for the O_2 contribution. The sequence is the same as in Figure 15 and the length is now 74% shorter than the combined lengths of the sequences used for S_3 and O_2 estimation in Sections 2.2.4 and 2.2.4.1, for $M = 20$. Note that since the O_2 kernel

is symmetric about the diagonal, only the $h_2(k,k+1),\ k=1,2,\cdots,M-1$ needs to be identified. The generalized equation for the complete a region, similar to (2.51), is given as:

$$\begin{bmatrix} y(q+1+4M(q-1)+p) \\ y(q+1+4M(q-1)+p+M) \\ y(q+1+4M(q-1)+p+2M) \\ y(q+1+4M(q-1)+p+3M) \end{bmatrix} = \begin{bmatrix} \beta_1^2\beta_2 & \beta_2^2\beta_1 & \beta_1\beta_2 \\ -\beta_2^2\beta_1 & -\beta_1^2\beta_2 & \beta_1\beta_2 \\ \beta_2^2\beta_1 & \beta_1^2\beta_2 & \beta_1\beta_2 \\ -\beta_1^2\beta_2 & -\beta_2^2\beta_1 & \beta_1\beta_2 \end{bmatrix} \begin{bmatrix} h_3(p,p+q,p+q) \\ h_3(p,p,p+q) \\ h_2(p,p+q) \end{bmatrix}$$

$$\begin{aligned} y_{OSq,p} &= u_{OSa} h_{OSq,p} \\ Y_{OSq} &= \left[y_{OSq,1}^T \cdots y_{OSq,M-q}^T\right]^T = U_{OSa} H_{OSa,q} \\ U_{OSa} &= \text{block}\{u_{OSa}\} \\ H_{OSa,q} &= \left[h_{OSq,1}^T \cdots h_{OSq,M-q}^T\right]^T \\ \mathcal{Y}_{OSa} &= \left[Y_{OS1}^T \cdots Y_{OS(M-\frac{M}{2})}^T\right]^T = \mathcal{U}_{OSa}\mathcal{H}_{OSa} \end{aligned} \tag{2.56}$$

In this equation the sub-script OS signifies that the S_3 and the O_2 kernels are identified simultaneously, $\mathcal{U}_{OSa}$ is a non-square block-diagonal matrix with U_{OSa} along the main diagonal. The column vector $\mathcal{H}_{OSa}$ represents the sub-diagonal and the second-order off-diagonal coefficients for the a regions. As in (2.53), the solution for the kernel coefficients is given as:

$$\mathcal{H}_{OSa} = (\mathcal{U}_{OSa}^T \mathcal{U}_{OSa})^{-1} \mathcal{U}_{OSa}^T \mathcal{Y}_{OSa} \tag{2.57}$$

Similarly, the equivalent of u_{OSa} for the b regions of the sequence is given as:

$$u_{OSb} = \begin{bmatrix} -\beta_2^3 & \beta_2^3 & -\beta_2^2 \\ \beta_1^2\beta_2 & -\beta_2^2\beta_1 & -\beta_1^2 \\ -\beta_1^3 & \beta_1^3 & -\beta_2^2 \\ \beta_2^2\beta_1 & -\beta_1^2\beta_2 & -\beta_1^2 \end{bmatrix}$$

The rest of the calculations proceed in a manner similar to that for the a region.

2.3 IDENTIFICATION ALGORITHMS

The following algorithms are used to identify Volterra kernels using the tailored sequences, and from the cross-correlation approach.

Algorithm 1 (Tailored Sequence)

1. The bias, linear, and nonlinear diagonal parameters are estimated using a four-pulse, five-level, $4M + 4$ length (84 point) sequence and equations (2.28), (2.29), (2.32), and (2.33).
2. The second-order off-diagonal kernel is estimated using the cross-correlation technique described in Section 2.2.4.1 (as in [30]). Pre-whitening is used to eliminate the contributions of the bias, linear, and second-order diagonal kernel. The $\frac{M(M-1)}{2} = 190$ off-diagonal coefficients are identified using a 1900 point RBS.
3. A tailored sequence with length $3M(M-1)$ (1140 points) is used to excite the system, and residuals are calculated by subtracting off the bias, linear, nonlinear diagonal, and second-order off-diagonal contributions. Third-order sub-diagonal coefficients are estimated from residuals obtained by pre-whitening using equations (2.42) and (2.43).
4. The third-order off-diagonal terms are estimated using a cross-correlation technique. Pre-whitening is used to eliminate the contributions of all the kernels previously identified. The $\frac{M(M-1)(M-2)}{6} = 1140$ third-order off-diagonal coefficients are identified using a $11,400$ point RBS.

Algorithm 2 (Cross-correlation)

1. The estimation of bias, linear, and nonlinear diagonal kernels is carried out using a 600 point (length $10NM$) random five-level sequence.
2. The second-order off-diagonal kernel is estimated as in Algorithm 1, step 2.
3. For the $M(M-1)$ unique third-order sub-diagonal kernels, a 3800 point (length $10M(M-1)$) RQS is employed. Pre-whitening is used to eliminate the contributions of the bias, linear, and nonlinear diagonal kernels.
4. The third-order off-diagonal terms are estimated as in Algorithm 1, step 4.

Algorithm 3 (Data-efficient Tailored Sequence for S_3)

1. The bias, linear, and nonlinear diagonal parameters are estimated using a four-pulse, five-level, $4M + 4$ length (84 point) sequence as in Algorithm 1, step 1.
2. The second-order off-diagonal kernel is estimated using a cross-correlation technique as in Algorithm 1, step 2.
3. A tailored sequence with length $2M^2$ (800 points) is used to excite the system, and residuals are calculated by subtracting off the bias, linear, nonlinear diagonal, and second-order off-diagonal contributions. Third-order sub-diagonal coefficients are estimated from residuals obtained by pre-whitening.
4. The third-order off-diagonal terms are estimated using a cross-correlation technique as in Algorithm 1, step 4.

Algorithm 4 (Data-efficient Identification of O_2 and S_3)

1. The estimation of bias, linear, and nonlinear diagonal kernels is carried out as in Algorithm 1, step 1.
2. The sequence used for the identification of S_3 in Algorithm 3 (length $2M^2$, 800 points) is used to calculate the third-order sub-diagonal and the second-order off-diagonal kernels as described in Section 2.2.6.2. Pre-whitening is first carried out to eliminate the contributions of the bias, the linear, and the nonlinear diagonal kernels.
3. The third-order off-diagonal terms are estimated as in Algorithm 1, step 4.

2.4 RESULTS

2.4.1 Nominal Identification

Table 2 shows the sum-squared coefficient errors (SSE) between the theoretical kernels obtained from the Carlemann linearization and those calculated from the identification algorithms using the nonlinear system (2.1) as the process (data generator). Due to the statistical nature of random sequences, the SSE values for all kernels obtained from a cross-correlation calculation, are reported as a mean of 100 runs.

Table 2: SSE values between the analytically obtained and identified Volterra kernels, for the nominal study

Kernel	Algorithm 1	Algorithm 2	Algorithm 2*	Algorithm 3	Algorithm 4
Linear	3.55×10^{-6}	6.67×10^{-3}	–	–	–
2^{nd} Diag.	9.15×10^{-8}	7.74×10^{-6}	–	–	–
2^{nd} Off-diag.	9.37×10^{-8}	5.48×10^{-7}	4.55×10^{-7}	9.37×10^{-8}	8.74×10^{-8}
3^{rd} Diag.	1.27×10^{-10}	1.14×10^{-7}	–	–	–
3^{rd} Sub-diag.	9.00×10^{-10}	4.56×10^{-8}	1.30×10^{-9}	9.48×10^{-10}	9.48×10^{-10}
3^{rd} Off-diag.	7.84×10^{-11}	3.78×10^{-10}	3.67×10^{-10}	7.84×10^{-11}	7.85×10^{-11}

It is observed that the tailored sequence algorithms consistently outperform the cross-correlation approach (Algorithm 2) based on the low values of the squared coefficient errors. The bias term values (not shown) were close to zero for all of the algorithms as expected. As a "fairer"comparison, the fourth column (Algorithm 2*) represents the coefficient SSE when the linear and nonlinear diagonal kernels from Algorithm 1 are used in place of the cross-correlation kernel estimates for pre-whitening prior to subsequent coefficient estimations using cross-correlation.

The numbers in Table 3 represent the % improvement in estimates of the tailored algorithms versus Algorithm 2 and Algorithm 2*. Table 4 shows the reduction in data requirement for the tailored algorithms when compared with Algorithm 2. The `n.a.` in the last column for O_2, indicates that a separate identification sequence is not necessary. This is because of the simultaneous identification of S_3 and O_2 in Algorithm 4.

It is observed in Figure 18 that the tailored algorithms capture the first-order Volterra kernel behavior admirably. The statistical nature of cross-correlation is evident, and the estimates are scattered around the theoretical kernel behavior. A graphical comparison of the nonlinear diagonal kernels, along with the corresponding theoretical kernels, is shown

Table 3: % Improvement in SSE values

Kernel	A_1 *vs.* A_2	A_1 *vs.* $A_{2^\star}$	A_3 *vs.* $A_{2^\star}$	A_4 *vs.* $A_{2^\star}$
Linear	99	–	–	–
2^{nd} Diag.	99	–	–	–
2^{nd} Off-diag.	83	79	79	81
3^{rd} Diag.	99	–	–	–
3^{rd} Sub-diag.	98	31	28	28
3^{rd} Off-diag.	79	78	78	78

Table 4: Data Reduction when compared to Algorithm 2

Kernel	Algorithm 1	Algorithm 3	Algorithm 4
Linear	7×	7×	7×
2^{nd} Diag.	7×	7×	7×
2^{nd} Off-diag.	–	–	n.a.
3^{rd} Diag.	7×	7×	7×
3^{rd} Sub-diag.	3.3×	4.75×	4.75×
3^{rd} Off-diag.	–	–	–

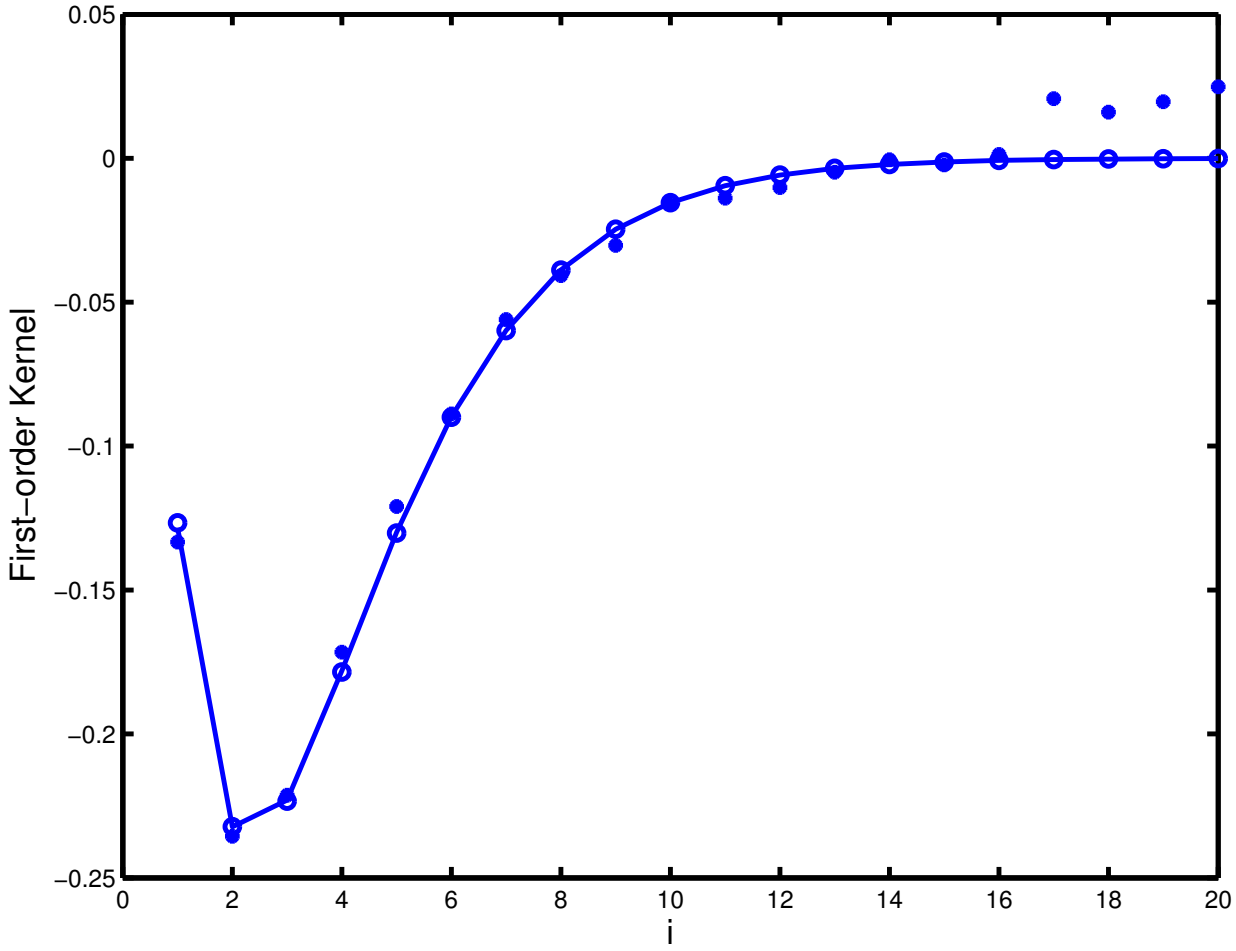

Figure 18: Comparison of the analytical (—), Algorithm 1 (o), and Algorithm 2 (•) first-order Volterra kernels for the nominal case

in Figure 19. The identified second-order kernel underpredicts slightly at the low indices. Similarly, the third-order diagonal kernel captures the nonlinear behavior qualitatively, *i.e.,* an initial dip followed by a return to zero. However the identified kernel does not adequately capture the magnitude of this dip. In both cases, the identified coefficients converge to the theoretical values after the first few coefficients.

For the sub-diagonal kernels, the data reduction is 3.3 times for Algorithm 1 versus Algorithm 2, and there is an improvement of 98% in the SSE value of Algorithm 1 over Algorithm 2. The SSE values from all the tailored algorithms are nearly identical. It is interesting to note that Algorithm 1 shows an improvement of 31% in the value of the coefficient SSE over Algorithm 2*. This improvement is strictly due to sequence tailoring, as both Algorithms use the same linear and nonlinear diagonal kernel estimates. The only difference lies in the input sequences for S_3 identification, thereby emphasizing the importance of input sequence design in the quality of the identified Volterra model estimates.

At first glance, one would expect the SSE values for O_2 from Algorithms 1 and 3, and Algorithm 2* to be similar as they are all identified from the same input sequences. The difference arises because pre-whitening in Algorithms 1 and 3 includes a contribution from the third-order diagonal term, unlike in Algorithm 2*, where the kernels are identified strictly on the basis of Volterra model order. Furthermore, since the O_2 kernel is identified explicitly in Algorithm 4, its SSE value is the lowest amongst all algorithms. The SSE values for the O_3 kernel for all the tailored identification algorithms are almost identical. Overall, the tailored algorithms and Algorithm 2* all outperform Algorithm 2. This highlights the importance of the linear and nonlinear diagonal kernels, in addition to tailored sequence design, in estimating subsequent kernels.

One reason for this quantitative difference between the analytically calculated and the identified Volterra kernels is that the analytically calculated Volterra kernels are obtained from the discretized Carlemann linearized model, which itself, is a third-order approximation of the actual nonlinear system (2.1). The quantitative differences are not due to the estimator equation calculations, but instead are due to the difference between the nonlinear system and the "theoretical" Volterra kernel outputs to the input sequence (2.22). Thus, these errors are an artifact of the case study that has been used. The polymerization reactor is

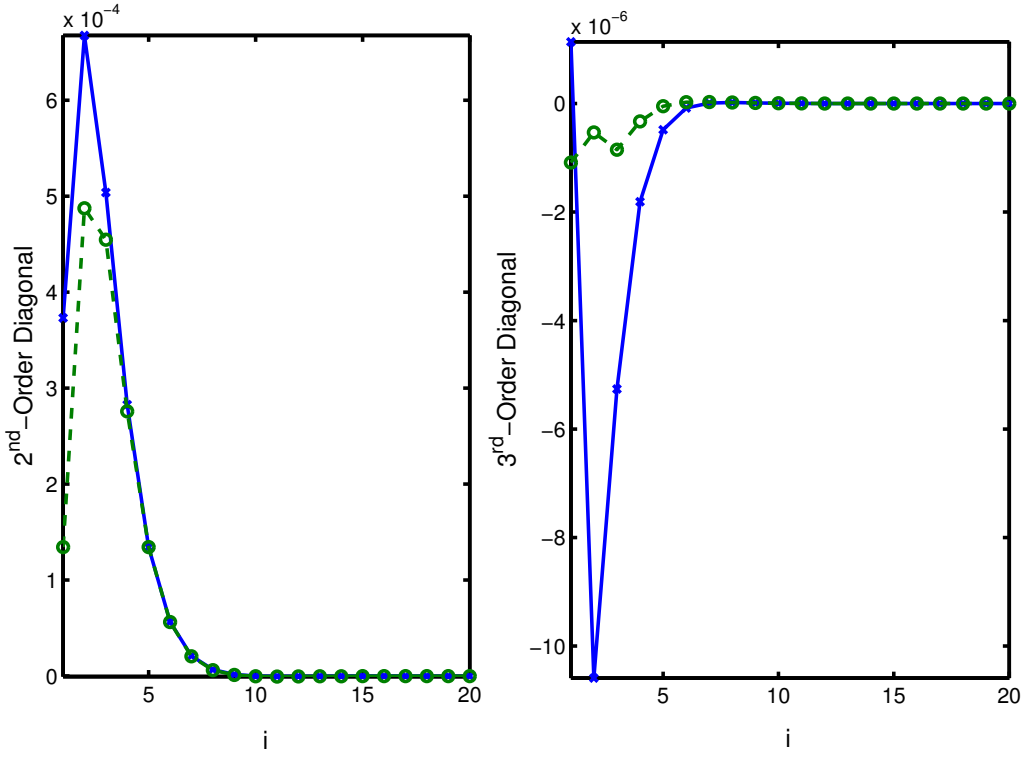

Figure 19: Comparison of the analytically calculated (-×-) and identified (Algorithm 1, $-\circ-$) second-order (left) and third-order (right) diagonal kernels

Table 5: A comparison of the SSE values between the identified and analytically calculated kernels

Kernel	Algorithm 3	Algorithm 4
Linear	1.95×10^{-30}	1.95×10^{-30}
D_2.	3.38×10^{-17}	3.38×10^{-17}
O_2	3.08×10^{-9}	5.43×10^{-10}
D_3	5.20×10^{-39}	5.20×10^{-39}
S_3	5.93×10^{-14}	4.87×10^{-14}
O_3	1.69×10^{-13}	1.68×10^{-13}

not governed by a true third-order Volterra model. The algorithm can identify the kernel coefficients exactly for systems that are governed by a third-order Volterra model. In the light of these facts, and to highlight the capabilities of the algorithm, subsequent analyses employ the analytically calculated Volterra kernels as the process data generator. Since all the tailored algorithms outperform the cross-correlation approach, all further analysis is reported for Algorithms 3 and 4, as they require the least amount of data of the tailored identification algorithms.

The nominal identification results are once again reported in Table 5 as sum squared errors between the identified and the analytically calculated Volterra kernels. Algorithms 3 and 4, effectively captured the complete linear and nonlinear diagonal kernel behavior, as is seen from the low SSE values. Since the O_2 kernel in Algorithm 4 was calculated explicitly, the SSE values are superior to those for Algorithm 3, where the O_2 kernel was calculated via cross-correlation. A comparison of the O_2 kernels is shown in Figure 20. Furthermore, since the O_2 kernel in Algorithm 4 is calculated explicitly, *i.e.*, without pre-whitening, there is no compounding of pre-whitening effects for subsequent kernel identification.

The third-order sub-diagonal kernel can be visualized as a two dimensional matrix by assigning the $h_3(i, j, j)$ coefficients to the (i, j), and the $h_3(i, i, j)$ coefficients to the (j, i),

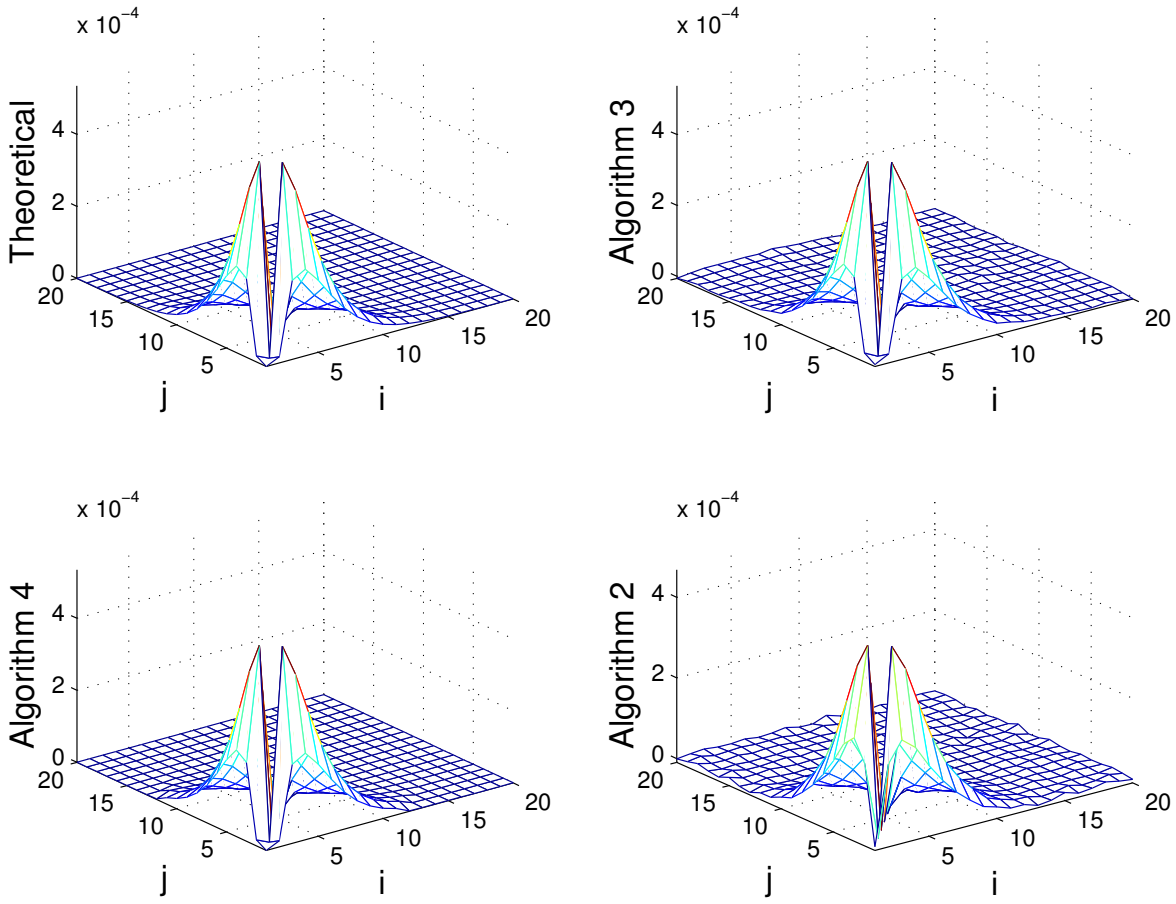

Figure 20: Comparison of the O_2 kernels (z-axis is kernel value). First row: theoretical (L), Algorithm 3 (R), second row: Algorithm 4 (L), Algorithm 2 (R)

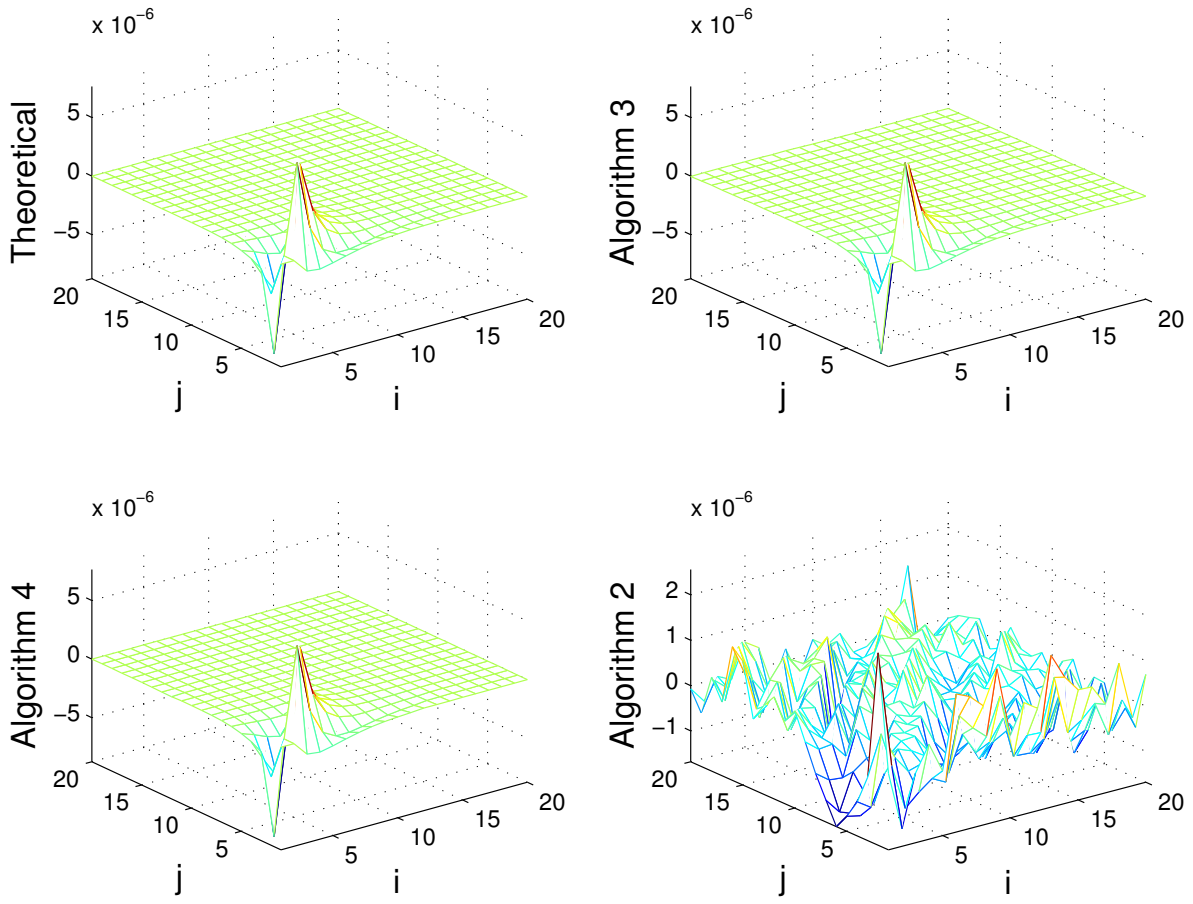

Figure 21: Comparison of the S_3 kernels (z-axis is kernel value). First row: theoretical (L), Algorithm 3 (R), second row: Algorithm 4 (L), Algorithm 2 (R)

elements. A comparison of the identified S_3 kernels is shown in Figure 21. From the figure, it is observed that the S_3 kernel from cross-correlation captures neither the quantitative, nor the qualitative, nature of the theoretical kernel adequately. Finally, the O_3 kernel for both the algorithms is calculated by the cross-correlation approach. Once again, due to the improved estimates from previously identified kernels ($S_3 + O_2$), the O_3 kernel from Algorithm 4 yields better estimates than the kernel from Algorithm 3.

In addition to improving the quality of kernel estimates, the tailored sequences are also significantly shorter than the corresponding sequences used for cross-correlation. Since the same sequence is used for identifying the S_3 and O_2 kernels for the simultaneous approach (Algorithm 4), there is a reduction of 86% in the data required to identify these two kernels, compared to the cross-correlation approach (*i.e.*, the O_2 cross-correlation sequence is no longer necessary).

2.4.2 Identification in the Presence of Noise

An additive uncorrelated white measurement noise with variance equal to 5% of the output was added to the output signal, and Volterra kernels were estimated. All kernel estimates are reported as a mean of 100 runs to average out the effects of the random noise. The results are reported in Table 6 as SSE values between the identified and analytical kernels. For noisy measurement sequences, the number of data points employed per coefficient estimated is an important quantity as the noise effects can be decreased by averaging. Consequently, five repeat units of the input sequence were used for linear and nonlinear diagonal estimation. The resulting sequence, even after five repeat units, was still 30% shorter than the one used in the cross-correlation approach. However, there is a considerable decrease in the quality of the linear and nonlinear diagonal kernel estimates in the presence of noise, for all algorithms. Figure 22 shows a comparison of the linear kernels.

In Algorithm 4, the O_2 kernel is identified simultaneously with the S_3 kernel; it can be observed that noise has a more significant impact on the O_2 than the S_3 kernel. This is consistent with (2.11) and (2.34) where the simultaneous identification of O_2 and S_3 leads to a trade-off in the minimization of the PEV expression. An increase in the coefficient in

Table 6: A comparison of the SSE values between the identified and analytically calculated kernels in the presence of noise

Kernel	Algorithm $2^\star$	Algorithm 3	Algorithm 4
Linear	9.41×10^{-4}	9.41×10^{-4}	9.41×10^{-4}
D_2	1.49×10^{-8}	1.49×10^{-8}	1.49×10^{-8}
O_2	7.25×10^{-7}	5.31×10^{-9}	1.59×10^{-6}
D_3	1.25×10^{-10}	1.25×10^{-10}	1.25×10^{-10}
S_3	4.21×10^{-9}	2.91×10^{-9}	2.76×10^{-9}
O_3	3.93×10^{-10}	8.24×10^{-12}	8.62×10^{-12}

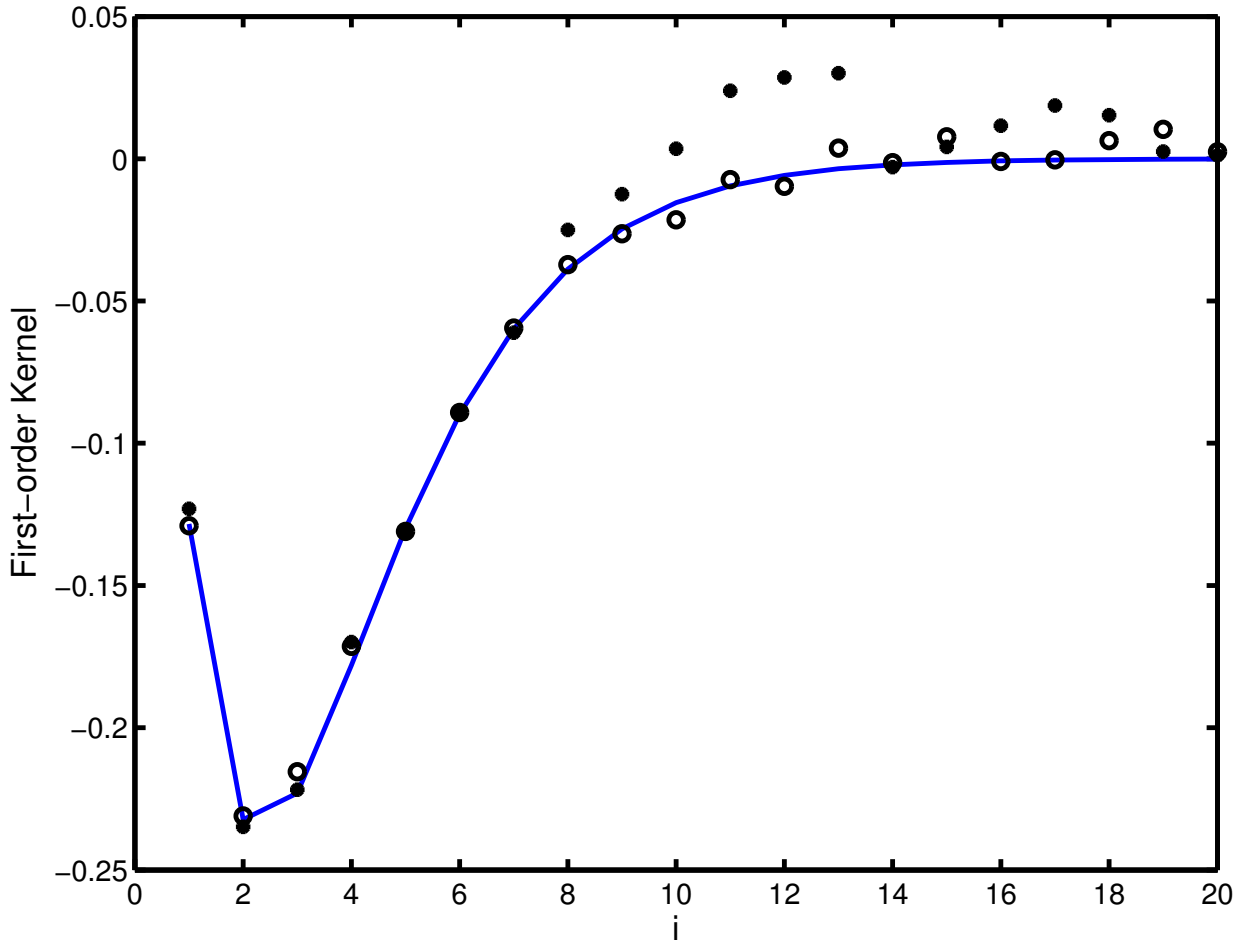

Figure 22: Comparison of the first-order Volterra kernel in the presence of measurement noise

(2.34) causes the S_3 contribution to dominate, leading to a decrease in error in S_3 according to (2.11). This results in O_2 PEV calculations being weighted comparatively less heavily, and hence, preferential noise-corruption of O_2 versus S_3 in Algorithm 2. Finally, the SSE value for O_3 is greater for Algorithm 4 due to the propagation of noise effects from previously identified kernels. Overall, with the exception of O_2 in Algorithm 4, kernel estimates from the tailored identification Algorithms outperform the estimates from cross-correlation.

2.4.3 Validation of the Identified Volterra Models

Validation of the identified Volterra models was carried out using step inputs of magnitude $v = \pm 30$. Figure 23 shows a comparison of the system response for the analytically calculated Volterra model as well as the linear, second-order, and third-order Volterra models for the nominal kernels. From the residuals presented in the legend (representing the SSE between the outputs from the nonlinear and the identified Volterra models summed for the two steps), it is observed that the third-order Volterra model output prediction captures the analytical Volterra kernel output behavior with very high accuracy. In addition, the SSE values are better for Algorithm 4 because of the improved estimates of the noise-free O_2 kernel which is calculated explicitly.

Similarly, Figure 23 also shows the validation results for the kernels identified in the presence of noise. There is a marked drop-off in performance in the presence of noise, more so for Algorithm 4 than for Algorithm 3. This is due to the decreased accuracy of the estimates in the O_2 kernel, and their effect on subsequent kernels due to pre-whitening as outlined in Section 2.4.2. However, it is interesting to note that the third-order Volterra model still outperforms the second-order Volterra model in the presence of noise, independent of the identification algorithm. The decrease in performance is attributed to error propagation due to pre-whitening.

Figure 24 shows the steady-state locus for the nonlinear polymerizer model, as well as the linear, second-, and third-order Volterra models in response to step inputs from -56 to $+56$. The linear model is unable to account for the asymmetry exhibited by the nonlinear model. Of the Volterra models, the third-order Volterra best captures the nonlinear system

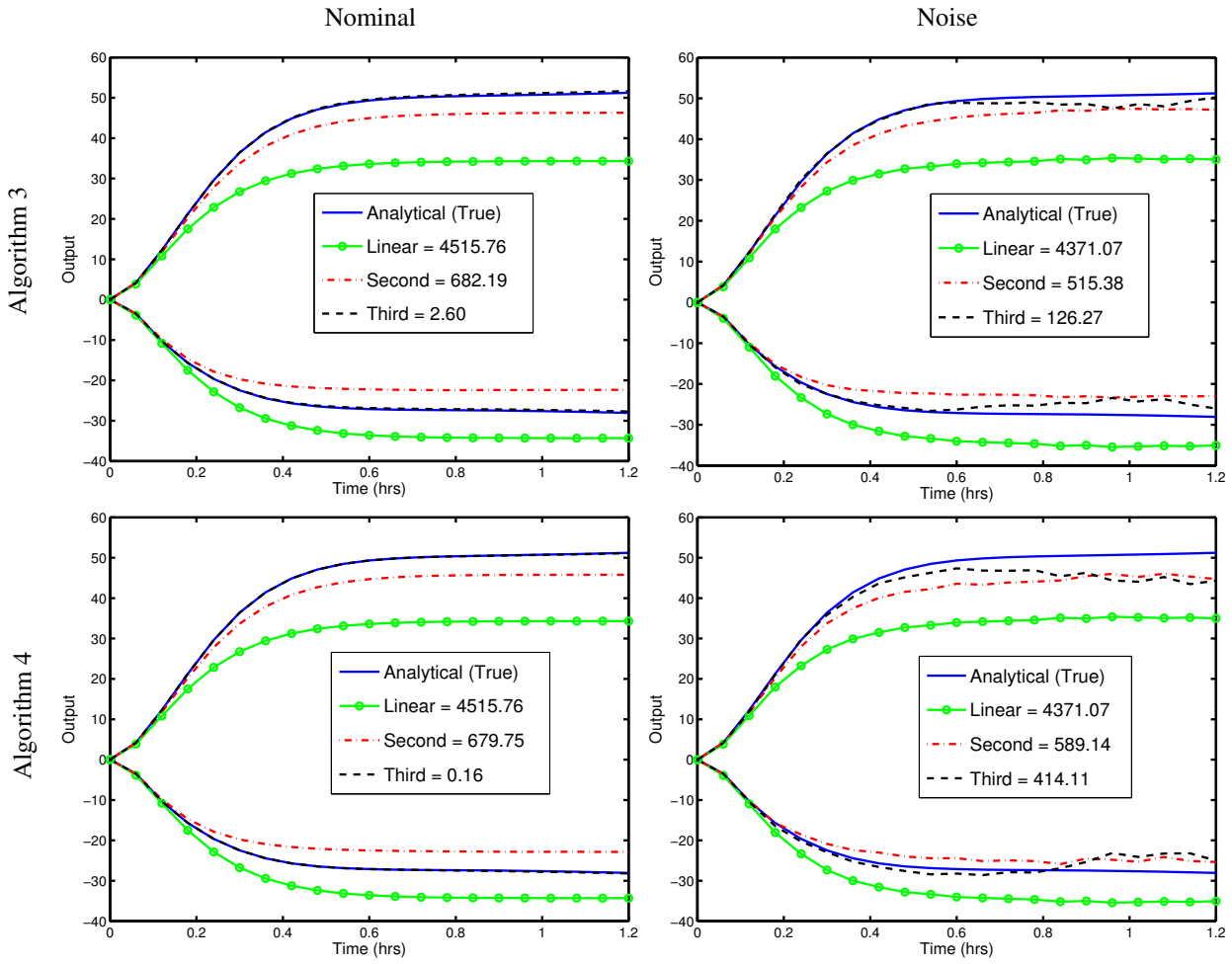

Figure 23: System response to step-inputs of magnitude $v = \pm 30$ for the nominal (left column) and the noise (right column) cases. Outputs from Algorithm 3 (top row) and Algorithm 4 (bottom row) are shown. Lines indicate the analytically calculated Volterra model (—), identified linear model ($- \circ -$), identified second-order Volterra model ($- \cdot -$), and identified third-order Volterra model ($- - -$)

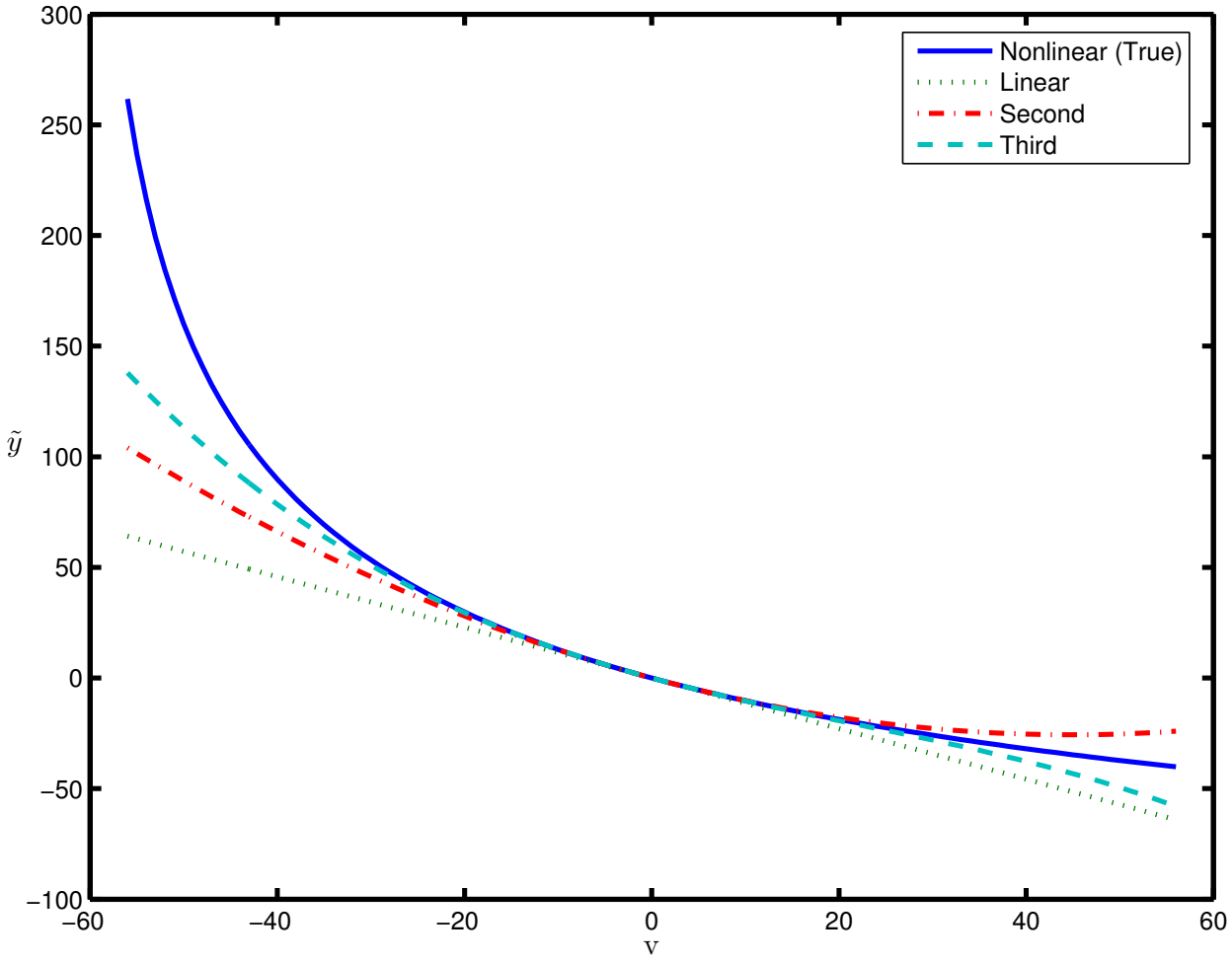

Figure 24: Steady-state locus of the nonlinear model, and the identified linear, second-order, and third-order Volterra models. $\tilde{y}$: scaled deviation number average molecular weight; v: scaled deviation initiator flow rate

behavior. The identified model does not exactly capture the polymerization system response because the latter is not governed by a third-order Volterra model structure. The results represent the best Volterra model for the system based on the input-output data-set.

Figure 25 shows the Integral Square Error (ISE) between the nonlinear and the identified Volterra models in response to steps of integer magnitude for $\nu \in [-56, 56]$ over 1.2 hr. Once again, the third-order Volterra model generally outperforms the linear and second-order Volterra models. ISE is potentially misleading, however. The second-order Volterra model at high v values is qualitatively incorrect (wrong gain sign, see Figure 24).

2.5 SUMMARY

In this Chapter, algorithms for the identification of third-order Volterra models were developed. Tailored input sequences were designed based on the PEV expression and these sequences exploited the Volterra model structure. The estimation of the bias, linear, and nonlinear diagonal kernels was carried out using a tailored four pulse sequence. The third-order sub-diagonal parameters were estimated using a novel sequence designed to preferentially excite the third-order sub-diagonal terms versus the off-diagonal terms. Random binary signals were used to estimate the second- and third-order off-diagonal kernels. Simultaneous identification of the second-order off-diagonal and third-order sub-diagonal kernels was also carried out, thereby completely eliminating the need for a separate sequence for the identification of the second-order off-diagonal kernel. Kernel estimates from this novel identification algorithm consistently outperformed those from the cross-correlation approach.

The effect of measurement noise on the quality of the model estimate was analyzed; while coefficient estimation error increased, the novel identification algorithm estimates were superior to the corresponding cross-correlation estimates. The off-diagonal coefficient estimates showed moderate noise sensitivity (due to the cross-correlation calculation), but improvement in other kernels uniformly improved off-diagonal model quality. In the validation studies, the third-order Volterra model performed significantly better than the lower-order Volterra models.

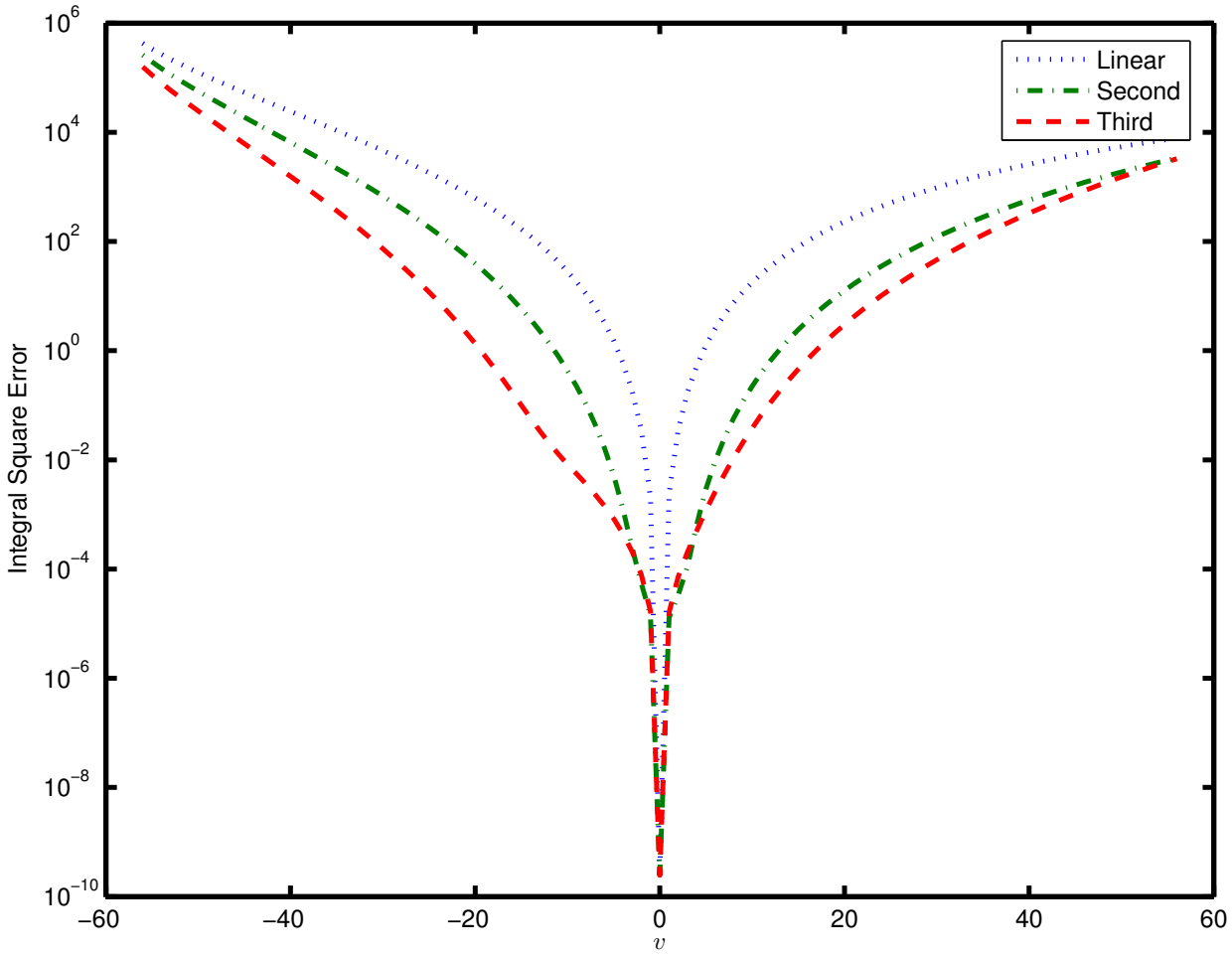

Figure 25: Steady-state step response ISE profiles between the nonlinear and the identified linear, second-order, and third-order Volterra models

One way to reduce the noise-sensitivity of the identified Volterra kernels, is to project them onto a set of basis functions, *e.g.,* the Laguerre basis [30, 41]. This projection also leads to a decrease in the number of parameters, and makes the resulting Volterra-Laguerre model less sensitive to measurement noise. The problem of optimal projection of Volterra models onto the orthonormal Laguerre basis is addressed in the next Chapter.

3.0 VOLTERRA-LAGUERRE MODELS

One of the main difficulties in the estimation of higher order Volterra models is the large number of unique model coefficients that need to be estimated, given by the relation:

$$\sum_{i=1}^{N} \binom{M+i-1}{i} \tag{3.1}$$

Here, N is the order of the Volterra model under consideration and M is the system memory. For the polymerization reactor study, this leads to 1770 parameters. Although it is possible to estimate these parameters, a large amount of data is required which may not always be easily available. In addition, it was observed in the previous chapter that while the quality of nominal Volterra kernels is excellent, there is a considerable decrease in the validation performance for Volterra kernels in the presence of noise. This is not surprising, as the Volterra model is a time series model, and this model class is known to be sensitive to measurement noise. The Volterra model offers several benefits, however: it is linear with respect to parameters, which makes the parameter estimation easier, and its structure is a nonlinear extension of the well known finite impulse response model. Finally tailored input sequence design is possible for the efficient identification of Volterra models.

One way to reduce the parameterization and noise sensitivity of Volterra models is to project them onto a set of orthonormal basis functions. Generally, a relatively small number of orthonormal functions are required, so that the projection operation significantly reduces the number of parameters needed to describe the original nonlinear system. A number of orthogonal basis functions have been considered in literature, such as the Laguerre [30, 41, 59], Kautz [65, 77], generalized basis functions, distorted sines [78], *etc.* In contrast to the Volterra model, common orthonormal functions (*e.g.,* Laguerre and Kautz) consist of a set

of smooth exponential filters so that models based on orthonormal bases have inherent noise filtering. The difference between the orthonormal basis functions lies in their complexity and in their qualitative behavior. As an example, only one parameter (the pole) is required to determine the dynamics for the Laguerre model whereas two poles need to be specified for the Kautz model. The Kautz model can capture oscillatory behavior when the two poles are chosen to be complex conjugates, at the cost of a more complicated model. Owing to the single pole, the Laguerre model is structurally incapable of manifesting oscillatory dynamics. However, the polymerization reactor in this work does not exhibit oscillatory behavior, and the single parameter makes the identification of Laguerre models comparatively easier. Consequently, the Laguerre basis is used in this work.

3.1 LAGUERRE MODEL

Since the Laguerre basis is orthonormal, pertinent mathematical preliminaries of orthonormal bases are presented [79]:

Definition 1. *If, in a space V, there exists a set of n elements ϕ_1, ϕ_2, ϕ_3,..., ϕ_n such that every element $v \in V$ can be written as a linear combination of ϕ_1,...,ϕ_n,* i.e.,

$$v = a_1\phi_1 + a_2\phi_2 +, \cdots + a_n\phi_n$$

where a_1, a_2,..., a_n are scalars not all zero, then the set $\{\phi_1, \phi_2, \phi_3, ..., \phi_n\}$ is called a basis of V and n is the dimensionality of V.

Definition 2. *If the basis set $\{\phi_1, \phi_2, \phi_3, ..., \phi_n\}$ satisfies*

$$\begin{aligned} < \phi_i, \phi_i > &= 1 \qquad \forall i = 1, ..., n \\ < \phi_i, \phi_j > &= 0 \qquad \forall i \neq j \end{aligned}$$

then the basis is an orthonormal basis.

In this definition, $< \cdot >$ denotes the inner product operation.

Definition 3. *Given an orthonormal basis, any real function $f(i)$, $i = 0, 1, ...$ satisfying $f(0) = 0$ can be expanded as [41]:*

$$\begin{aligned} f(i) &= \sum_{k=1}^{\infty} a_k \phi_k(i) \\ a_k &= \sum_{i=0}^{\infty} f(i)\phi_k(i) \end{aligned}$$

In a similar manner, nonlinear functions can be expressed as a linear combination of Laguerre filters. In discrete time, the Laguerre basis function is given as:

$$\phi_j(i) = \sqrt{1-\alpha^2} \left\{ \sum_{k=0}^{j-1} (-\alpha)^k \binom{j-1}{k} \binom{i+k-1}{k-1} \alpha^{i-j+k} U(i-j+k) \right\} \tag{3.2}$$

In this equation, α represents the Laguerre pole and $U(\cdot)$ represents the unit step function. The parameter α controls the exponential rate of decay of the Laguerre functions and thus specifies the dynamic behavior of the Laguerre model. It lies in the range, $0 \leq \alpha < 1$. Figure 26 shows the effect of the Laguerre pole on the dynamics of discrete Laguerre filters. The dynamic behavior of one Laguerre filter is shown for α values of 0.1 and 0.9. Two Laguerre filters are shown for $\alpha = 0.5$. It is observed that all Laguerre filters exponentially decay to zero. However, an increase in α is accompanied by a slower rate of exponential decay. Furthermore, there are $n-1$ zero crossings for a Laguerre filter of order n as is observed for $\alpha = 0.5$. The discrete nature of the filters is also evident for the filter corresponding to $\alpha = 0.1$.

Akin to Definition 3, the output from a Laguerre model can be given as:

$$\hat{y}_l(k) = c_0 + \sum_{i=1}^{\infty} c_i \ell_i(k) + \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} c_{ij}\ell_i(k)\ell_j(k) + \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\sum_{n=1}^{\infty} c_{ijn}\ell_i(k)\ell_j(k)\ell_n(k) \tag{3.3}$$

$$\ell_j(k) = \sum_{i=1}^{u_l} \Phi_j(i) u(k-i) \tag{3.4}$$

In (3.3), $\hat{y}_l$ is the Laguerre output, c represents the Laguerre coefficients (similar to a_k in Definition 3), and u_l is the total length of the input sequence u. The practical advantage of the Laguerre basis is its efficiency because instead of the infinite sum used in (3.3), it is often possible to truncate this sum to a relatively small number of terms [25]. As the number

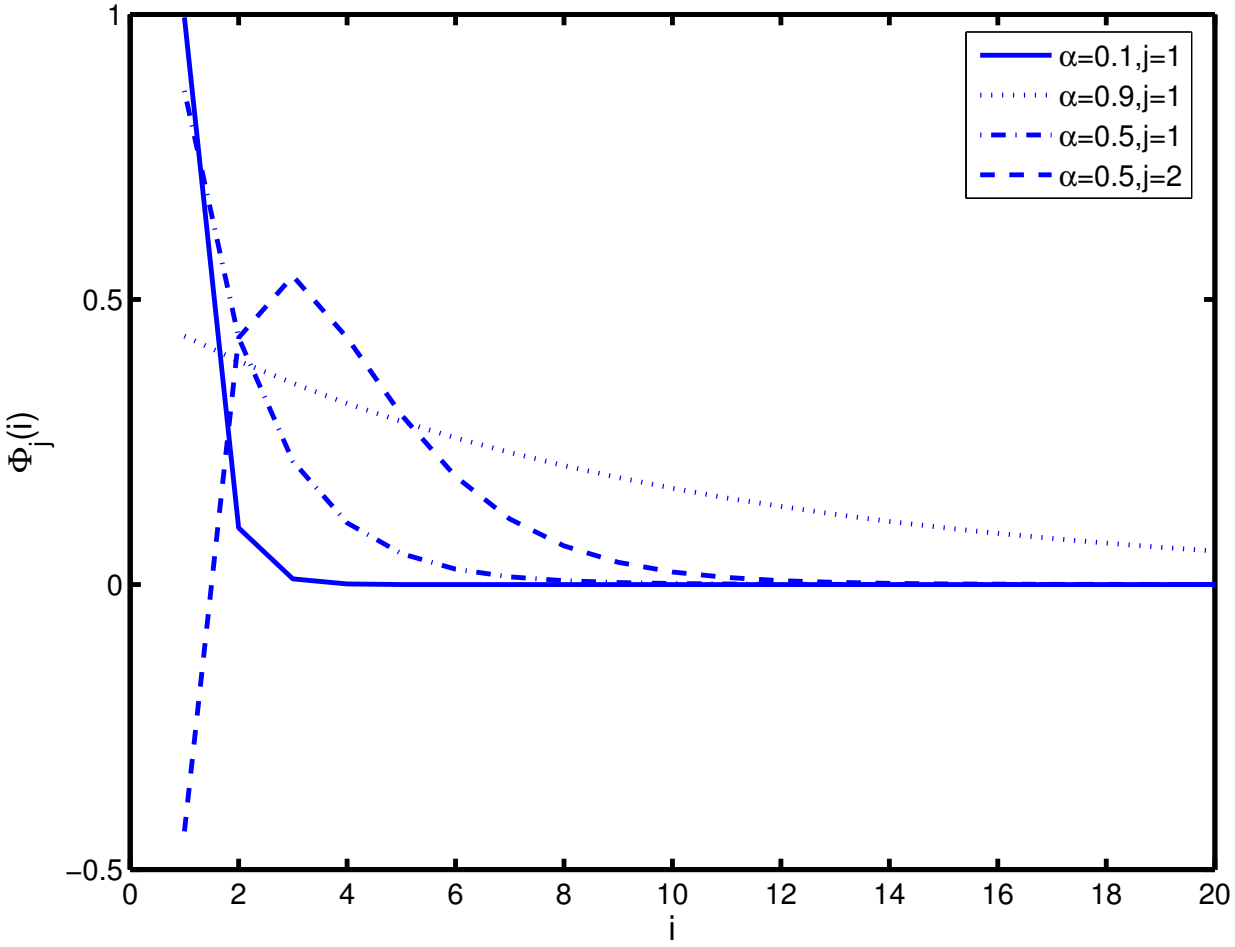

Figure 26: Laguerre filters for different values of the Laguerre pole

of filters included in the Laguerre model is increased, the truncation error from using a finite number of Laguerre filters, decreases. Also, for a given Laguerre model accuracy, the truncation error of the Laguerre approximation can be reduced by properly choosing α [59]. The total number of parameters, N_p, in a third-order Laguerre model with L filters is given as:

$$N_p = \frac{(L+1)(L+2)(L+3)}{6} \tag{3.5}$$

The Laguerre model can also be written in state-space form, which is a useful structure for predictive control. The third-order discrete time state-space representation is given as [59]:

$$\ell(k+1) = A(\alpha)\ell(k) + B(\alpha)u(k) \tag{3.6}$$

$$\hat{y}_l(k) = C^T\ell(k) + \ell^T(k)D\ell(k) + \sum_{i=1}^{L}[\ell(k)^T E_i \ell(k)]\ell_i(k) \tag{3.7}$$

This structure reveals that the Laguerre model, like the Volterra model, is a member of the Wiener class of models as it is composed of a linear dynamic element (3.6), followed by a static memoryless output nonlinearity (3.7). In equations (3.6) and (3.7),

$$A(\alpha) = \begin{bmatrix} \alpha & 0 & 0 & \dots & 0 \\ (1-\alpha^2) & \alpha & 0 & \dots & 0 \\ (-\alpha)(1-\alpha^2) & (1-\alpha^2) & \alpha & \dots & 0 \\ \vdots & \vdots & \dots & \ddots & \vdots \\ (-\alpha)^{L-2}(1-\alpha^2) & (-\alpha)^{L-3}(1-\alpha^2) & \dots & \dots & \alpha \end{bmatrix} \tag{3.8}$$

$$B(\alpha) = \sqrt{(1-\alpha^2)} \begin{bmatrix} 1 \\ -\alpha \\ (-\alpha)^2 \\ \vdots \\ (-\alpha)^{L-1} \end{bmatrix} \tag{3.9}$$

$$C^T = [c_1 \cdots c_L] \tag{3.10}$$

$$D = \begin{bmatrix} c_{11} & \cdots & c_{1L} \\ \vdots & \ddots & \vdots \\ c_{L1} & \cdots & c_{LL} \end{bmatrix} \tag{3.11}$$

$$E_i = \begin{bmatrix} c_{11i} & \cdots & c_{1Li} \\ \vdots & \ddots & \vdots \\ c_{L1i} & \cdots & c_{LLi} \end{bmatrix} \tag{3.12}$$

Here, $A(\alpha)$ and $B(\alpha)$ only depend on α, *i.e.,* the dynamic behavior of the Laguerre filters is set by the pole (α) value. This structure is shown in Figure 27, adapted from [59]. The Laguerre model, given by (3.6), is shown in the z-domain. The top part of the memoryless nonlinearity block includes the linear contributions to the output and has a total of L terms. The middle represents the second-order combinations of Laguerre filter outputs *i.e.,* ℓ_1^2, $\ell_1\ell_2$,$\cdots$, $\ell_1\ell_L$, ℓ_2^2,$\cdots$, ℓ_L^2. The subsequent parts of the memoryless nonlinearity block represent the higher-order contributions.

3.2 VOLTERRA KERNEL PROJECTION ONTO THE LAGUERRE BASIS

3.2.1 Volterra-Laguerre Model

Since the Laguerre basis is orthonormal in the interval $[0, \infty)$ and is complete in the $\mathbf{L}_2[0, \infty)$ space [59], the following Theorem holds [41]:

Theorem 1. *The N^{th}-order kernel $h_N(i_1, ...i_N)$ can be expanded by a Laguerre series as:*

$$h_N(i_1, ...i_N) = \sum_{j_1=1}^{\infty} \cdots \sum_{j_N=1}^{\infty} c_N(j_1 \cdots j_N)\phi_{j_1}(i_1), \cdots, \phi_{j_N}(i_N) \tag{3.13}$$

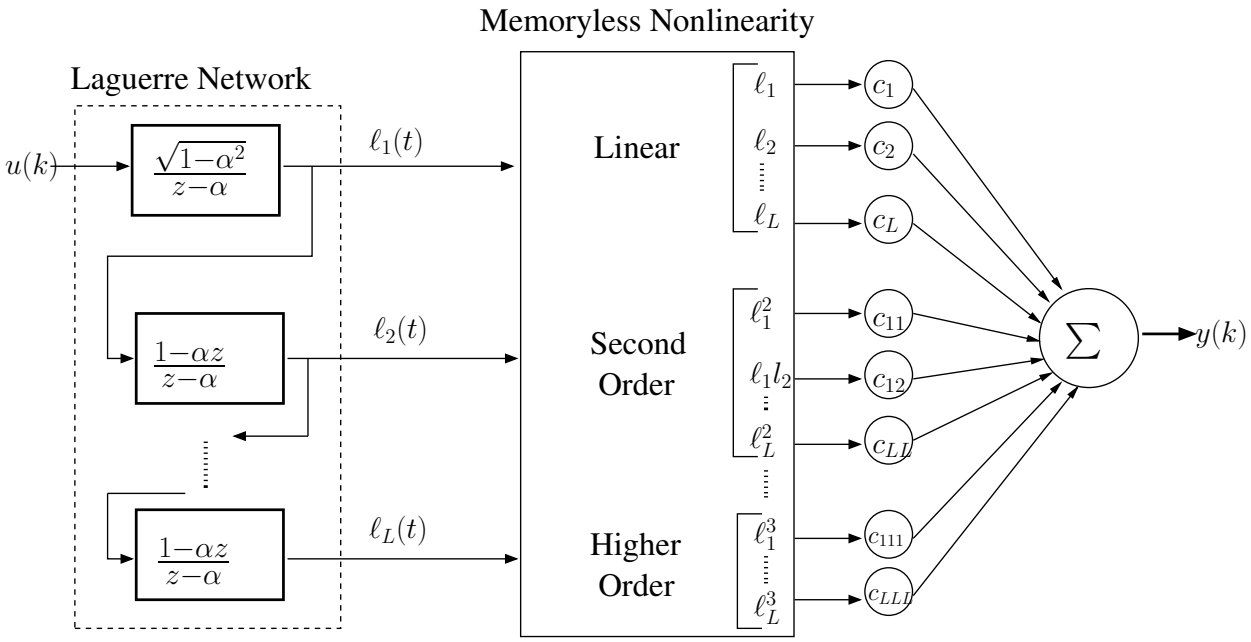

Figure 27: Schematic of the nonlinear Laguerre model structure (adapted from [59])

where

$$c_N(j_1, ...j_N) = \sum_{i_1=1}^{\infty} \cdots \sum_{i_N=1}^{\infty} h_N(i_1, \cdots, i_N)\phi_{j_1}(i_1) \cdots \phi_{j_N}(i_N) \tag{3.14}$$

if the kernel is stably separable and strictly proper.

Definition 4. *A kernel is separable if [41]:*

$$h_N(i_1, ..., i_N) = \sum_{j=1}^{q} \nu_{1j}(i_1)\nu_{2j}(i_2) \cdots \nu_{Nj}(i_N) \tag{3.15}$$

where q is some finite number and $\nu_{gj}(i_g)$ is a single variable real function $g = 1, 2, \cdots, N$. If each $\nu_{gj}(i_g)$ also satisfies

$$\sum_{i_g=0}^{\infty} |\nu_{gj}(i_g)| < \infty \tag{3.16}$$

the kernel is called stably separable.

Definition 4 states that the the Volterra kernel can be deconstructed in terms of $\nu_{gj}(i_g)$ (which can be thought of as pseudo-input terms), and (3.16) states that if each of $\nu_{gj}(i_g)$ is bounded, then the system is stable. Definition 4 and (3.16) state that the Volterra kernels are BIBO stable, *i.e.,* for a bounded input, the output of the Volterra model is bounded.

Definition 5. *A symmetric kernel is called strictly proper if*

$$h_N(i_1, \cdots, i_N) = 0, \ \textit{for } i_1, i_2, \cdots, i_N = 0 \tag{3.17}$$

This definition implies that the Volterra model has no direct feed, *i.e.,* the current output is a function of only the past inputs, and does not depend on the current input. The polymerization reactor case-study is a fading memory system and the Volterra kernels for this system satisfy the conditions of Definitions 4 and 5. Hence the Volterra kernels can be projected onto the Laguerre basis. As discussed before, instead of the infinite sum in (3.13), only L terms are retained, so that:

$$h_N(i_1, \cdots, i_N) \ = \ \sum_{j_1=1}^{L} \cdots \sum_{j_N=1}^{L} c_N(j_1, \cdots, j_N)\phi_{j_1}(i_1) \cdots \phi_{j_N}(i_N) \tag{3.18}$$

In this equation, $c_N(j_1, \cdots, j_N)$ represent the Laguerre coefficients, and they are obtained using the Volterra kernels using the following equation:

$$c_N(j_1, \cdots, j_N) = \sum_{j_1=1}^{L} \cdots \sum_{j_N=1}^{L} h_N(i_1, \cdots, i_N)\phi_{j_1}(i_1) \cdots \phi_{j_N}(i_N) \tag{3.19}$$

The Laguerre coefficients thus obtained are equivalent to the C, D, and E matrices in (3.7). The dynamics of the VL model are determined by α and are given by (3.6).

The main advantage of the Laguerre basis is that a system can be adequately described with $L << M$, so that the number of parameters required to characterize the Laguerre model decreases considerably. Another advantage of using the Laguerre basis is that α and L are the only two variables that need to be specified to completely describe a VL approximation of a given Volterra model. However, as seen before (Figure 26) L and α are coupled, *i.e.,* the truncation error is a function of both L and α.

3.2.2 Determination of Laguerre Model Parameters

Since L and α are coupled, the appropriate selection of these two parameters is critical for an accurate and compact VL model description. The selection of the number of Laguerre filters has generally been carried out heuristically. This choice is a balance between increased model accuracy due to an increase in the number of filters, and the corresponding increase in the number of parameters. One of the approaches to determine α is the brute force technique, *i.e.,* for evenly-spaced α in the range $[0, 1)$, what value minimizes the SSE between the Volterra and the Volterra-Laguerre model output predictions. This approach assumes that the appropriate number of Laguerre filters has already been selected, and it has been successfully employed in literature [19, 30, 34, 65]. Less computationally intensive methods have also been employed. Wahlberg proposed a methodology to collapse ARX models onto the orthonormal Laguerre basis [58]. The Laguerre pole was selected based on the time-constant of the system and was chosen to avoid a slow convergence rate of the Laguerre filters. Fu and Dumont used an optimization-based approach to estimate the optimal Laguerre pole for linear systems [80]. Their objective function ensured that the coefficients of higher-order Laguerre functions rapidly decayed to zero. In their objective function, the weight

on each additional Laguerre filter increased linearly. Sabatini used a least-squares approach to determine the optimal α for a Laguerre model. Their objective function minimized the truncation error from using a finite number of Laguerre filters, and they converted the optimization to a roots finding problem. A hybrid genetic algorithm (GA) based approach was used, whereby the GA was used for fast convergence to reach a neighboring region of solution and then a Newton Raphson method was used to attain convergence within the region identified by the GA [81]. Malti *et al.* minimized the SSE between the system and Laguerre model outputs. They used a Newton Raphson technique to calculate the optimal Laguerre pole [82]. The authors provided analytical expressions for the gradients and Hessians of the optimization problem. A drawback of the above approaches is that they do not address the projection of nonlinear Volterra models, and their results are demonstrated for relatively simple numerical examples. A notable exception is the approach by Campello *et al.* [83]. Their results are an extension of those presented in [59] for nonlinear systems. However, this approach is strictly optimal only for infinite dimensional expansions. This approach is not conducive for practical implementation, as the high model dimensionality (L) could pose challenges for control.

3.2.3 Optimal Projection of Volterra kernels onto the Laguerre Basis

One of the main disadvantages of the approaches for determining the optimal α presented in the previous Section is that they are valid only for linear time invariant (LTI) systems. The objective function for optimization in these problems could be broadly classified into two main categories given as follows:

- Minimization of model complexity: In this class of problems the objective function is formulated to reduce the number of model parameters, *e.g.,* Dumont and Fu penalize the addition of Laguerre filters [59]. However these approaches do not explicitly address the accuracy of the projection.
- Maximization of model accuracy: Here the objective function minimizes the SSE between the nonlinear and (Volterra-)Laguerre model predictions [30, 41, 65]. However these

approaches do not explicitly address the selection of the number of Laguerre filters, and thus leave the question of the size of the resultant model unaddressed.

As discussed in Chapter 1, a nonlinear model developed for the purpose of control should be both accurate and parsimonious. Thus, a different approach is selected to determine the optimal Laguerre pole in this work. The key motivation is that the identified VL model must take into account optimality for both α and L.

Generally, an increase in the number of Laguerre filters leads to a higher quality VL model. However, this improvement in quality is at the cost of increased model parameterization. In such a scenario, statistical information criteria are useful as they can be used to determine the quality of model fit with respect to both model size and model accuracy. Information criteria are generally composed of a sum of two terms. The first term takes into account the accuracy with which the model predicts the data, and this term attains an increasingly negative magnitude as the model becomes more accurate. The second term penalizes the addition of new parameters in the resulting model. Thus information criteria enable a balance between model accuracy and model complexity. The commonly used information criteria are the Akaike Information Criterion, the Bayes Information Criterion, Final Prediction Error, and Law of Iterated Logarithmic Criterion [84].

In this work the Akaike Information Criterion is used as a metric to trade-off model size (set by the number of Laguerre filters) and model quality (characterized by both the number of Laguerre filters and the Laguerre pole) as measured by the sum-squared error between the Volterra and the Volterra-Laguerre model outputs. The Akaike Information Criterion (AIC) is given as [85]:

$$AIC = K \log\left(\frac{SSE(\alpha, L)}{K}\right) + 2N_p \tag{3.20}$$

The first term accounts for model accuracy, and K is a measure of the amount of data used to determine the model. The second term penalizes the complexity of the resultant model. In the context of this work, K is the number of Volterra model coefficients that are to be projected, SSE is the sum squared error between the original Volterra and the projected VL model outputs, and N_p is the number of parameters in the resulting VL model. For a given data-set, the SSE depends on both the number of Laguerre filters and the corresponding

Laguerre pole. Each pair of α and L characterizes a VL model that has a unique AIC value associated with it. The model with the lowest AIC value represents the optimal balance between the number of parameters and model quality.

In this work, the optimal Laguerre pole for a given L is calculated for the complete nonlinear Volterra model, a problem which has not received attention before. The SSE term in the AIC expression is given as follows:

$$SSE(\alpha, L) = \left(\hat{y}_v(k) - \hat{y}_l(k, \alpha, L)\right)^2 \tag{3.21}$$

In this equation $\hat{y}$ and $\hat{y}_l(\alpha, L)$ represent the Volterra and VL model outputs, respectively. The dynamics of the VL model are set by α in (3.6), whereas L determines the size of the output matrices C, D, and E in (3.7). Thus the optimization problem can be formulated as:

$$\begin{aligned} J(\alpha, L) \quad &= \quad \min_{\alpha, L} K \log\left(\frac{\left(\hat{y}_v(k) - \hat{y}_l(k, \alpha, L)\right)^2}{K}\right) + 2N_p \\ \texttt{s.t.} \quad &\quad 0 \leq \alpha < 1 \end{aligned} \tag{3.22}$$

This is a constrained nonlinear optimization problem due to the nonlinear dependence of the Laguerre filters on α.

The complete algorithm for determining the optimal number of Laguerre filters and Laguerre pole is as shown in Figure 28. The Algorithm is initialized with $L = 1$. The optimization routine determines an α that minimizes the SSE for the training data-set, which could be an impulse, step, or multi-level input. In general, the training SSE would depend on the data-set, but preliminary investigations for impulse and random data-sets revealed SSE to be a convex function of α. For an (α, L) pair, an AIC value is calculated, and the number of Laguerre filters is increased until a minimum in AIC is reached. To avoid over-parameterization the upper bound for the number of Laguerre filters is the Volterra model memory M. Thus another approach to determine the optimal α and L is to select the lowest AIC value corresponding to L in the range $[1, M]$. The combination of L and corresponding α sets all the parameters needed for the VL model. In the above optimization, L is an integer valued variable. Thus, optimizing over both the α and L range would lead to a mixed-integer nonlinear programming problem (MINLP). In order to restrict the complexity of the optimization problem, the Laguerre space is discretized, *i.e.,* the algorithm increments the addition of Laguerre filters.

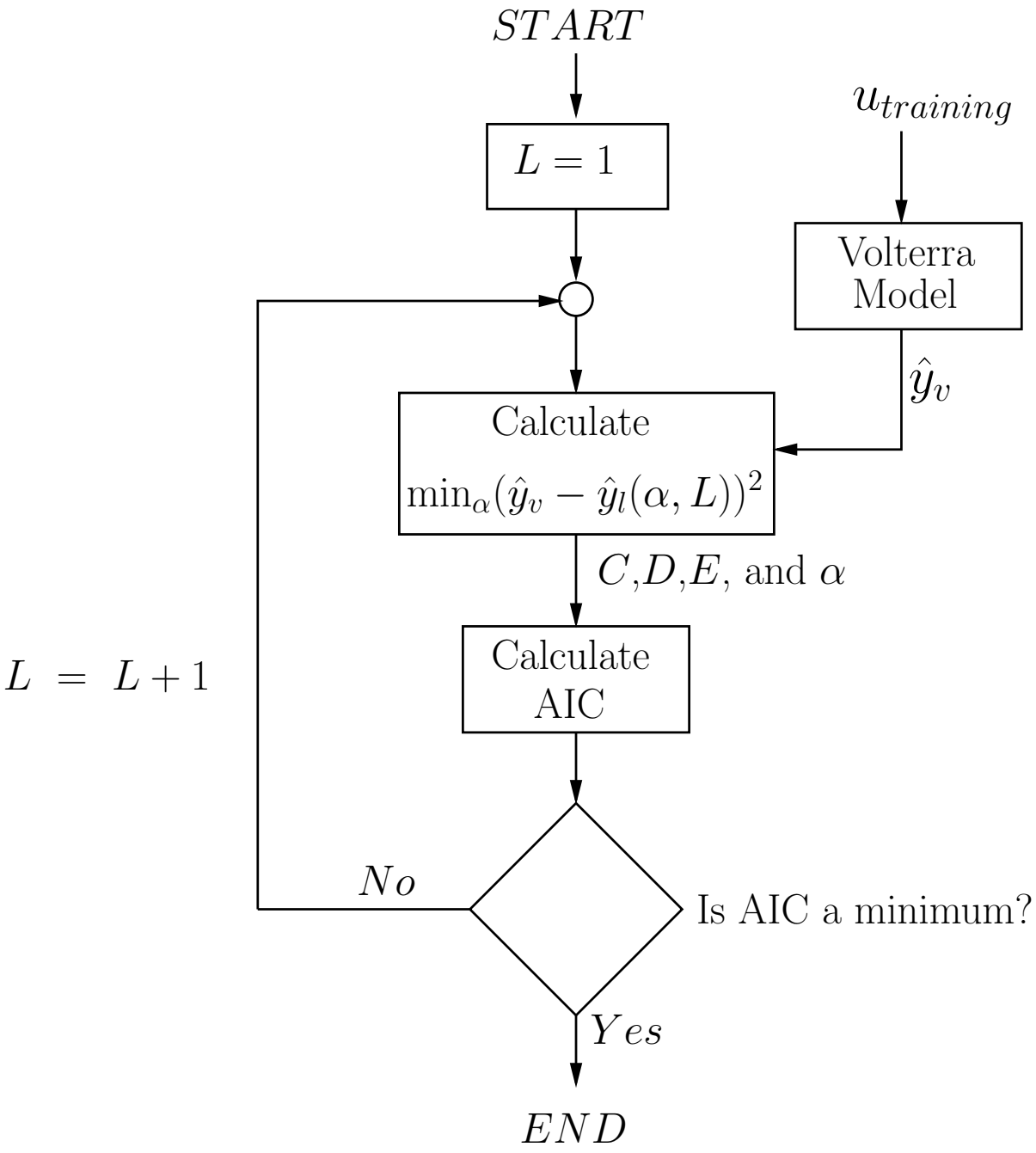

Figure 28: A schematic of the Algorithm for determining the optimal Laguerre pole

3.3 RESULTS

The Volterra kernels identified in the previous chapter can now be projected onto the Laguerre basis using the approach presented in Section 3.2.3. The polymerization reactor case study presented in Section 2.1 continues to serve as the simulation case study. Since Algorithm 4 required the least amount of data, and Algorithm 3 showed the best performance in the presence of noise, all results in this Section are presented using Volterra kernels identified via these two algorithms. As the analytical Volterra kernels (Section 2.1.1) are a given, they are used to validate the performance of the identified Volterra and the VL models to enable a fair comparison.

3.3.1 Projection of the Full Volterra Model onto the Laguerre Basis

3.3.1.1 Nominal Identification In order to project the Volterra kernels onto the Laguerre basis, the projection algorithm was initialized with one filter ($L = 1$). An impulse response was selected to generate the training data-set for two main reasons: first, the Volterra model is a nonlinear extension of the FIR model; second, all of the tailored input sequences used to identify the Volterra models were based on impulses. The magnitude of the impulse was selected as 28. The SSE values between the Volterra and the VL model outputs were found to be relatively insensitive to the magnitude of the impulse input. Hence a value of 28 was selected as it represented the mid-point of the positive half of the input range.

Using the projection algorithm, the minimum AIC value was obtained for $L = 19$ with an optimal pole of 0.0393. A visualization of the AIC results can be found in Figure 29. The sub-plot of the SSE between the Volterra and Volterra-Laguerre models for the training data-set (Figure 29, middle) shows that the SSE value is negligible for $L = 19$, and small for $L = 2$. A plot of the output residuals between the Volterra and the VL model (Figure 29, bottom) confirms this observation; the residuals are close to zero for the entire duration of the training data-set. However, 19 filters implies that the dimensionality of the VL model approaches that of the original Volterra model ($M = 20$). This is a somewhat contrasting

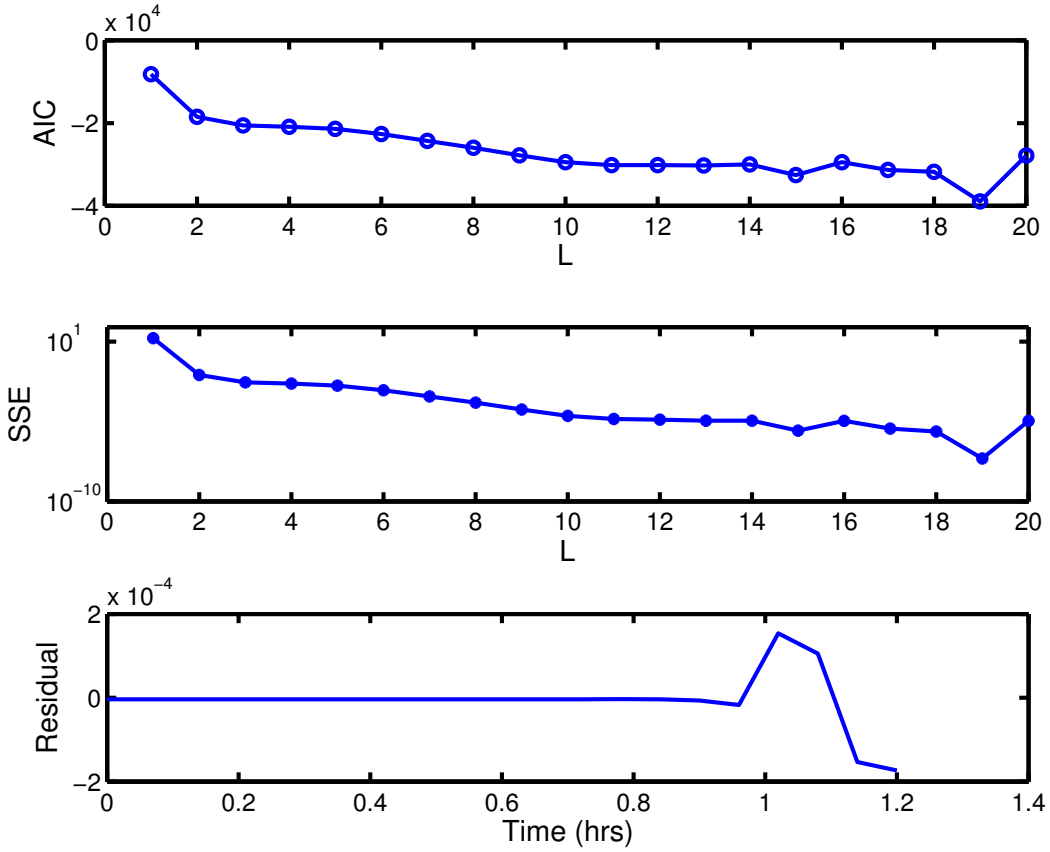

Figure 29: Projection of Volterra kernels identified from Algorithm 3 onto the Laguerre basis. AIC (top), SSE (middle), and the output residuals (bottom) between the Volterra and VL model for the training data-set

Table 7: Optimal Laguerre parameters from the projection algorithm for the nominal case

	Algorithm 3	Algorithm 4
Training SSE	0.0508	0.0486
L	2	2
α	0.5624	0.5625

result for the nominal case, as the SSE value is small for $L = 2$ and the residuals are close to zero.

Figure 30 shows the individual contributions of the two terms that constitute the AIC. The first term, based on the SSE value, is given by $K \log(\frac{SSE}{K})$, whereas the second term, based on the number of parameters, is given by $2N_p$. For the nominal case, since the SSE value is low for two filters, the contribution of the first term dominates that of the second term. The addition of subsequent filters improves the SSE value so that the first term, and hence the AIC, continue decreasing. This contribution is two orders of magnitude greater than that of the second term, so that the effect of additional parameters with added filters has no significant effect on the AIC value.

For $L = 2$, the projection SSE was 0.0508 for a sequence consisting of 21 data-points. For the impulse input, on a per coefficient basis, the output is within 1% of the analytically calculated Volterra kernel output. Hence, the incremental accuracy improvement provided by additional filters was not justifiable. Consequently, 2 filters were selected for the VL model and the optimal α was 0.5624. The results for kernels from both Algorithms are shown in Table 7. Since the nominal performance of the Volterra kernels from Algorithms 3 and 4 is similar, the corresponding VL models are similar as well. The outputs for the nonlinear, Volterra, and the Volterra-Laguerre models for the training data-set are shown in Figure 31 for kernels identified from both Algorithms 3 and 4.

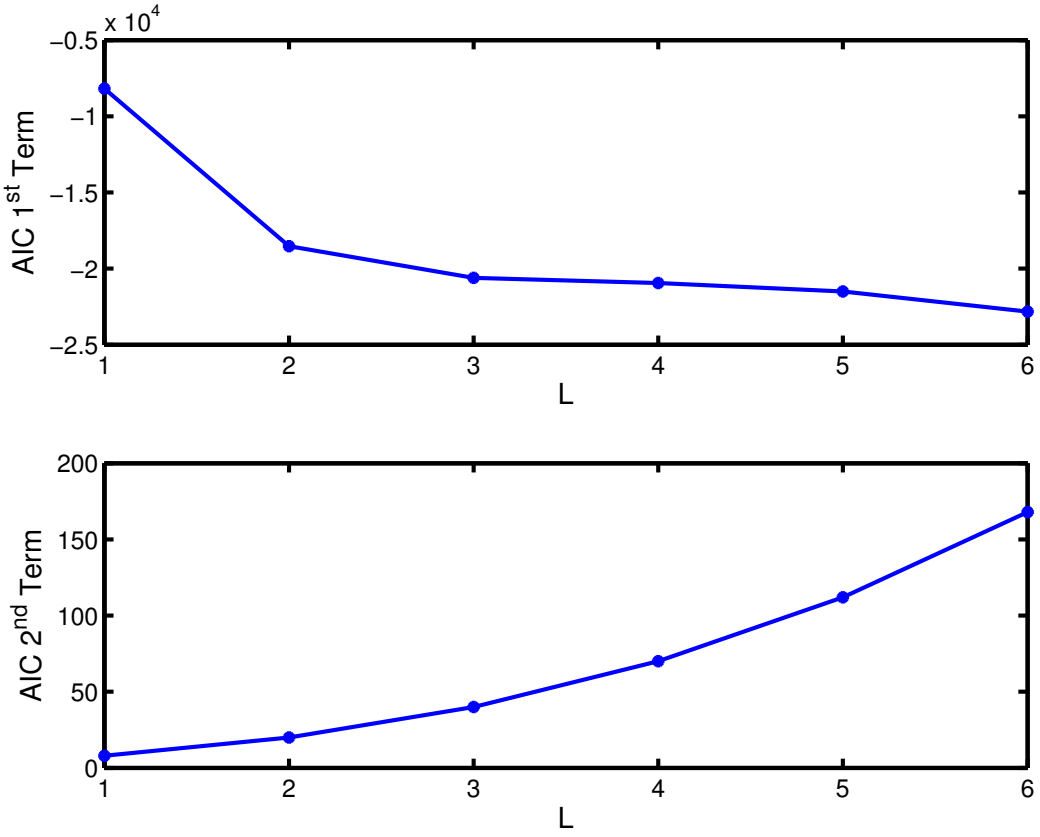

Figure 30: Relative contributions of the two terms that make up the AIC; first term (top) and second term (bottom)

Validation of the VL models was carried out using a random multi-level sequence consisting of 220 data-points characterized by the following distribution:

$$u_{random-val}(k) = \begin{cases} 56 & \text{with probability } \frac{1}{10} \\ 42 & \text{with probability } \frac{1}{10} \\ 28 & \text{with probability } \frac{3}{10} \\ -28 & \text{with probability } \frac{3}{10} \\ -42 & \text{with probability } \frac{1}{10} \\ -56 & \text{with probability } \frac{1}{10} \end{cases} \tag{3.23}$$

The first 20 points were used to initialize the input sequence and thereby eliminate coefficient error bias related to initial condition effects. To average out the sequence effects due to the random input, results from 100 input sequences given by the distribution in (3.23) were considered. The complete set of validation results are presented in Table 8. The first row represents results for the training data-set, *i.e.,* the SSE values between the analytically calculated and the VL, and the analytically calculated and Volterra model outputs. The subsequent rows present a representative result (Figure 32), the mean, and the range of the results from the random sequence, respectively. Since the Volterra kernels from Algorithm 4 are identified with the least amount of data (*i.e.,* the most efficient identification algorithm), all subsequent figures are shown for VL models identified from Volterra kernels identified from Algorithm 4. Also, the inputs and outputs in all the figures considered in this chapter are reported in scaled deviation form.

Since validation of the identified Volterra and VL models is carried out against analytically obtained Volterra kernels, the nominal performance of the Volterra models for the training data-set is excellent (SSE $= 1.36 \times 10^{-10}$ in Figure 31). Although worse than the Volterra model, the performance of the VL model is still admirable and on a per-coefficient basis predictions are within 1% of the analytically calculated Volterra kernel output values.

For the validation performance using the random sequence, with a representative result shown in Figure 32, the Volterra model continues to outperform the VL model. The VL model does not accurately match the peaks in the analytical Volterra model output (Figure 32, top). There is a considerable loss in the performance of the VL model when compared

Table 8: SSE values between the analytical Volterra and VL, and analytical Volterra and Volterra models for the nominal case

	Algorithm 3		Algorithm 4	
Sequence	Volterra-Laguerre	Volterra	Volterra-Laguerre	Volterra
Training	0.05	0.00	0.05	0.00
Random	1315.52	3.70	1317.36	1.59
mean	1252.00	3.59	1274.89	1.36
min	810.76	1.74	813.79	0.64
max	1708.06	6.34	1865.72	2.40

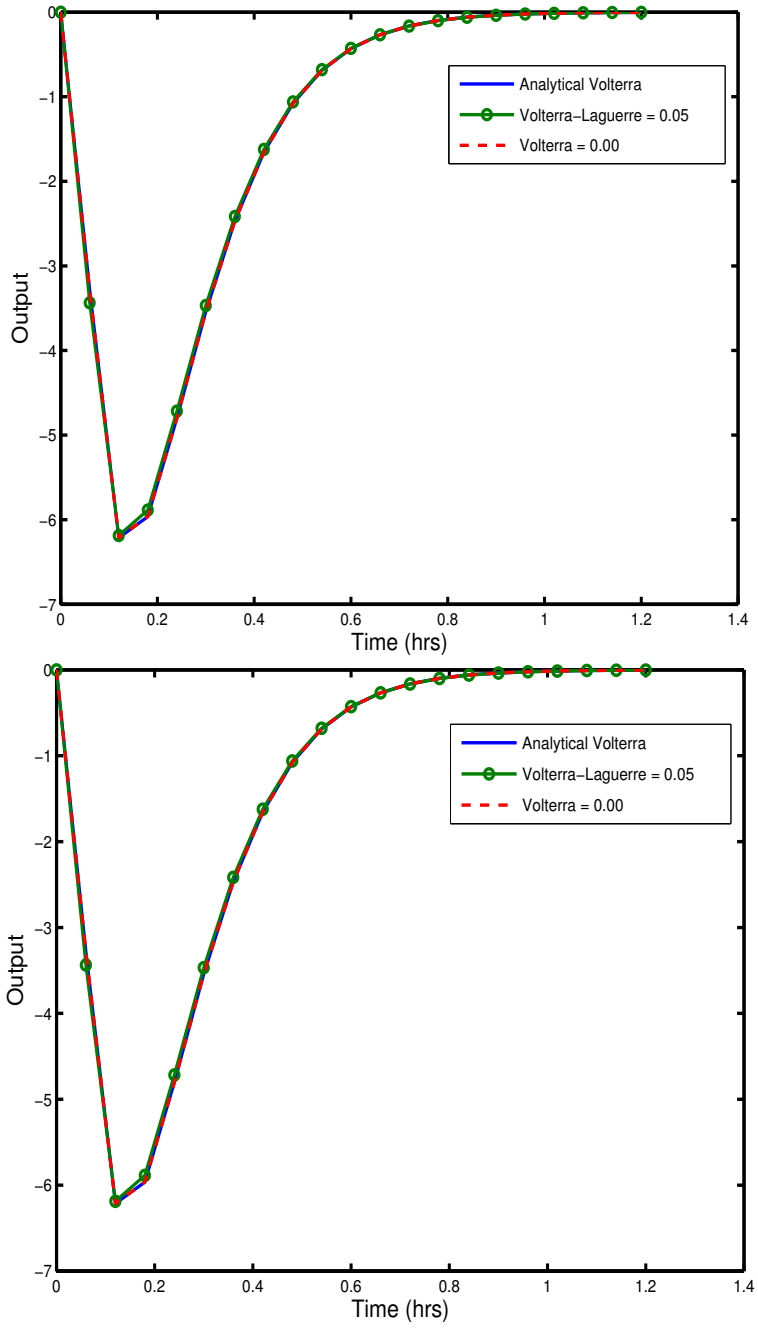

Figure 31: Output from the analytically obtained Volterra kernels (—), VL (−o−), and Volterra (− − −) model for an impulse input of magnitude 28 in the nominal case. Results from both Algorithm 3 (top) and Algorithm 4 (bottom) are shown. The input and the output are in scaled deviation form.

Table 9: Optimal Laguerre parameters from the projection algorithm for noise corrupted Volterra kernels

	Algorithm 3	Algorithm 4
Training SSE	0.2752	0.2687
L	2	2
α	0.5688	0.5701

to the original Volterra model. Since analytically calculated Volterra kernel output was used to validate the models, the excellent performance of the identified Volterra models is not surprising.

3.3.1.2 Identification in the Presence of Noise As in the nominal case, the kernels from both Algorithm 3 and Algorithm 4 were used to obtain a VL model. The noise was inherent in the identification of the Volterra kernel, *i.e.* no noise was added in the projection of Volterra kernels onto the Laguerre basis. The characteristics of the noise are described in Section 2.4.2. Once again, an impulse response of magnitude 28 was used to generate the training data-set. The results from the projection algorithm are shown in Table 9. Two Laguerre filters were found to be sufficient in the noise case. The optimal pole locations do not significantly deviate from their nominal counterparts (see Table 7). However, as the behavior of the Volterra kernels from Algorithms 3 and 4 are different in the presence of noise, the corresponding pole locations for the VL model are also slightly different. The SSE errors for the training data-set are also higher than their nominal counterparts, as expected.

The output for the analytically calculated, Volterra, and the VL models for the training data-set are shown in Figure 33. Here the excellent noise filtering capabilities of the Laguerre basis are evident. The VL model captures the nonlinear model behavior for the training data-set accurately, whereas the Volterra model shows some deviations about the nonlinear model,

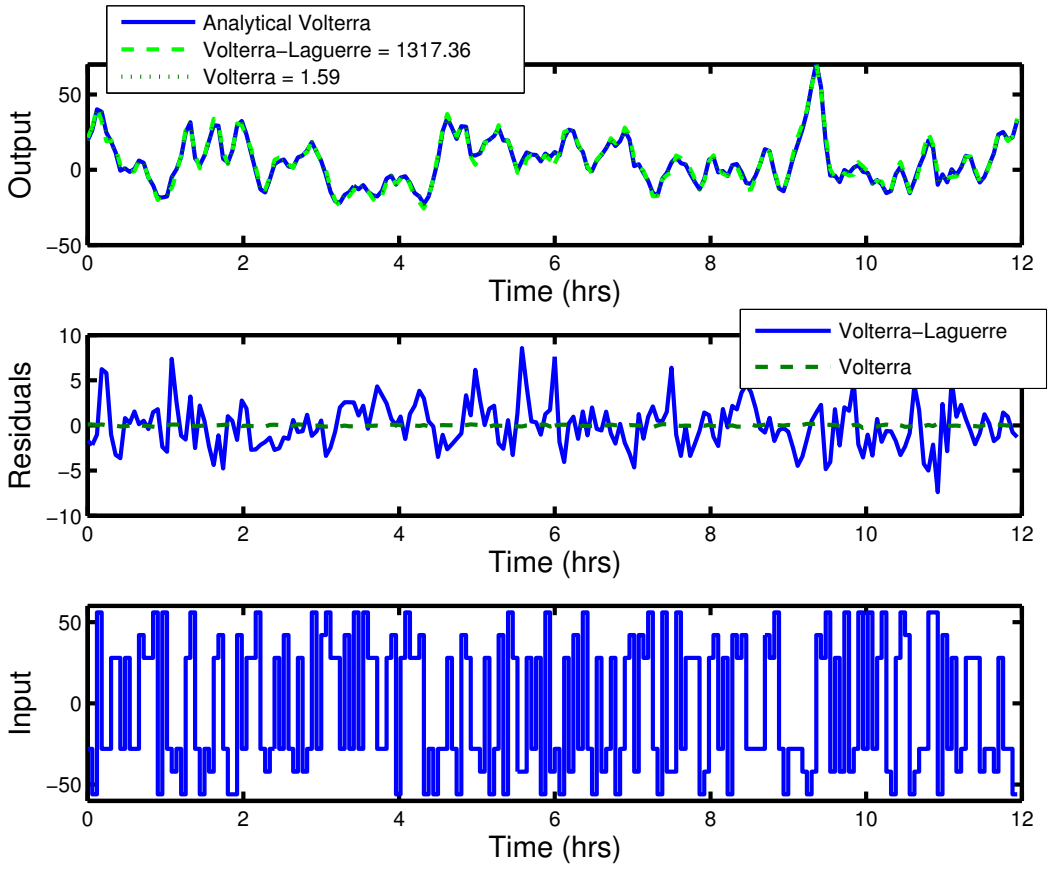

Figure 32: Validation performance of identified models with a random sequence, nominal case. Top: analytical Volterra (—), VL (− − −), and Volterra (· · ·) model outputs. The legend represents the SSE values between the analytically obtained Volterra kernels and the identified models. Middle: residuals between the analytical Volterra and the VL (—) and Volterra (− − −) models. Bottom: random input sequence used for validation. The input and the output are in scaled deviation form.

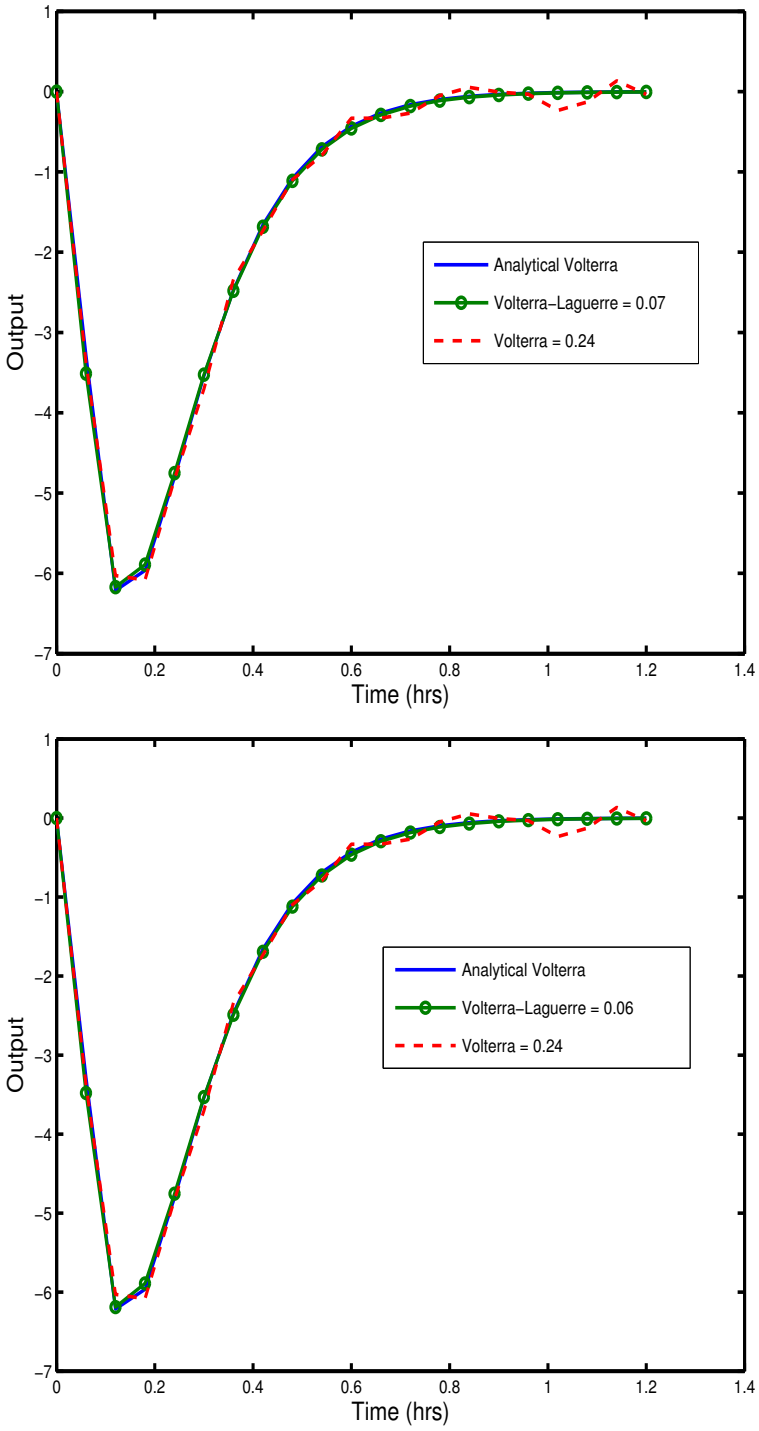

Figure 33: Output from the analytical Volterra kernels (—), VL (−o−), and Volterra (− − −) model for an impulse input of magnitude 28 in the projection of noise-corrupted Volterra kernels. Results from both Algorithm 3 (top) and Algorithm 4 (bottom) are shown. The input and the output are in scaled deviation form.

Table 10: SSE values between the analytical Volterra and VL, and analytical Volterra and Volterra models, in the presence of noise

	Algorithm 3		Algorithm 4	
Sequence	Volterra-Laguerre	Volterra	Volterra-Laguerre	Volterra
Training	0.07	0.24	0.06	0.24
Random	1330.52	37231.19	1558.33	35330.54
mean	1360.26	28862.73	1623.73	27378.74
min	738.65	19154.77	1033.00	19227.09
max	2290.10	42645.13	2475.41	38356.62

especially at later time-points (where the signal to noise ratio was low), for kernels identified using both Algorithms 3 and 4.

The validation SSE results are shown in Table 10. The first row presents the SSE values for the training data-set, whereas the subsequent rows present the results for the random sequence. As in the nominal case, results from 100 inputs are shown, and the distribution of the random sequence is as shown in (3.23). Figure 34 shows the output responses of the analytical Volterra, VL, and Volterra model outputs to the random validation sequence. The excellent noise filtering capabilities of the VL model are evident. When compared to the nominal case (see Table 8), the performance of the VL model does not degrade as significantly as that of the Volterra model. Comparing the means, the loss in performance due to noise is 9% and 26% for the VL models from Algorithm 3 and 4, respectively. On the other hand, the decrease in performance for the Volterra model is five orders of magnitude. Since the Laguerre model is a set of exponential filters, the VL model shows excellent noise filtering abilities, and consequently an improved performance.

Thus the VL model outperforms the Volterra model in the presence of noise. In addition, it was found that both the number of Laguerre filters (2) and the Laguerre pole remained relatively unchanged for the nominal, as well as the noise cases (see Tables 7 and 9).

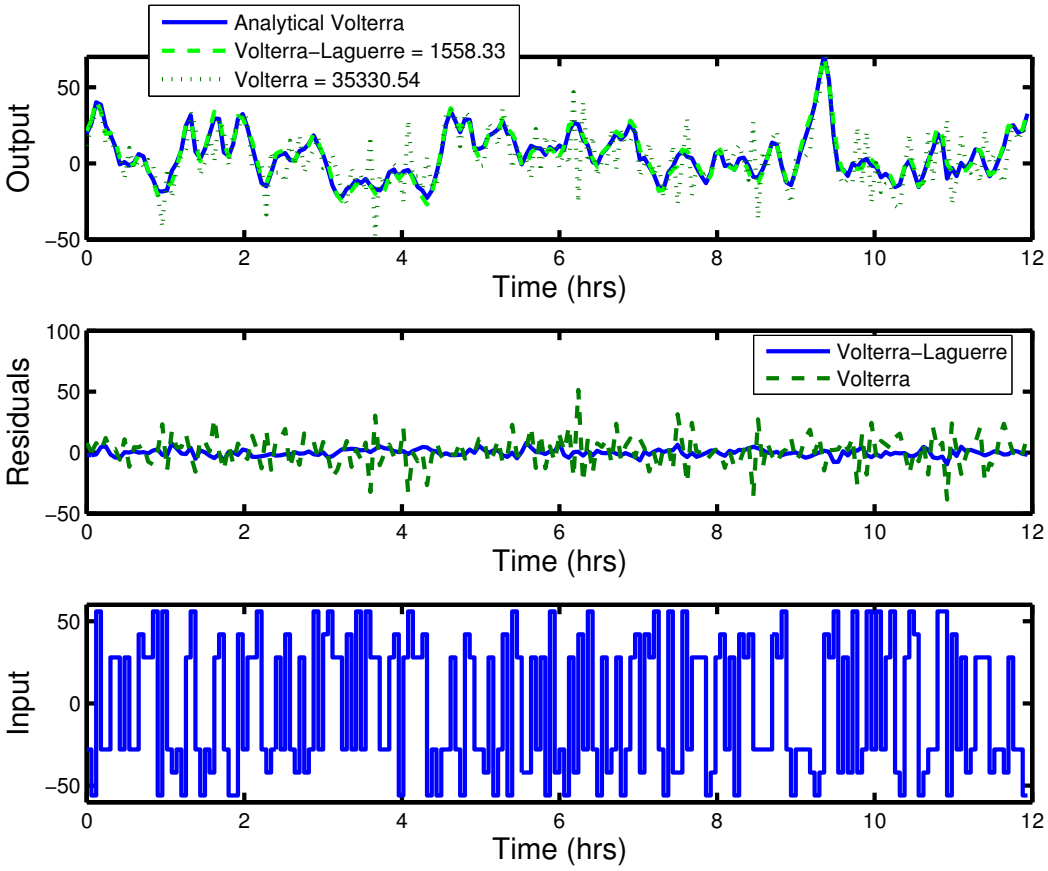

Figure 34: Validation performance of identified models with a random sequence in the projection of noise-corrupted Volterra kernels. Top: analytical Volterra (—), VL (− − −), and Volterra ($\cdots$) model outputs. The legend represents the SSE values between the analytically obtained Volterra kernels and the identified models. Middle: residuals between the analytical Volterra and the VL (—) and Volterra (− − −) models. Bottom: random input sequence used for validation. The input and the output are in scaled deviation form.

Furthermore, the improved performance is accompanied by a considerable reduction in the number of parameters.

3.3.2 Performance of the Volterra-Laguerre Model

Normally, the identification of the process model is carried out at a particular point in the operating space. This point is generally selected as the one where the nominal plant operation is carried out. The process model must be able to successfully represent the process behavior over the entire operating range. If the process is linear, then the model identified at a single operating point is valid over the entire range. However, since the identified VL model is nonlinear, the performance of the VL model over the operating range was analyzed. The identified models were first initialized to a steady state corresponding to an input of magnitude ν. The range for ν was $[-40\ , 40]$ in increments of 4 units. After reaching steady state, the process was subjected to a random input consisting of 220 data-points characterized by the following distribution:

$$u_{random}(k) = \begin{cases} \nu - 10 & \text{with probability } \frac{1}{3} \\ \nu & \text{with probability } \frac{1}{3} \\ \nu + 10 & \text{with probability } \frac{1}{3} \end{cases} \tag{3.24}$$

The first 20 points were used to initialize the input sequence and thereby eliminate coefficient error bias related to initial condition effects. In order to average out the effects of the random sequence, SSE results are reported for 100 runs in Table 11.

Figures 35 and 36 show the response of the analytically calculated Volterra, identified VL, and Volterra models, to a random sequence given by the distribution in (3.24), around steady states corresponding to inputs of magnitude 28 and -28, respectively. Once again, the nominal Volterra model outperforms the VL model. However, observing the plot of the residuals (Figures 35 and 36, middle) the VL model does an excellent job of capturing the analytical Volterra model behavior. In addition, the nominal performance of the VL models from both algorithms is comparable.

Figure 37 shows the SSE between the analytically calculated Volterra, and the VL Volterra models over the operating space. It is found that, the Volterra model is uniformly

Table 11: SSE values between the analytical Volterra and VL, and analytical Volterra and Volterra models around steady-states of ± 28 for the nominal case

	Algorithm 3		Algorithm 4	
Sequence	Volterra-Laguerre	Volterra	Volterra-Laguerre	Volterra
$\nu = 28$	35.45	0.44	36.96	0.13
mean	33.51	0.52	35.66	0.17
min	27.45	0.31	25.01	0.11
max	41.84	0.94	48.94	0.23
$\nu = -28$	56.92	1.33	53.45	0.07
mean	50.24	0.78	47.61	0.06
min	32.09	0.38	33.40	0.04
max	75.49	1.67	64.71	0.08

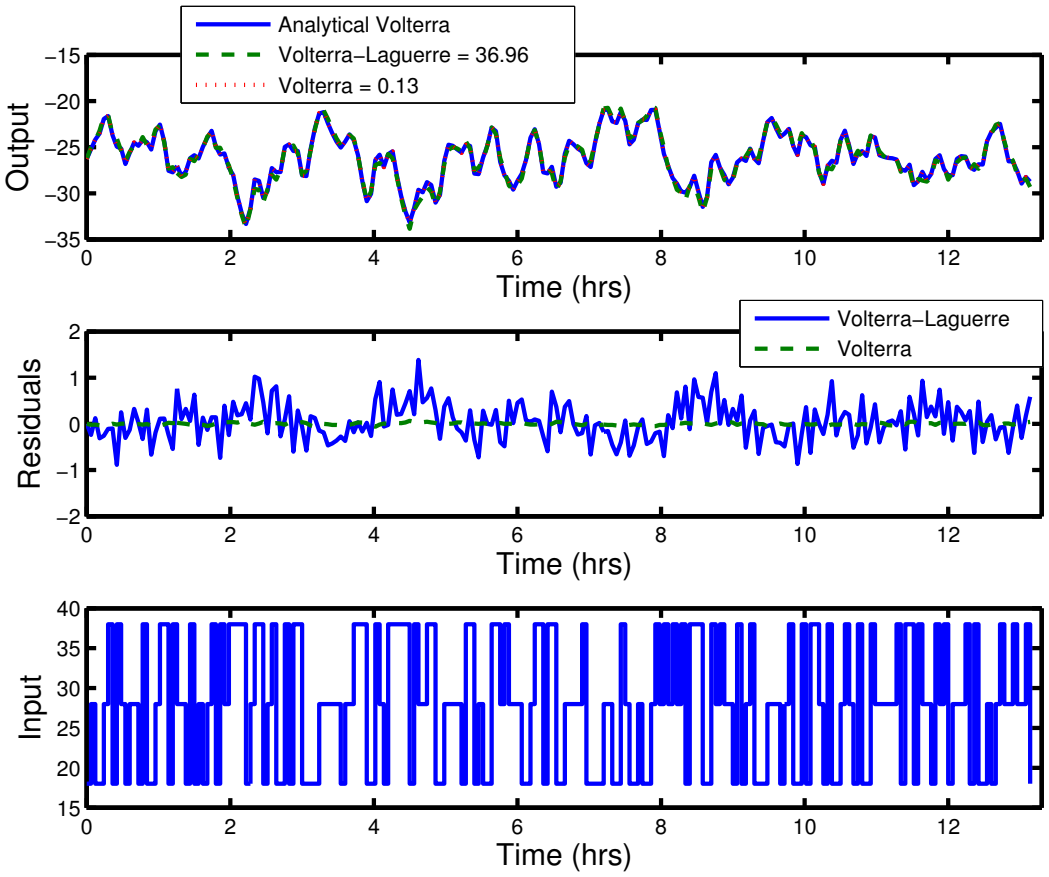

Figure 35: Validation performance of the identified models subjected to a random input around a steady-state corresponding to an input of $\nu = 28$. Top: analytical Volterra (—), VL (− − −), and Volterra ($\cdots$) model outputs. The legend represents the SSE values between the analytically obtained Volterra kernels and the identified models. Middle: residuals between the analytical Volterra and the VL (—) and Volterra models (− − −). Bottom: random input sequence used for validation. The input and the output are in scaled deviation form.

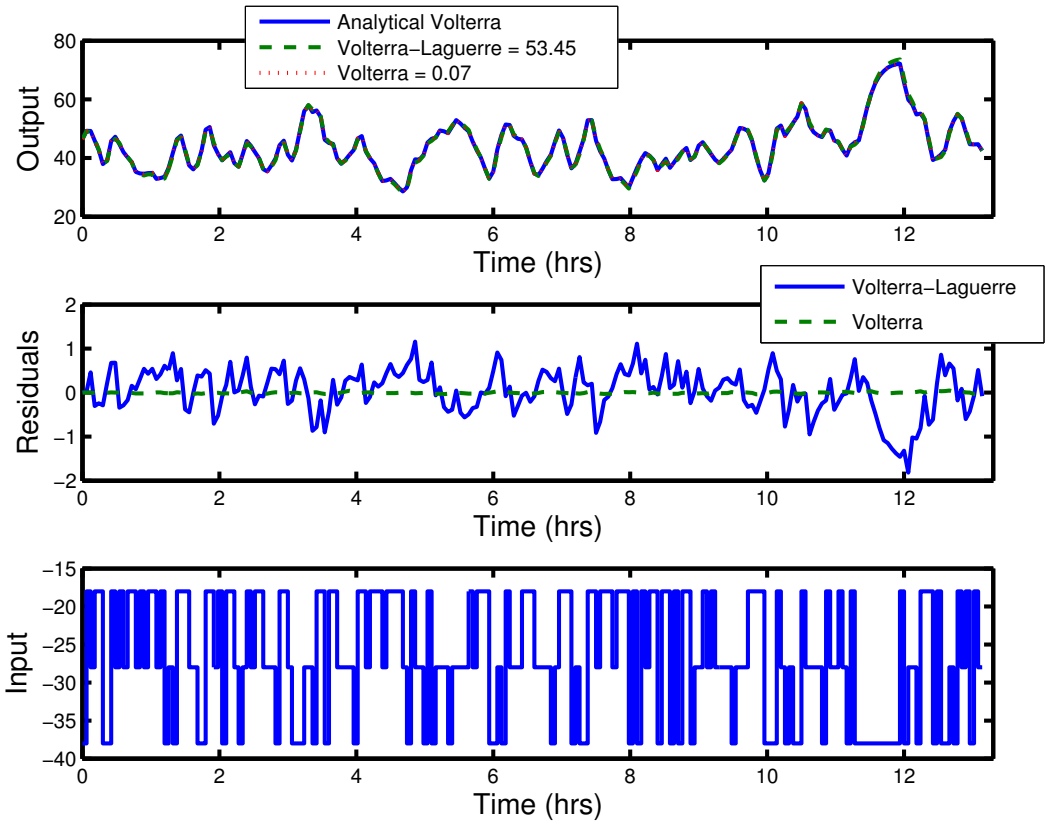

Figure 36: Validation performance of the identified models subjected to a random input around a steady-state corresponding to an input of $\nu = -28$. Top: analytical Volterra (—), VL (− − −), and Volterra ($\cdot\cdot\cdot$) model outputs. The legend represents the SSE values between the analytically obtained Volterra kernels and the identified models. Middle: residuals between the analytical Volterra and the VL (—) and Volterra models (− − −). Bottom: random input sequence used for validation. The input and the output are in scaled deviation form.

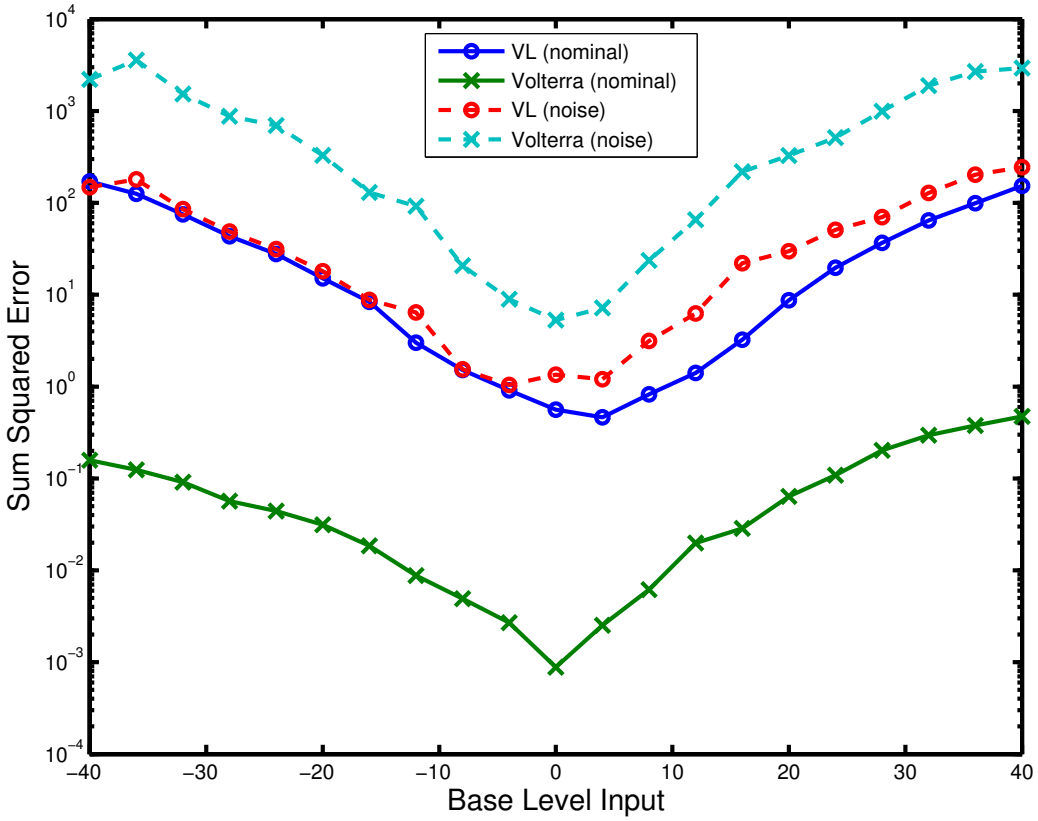

Figure 37: Variation in the SSE value between the analytically calculated and the VL (—o— nominal and −−o−− noise) and the Volterra (—x— nominal, −−x−− noise) model outputs for random inputs with steady-state varying in the range $[-40\ -36 \cdots 36\ 40]$

Table 12: SSE values between the analytically obtained Volterra kernels and the Volterra and VL models around steady-states of ± 28 for noise-corrupted Volterra kernels

	Algorithm 3		Algorithm 4	
Sequence	Volterra-Laguerre	Volterra	Volterra-Laguerre	Volterra
$\nu = 28$	40.88	752.77	76.19	792.42
mean	40.63	888.06	73.20	923.20
min	29.99	567.14	54.74	547.79
max	56.56	1481.23	106.57	1440.23
$\nu = -28$	71.02	956.76	59.41	961.35
mean	59.23	964.67	54.29	982.11
min	42.67	618.15	41.41	674.63
max	80.15	1472.37	73.59	1625.73

better than that of the VL model, for the nominal case. However, there is a qualitative similarity in the steady-state outputs from the Volterra and VL models. Once again, the excellent performance of the Volterra model is unsurprising given that analytically calculated Volterra kernel output is used to validate the identified Volterra and VL models.

For analysis with noise-corrupted Volterra kernels, the SSE values are shown in Table 12. It is observed that the VL model significantly outperforms the Volterra model. The system response to $\nu = \pm 28$ is shown in Figures 38 and 39, respectively. As before, noise is inherent in the identification of the Volterra kernels, and noise was not added in the projection of Volterra kernels onto the Laguerre space. The Volterra model performance decreases substantially in the presence of noise, and the sub-plot of the residuals confirms this observation. Compared to the nominal case, the performance of the VL models decreases only slightly. Comparing the means, the decrease in performance of VL models is 20% around a steady-state of $\nu = -28$, whereas the Volterra model shows a three orders of magnitude decrease in SSE performance, compared to the nominal case. Once again, the advantage of

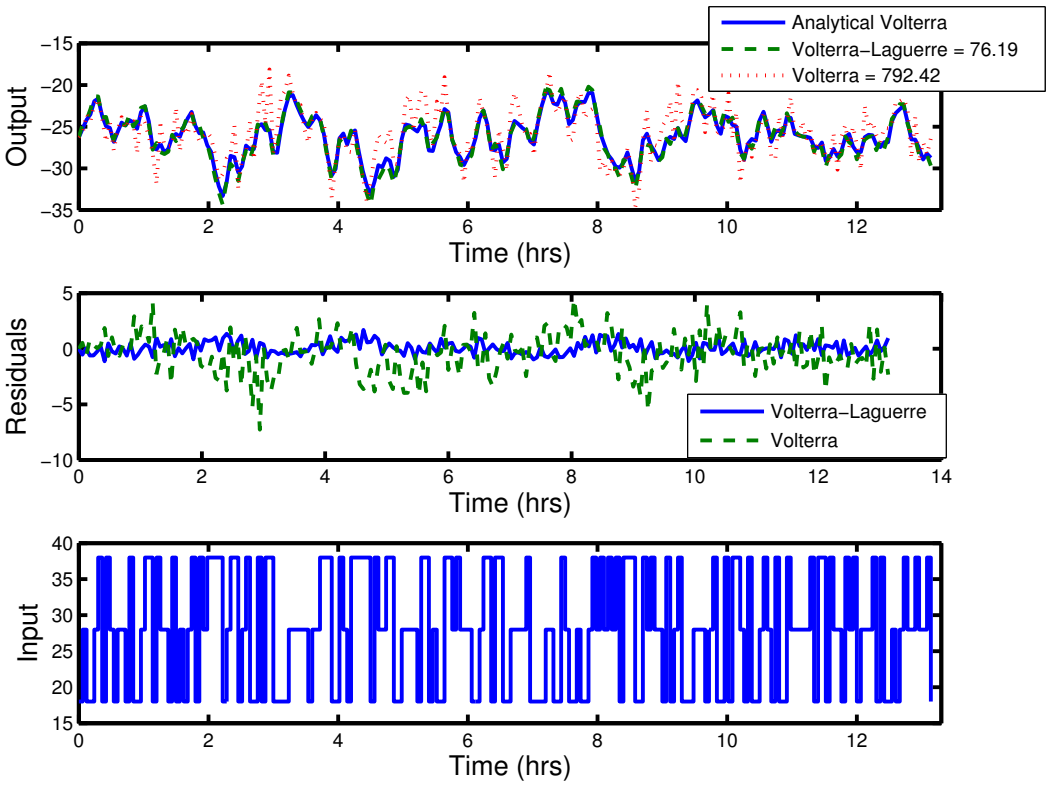

Figure 38: Validation performance of the identified noise-corrupted models subjected to a random input around a steady-state corresponding to an input of $\nu = 28$. Top: analytical Volterra (—), VL (– – –), and Volterra ($\cdots$) model outputs. The legend represents the SSE values between the analytically obtained Volterra kernels and the identified models. Middle: residuals between the analytical Volterra and the VL (—) and Volterra (– – –) models. Bottom: random input sequence used for validation. The input and the output are in scaled deviation form.

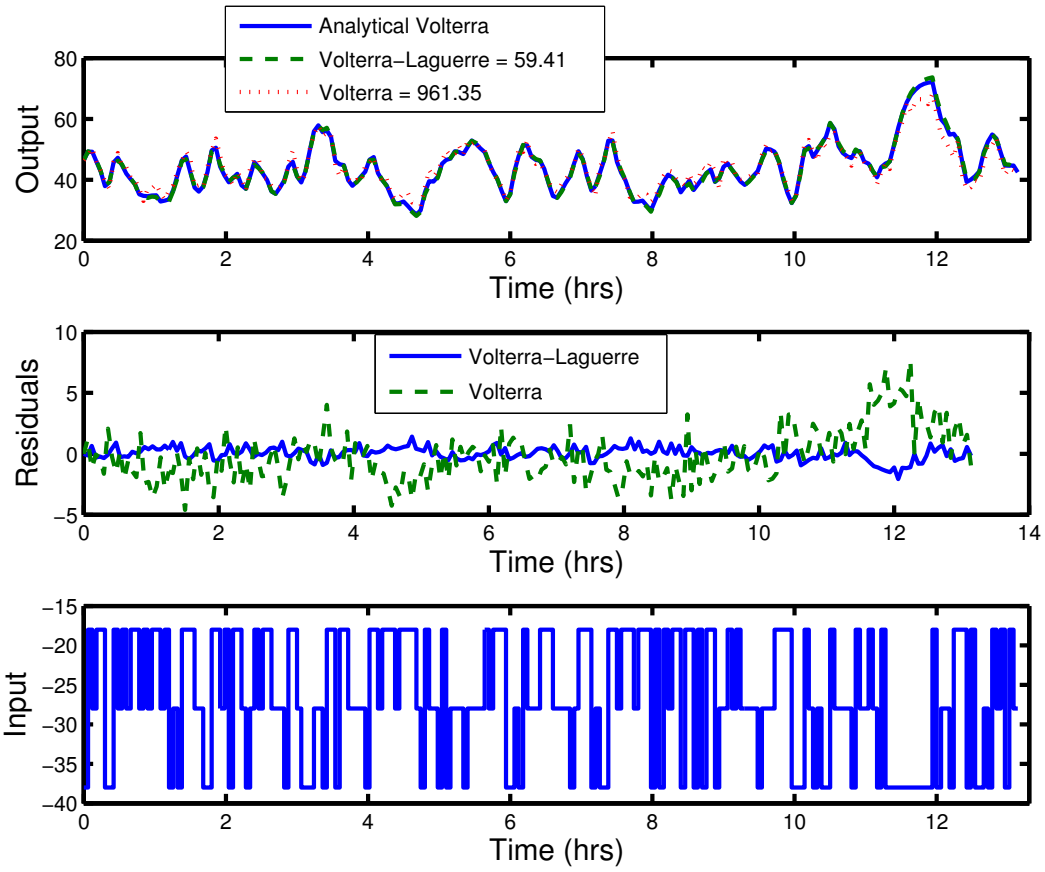

Figure 39: Validation performance of the identified noise-corrupted models subjected to a random input around a steady-state corresponding to an input of $\nu = -28$. Top: analytical Volterra (—), VL (− − −), and Volterra ($\cdots$) model outputs. The legend represents the SSE values between the analytically obtained Volterra kernels and the identified models. Middle: residuals between the analytical Volterra and the VL (—) and Volterra (− − −) models. Bottom: random input sequence used for validation. The input and the output are in scaled deviation form.

the VL models in the presence of noise is attributed to the noise filtering of the Laguerre model structure.

Similarly, Figure 37 shows that the VL model outperforms the Volterra model over the entire range of ν values in the noise case. Furthermore, there is minimal degradation in the performance of the VL models for the noise case. In contrast, the Volterra model shows a drastic decrease in validation performance over the range of input steady-states.

Thus, for the nominal case, both the Volterra and VL models identified at the nominal point given in Table 1 are valid over the entire operating range. There is a considerable decrease in performance of the Volterra model in the presence of noise, whereas the VL model showed excellent noise filtering characteristics.

3.3.3 Projection of a Reduced Volterra Model

In Section 3.3.1.1, the complete Volterra model was projected onto the Laguerre basis. It was shown that the resulting VL model was robust to measurement noise in the identification of the Volterra kernel. However, one key difficulty that still remains is the amount of data required to identify the complete Volterra model. The total number of unique coefficients in the full third-order Volterra model is $1,770$, and these required a considerable amount of identification data. The parameterization also made the Volterra model more susceptible to noise, as was observed in Section 3.3.1.2. One way to reduce the amount of data required would be to identify only certain structural components of the Volterra model. A similar approach was also used in [48, 86]. Both Pearson et al., and Florian Jr. and Parker identified only the linear and diagonal components of a second- and third-order Volterra model, respectively.

Different combinations of the Volterra kernels could be projected onto the Laguerre space, *e.g.*, $L+D_2+D_3$, $L+D_2+D_3+S_3$, $L+D_2+D_3+O_2+O_3$, $L+D_2+D_3+S_3+O_2+O_3$, *etc.* Of these combinations, $L+D_2+D_3$, represents the most drastic reduction in the identification data requirement, while preserving the third-order model structure. Thus, only the linear and nonlinear diagonal kernels, representing 60 of the $1,770$ parameters, were projected onto the Laguerre basis using the projection algorithm in Section 3.2.3. The training data-set was

Table 13: Optimal Laguerre parameters from the projection algorithm using the linear plus nonlinear diagonal Volterra kernels

	Reduced Volterra-Laguerre Model	
	Nominal	Noise
Training SSE	0.0886	0.3362
L	2	2
α	0.5697	0.5762

the same as that used in Section 3.3.1.1 and the results are shown in Table 13. Since the linear and nonlinear diagonal kernels are the same for both Algorithm 3 and Algorithm 4, the results from the projection algorithm are identical. Hence only one resultant VL model is obtained. A representative plot of the residuals, SSE, and AIC value for the noise-corrupted Volterra kernels is shown in Figure 40. Unlike the projection of the full Volterra model, a distinct minimum is obtained in AIC at $L = 2$. The SSE sub-plot (Figure 40, middle) shows that the SSE value is fairly small for $L = 2$. Figure 41 shows the contribution of the two terms that constitute the AIC. Only 60 Volterra kernel coefficients were projected. This decrease in the value of K ensures that the first term does not dominate the contribution of the second term. As a result, the addition of subsequent filters is penalized, and a minimum in AIC is achieved at 2 filters. Furthermore, on comparing with Tables 7 and 9, it is observed that the location of the optimal pole does not shift significantly, for either the nominal or the noise case. The validation results for the reduced VL model forced with a random multi-level sequence with the same distribution as in (3.23) are shown in Table 14. Once again, in order to average out the effects of the random sequence, 100 runs of the random sequence were used. The first row in Table 14 represents results for the training data-set, whereas the subsequent rows represent results for the random multi-level sequence. Figures 42 and 43 present the outputs for the analytically obtained Volterra kernels, reduced Volterra, and reduced VL models for the nominal and noise cases, respectively. For the random sequence,

Table 14: SSE values between the analytically obtained Volterra kernels and the identified Volterra and VL models for the training and random input sequences

	Nominal		Noise	
Sequence	Volterra-Laguerre	Volterra	Volterra-Laguerre	Volterra
Training	0.09	0.00	0.15	0.24
Random	3166.68	3655.85	3200.03	3650.79
mean	3507.92	3552.16	3795.40	3772.10
min	1268.78	1706.31	1782.40	1932.52
max	7405.84	7852.58	13923.78	15737.36

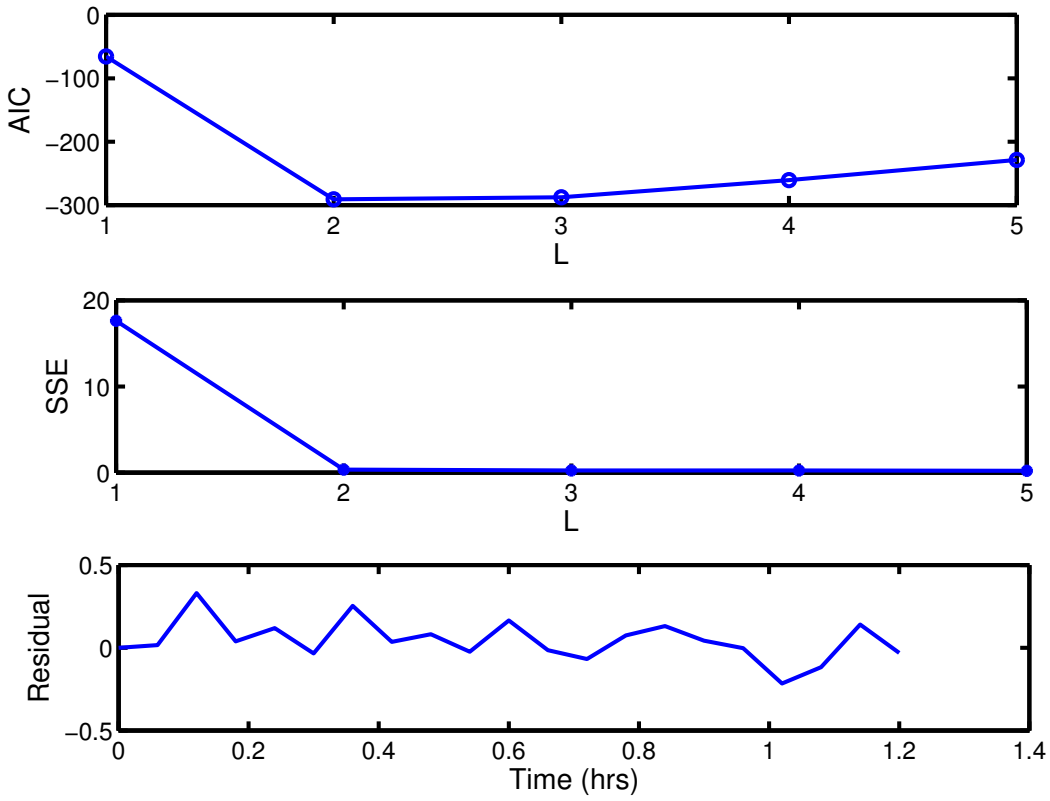

Figure 40: Projection of noise-corrupted linear and nonlinear diagonal Volterra kernels onto the Laguerre basis. AIC (top), SSE (middle), and the residuals (bottom) between the Volterra and VL model outputs. An impulse signal of magnitude 28 was used as the input to generate the training data-set. The input and the output are in scaled deviation form.

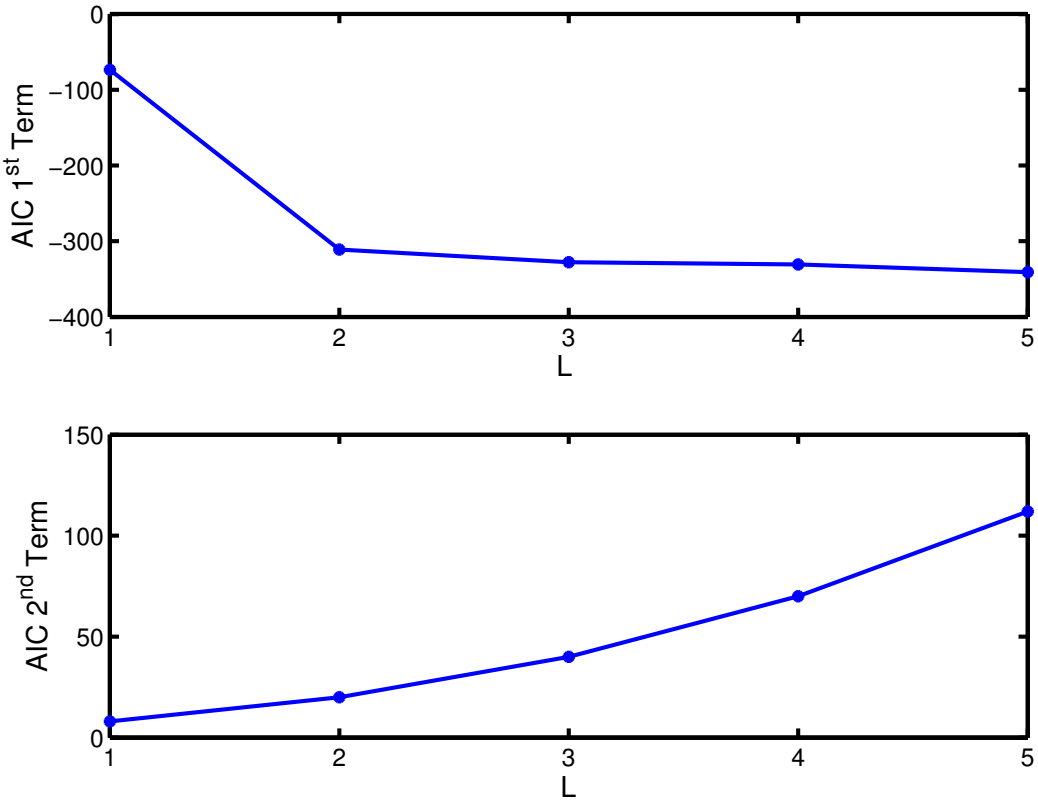

Figure 41: Relative contributions of the two terms that make up the AIC; first term (top) and second term (bottom), for the projection of reduced Volterra kernels onto the Laguerre basis.

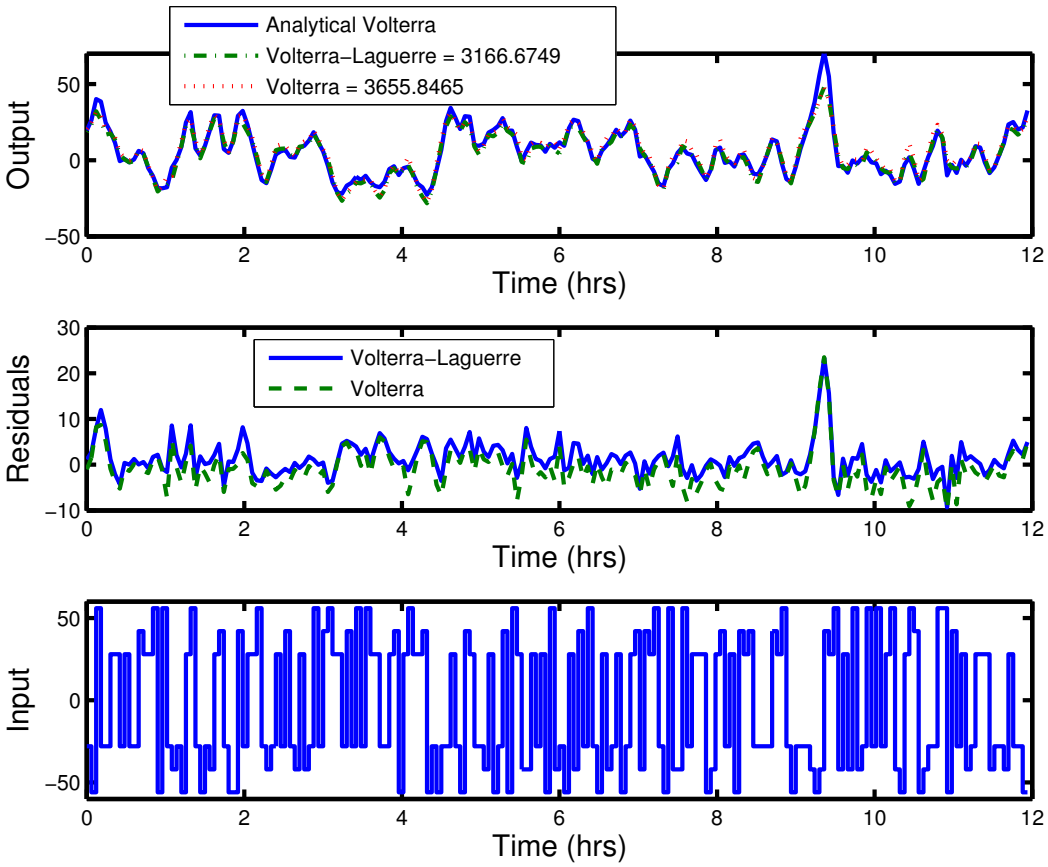

Figure 42: Validation performance of the reduced models (nominal case) in response to a random input sequence. Top: analytical Volterra (—), reduced VL (− − −), and reduced Volterra (···) model outputs. The legend represents the SSE values between the analytically obtained Volterra kernels and the identified models. Middle: residuals between the analytical Volterra and the reduced VL (—) and reduced Volterra (− − −) models. Bottom: random input sequence used for validation. The input and the output are in scaled deviation form.

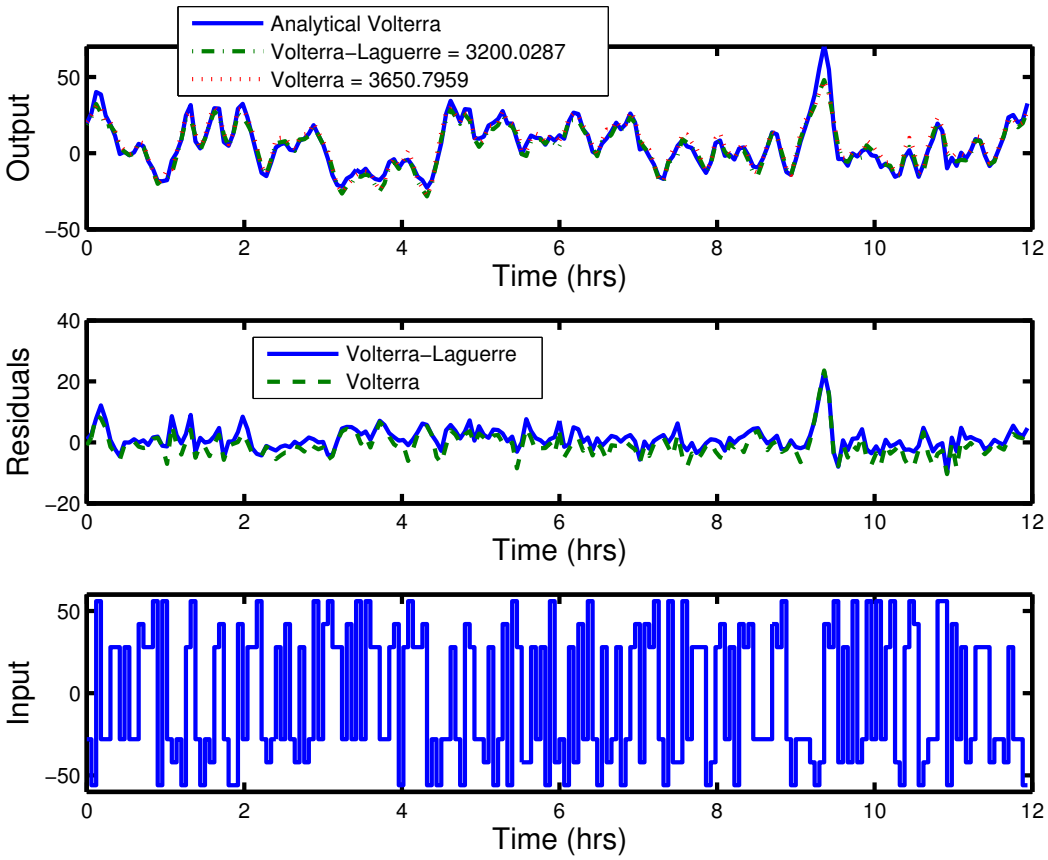

Figure 43: Validation performance of reduced noise-corrupted models in response to a random input sequence. Top: analytical Volterra (—), reduced VL (− − −), and reduced Volterra (· · ·) model outputs. The legend represents the SSE values between the analytically obtained Volterra kernels and the noise corrupted identified models. Middle: residuals between the analytical Volterra and the noise corrupted reduced VL (—) and reduced Volterra (− − −) models. Bottom: random input sequence used for validation. The input and the output are in scaled deviation form.

the mean of 100 runs for both the nominal and noise cases yields comparable performance for the VL and Volterra models. The advantage of the Volterra model is now not as pronounced owing to the bias error that arises from using only the linear and nonlinear diagonal kernels.

The performance of both reduced models is considerably better than that of the full noise-corrupted Volterra model (SSE of 3795.40 for VL, 3772.10 for Volterra, versus $19,227$ for the best full Volterra model). This is an extremely important result as it suggests that reduction in the number of parameters yields a considerable improvement in performance in the presence of noise. The error in the noise-corrupted full Volterra model is associated with large magnitude input excitation of higher-order noise-sensitive kernels, *e.g.,* O_2, S_3, and especially O_3. In addition, there is a significant reduction in the number of Volterra coefficients that require identification (60 versus $1,770$). Since noise, (*e.g.*, measurement noise, is prevalent in most real physical systems, it is advisable to use a VL model due to its inherent noise filtering capabilities. Furthermore, the reduced parameterization of the resultant VL models makes them attractive candidates for use in model-based control. Thus projection of the Volterra kernels onto the Laguerre basis provides considerable benefits.

3.4 SUMMARY

In this chapter the problem of projecting the Volterra kernels onto the Laguerre basis was addressed. This projection was accomplished in an optimal manner, using AIC as the criterion for optimality. The AIC incorporates terms for both the accuracy of projection onto the Laguerre basis and the size of the resulting Volterra-Laguerre model. It was found that only two filters were required to capture the complete nonlinear reactor behavior and excellent validation performance was achieved. In addition, the location of the optimal Laguerre pole was found to be relatively insensitive to the presence of noise. The excellent noise filtering capabilities of the Volterra-Laguerre model were also evident when the model was used to track random inputs about different steady-state operating points over the reactor operating range. A reduced VL model was also obtained by projecting only the linear

and nonlinear diagonal kernels. Similar to the full VL model, the number of Laguerre filters and the location of the Laguerre pole in the linear plus nonlinear diagonal case were found to be insensitive to the presence of noise. Although the reduced VL model was outperformed by the full Volterra model in the open-loop noise-free case, the true test of practical utility of the reduced models is their performance in closed-loop. Evaluation of the closed-loop performance for both the full and reduced VL models forms the basis of the next chapter.

4.0 CLOSED-LOOP PERFORMANCE OF VOLTERRA-LAGUERRE MODELS

For process control applications, it may not be necessary for a model to describe the complete set of dynamic process behaviors. As an example, Balasubramhanya and Doyle III [22] used traveling wave phenomena to develop a low-order nonlinear model that captured the dynamic behavior of a high purity distillation column to sufficient accuracy for use in model-based control strategies. As discussed in Chapter 1, such reduced models are known as control-relevant models. In this chapter the performance of the reduced VL model is analyzed in closed-loop to assess its applicability as a control-relevant model. In Section 3.3.3, it was observed that while the reduced Volterra-Laguerre models showed adequate nominal performance and low noise sensitivity, their performance was compromised when compared with a full Volterra-Laguerre model. However, the number of coefficients that need to be identified is significantly reduced from 1771 for the full, to just 60 for the reduced Volterra model. In this Chapter the closed-loop performance of the reduced VL models is compared to the full VL model, and the capability of the reduced VL model to serve as a control-relevant model is assessed.

4.1 VOLTERRA-LAGUERRE MODELS FROM SIMULATED POLYMERIZATION REACTOR DATA

In Chapter 3, VL models were constructed by projecting Volterra kernels identified using data from simulating analytical Volterra kernels. This provided the best way to evaluate the performance of the projection algorithm, as the analytical Volterra kernels were calculated

Table 15: Optimal Laguerre parameters from the projection algorithm using Volterra kernels identified from simulated polymerization reactor data

	Full Volterra-Laguerre Model		Reduced Volterra-Laguerre Model	
	Nominal	Noise	Nominal	Noise
Training SSE	0.0039	0.1444	0.1505	0.3287
L	2	2	2	2
α	0.5651	0.5587	0.5718	0.5658

explicitly. However, in practice analytical Volterra kernels are rarely, if ever, available. Thus the performance of the projection algorithm is evaluated in this chapter by projecting Volterra kernels identified using data from the simulated polymerization reactor (Equation 2.1). Since kernels from Algorithm 4 require the least amount of data to be identified, these full kernels are projected onto the Laguerre basis. Reduced kernels, *i.e.,* only the linear and nonlinear diagonal kernels, are also projected for comparison.

An impulse input of magnitude 28 was used to generate the training data-set. The results from the projection algorithm (Section 3.2.3), are shown in Table 15 for the full and reduced Volterra kernel projection. In this table, noise refers to the projection of noise-corrupted Volterra kernels, and no noise was added during the projection of Volterra kernels onto the Laguerre basis. Two Laguerre filters were found to be sufficient. For both the full and reduced Volterra model projection, the location of the optimal α did not vary significantly from the nominal to the noise case. The training SSE results are not surprising. The SSE values for both the full and reduced Volterra projection were higher in the noise than in the nominal case. Also, because of the bias error introduced from using only the linear and nonlinear diagonal kernels, the training SSEs for the reduced Volterra kernel projection were greater than their full Volterra model counterparts.

Table 16: SSE values between simulated polymerization reactor NAMW and the identified VL model predictions using a random input sequence (3.23).

	Nominal				Noise			
	Full		Reduced		Full		Reduced	
	Volterra	VL	Volterra	VL	Volterra	VL	Volterra	VL
mean	193.17	183.1205	3911.28	3728.18	24,627.06	306.18	3995.88	3712.46
min	63.79	52.64	985.94	702.10	13881.8	77.97	1058.95	579.77
max	697.97	713.78	15996.01	15600.50	35702.70	2520.39	25665.13	24191.40

Validation of the identified VL models was also carried out using a random sequence consisting of 220 data-points. This sequence had the distribution as in (3.23). The first 20 points were used to initialize the input sequence and thereby eliminate coefficient error bias related to initial condition effects. To average out the sequence effects due to the random input, results from 100 input sequences given by the distribution in (3.23), were considered. The range and mean of the validation results from the 100 sequences are presented in Table 16 and representative results are shown in Figures 44 and 45 for the nominal and the noise cases, respectively. In these two figures, and in all subsequent figures in this chapter, the inputs and outputs are given in scaled deviation form. The full VL model does an admirable job of capturing the nonlinear reactor behavior. The performance of the reduced VL model degrades slightly, owing to the bias error from using only the linear and nonlinear diagonal kernels. A decrease in performance is observed for both the full and reduced VL models obtained from noise-corrupted Volterra kernels, when compared to their nominal counterparts. Overall, all of the VL models outperform their Volterra model counterparts with the difference in performance more pronounced for the noise case.

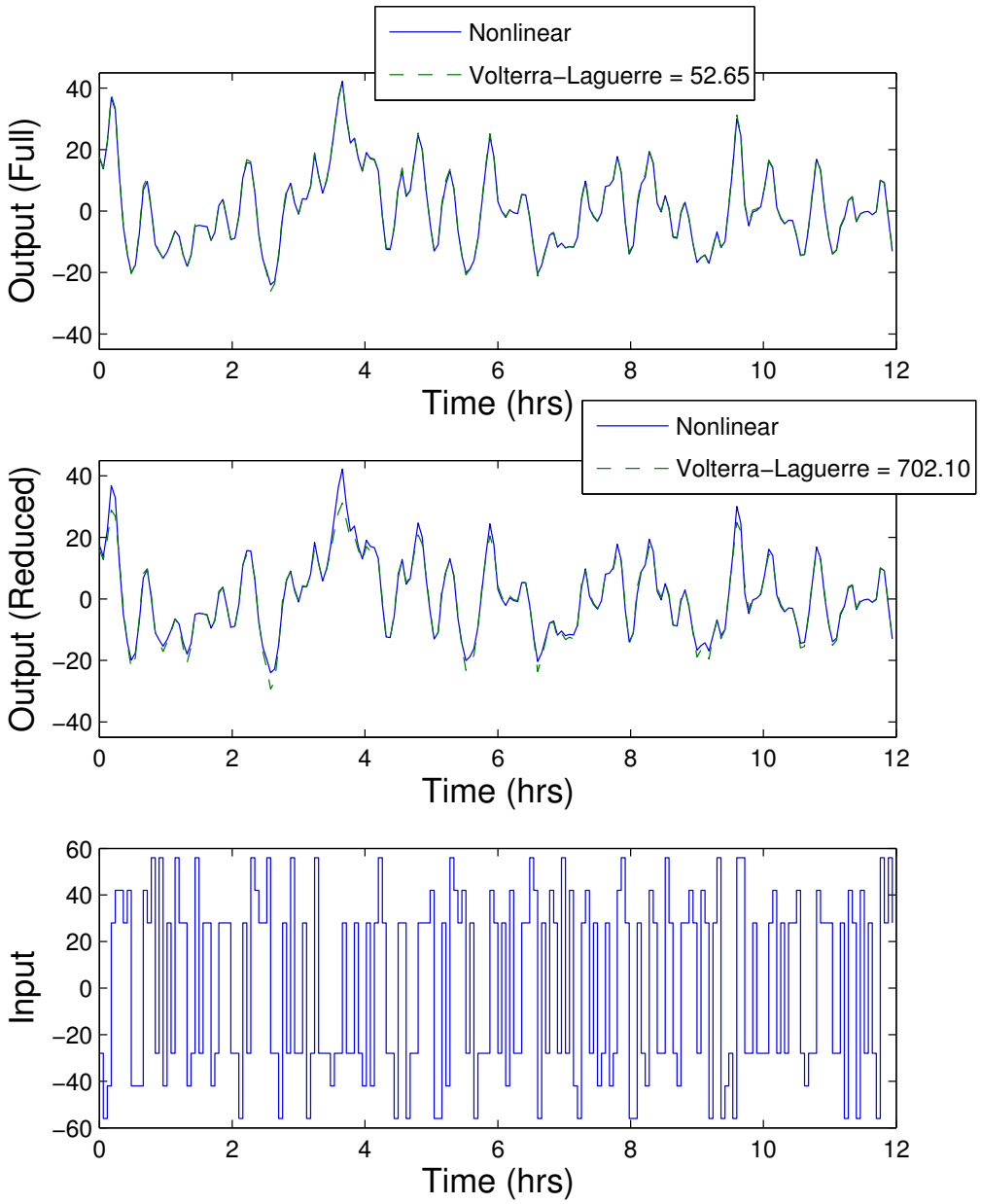

Figure 44: Validation of the identified VL model (nominal case) using a random sequence. Top and middle: nonlinear (solid) and full VL (top, dashed) or reduced VL (middle, dashed) model outputs. The legend represents the SSE value between the nonlinear and the identified VL model. Bottom: random six-level sequence used for validation. The input and the two outputs are in scaled deviation form.

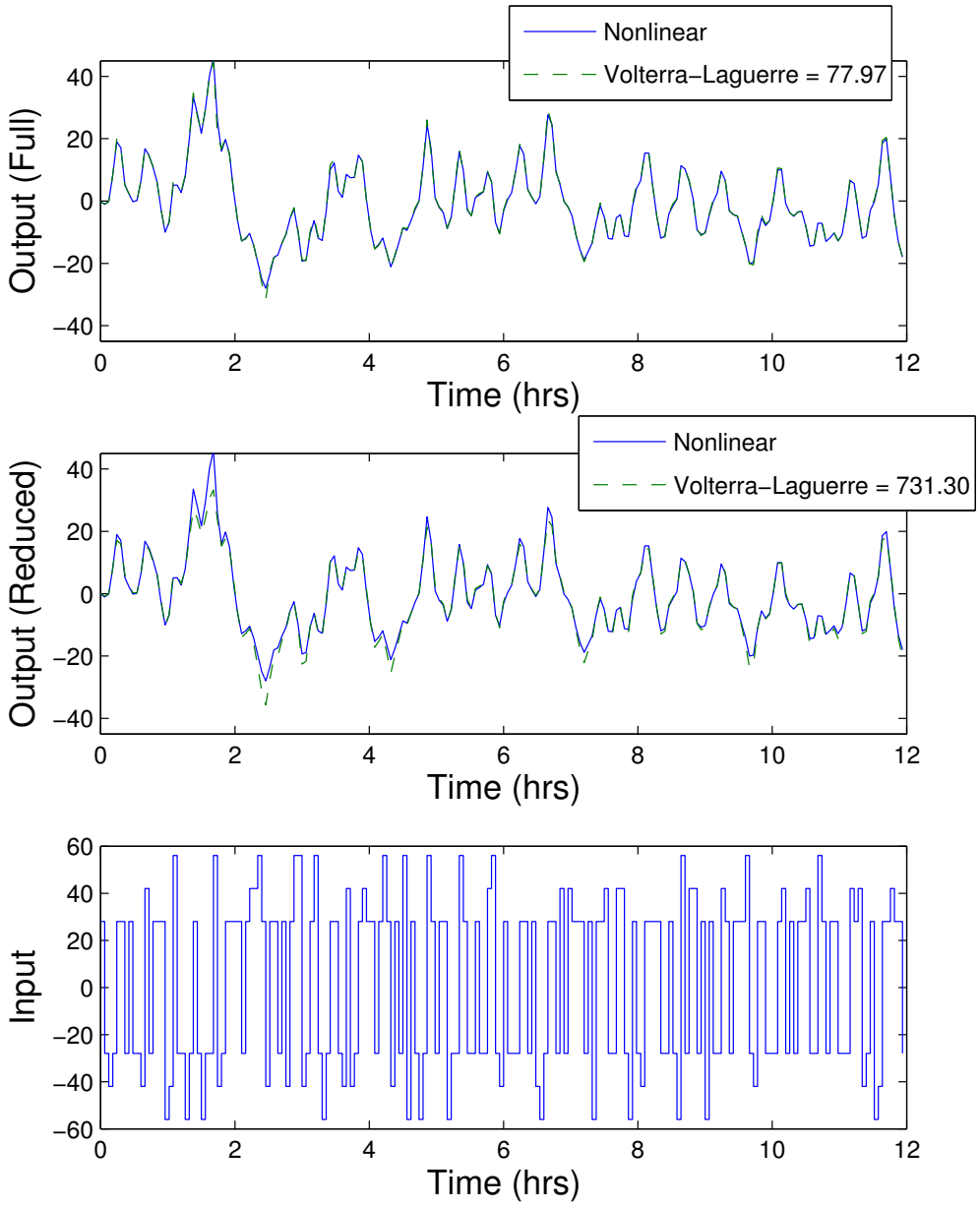

Figure 45: Validation of the identified VL model obtained from noise-corrupted Volterra models, using a random sequence. Top and middle: nonlinear (solid) and full VL (top, dashed) or reduced VL (middle, dashed) model outputs. The legend represents the SSE value between the nonlinear and the identified VL model Bottom: random six-level sequence used for validation. The input and the two outputs are in scaled deviation form.

So far, only open-loop performance of the identified VL models was considered. Since an important test of model quality is the performance of the identified model in closed-loop, the next section discusses the closed-loop performance of the full and reduced VL models.

4.2 CLOSED-LOOP PERFORMANCE OF VOLTERRA-LAGUERRE MODELS

4.2.1 Model Predictive Control

Model predictive control refers to a broad class of control schemes in which a controller uses a process model to generate a manipulated variable profile that minimizes an open-loop performance objective over a future prediction horizon. This optimization uses a model prediction of the future plant behavior. The manipulated variable move thus obtained is implemented on the process until a new measurement becomes available. Process measurements are used to update the optimization at the next time interval, thus incorporating feedback in the control scheme. Optimization of the open-loop performance objective over the prediction horizon, at each time-step, is the defining feature of MPC which separates it from other control schemes [87, 88]. Mathematically, the MPC problem can be represented as:

$$
\begin{aligned}
\min_{\Delta u(k|k)} \quad & J_{MPC}(\Delta u(k|k)) \\
\text{s.t.} \quad & x(k+1) = f(x(k), u(k)) \\
& y(k) = g(x(k), u(k)) \\
& \varsigma(x(k), u(k)) = 0 \\
& \varphi(x(k), u(k)) \geq 0 \\
& x(k=0) = x_0 \\
\text{where} \quad & k \in [t_0, t_0 + p\Delta t_s]
\end{aligned}
\tag{4.1}
$$

The input and output are given by u and y respectively. The time interval is from the current time, t_0, to $t_0 + p\Delta t_s$, where p is the prediction horizon and Δt_s is the sampling time. The scalar function J_{MPC} serves as the performance objective for the optimization problem. The functions f and g describe the process, whereas ς and φ are input and state equality and inequality constraints, respectively.

Figure 46 presents a schematic of the MPC algorithm implementation. The first part of the MPC algorithm is the specification of the reference trajectory, which may be as simple as a step change to a new set-point or, as is common in batch processes, a trajectory that the system should follow. At the present time, k, the reference trajectory has a value $r(k)$ and a predicted value of $r(k+1|k)$. Also at k, consider the predicted process output over a future prediction horizon, p. A model of the process is used to obtain the projected behavior of the output over the prediction horizon by simulating the effects of past inputs applied to the actual process (value $\hat{y}(k)$ at the current time). The same model is used to calculate a sequence of m current and future manipulated variable moves, in order to satisfy a user-specified objective function. Here, m is the move horizon. A common objective function is to minimize the sum of squared deviations of the predicted controlled variable values from a (possibly time-varying) reference trajectory, over the prediction horizon, based on system information available at the current time, k. The minimization also takes into account constraints that may be present on the state and/or the manipulated variable. Conceptually, the problem is similar to constructing and utilizing the relaxed inverse of the controller model to determine the sequence of m moves that most closely achieve the specified output behavior over the predicted horizon [88].

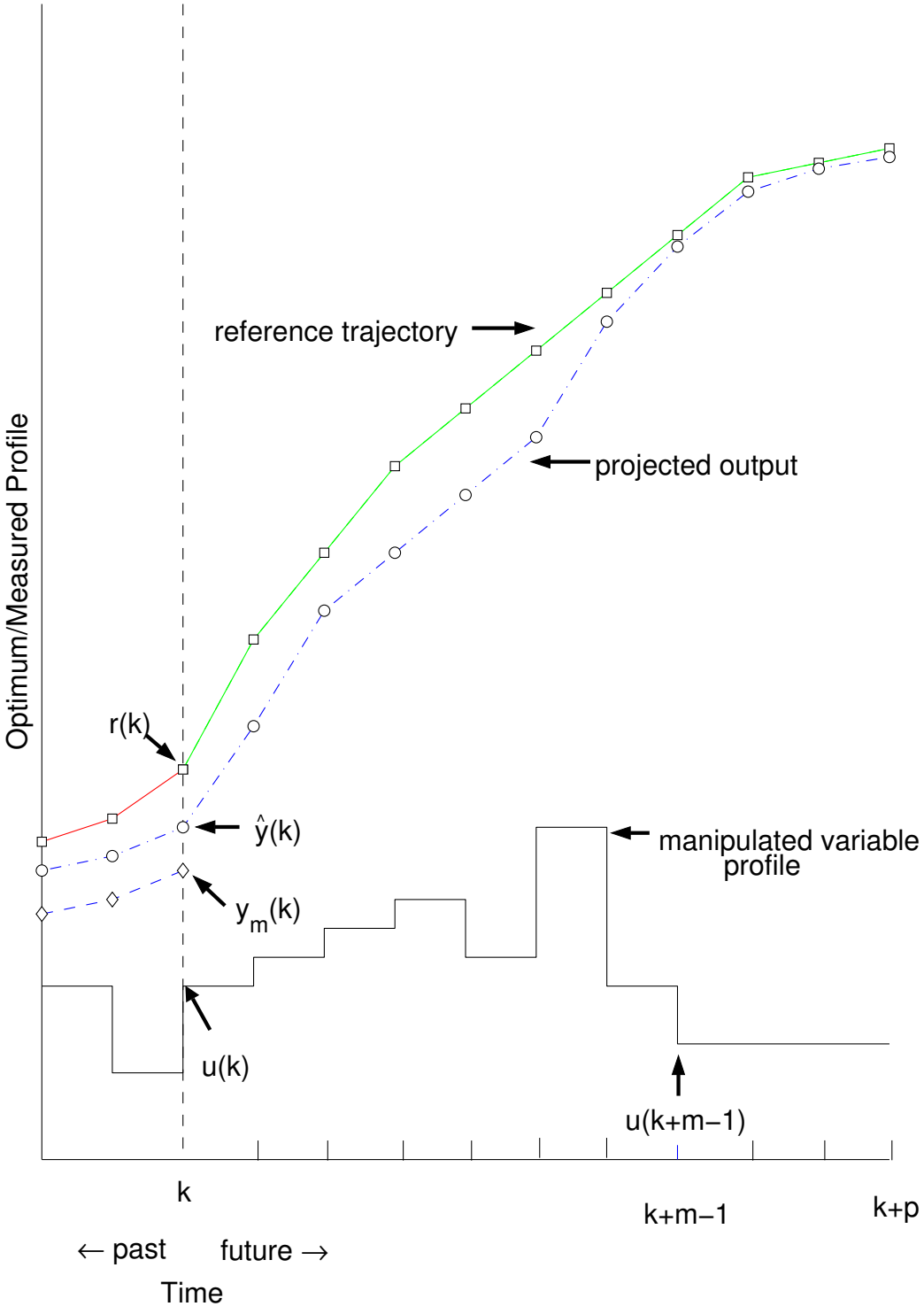

Figure 46: MPC algorithm schematic (adapted from [88])

Due to unmodeled disturbances and modeling errors, there may be deviations between the actual observed output, $y_m(k)$, and the predicted output behavior. Due to this deviation, the computed future manipulated variable moves may no longer be appropriate, and hence, only the first of the computed manipulated variable moves, $\Delta u(k|k)$, is implemented on the actual process. The error, $d(k|k) = y_m(k) - \hat{y}(k|k-1)$, is calculated and is used to update the future predictions. At the next time instant, $k+1$, the process measurement is again taken and the horizon is shifted forward by one step. Based on this new horizon, and using the updated system information, the optimization is carried out again at the next time step, and the algorithm repeats.

4.2.1.1 Linear and Nonlinear Model Predictive Control MPC algorithms have been classified most generally on the basis of the type of controller used. In the most general implementation of MPC, the model is a nonlinear function of the system states, manipulated inputs, and/or outputs. The objective function and system constraints could be nonlinear as well, and this particular implementation is known as Nonlinear Model Predictive Control (NMPC). Linear Model Predictive Control (LMPC) refers to the particular case when the controller employs a linear model. A quadratic objective function is commonly used, which is one of the most convenient forms for the on-line optimization problem; the input and output constraints, if present, are linear in the decision variables. However, as pointed out by Biegler and Rawlings, the constrained LMPC problem is no longer linear [89].

In the case of LMPC, the most commonly used model form is the discrete-time finite convolution model. This convolution model is based on either step or impulse response models in which case the output is represented as:

$$\begin{aligned} y_{impulse}(k) &= \sum_{i=0}^{M} h(i)u(k-i) && \text{FIR or Linear Volterra Model} \\ y_{step}(k) &= \sum_{i=0}^{M} s(i)\Delta u(k-i) && \text{Step Response Model} \\ \Delta u(k) &= u(k) - u(k-1) \end{aligned}$$

Here the parameters $h(i)$ and $s(i)$ are the impulse and step response coefficients, respectively, which are usually obtained from plant data. The coefficients in $h(i)$ are, in fact, the first

order Volterra model coefficients. The model memory, M, denotes the window over which past inputs affect the output significantly, and it is the same as the Volterra model memory. Other model forms that have been used are the state-space format and the discrete transfer function format, and a description of these model forms can be found in [88].

Considering NMPC, a nonlinear optimization problem needs to be solved at every time step. The general methodology is to convert the objective function and the constraints into a nonlinear programming problem. This approach has been used by a number of authors [90, 91, 92, 93, 94], and a representative example is the control of a CSTR by Eaton and Rawlings [95]. However, NMPC can be computationally demanding, and the optimization problem may not be strictly convex, so that the solution may converge to local minima. If the algorithm does converge to the global minimum, it may take a long time to do so [69]. This makes the on-line implementation of NMPC a non-trivial task. However, by properly selecting the model structure for control, the optimization problem can be greatly simplified. Using the Volterra-Laguerre model, several researchers have developed analytical solutions to the optimization problem under special conditions [15, 36, 96, 97]. Obtaining a global analytical solution to the nonlinear optimization problem is an area of on-going research.

4.2.2 Model Predictive Controller Design

In this section, an on-line closed-loop controller is designed using the Model Predictive Control framework. The essential components of this strategy are summarized below.

Objective Function: The 2-norm squared objective function used in this work is of the following form:

$$\min_{\Delta\mathcal{U}(k|k)} \left\{ \left|\left| \Gamma_y \left[\hat{\mathcal{Y}}(k+1|k) - \mathcal{R}(k+1|k) \right] \right|\right|_2^2 + \left|\left| \Gamma_u \Delta\mathcal{U}(k|k) \right|\right|_2^2 \right\} \tag{4.2}$$

In this equation, the first term denotes the projected deviations of the controlled variable, $\hat{\mathcal{Y}}(k+1|k)$, from the reference trajectory, $\mathcal{R}(k+1|k)$, (both of which are vectors of length p for the single input single output polymerizer case study); the second term penalizes the manipulated variable moves ($\Delta\mathcal{U}(k|k)$, length m). The relative importance of trajectory tracking and control effort can be tuned by altering the values of the diagonal weights, Γ_y and Γ_u, respectively.

Specification of the Reference Trajectory: The reference trajectory is generally user specified, and in the case of the polymerization reactor, it could be a step-up or a step-down to a polymer with a different NAMW.

Output Trajectory Prediction: The process is described by the set of coupled nonlinear differential equations given by (2.1). In order to design a suitable controller for this process, the controller requires a model to predict the process behavior. The VL model given by (3.6) and (3.7) serves as the model for controller design. Based on the VL model, future values of the Laguerre state variable can be calculated with a knowledge of the current state, the past input, and the future input changes.

Considering the input, the controller determines the moves extending until the control horizon, *i.e.,* the next m moves only, so that the vector of current and future manipulated variables is given as:

$$\Delta\mathcal{U}(k|k) = [\Delta u(k|k)\ \Delta u(k+1|k)\ ....\ \Delta u(k+m-1|k)]^T \tag{4.3}$$

Beyond m, the input remains constant so that $\Delta u(k+j|k) = 0 \ \ \forall\ j \geq m$.

Equation (3.6) can be re-written in terms of the past input and the current input change as:

$$\ell(k+1|k) = A\ell(k) + Bu(k-1) + B\Delta u(k|k) \tag{4.4}$$

For the next step, the Laguerre states can be written as follows:

$$\begin{aligned}
\ell(k+2|k) &= A\ell(k+1|k) + Bu(k) + B\Delta u(k+1|k) \\
&= A[A\ell(k) + Bu(k-1) + B\Delta u(k|k)] + B[u(k-1) + \Delta u(k|k)] + B\Delta u(k+1|k) \\
&= A^2\ell(k) + (A+I)Bu(k-1) + (A+I)B\Delta u(k|k) + B\Delta u(k+1|k)
\end{aligned} \tag{4.5}$$

Generalizing, the state-space model for future time-steps can be written as:

$$\begin{aligned}
\ell(k+i|k) &= A^i\ell(k+i|k) + \bar{A}_i u(k-1) + \sum_{j=0}^{i-1} \bar{A}_{i-j} B\Delta u(k+j|k) \\
\text{where}\ \ \bar{A}_j &= A^{j-1} + A^{j-2} + \cdots + I
\end{aligned} \tag{4.6}$$

The future output in terms of the future Laguerre states can be written as:

$$\hat{y}_l(k+i|k) = C^T\ell(k+i|k)+\ell^T(k+i|k)D\ell(k+i|k)+\sum_{j=1}^{L}[\ell(k+i|k)^T E_j\ell(k+i|k)]\ell_j(k+i|k) \quad (4.7)$$

The controlled variable in (4.2) can now be written as:

$$\hat{\mathcal{Y}}(k+1|k) = [\hat{y}_l(k+1|k)\ \hat{y}_l(k+2|k)\ \cdots \hat{y}_l(k+p|k)]^T \quad (4.8)$$

The unmodeled effects at the current sampling time are computed as the difference between the plant measurement obtained from the actual nonlinear model, $y_m(k)$, and the model prediction, $\hat{y}_l(k|k-1)$. In the absence of any further information about these disturbances at future times, it is assumed that the predicted values of the disturbances are equal to the current values over the prediction horizon (the so-called "stepwise constant" assumption). Using (4.6) and (4.7), the objective function in (4.2) is expressed in terms of the current Laguerre state $\ell(k)$, the past input $u(k-1)$, the current and future reference $\mathcal{R}(k+1|k)$, and the future manipulated variable changes $\Delta\mathcal{U}(k|k)$. With the exception of $\Delta\mathcal{U}(k|k)$, all other quantities are known at time k.

4.2.3 Results

The MPC algorithm was implemented in MATLAB (©The Mathworks, Natick MA, 2006), and the routine *fmincon* was used to solve the optimization problem. For results in this section, the full VL model derived from projecting Volterra kernels from Algorithm 4 is considered. This full VL model is then used as a basis to compare the closed-loop performance of the reduced VL model.

The upper half of Figure 47 shows the nonlinear reactor (plant) outputs from both the full and reduced VL model-based controllers (henceforth referred to as full and reduced VL controllers, respectively), in response to a set-point of +40 in the scaled polymer NAMW. The SSE between the reference and the plant output from the VL controllers is shown in the legend. The bottom half of the figure shows the control moves. It is observed that while the full model has a smooth approach to the set-point, the reduced model exhibits an overshoot of 7.5%. The rise times, defined as the the time required to first attain the

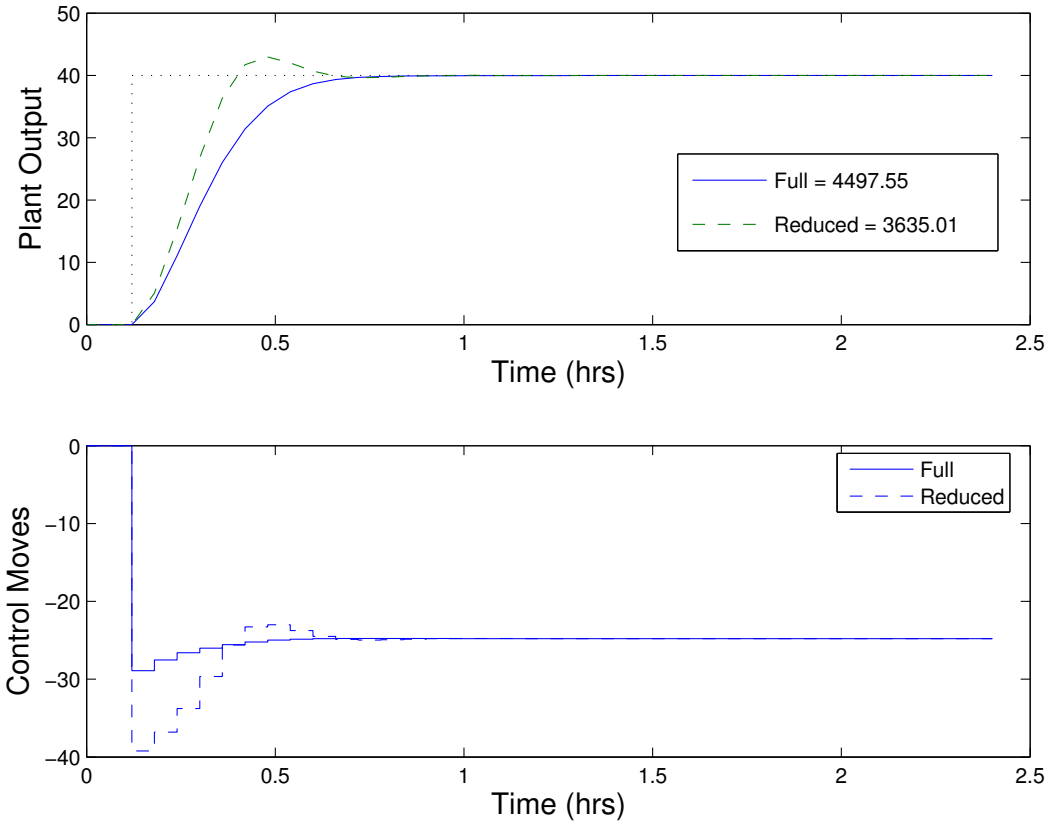

Figure 47: Set-point tracking performance of VL model based controllers with no measurement noise. Top: reference trajectory $\mathcal{R}(k+1|k) = 40$ (dotted), plant output under the full VL controller (solid), and plant output under the reduced VL controller (dashed). The SSEs between the reference and the plant outputs from the VL controllers are shown in the legend. Bottom: control moves for the full VL (solid) and the reduced VL (dashed) controllers. The input and the output are in scaled deviation form. The controller parameters were $m = 1$, $p = 10$, $\Gamma_y = 5I_{p\times p}$, and $\Gamma_u = I_{m\times m}$

final value, are also different for the outputs from the two controllers. The rise time for the reduced controller, 0.42 hr, is half of that from the full VL controller. Consequently, the SSE for the reduced VL controller is lower than that of the full VL controller. Finally, the 99% settling time, defined as the time at which the plant output enters and remains within 1% of the final value, is 0.72 hr for both the controllers. The results thus presented are generated with ad hoc tuning rules, *i.e.,* values of m, p, Γ_y, and Γ_u were selected that showed good performance for the two controllers. Determination of the optimal tuning parameters is a non-trivial problem as it requires the solution of a mixed integer nonlinear programming problem. The speed of response, (and hence the rise time) can certainly be improved by changing the controller parameters.

Similarly, Figure 48 shows the performance of the full and reduced VL controllers for a set-point of -40 in scaled NAMW. The performance of the two controllers is similar and neither of the two controllers exhibits an overshoot. The settling time of the reduced VL controller (1.56 hr) is less than that of the full VL controller (2.34 hr).

A choice of $\Gamma_y = I_{p\times p}$ and $\Gamma_u = I_{m\times m}$ showed the best combination of low overshoot and a fast response time, so these values were used for all subsequent analysis. Furthermore, keeping Γ_y and Γ_u constant enables evaluation of the effect of control and prediction horizons on the closed-loop performance.

Figure 49 shows the effect of the prediction horizon on the set-point tracking performance for a set-point of 40 in scaled NAMW. The left column shows the plant output responses (top) and the control moves (bottom) for the full and reduced VL controllers for $p = 2$. The rise times for both controllers were identical (0.42 hr), however the settling time of the reduced VL controller (1.68 hr) was greater than that of the full VL controller (0.9 hr). Both controllers exhibit overshoot, but the overshoot in the reduced VL controller (48%) is considerably higher than that of the full VL controller (20%). The reduced VL controller also shows a damped oscillatory behavior. In the context of the polymerization reactor, both a large overshoot and slow rise and settling times are undesirable, as they signify a product with uneven NAMW distribution. The right column of Figure 49 shows the response for $p = 10$. Both controllers lead to outputs that successfully track the set point and exhibit lower overshoot, 5% and 20% for the full and reduced VL controllers, respectively. The

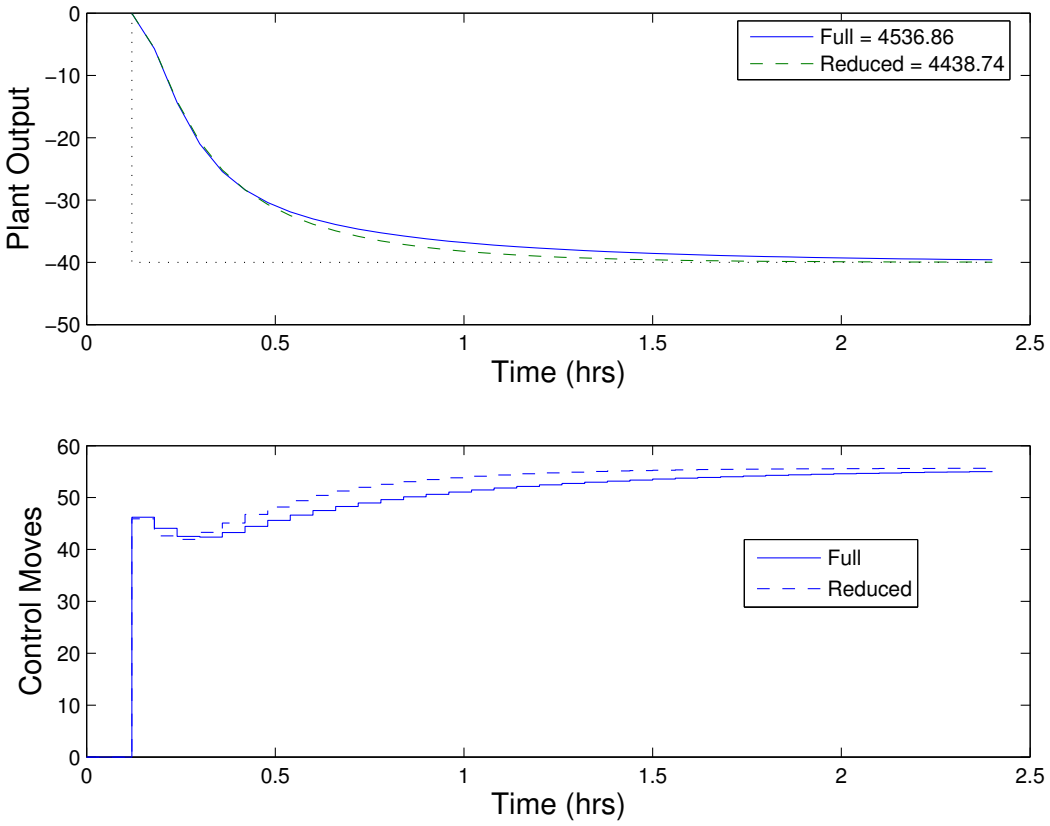

Figure 48: Set-point tracking performance of VL model based controllers with no measurement noise. Top: reference trajectory $\mathcal{R}(k+1|k) = -40$ (dotted), plant output under the full VL controller (solid), and plant output under the reduced VL controller (dashed). The SSEs between the reference and the plant outputs from the VL controllers are shown in the legend. Bottom: control moves for the full VL (solid) and the reduced VL (dashed) controllers. The input and the output are in scaled deviation form. The controller parameters were $m = 1$, $p = 10$, $\Gamma_y = 5I_{p\times p}$, and $\Gamma_u = I_{m\times m}$

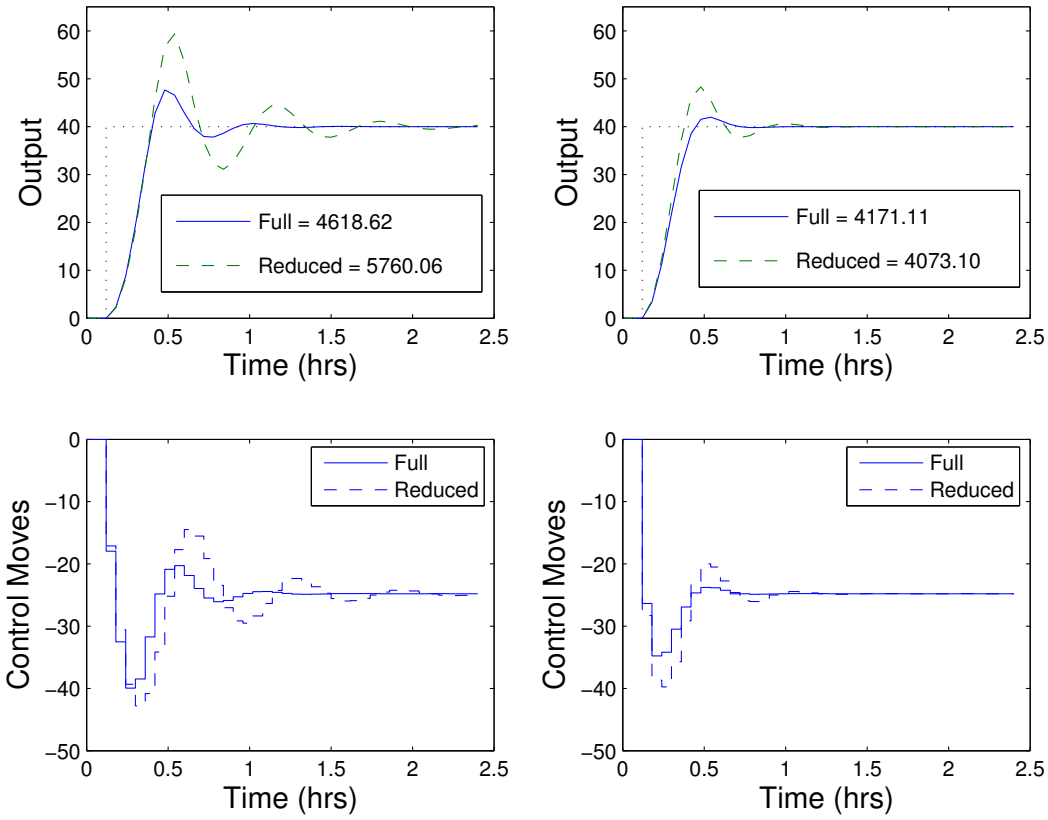

Figure 49: Set-point tracking performance of the VL model based controllers with different prediction horizons, $p = 2$ (left column) and $p = 10$ (right column). Top: reference trajectory $\mathcal{R}(k+1|k) = 40$ (dotted), plant output under the full VL controller (solid), and plant output under the reduced VL controller (dashed). The SSEs between the reference and the plant outputs from the VL controllers are shown in the legend. Bottom: control moves for the full VL (solid) and the reduced VL (dashed) controllers. The controller parameters were $m = 2$, $\Gamma_y = I_{p\times p}$, and $\Gamma_u = I_{m\times m}$

settling times of the full and reduced VL controllers are reduced to 0.66 hr and 0.90 hr, respectively. This is because with a larger prediction horizon, the controller can make more accurate predictions of the future NAMW behavior, thus leading to improved control. Thus a value of $p = 10$ was selected for all further analysis.

Figure 50 shows the plant output obtained from the full and reduced VL controllers in response to a set-point change of +40 in scaled NAMW for different values of the control horizon, m. For $m = 1$, the full VL controller was sluggish but showed no overshoot. The response of the reduced controller was faster with a rise time 0.42 hr, but showed an overshoot of 8%. The settling times of the full and reduced controllers were 0.72 hr and 0.84 hr, respectively. For $m = 2$, the controller response was not as sluggish as for $m = 1$, and the rise time of the full VL controller improved to 0.48 hr, at the cost of an overshoot of 5%. The rise time of the reduced controller remained unchanged but its overshoot increased to 20%. For $m = 3$ and $m = 5$, no appreciable change in either the rise or the settling time was observed. However, the overshoot for the reduced VL controller increased to 25%.

For a set-point of -40 in scaled NAMW, the full VL controller showed a sluggish response for $m = 1$ (settling time 2.40 hr, Figure 51, top left) whereas the settling time of the reduced VL controller was 1.56 hr. For $m = 2$, the settling time of the full VL controller improved to 2.04 hr, and subsequent increase in m did not result in a noticeable decrease in settling time of either the full, or the reduced, VL controller. For all subsequent analysis a value of $m = 2$ was selected as the controllers showed a fast response.

Figure 52 shows the closed-loop SSE profiles between the reference and plant outputs under the full and reduced VL controllers, in response to steps of magnitude between $[-40, 40]$ in increments of 8 in scaled NAMW units over 2.4 hr. It is observed that over the entire range of inputs, the performance of the full and reduced VL controllers is comparable. There is a slight difference between the two curves in the set-point range of +20 to +40 where the reduced VL controller outperforms the full VL controller. The improved performance of the reduced VL controller is attributed to its faster rise time, and hence a faster approach to the reference set-point (as in Figure 47).

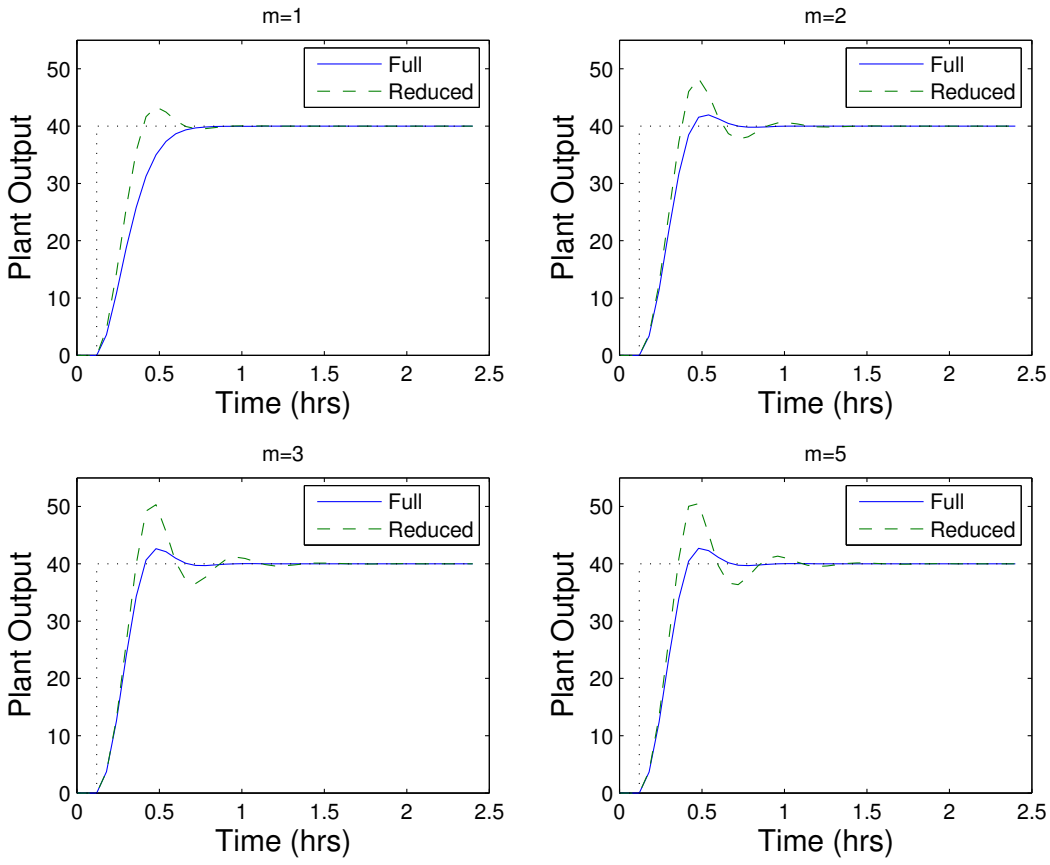

Figure 50: Set-point tracking performance of the VL model based controllers with different control horizons. Top row (left to right) $m = 1$ and $m = 2$, bottom row $m = 3$ and $m = 5$. Reference trajectory $\mathcal{R}(k+1|k) = 40$ (dotted), plant output under the full VL controller (solid), and plant output under the reduced VL controller (dashed) are shown. The input and the output are in scaled deviation form. The controller parameters were $p = 10$, $\Gamma_y = I_{p\times p}$, and $\Gamma_u = I_{m\times m}$

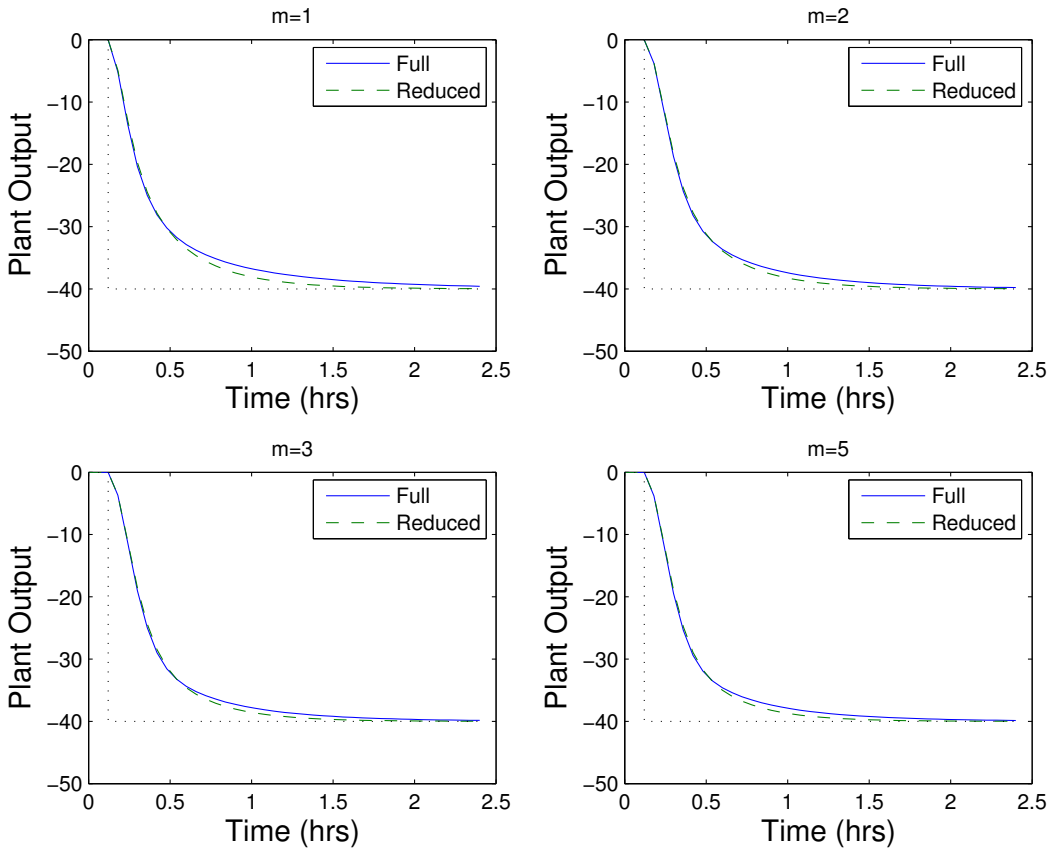

Figure 51: Set-point tracking performance of the VL model based controllers with different control horizons. Top row (left to right) $m = 1$ and $m = 2$; bottom row $m = 3$ and $m = 5$. Reference trajectory $\mathcal{R}(k+1|k) = -40$ (dotted), plant output under the full VL controller (solid), and plant output under the reduced VL controller (dashed) are shown. The input and the output are in scaled deviation form. The controller parameters were $p = 10$, $\Gamma_y = I_{p\times p}$, and $\Gamma_u = I_{m\times m}$

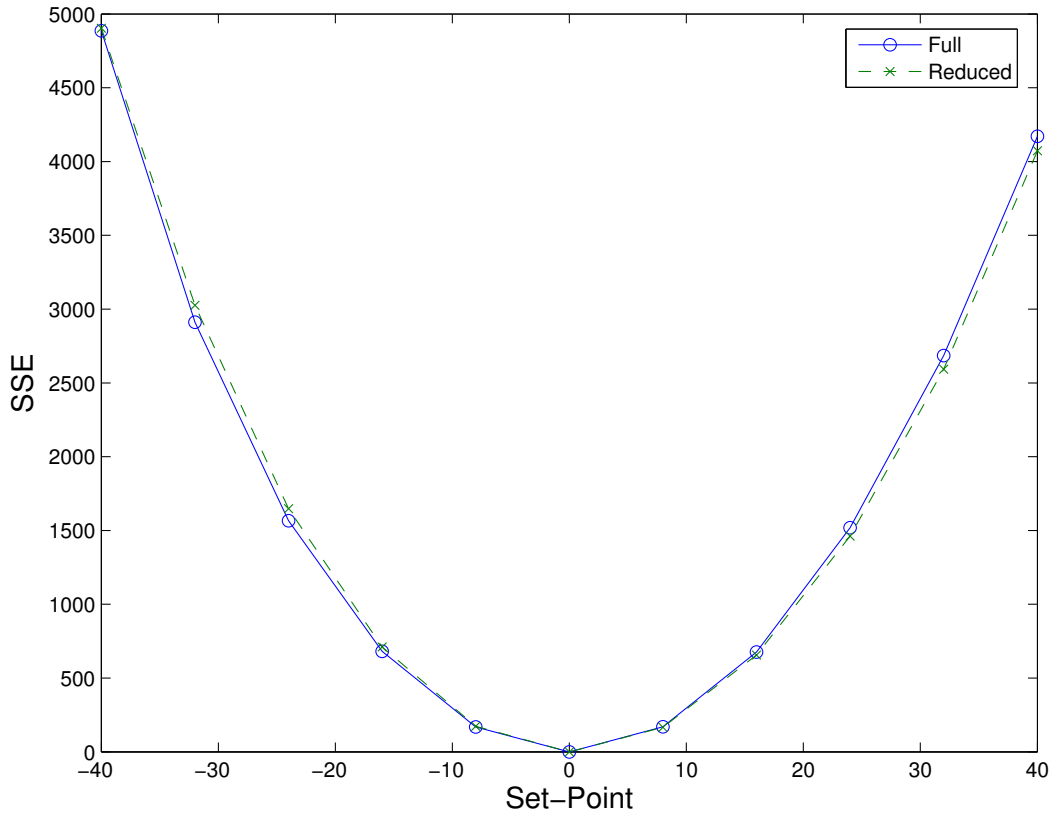

Figure 52: Sum Squared Error comparison of the plant outputs under the full (–o–) and reduced (−−x−−) VL controllers with no measurement noise in the set-point range $[-40, 40]$. The controller parameters were $m = 2$, $p = 10$, $\Gamma_y = I_{p\times p}$, and $\Gamma_u = I_{m\times m}$

Figure 53 shows the SSE profiles for both the full and reduced VL controllers derived from the nominal Volterra model, and for VL models obtained from noise corrupted Volterra kernels. It is observed that for both the full and reduced VL controllers, the control performance is unaffected by noise in the identification of the Volterra kernels.

In order to evaluate the performance of the VL controllers in the presence of measurement noise, a 10% random additive measurement noise was added to the plant output. All other controller parameters were the same as in the nominal case, *i.e.*, $m = 2$, $p = 10$, $\Gamma_y = I_{p\times p}$, and $\Gamma_u = I_{m\times m}$. Figure 54 shows the plant outputs for a set-point of +40 in scaled NAMW. The control moves and the noise sequence are also shown (Figure 54, middle and bottom, respectively). The reduced VL controller exhibits an overshoot of 25% as compared to 8% for the full VL controller. The SSE is 4354.14 between the reference and the output under the full VL controller, and is 4415.98 between the reference and the output under the reduced VL controller. The corresponding SSE values in the absence of measurement noise were 4170.26 for the full, and 4082.35 for the reduced VL controller. Thus even in the presence of 10% random additive measurement noise, the performance of the full and reduced VL controllers decreased by only 4.4% and 8.3%, respectively.

Figure 55 shows the plant outputs for a set-point of −40 in scaled NAMW. The settling times of the full and reduced VL controllers are similar (1.14 *hr* for the reduced, and 1.2 *hr* for the full VL controller). The SSE is 4833.28 between the reference and the output under the full VL controller, and is 4936.51 between the reference and the output under the reduced VL controller. These values are within 2% of their corresponding nominal counterparts, 4768.24 and 4846.49 for the full and reduced VL controllers, respectively. Thus, both the VL controllers display excellent tracking of negative set-points. This improvement over the positive set-point is attributed to the absence of overshoot in the plant outputs while tracking negative set-points.

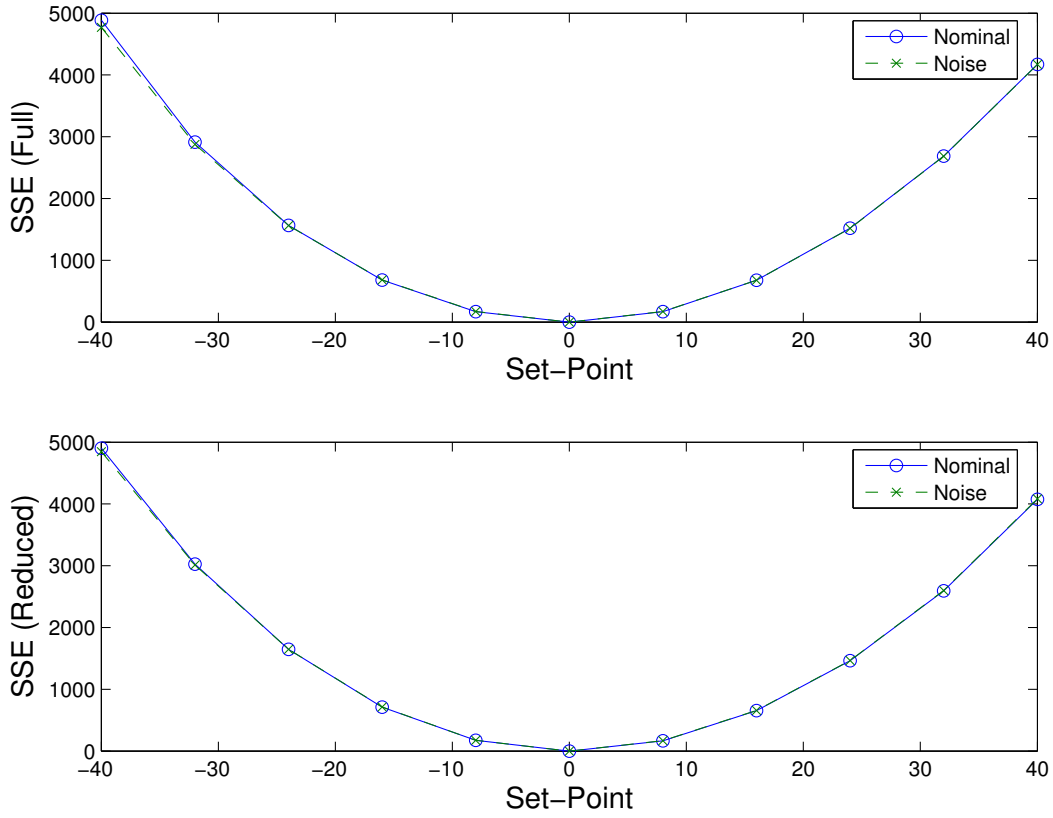

Figure 53: Sum Squared Error comparison of the plant outputs under the full (top) and reduced (bottom) VL controllers with no measurement noise in the set-point range $[-40, 40]$. Plant output from controllers based on nominal VL models (–o–) and from controllers based on VL models identified from noise-corrupted Volterra kernels (––x––) are shown. The controller parameters were $m = 2$, $p = 10$, $\Gamma_y = I_{p\times p}$, and $\Gamma_u = I_{m\times m}$

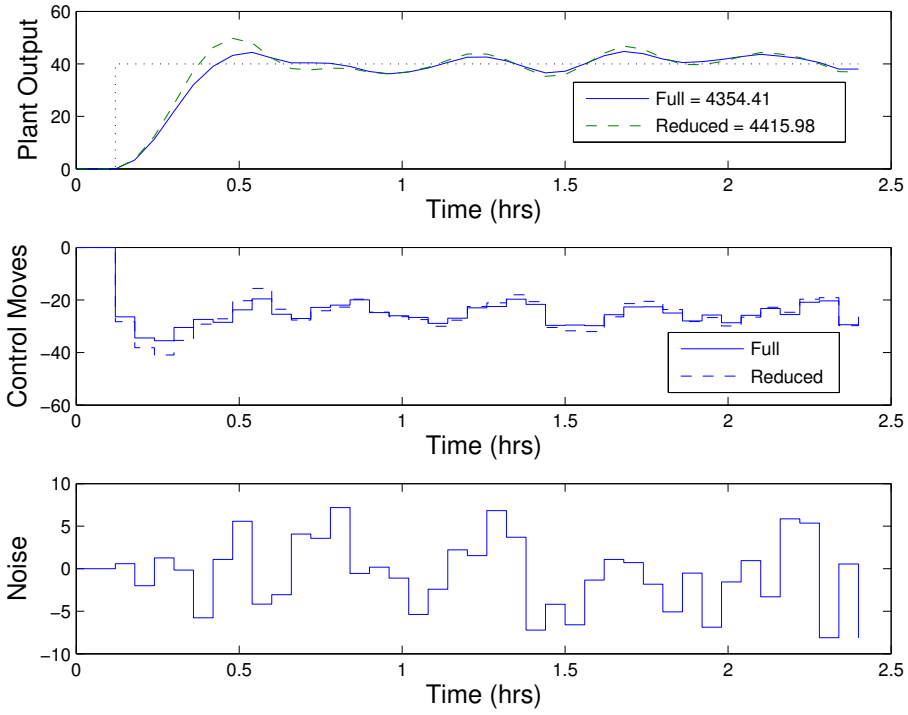

Figure 54: Set-point tracking performance of VL model based controllers with 10% random additive measurement noise. Top: reference trajectory, $\mathcal{R}(k+1|k) = 40$ (dotted), plant output under the full VL controller (solid), and plant output under the reduced VL controller (dashed). The SSE between the reference and the plant outputs from the VL controllers are shown in the legend. Middle: control moves for the full VL (solid) and the reduced VL (dashed) controllers. Bottom: noise sequence for the VL controllers. The input and the output are in scaled deviation form. The controller parameters were $m = 2$, $p = 10$, $\Gamma_y = I_{p\times p}$, and $\Gamma_u = I_{m\times m}$

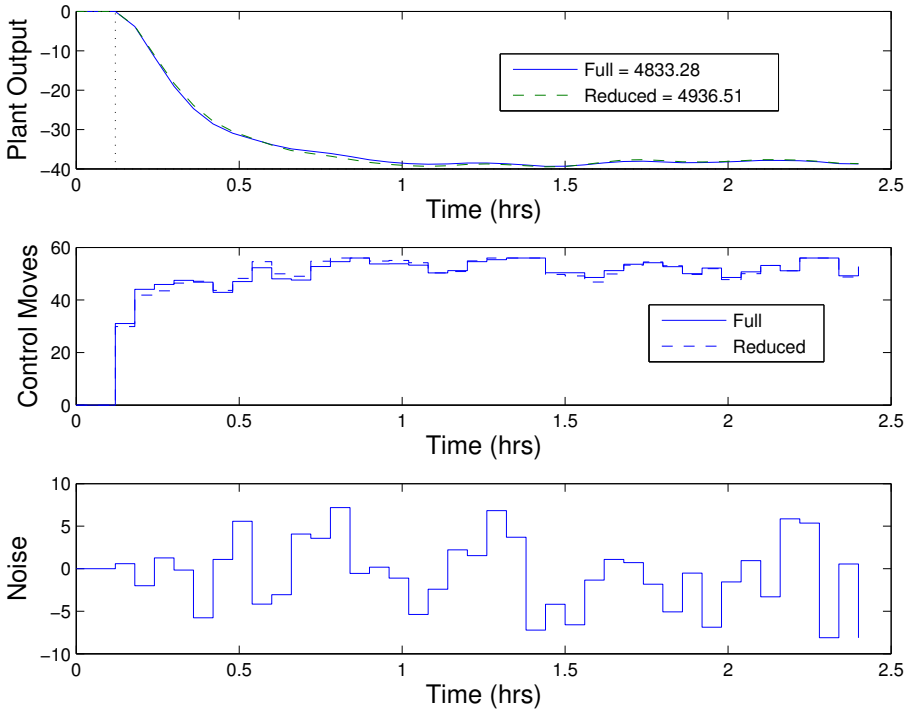

Figure 55: Set-point tracking performance of VL model based controllers with 10% random additive measurement noise. Top: reference trajectory, $\mathcal{R}(k+1|k) = -40$ (dotted), plant output under the full VL controller (solid), and plant output under the reduced VL controller (dashed). The SSEs between the reference and the plant outputs from the VL controllers are shown in the legend. Middle: control moves for the full VL (solid) and the reduced VL (dashed) controllers. Bottom: noise sequence for the VL controllers. The input and the output are in scaled deviation form. The controller parameters were $m = 2$, $p = 10$, $\Gamma_y = I_{p\times p}$, and $\Gamma_u = I_{m\times m}$

Figure 56 shows the SSE profiles for both the full and reduced VL controllers in the presence of the 10% additive random measurement noise, in response to steps of magnitude between $[-40, 40]$ in increments of 8 units in scaled NAMW over 2.4 hr. It is observed that the reduced VL controller performance is similar to that of the full VL controller over the range of set-points. The reduced Volterra Laguerre models are thus shown to be useful control-relevant models.

4.3 SUMMARY

In this chapter the performance of NMPC controllers synthesized from the VL models was evaluated. Using the projection algorithm in Section 3.2.3, models were obtained from Volterra kernels identified using data from the polymerization reactor. The effect of prediction and control horizons on control performance was analyzed. Overall, the VL controllers showed excellent set-point tracking performance over the operating range in both the absence, and presence, of measurement noise. Although the reduced VL controller showed higher overshoot than the full VL controllers for positive set-points, the steady-state closed-loop performance of the two controllers was comparable.

A distinct advantage in favor of the reduced VL models is that the number of coefficients requiring identification is 29× less than that required for the full Volterra model. This reduction in the number of coefficients results in a corresponding decrease in the amount of data required for identification. An additional advantage of the VL model structure is its linearity with respect to the parameters. This makes it possible to update the model parameters of a reduced VL model on-line using a recursive least squares (RLS) algorithm [15].

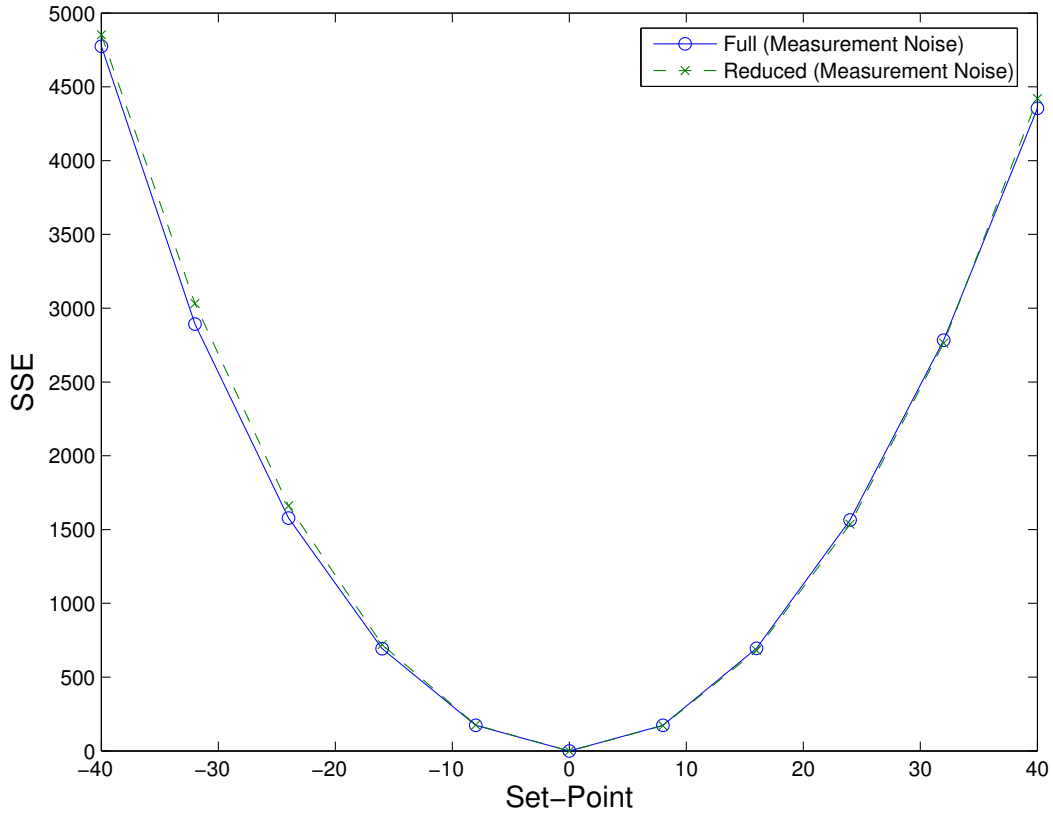

Figure 56: Sum Squared Error comparison of the plant outputs under the full (–o–) and reduced (––x––) VL controllers with 10% random additive measurement noise in the set-point range $[-40, 40]$. The controller parameters were $m = 2$, $p = 10$, $\Gamma_y = I_{p\times p}$, and $\Gamma_u = I_{m\times m}$

5.0 SUMMARY AND RECOMMENDATIONS

This dissertation addressed the problem of nonlinear system identification using third-order Volterra and Volterra-Laguerre models. Input sequences were designed for the efficient identification of Volterra models. The Volterra models were then projected optimally onto the Laguerre basis. The resultant Volterra-Laguerre models were validated in both open-loop and closed-loop. The contributions of this dissertation have been presented in the summary section at the end of each chapter, and the main contributions are restated in the next section.

5.1 SUMMARY

Tailored plant-friendly input sequences were designed for the efficient identification of third-order Volterra models. The Congalidis, Richards, and Ray polymerization reactor served as the simulation case-study [75]. The tailored sequences were designed based on the PEV expression, and these sequences simplified the identification by exploiting the structure of the Volterra model. Improvements in the sequence design were also evaluated, and this resulted in the simultaneous identification of the third-order sub-diagonal and second-order off-diagonal kernels. A significant reduction in the data requirement for identification was achieved, as the need for a separate second-order off-diagonal sequence was eliminated. All of the tailored input sequences were shorter than, and provided more accurate kernel estimates than, the corresponding sequences based on cross-correlation.

While the Volterra models showed excellent nominal performance, there was a slight degradation in performance in the presence of noise. In order to reduce the noise sensitivity

and high parameterization of the Volterra model, the Volterra kernels were projected onto the orthonormal Laguerre basis. This projection was optimal, using the Akaike Information Criterion as the optimality metric. AIC took into account both the accuracy and size of the resultant Volterra-Laguerre model. Reduced VL models were also obtained by projecting only the linear plus nonlinear diagonal Volterra kernels onto the Laguerre basis. All of the identified Volterra-Laguerre models showed excellent validation performance, across a broad operating range. The defining feature of the identified Volterra-Laguerre models was their excellent predictive performance, even when generated from the noise-corrupted Volterra kernels. Two Laguerre filters were sufficient for the full and reduced Volterra-Laguerre models, so that a significant reduction in the model parameterization was achieved.

The identified Volterra-Laguerre models were employed in closed-loop by designing controllers based on the model predictive control algorithm. The full and reduced Volterra-Laguerre controllers showed excellent set-point tracking performance in both the absence and presence of measurement noise. Over the operating range of the reactor, the closed-loop performance of both the controllers was comparable. The reduced Volterra-Laguerre models were thus, a suitable control-relevant alternative to the full Volterra and Volterra-Laguerre models.

5.2 RECOMMENDATIONS FOR FUTURE WORK

5.2.1 Identification of Multi-Input Multi-Output Volterra Models

All of the work in this dissertation was carried out for the SISO polymerization reactor presented in Section 2.1. The algorithms developed for Volterra model identification could, in principle, be extended to multi-input multi-output (MIMO) systems as well, but this extension is accompanied by a significant increase in the number of parameters that need to be identified. For a b-input c-output third-order Volterra model, the number of unique Volterra model coefficients that need to be identified is given as:

$$c\sum_{i=1}^{3}\binom{bM+i-1}{i} = c\left(1+bM+\frac{bM(bM+1)}{2!}+\frac{bM(bM+1)(bM+2)}{3!}\right) \tag{5.1}$$

Assuming $M = 20$ and $b = c = 2$, leads to $24,862$ unique coefficients that need to be identified, which is quite a significant challenge. One way to reduce the number of coefficients to be identified is to consider only reduced Volterra kernel identification, *i.e.,* identify only the linear and nonlinear diagonal kernels. For the hypothetical 2×2 system, the third-order diagonal kernel would now include contributions from $u_1^3(k-j)$, $u_1^2(k-j)u_2(k-j)$, $u_1(k-j)u_2^2(k-j)$, and $u_2^3(k-j) \;\; \forall\, j = 1, 2, \cdots, M]$.

Another approach to reducing the number of identified coefficients would be to consider only those input interactions that contribute significantly to the process output. Korenberg *et al.* proposed an orthogonal regression analysis based algorithm that accomplishes this goal [98]. The advantage of the orthogonal regression analysis is that entering a new interaction term in the model does not alter the values of identified coefficients from previously included terms. The best model component to be introduced into the model can be obtained by calculation of the SSE contribution from regression of candidate terms; the term which results in the largest decrease in SSE is introduced [99]. When used in conjunction with the Laguerre basis, a significant reduction in the number of identified coefficients for a MIMO Volterra model could be achieved.

5.2.2 Parameter Update using Recursive Least Squares

One of the main differences between the reduced Volterra and the reduced VL models is that as opposed to the reduced Volterra models, the reduced VL model is a full parameter model, *i.e.,* there are no zeros in the parameter matrices C, D, and E in (3.7). In addition, since the VL model is linear with respect to the parameters, (3.7) can be re-written as:

$$\begin{aligned} \hat{y}_\ell(k) &= \Phi^T(k)\hat{\Theta}(k) \qquad (5.2)\\ \Phi^T(k) &= \big[\ell_1(k)\ \ell_2(k)\ \ell_1(k)\ell_1(k)\ \ell_1(k)\ell_2(k)\ \ell_2(k)\ell_2(k)\cdots \\ &\quad\ \ell_1(k)\ell_1(k)\ell_1(k)\ \ell_1(k)\ell_1(k)\ell_2(k)\ \ell_1(k)\ell_2(k)\ell_2(k)\ \ell_2(k)\ell_2(k)\ell_2(k)\big] \\ \hat{\Theta} &= [c_1\ c_2\ c_{11}\ 2c_{12}\ c_{22}\ c_{111}\ 3c_{112}\ 3c_{122}\ c_{222}] \end{aligned}$$

Since only two Laguerre filters are required, the parameter set, Θ, is of a manageable size. Furthermore, in Chapters 3 and 4 it was observed that the location of the Laguerre pole

does not change significantly when using a reduced VL model as compared to a full VL model. Consequently the parameters of the reduced VL model could be updated on-line using a recursive least squares algorithm [18], so that the reduced VL model output could asymptotically approach the output of the full VL model. The RLS parameter update could also account for disturbances that could create a mismatch between the process and the identified VL model.

The parameter update algorithm can be written as [96]:

$$\Omega(k) = \Upsilon^{-1} I - \left[\frac{\Omega(k-1)\Phi(k)\Phi^T(k)}{\Upsilon + \Phi^T(k)\Omega(k-1)\Phi(k)} \right] \Omega(k-1) \tag{5.3}$$

$$\mathcal{E}(k) = y(k) - \Phi^T(k)\hat{\Theta}(k) \tag{5.4}$$

$$\hat{\Theta}(k) = \hat{\Theta}(k-1) + \Omega(k)\Phi(k)\mathcal{E}(k) \tag{5.5}$$

In (5.3), Ω and Υ are the covariance matrix and the forgetting factor respectively. The initial covariance matrix, *i.e.,* $\Omega(k=0)$ and the forgetting factor are adjustable parameters that can be chosen to determine the rate of convergence of the parameter estimates. In (5.4), $\mathcal{E}$ represents the error vector between the process and identified model outputs. Finally the parameter update can be carried out, recursively, as in (5.5).

Thus using this RLS algorithm, the reduced VL model parameters could be updated on-line in the presence of disturbances, and the reduced VL outputs could asymptotically approach the full VL model or process outputs.

5.2.3 Simultaneous Model Predictive Control and Identification

The Volterra and the Volterra-Laguerre models considered in this work were identified in open-loop, and the resultant VL model was used in closed-loop for controller design. An alternative to open-loop identification is to identify the models in closed-loop. Closed-loop identification has been addressed extensively in a linear control setting [2, 18, 100]. However, the identification of closed-loop nonlinear models is a significantly more challenging problem [101]. The key requirement in system identification is that the input must be persistently exciting (PE) [18]. This PE condition is difficult to satisfy in closed-loop because the PE properties of the input depend on the closed-loop properties, which in turn depend on the

model to be identified in the first place. Dithering of the process inputs, *i.e.,* addition of an external signal such as a sum of sinusoids at various frequencies, can be used to guarantee PE but this can compromise closed-loop control performance [12, 18].

An alternative approach is used by Genceli and Nikolaou termed as simultaneous model predictive control and identification (MPCI). MPCI is an extension of the MPC algorithm, and it includes a PE condition on the inputs. At each time step the MPCI controller minimizes a standard quadratic objective function subject to the standard MPC constraints (Equation 4.2), and additional input constraints. The modified objective function is given as [101]:

$$\min_{\Delta\mathcal{U}(k|k)} \quad \left\{ \left|\left| \Gamma_y \left[\hat{\mathcal{Y}}(k+1|k) - \mathcal{R}(k+1|k) \right] \right|\right|_2^2 + \left|\left| \Gamma_u \Delta\mathcal{U}(k|k) \right|\right|_2^2 \right\} + w_{\emptyset}\emptyset^2 - w_{\rho}\rho \quad (5.6)$$

$$\text{s.t.} \quad \mathcal{I}_{\mathcal{M}} \succeq (\rho - \emptyset) I \succ \mathbf{0} \quad (5.7)$$

The first two terms in (5.6) are the same as in (4.2), whereas the last two terms arise due to the persistence of excitation condition. The relative importance of the terms can be adjusted by the weights $w_{\emptyset}$ and w_{ρ}. The constraint equation (5.7) guarantees that the input will be PE. $\mathcal{I}_{\mathcal{M}}$ represents the information matrix and includes the past input terms, whereas the identity, A $\succeq$ B ➔ A-B is positive semi-definite, whereas A $\succ$ B ➔ A-B is strictly positive definite.

The MPCI design can be applied to the reduced VL models and the advantage of closed-loop identification would be that the controller could take into account closed-loop performance objectives in model design, and potentially identify situation-specific models thereby improving the control performance.

5.2.4 Analytical Solution to the Model Predictive Control Problem

The objective function for the MPC optimization problem used in this work is given as (4.2):

$$\min_{\Delta\mathcal{U}(k|k)} \quad \left\{ \left|\left| \Gamma_y \left[\hat{\mathcal{Y}}(k+1|k) - \mathcal{R}(k+1|k) \right] \right|\right|_2^2 + \left|\left| \Gamma_u \Delta\mathcal{U}(k|k) \right|\right|_2^2 \right\}$$

The optimization is carried out with $\Delta\mathcal{U}(k|k)$ as the manipulated variable. Analytical solutions to (4.2) have been reported for $m = 1$ [15, 59, 102], and for higher values of

m, a local linearization was employed to approximate the future control moves [103, 104]. The difficulty in obtaining an analytical solution arises due to the complexity in (4.2), *e.g.,* for the third-order VL models in this work, the objective function contains terms of the order $\Delta\mathcal{U}^6(k|k)$. In order to obtain the optimal input (4.2) needs to be differentiated so that a set of m polynomials in $\Delta\mathcal{U}(k|k)$ need to be solved.

$$\Pi_i(\Delta u(k+i|k)) = 0 \quad \forall\ 0 \leq i \leq m-1 \tag{5.8}$$

This set of polynomials may contain terms up to a maximum order of five, and may include a combination of input terms of different orders.

One approach to solving the set of polynomial equations is to use Gröbner bases [105, 97]. Using Gröbner bases, the set of polynomial equations in (5.8) can be converted into a possibly larger set of polynomial equations such as:

$$\begin{aligned}
\varrho_i(\Delta u(k+1|k), \Delta u(k+2|k), \cdots, \Delta u(k+m-1|k)) &= 0 \quad 1 \leq i \leq \iota_1 \\
\varrho_j(\Delta u(k+2|k), \Delta u(k+3|k), \cdots, \Delta u(k+m-1|k)) &= 0 \quad \iota_1 + 1 \leq j \leq \iota_2 \\
&\vdots \qquad\qquad \vdots \\
\varrho_x(\Delta u(k+m-1|k)) &= 0 \quad x = 1 + \sum_{i=1}^{m-1} \iota_i
\end{aligned} \tag{5.9}$$

The advantage of using this approach is that (5.9) now consists of ι_1 equations that are a function of all manipulated variable moves, ι_2 equations that are a function of all but $\Delta u(k|k)$, and so on until the last equation is a function only of $\Delta u(k+m-1|k)$. This approach is thus similar to the Gaussian elimination technique used to solve a system of linear simultaneous equations. An important choice that has to be made in the solution of this set of equations is the ordering, *i.e.,* the order in which the $\Delta u(k+i|k)$ are successively eliminated. The use of Gröbner bases thus simplifies a nonlinear optimization problem into one that requires solving for the roots of a set of polynomial equations. Also, this approach leads to all possible extrema, so that a simple back-substitution of the input in (4.2) can enable determination of the global minimum. Thus, a global minimum to the nonlinear optimization problem can be obtained.

The complexity of the system of equations in (5.9), and thus the ease with which the solution can be obtained, depends on the choice of ordering that is employed. In addition, this choice may also be determined by the input-output behavior of the process [97]. The choice of m is also critical, as higher values of m would lead to an increase in problem size and potential computational difficulties. However, it was found that for the polymerization reactor case study only 2 Laguerre filters were necessary and good closed-loop control performance was achieved using $m = 2$. Thus the Gröbner basis approach to solving the optimization problem in (4.2) shows great promise both in general, and for the polymerization reactor case-study specifically.

APPENDIX A

NOMENCLATURE

Abbreviations	
AIC	Akaike's information criterion
ANN	artificial neural network
ARMA	auto-regressive moving average
ARMAX	auto-regressive moving average with exogenous inputs
ARX	auto-regressive with exogenous inputs
BIBO	bounded-input bounded-output
CSRS	constant-switching-pace symmetric random signal
CSTR	continuous stirred tank reactor
DFT	discrete Fourier transform
FIR	finite impulse response
GA	genetic algorithm
GWN	Gaussian white noise
ISE	integral squared error
LMPC	linear model predictive control
LTI	linear time invariant
MIMO	multiple-input multiple-output
MINLP	mixed-integer nonlinear program

MPC	model predictive control
MPCI	model predictive control and identification
NAMW	number average molecular weight
NLP	nonlinear program
NMPC	nonlinear MPC
PE	persistently exciting
PEV	prediction error variance
PRBS	pseudo-random binary signal
ODE	ordinary differential equation(s)
QR	quadratic regulator
RBS	random binary signal
RKHS	reduced kernel Hilbert space
RLS	recursive least squares
RQS	random quartenary sequence
SISO	single input single output
SSE	sum squared error
VL	Volterra-Laguerre

General Notation	
A, B, C, D, E	Volterra-Laguerre model state-space matrices
A_c, B_c, C_c, N_c	Carlemann linearization model matrices
$\hat{A}_0, \hat{A}_1, \hat{A}_2, \hat{B}_1, \hat{B}_2, \hat{B}_3$	discretized Carlemann linearization model matrices
c_i	linear Volterra-Laguerre or Laguerre coefficients
c_{ij}	second-order Volterra-Laguerre or Laguerre coefficients
c_{ijl}	third-order Volterra-Laguerre or Laguerre coefficients
C_{O2}	coefficient of the second-order off-diagonal kernel in PEV expression
C_{S3}	coefficient of the third-order sub-diagonal kernel in PEV expression
$D_2(k)$	second-order diagonal kernel
$D_3(k)$	third-order diagonal kernel
d	disturbance
$E(\cdot)$	expectation operator
F_f	friendliness factor
h_i	scaled Volterra kernel of order i
h_0	Volterra kernel bias term
h_{isym}	symmetric/triangular Volterra kernel of order i
h_{iasym}	asymmetric/full Volterra kernel of order i
J	objective function
k	discrete time index
$\mathbf{L}$	L_2 space
L	number of Laguerre filters used in projection
ℓ	Laguerre states
M	Volterra model memory
m	control move horizon

m_k	k^{th} moment of the input sequence
N	Volterra model order
N_L	total length of the input sequence
n_t	total number of transitions in input sequence
$\emptyset$	PE condition variable in MPCI objective function
$O_2(k)$	second-order off-diagonal kernel
$O_3(k)$	third-order off-diagonal kernel
p	prediction horizon (except when used as a counting index)
q	Carlemann linearized model state variable (except when used as a counting index)
$\tilde{q}$	discretized Carlemann linearized model state variable
$\mathbf{R}$	real space
r	reference signal
S_3	third-order sub-diagonal kernel
$s(i)$	step-response coefficients
t	time
$U(\cdot)$	unit step function
u	input sequence
$\bar{u}$	mean of the input sequence
v	scaled input corresponding to operating points over the reactor input range
Δu	manipulated variable move
$w_\emptyset, w_\rho$	tuning weights for $\emptyset$ and ρ in MPCI objective function
x	process model state variable
x_{i0}	i^{th} model state nominal operating condition
y	controlled variable
$\hat{y}$	output prediction

y_m	measured output
y_c	output from Carlemann linearized model
$\tilde{y}$	discretized output from Carlemann linearized model
z	identification residual

Calligraphic letters	
$\mathcal{C}$	controller
$\mathcal{E}$	error vector in RLS formulation
$\mathcal{G}(\cdot)$	static nonlinearity
$\mathcal{H}_a$	sub-diagonal coefficient vector for a regions of reduced sequence for sub-diagonal kernel identification
$\mathcal{H}_b$	sub-diagonal coefficient vector for b regions of reduced sequence for sub-diagonal kernel identification
$\mathcal{H}_{OS}$	sub-diagonal coefficient vector for simultaneous O_2 and S_3 kernel identification
$\mathcal{H}_S$	sub-diagonal coefficient vector for S_3 kernel identification
$\mathcal{I}_{\mathcal{M}}$	information matrix in MPCI objective function
$\mathcal{L}(t)$	linear dynamic element
$\mathcal{M}$	model
$\mathcal{P}$	plant
$\mathcal{R}$	vector of future reference values
$\mathcal{U}$	vector of future manipulated variable values
$\Delta\mathcal{U}$	vector of future manipulated variable changes
$\mathcal{U}_a$	block diagonal input matrix for a regions of reduced sequence for sub-diagonal kernel identification
$\mathcal{U}_b$	block diagonal input matrix for b regions of reduced sequence for S_3 kernel identification
$\mathcal{U}_S$	block diagonal input matrix for S_3 kernel identification
$\mathcal{U}_{OS}$	block diagonal input matrix for a regions of sequence for simultaneous O_2 and S_3 kernel identification
$\mathcal{W}$	output vector for b regions of reduced sequence for S_3 kernel identification
$\hat{\mathcal{Y}}$	vector of predicted future outputs

$\mathcal{Y}_{redsd}$	output vector for a regions of reduced sequence for S_3 kernel identification
$\mathcal{Y}_{OS}$	output vector for simultaneous O_2 and S_3 kernel identification
$\mathcal{Z}_S$	output vector for S_3 kernel identification

Greek Letters	
α	Laguerre pole
β	pulse magnitude of reduced sequence for S_3 kernel identification
Γ_u	manipulated variable weighting matrix
Γ_y	controlled variable weighting matrix
γ	pulse magnitude of sequence for linear and nonlinear diagonal kernel identification
δ_i	Volterra kernel coefficient estimation error for the i^{th}-order kernel contribution
κ	kurtosis
λ	pulse magnitude of sequence for third-order sub-diagonal kernel identification
ν	scaled input in deviation variables corresponding to steady-states in reactor operating range
Ω	covarianc matrix in RLS
Π	set of polynomials in $\Delta\mathcal{U}$ from the MPC objective function
Ξ	gap between successive pulses for the full S_3 sequence
ρ	condition for PE in the MPCI objective function
ϱ	set of polynomials obtained from Gröbner bases
ϕ_j	j^{th}-Laguerre function
φ	inequality constraints on the input
σ_p^2	prediction error variance
σ_u^2	input signal variance
θ	parameter set
Θ	parameter set in RLS
Υ	forgetting factor
ς	equality constraints on the input

APPENDIX B

ESTIMATORS BASED ON REDUCED S_3 SEQUENCE

Consider the first $2M+2$ data-points of u_1 in (2.50). The output corresponding to this section of the input sequence is as follows:

$$\hat{z}_{red-sd}(k) = \begin{cases} 0 & k = 1 \\ 0 & k = 2 \\ h_3(k-2,k-1,k-1)\beta_1^2\beta_2 & 3 \le k \le M+1 \\ +h_3(k-2,k-2,k-1)\beta_2^2\beta_1 & \\ -h_3(1,M,M)\beta_2^3 + h_3(1,1,M)\beta_2^3 & k = M+2 \\ -h_3(k-M-2,k-M-1,k-M-1)\beta_2^2\beta_1 & M+3 \le k \le 2M+1 \\ -h_3(k-M-2,k-M-2,k-M-1)\beta_1^2\beta_2 & \\ h_3(1,M,M)\beta_2\beta_1^2 - h_3(1,1,M)\beta_1\beta_2^2 & k = 2M+2 \end{cases} \tag{B.1}$$

It is possible to calculate the two contributions separately because the sequence from $2M+1$ to $4M$ is the same as the sequence from 1 to $2M$, with the exception that the order of the pulses β_1 and β_2 is now reversed. The estimates for the third-order sub-diagonal coefficients can be obtained by minimizing the following sum-squared prediction error:

$$J_{red-sd} = \sum_{k=1}^{2M+2} \{z_{red-sd}(k) - \hat{z}_{red-sd}(k)\}^2 \tag{B.2}$$

In Equation (B.1) there are two sets of coefficients that need to be calculated. The first set corresponds to the coefficients in the range $3 \le k \le M+1$ and $M+3 \le k \le 2M+1$, and the second set corresponds to the $h_3(1,M,M)$ and $h_3(1,1,M)$ coefficients.

Consider the calculation for the $h_3(1, M, M)$ and $h_3(1, 1, M)$ coefficients. Applying the conditions,

$$\begin{aligned} \frac{\partial J_{red-sd}}{\partial h_3(1, M, M)} &= 0 \\ \frac{\partial J_{red-sd}}{\partial h_3(1, 1, M)} &= 0 \end{aligned}$$

yields:

$$\begin{aligned} -\beta_2\{\beta_2^4 + \beta_1^4\}h_3(1, M, M) + & \\ \beta_2^2\{\beta_2^3 + \beta_1^3\}h_3(1, 1, M) &= \beta_2^3 z_{red-sd}(M + 2) + \beta_1^2 z_{red-sd}(2M + 2) \quad \text{(B.3)} \\ -\beta_2\{\beta_2^3 + \beta_1^3\}h_3(1, M, M) + & \\ \beta_2^2\{\beta_2^2 + \beta_1^2\}h_3(1, 1, M) &= \beta_2 z_{red-sd}(M + 2) - \beta_1 z_{red-sd}(2M + 2) \quad \text{(B.4)} \end{aligned}$$

Solving Equations (B.3) and (B.4) simultaneously returns the following:

$$h_3(1, M, M) = \frac{\beta_1 z_{red-sd}(M + 2) + \beta_2 z_{red-sd}(2M + 2)}{\beta_1\beta_2^2(\beta_1 - \beta_2)} \quad \text{(B.5)}$$

$$h_3(1, 1, M) = \frac{\beta_1^2 z_{red-sd}(M + 2) + \beta_2^2 z_{red-sd}(2M + 2)}{\beta_1\beta_2^3(\beta_1 - \beta_2)} \quad \text{(B.6)}$$

Proceeding in a similar manner, the coefficients for the second half of the sequence, *i.e.*, for points from $2M + 1$ to $4M$, are given as:

$$h_3(1, M, M) = \frac{\beta_2 z_{red-sd}(3M + 2) + \beta_1 z_{red-sd}(4M + 2)}{\beta_2\beta_1^2(\beta_2 - \beta_1)} \quad \text{(B.7)}$$

$$h_3(1, 1, M) = \frac{\beta_2^2 z_{red-sd}(3M + 2) + \beta_1^2 z_{red-sd}(4M + 2)}{\beta_2\beta_1^3(\beta_2 - \beta_1)} \quad \text{(B.8)}$$

In a similar manner the coefficients corresponding to the range $3 \leq k \leq M + 1$ and $M + 3 \leq k \leq 2M + 1$ can be derived.

However, when the entire sequence of length $4M$ is considered, there are four sets of equations (two corresponding to each half of length $2M$) for two sets of coefficients. This leads to an overdetermined system of equations which was solved using a least-squares calculation as shown in Section 2.2.6.1. The coefficients obtained in this manner are an average of the coefficients obtained from the two half sequences, *i.e.*, from 1 to $2M$ and from $2M + 1$ to $4M$.

APPENDIX C

TAYLOR EXPANSION OF THE REACTOR SYSTEM

In order to obtain analytical Volterra kernels from (2.1), the system was first written in scaled deviation form such that:

$$\begin{aligned} v &= \frac{u - u_0}{001} \\ \tilde{y} &= \frac{y - y_0}{100} \\ q_i &= \frac{x_i - x_{i0}}{x_{i0}} \quad \forall i = 1, \cdots, 4 \end{aligned} \tag{C.1}$$

The nominal operating conditions are given in Table 1. Writing the original system in deviation variables yields:

$$\begin{aligned} x_1 &= 5.128q_1 + 5.128 \\ x_2 &= 0.479q_2 + 0.479 \\ x_3 &= 00623q_3 + 00623 \\ x_4 &= 87.309q_4 + 87.309 \\ u &= 001v + 0605 \\ y &= 100\tilde{y} + 14012 \end{aligned} \tag{C.2}$$

Also,

$$\dot{x}_i = \dot{q}_i x_{i0} \tag{C.3}$$

Substituting (C.2) and (C.3) in (2.1) results in the following system of equations:

$$\begin{aligned}
\dot{q}_1 &= 11.70 - (1+q_1)\left[10 + 1.7(1+q_2)^{1/2}\right] \\
\dot{q}_2 &= 0.1669v - 10.1022q_2 \\
\dot{q}_3 &= 1.374(1+q_1)(1+q_2)^{1/2} - 10q_3 + 8.626q_2 - 1.3740 \\
\dot{q}_4 &= 10(1+q_1)(1+q_2)^{1/2} - 10(1+q_4) \\
\tilde{y} &= 140.12\frac{q_4 - q_3}{1+q_3}
\end{aligned} \tag{C.4}$$

The equation for $\dot{q}_2$ is the only linear equation in (C.4). In order to linearize the nonlinear terms in (C.4), the following mathematical relations are employed [106]:

$$(1+a)^{1/2} = 1 + \frac{1}{2}a - \frac{1}{8}a^2 + \frac{1}{16}a^3 - \cdots \tag{C.5}$$

$$\frac{1}{1+a} = 1 - a + a^2 - a^3 + \cdots \tag{C.6}$$

Since a third-order Taylor series approximation is being developed, all terms higher than third-order, are neglected in (C.5) and (C.6). Substituting (C.5) in the equations for $\dot{q}_1$, $\dot{q}_3$, and $\dot{q}_4$ in (C.4) and simplifying yields:

$$\begin{aligned}
\dot{q}_1 &= 11.70 - (1+q_1)[10 + 1.7(\frac{1}{2}q_2 - \frac{1}{8}q_2^2 + \frac{1}{16}q_2^3)] \\
&= -11.7q_1 - 0.85q_2 + 0.2125q_2^2 - 0.10625q_2^3 - 0.85q_1q_2 + 0.2125q_1q_2^2
\end{aligned} \tag{C.7}$$

$$\begin{aligned}
\dot{q}_3 &= 1.374(1+q_1)(\frac{1}{2}q_2 - \frac{1}{8}q_2^2 + \frac{1}{16}q_2^3) - 10q_3 + 8.626q_2 - 1.3740 \\
&= 1.3740q_1 + 9.31295q_2 - 10q_3 + 0.687q_1q_2 - 0.17175q_2^2 \\
&\quad -0.17175q_1q_2^2 + 085875q_2^3
\end{aligned} \tag{C.8}$$

$$\begin{aligned}
\dot{q}_4 &= 10(1+q_1)(\frac{1}{2}q_2 - \frac{1}{8}q_2^2 + \frac{1}{16}q_2^3) - 10(1+q_4) \\
&= 10q_1 + 5q_2 - 10q_4 + 5q_1q_2 - 1.25q_2^2 - 1.25q_1q_2^2 + 0.625q_2^3
\end{aligned} \tag{C.9}$$

In (C.7), (C.8), and (C.9), terms with an overall order greater than three are neglected. Similarly, substituting (C.6) in the equation for $\tilde{y}$ in (C.4), and neglecting terms greater than three, yields:

$$\begin{aligned}
\tilde{y} &= 140.12(q_4 - q_3)(1 - q_3 + q_3^2 - q_3^3) \\
&= 140.12(q_4 - q_4q_3 + q_4q_3^2 - q_3 + q_3^2 - q_3^3)
\end{aligned} \tag{C.10}$$

Thus the system of equations given in (2.3) which is a third-order Taylor series expansion of (2.1), is obtained.

APPENDIX D

CARLEMANN STATE-SPACE MATRICES

The Carlemannn linearized system for (2.1) in state-space form is given as:

$$
\begin{aligned}
\dot{\tilde{q}} &= A_c\tilde{q} + N_c\tilde{q}v + B_c v \\
\tilde{y}_c &= C_c\tilde{q}
\end{aligned}
$$

The matrices are given as: $A_c(:, 1:12) =$

$$
\begin{bmatrix}
-11.700 & -0.85028 & 0 & 0 & 0 & -0.85028 & 0 & 0 & 0.21250 & 0 & 0 & 0 \\
0 & -10.102 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1.3740 & 9.3130 & -10 & 0 & 0 & 0.68700 & 0 & 0 & -0.17175 & 0 & 0 & 0 \\
10 & 5.0 & 0 & -10 & 0 & 5.0 & 0 & 0 & -1.2500 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -23.401 & -1.7006 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -21.803 & 0 & 0 & -0.85028 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1.3740 & 9.3130 & -21.701 & 0 & 0 & -0.85028 & 0 & 0 \\
0 & 0 & 0 & 0 & 10 & 5.01 & 0 & -21.701 & 0 & 0 & -0.85028 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -20.204 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.3740 & 0 & 0 & 9.3130 & -20.102 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 5.01 & 0 & -20.102 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2.7481 & 0 & 0 & 18.626 & 0 & -20 \\
0 & 0 & 0 & 0 & 0 & 0 & 10 & 1.3740 & 0 & 5.01 & 9.3130 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 10 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$A_c(:, 13:24) =$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0.21250 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1.2500 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1.7000 & 0 & 0 & 0.42500 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -0.85000 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.68790 & 0 & 0 & -0.17175 & -0.85000 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 5.0 & 0 & 0 & -1.2500 & 0 & -0.85000 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.68700 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 5.0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.3740 & 0 & 0 & 0 & 0 \\
-20 & 0 & 0 & 0 & 0 & 0 & 0 & 5.0 & 0.68700 & 0 & 0 & 0 \\
0 & -20 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\
0 & 0 & -35.100 & -2.5500 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -33.502 & 0 & 0 & -1.7000 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1.3740 & 9.3130 & -33.400 & 0 & 0 & -1.7000 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 5.0 & 0 & -33.400 & 0 & 0 & -1.7000 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -31.904 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1.3740 & 0 & 0 & 9.3130 & -31.802 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 10 & 0 & 0 & 5.0 & 0 & -31.802 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2.7480 & 0 & 0 & 18.626 & 0 & -31.700 & 0 & 0 \\
0 & 0 & 0 & 0 & 10 & 1.3740 & 0 & 5.0 & 9.3130 & 0 & -31.700 & 0 \\
0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 10 & 0 & 0 & -31.700 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.3740 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.7482 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 1.3740 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4.1220 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 2.7480 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 1.3740 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 30
\end{bmatrix}$$

$A_c(:, 25:34) =$

$$\begin{bmatrix}
-0.10625 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
085875 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.62500 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.21250 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.21250 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.21250 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-0.17175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1.2500 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -0.34300 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1.2500 & -0.17175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -2.5000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-0.85000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -0.85000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.85000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -0.85000 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -0.85000 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -0.85000 & 0 & 0 & 0 & 0 \\
-30.307 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9.3130 & -30.204 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5.0 & 0 & -30.204 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 18.626 & 0 & -30.102 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 5.0 & 9.3130 & 0 & -30.102 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 10 & 0 & 0 & -30.102 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 27.938 & 0 & 0 & -30 & 0 & 0 & 0 \\
0 & 0 & 0 & 5.0 & 18.626 & 0 & 0 & -30 & 0 & 0 \\
0 & 0 & 0 & 0 & 10 & 9.3130 & 0 & 0 & -30 & 0 \\
0 & 0 & 0 & 0 & 0 & 15.0 & 0 & 0 & 0 & -30
\end{bmatrix}$$

$$B_c = \begin{bmatrix} 0 \\ 0.16698 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$C_c^T = \begin{bmatrix} 0 \\ 0 \\ -140.12 \\ 140.12 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 140.12 \\ -140.12 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -140.12 \\ 140.12 \\ 0 \\ 0 \end{bmatrix}$$

$N_c(:, 1:12) =$

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.16698 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.33396 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.16698 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.16698 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.16698 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.33800 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.16698 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.16698 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.50070 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.33380 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.33380 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.16698 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$N_c(:, 13:24) =$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.16698 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.16698 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$N_c(:, 25:34) = zeros(34, 10)$$

BIBLIOGRAPHY

[1] M. Nikolaou. NSF/NIST workshop on identification and adaptive control. Process measurement and control: Industry needs. *Comput. Chem. Eng.*, 23:217–227, 1998.

[2] K. J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, Reading, MA, second edition, 1989.

[3] M. Morari and E. Zafiriou. *Robust Process Control*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[4] M. A. Hussain. Review of the applications of neural networks in chemical process control simulation and on-line implementation. *Artif. Intell. Eng.*, 13(1):55–68, 1999.

[5] R. K. Pearson, B. A. Ogunnaike, and F. J. Doyle III. Identification of discrete convolution models for nonlinear processes. *IEEE Trans. Acoust., Speech, and Signal Processing*, 1995.

[6] D. E. Steinmeyer and M. L. Shuler. Structured model for *Saccharomyces cerevisiae*. *Chem. Eng. Sci.*, 44:2017–2030, 1989.

[7] J. W. Jeong, J. Snay, and M. M. Ataai. A mathematical model for examining growth and sporulation processes of *Bacillus subtilis*. *Biotech. Bioeng.*, 35:160–184, 1990.

[8] M. M. Domach, S. K. Leung, R. E. Cahn, G. G. Cocks, and M. L. Shuler. Computer model for glucose-limited growth of a single cell of *Escherichia coli B/r-A*. *Biotechnology and Bioengineering*, 26:203–216, 1984.

[9] P. A. Wisnewski, F. J. Doyle III, and F. Kayihan. Fundamental continuous pulp digester model for simulation and control. *AIChE J.*, 43:3175–3192, 1997.

[10] T. J. Crowley, E. S. Meadows, E. Kostoulas, and F. J. Doyle III. Control of particle size distribution described by a population balance model of semibatch emulsion polymerization. *J. Proc. Control*, 10:419–432, 2000.

[11] A. M. Zamamiri, Y. Zhang, M. A. Henson, and M. A. Hjortsø. Dynamics analysis of an age distribution model of oscillating yeast cultures. *Chem. Eng. Sci.*, 57:2169–2181, 2002.

[12] M. Shouche, H. Genceli, P. Vuthandam, and M. Nikolaou. Simultaneous constrained model predictive control and identification of DARX processes. *Automatica*, 34:1521–2530, 1998.

[13] A. Aoyama, F.J. Doyle III, and V. Venkatasubramanian. Control-affine fuzzy neural network approach for nonlinear process control. *J. Proc. Cont.*, 5:375–386, 1995.

[14] P. Agrawal, G. Koshy, and M. Ramseier. An algorithm for operating a fed-batch fermenter at optimum specfic-growth rate. *Biotech. Bioeng.*, 33:115–125, 1989.

[15] R. S. Parker and F. J. Doyle III. Optimal control of a continuous bioreactor using an empirical non-linear model. *Ind. Eng. Chem. Res.*, 40:1939–1951, 2001.

[16] F. J. Doyle III, B. A. Ogunnaike, and R. K. Pearson. Nonlinear model-based control using second order Volterra models. *Automatica*, (31):697–714, 1995.

[17] B. Maner and F. J. Doyle III. Simulated polymerization control using autoregressive plus Volterra–based MPC. *AIChE J.*, 43:1763–1784, 1997.

[18] L. Ljung. *System Identification: Theory for the User*. Prentice Hall PTR, Upper Saddle River, NJ, 2nd edition, 1999.

[19] R. S. Parker. *Model-Based Analysis and Control for Biosystems*. PhD thesis, Department of Chemical Engineering, University of Delaware, 1999.

[20] M. A. Henson. Nonlinear model predictive control: Current status and future directions. *Comput. Chem. Eng.*, 23:187–202, 1998.

[21] S. Skogestad and M. Morari. Understanding the dynamic behavior of distillation columns. *Ind. Eng. Chem. Res*, 27(10):1848–1862, 1988.

[22] L. S. Balasubramhanya and F. J. Doyle III. Nonlinear control of a high-purity distillation column using a traveling wave model. *AIChE J.*, 43(3):703–714, 1997.

[23] S. J. Qin and T. A. Badgwell. An overview of nonlinear MPC applications. In F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control: Assessment and Future Directions*. Birkhäuser, 1999.

[24] L. A. Zadeh. From circuit theory to systems theory. *IRE Proceedings*, 50:856–865.

[25] F. J. Doyle III, R. K. Pearson, and B. A. Ogunnaike. *Identification and Control Using Volterra Models*. Springer-Verlag, 2002.

[26] R. K. Pearson. Selecting nonlinear model structures for computer control. *J. Proc. Cont.*, 13:1–26, 2003.

[27] R. K. Pearson. Nonlinear input/output modeling. *J. Proc. Cont.*, 5(4):197–211, 1995.

[28] R. D. Nowak and B. D. Van Veen. Efficent methods for identification of Volterra filter models. *Signal Processing*, 38:417–428, 1994.

[29] D. E. Rivera, H. Lee, M. W. Braun, and H. D. Mittelmann. Plant-friendly system identification: A challenge for the process industries. In *IFAC Symposium on System Identification (SYSID 2003)*, Rotterdam, The Netherlands, 2003.

[30] R. S. Parker, D. Heemstra, F. J. Doyle III, R. K. Pearson, and B. A. Ogunnaike. The identification of nonlinear models for process control using tailored "plant–friendly" input sequences. *J. Proc. Control*, 11, Sp. Issue SI:237–250, 2001.

[31] M. W. Braun, R. Ortiz-Mojica, and D. E. Rivera. Application of minimum crest factor multisinusoidal signals for "plant-friendly" identification of nonlinear process systems. *Control Engineering Practice*, 10(3):301–313, March 2002.

[32] K. S. Narendra and P. G. Gallman. An iterative method for the identification of the nonlinear systems using the Hammerstein model. *IEEE Trans. Aut. Control*, 12:546, 1966.

[33] L. Ljung and T. Glad. *Modeling of Dynamic Systems*. Prentice Hall Information and System Sciences. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[34] D. G. Heemstra. Practical nonlinear model identification and control implementation. Master's thesis, Purdue University, 1996.

[35] J. D. Rubb and R. S. Parker. Glucose control in type I diabetic patients: A Volterra model-based approach. In *Proc. ADCHEM 2003*, 2003.

[36] R. S. Parker. Efficient nonlinear model predictive control: Exploiting the Volterra–Laguerre model structure. In *Proceedings of CPC VI*. CACHE Corporation, AIChE Symposia Series, 2002.

[37] Nonlinear Model Predictive Control Using Hammerstein Models, author = K. P. Fruzzetti and A. Palazoglu and K. A. McDonald, journal = J. Proc. Cont., year = 1995.

[38] S. J. Norquay, A. Palazoglu, and J. A. Romagnoli. Nonlinear model predictive control of pH neutralisation using Wiener models. In *IFAC World Congress*, pages 31–36, San Francisco, USA, 1996.

[39] R. S. Patwardhan, S. Lakshminarayanan, and S. L. Shah. Nonlinear model predictive control using PLS based Hammerstein and Weiner models. AIChE Conference, November 1997.

[40] W. J. Rugh. *Nonlinear System Theory - The Volterra/Wiener Approach*. The Johns Hopkins University Press, Baltimore, MD, 1981.

[41] Q. Zheng and E. Zafiriou. Volterra-Laguerre models for nonlinear process identification with application to a fluid catalytic cracking unit. *Ind. Eng. Chem. Res.*, 43:340–348, 2004.

[42] N. Wiener. *Nonlinear Problems in Random Theory*. Wiley, New York, 1958.

[43] Y. W. Lee and M. Schetzen. Measurement of the Wiener kernels of a non-linear system by cross-correlation. *Int. J. Control*, 2:237–254, 1965.

[44] P. Z. Marmarelis and V. Z. Marmarelis. *Analysis of Physiological Systems: The White-Noise Approach*. Plenum Press, NY, 1978.

[45] M. Schetzen. *The Volterra and Wiener Theories of Nonlinear Systems*. John Wiley & Sons, New York, NY, 1980.

[46] T. Koh and E. J. Powers. Second-order Volterra filtering and its application to nonlinear system identification. *IEEE Acoust. Speech Sig. Proc.*, ASSP-33(6):1445–1455, 1985.

[47] F. J. Doyle III, B. A. Ogunnaike, and R. K. Pearson. Nonlinear model-based control using second-order Volterra models. *Automatica*, 31:697–714, 1995.

[48] R. K. Pearson, B. A. Ogunnaike, and F. J. Doyle III. Identification of structurally constrained second-order Volterra models. *IEEE Trans. Acoust. Speech Sig. Proc.*, 44:2837–2846, 1996.

[49] M. Korenberg. Identifying nonlinear difference equation and functional expansion representations: The fast orthogonal algorithm. *Ann. Biomed. Eng.*, 16:123–142, 1988.

[50] Q. Zhang, B. Suki, D. T. Westwick, and K. R. Lutchen. Factors affecting Volterra kernel estimation: Emphasis on lung tissue viscoelasticity. *Annals of Biomedical Engineering*, 26:103–116, 1998.

[51] W. M. Ling and D. E. Rivera. A methodology for control-relevant nonlinear system identification using restricted complexity models. *J. Proc. Cont.*, 11:209–222, 2001.

[52] T. J. Dodd and R. F. Harrison. A new solution to Volterra series estimation. In *Proc. 2002 IFAC World Congress*, Barcelona, Spain, 2002.

[53] R. P. Parker Jr. and M. Tummala. Identification of Volterra systems with a polynomial neural network. In *Proc. 1992 IEEE Int. Conf. Acoust., Speech, Signal Processing*, volume 4, pages 561–564, San Francisco, CA, 1992.

[54] G. P. Liu, K. Kadirkamanathan, and S. A. Billings. On-line identification of nonlinear systems using Volterra polynomial basis function neural networks. *Neural Networks*, 11:1645–1657, 1998.

[55] D. Aiordachioaie, E. Ceanga, R. De Keyser, and Y. Naka. Detection and classification of nonlinearities based on Volterra kernels processing. *Engineering Applications of Artificial Intelligence*, 14:497–503, 2001.

[56] J. G. Németh, I. Kollár, and J. Schoukens. Identification of Volterra kernels using interpolation. *IEEE Trans. on Instrumentation and Measurement*, 51(4):770–775, 2002.

[57] A. Watanabe and L. Stark. Kernel method for nonlinear analysis: Identification of a biological control system. *Math. Biosci.*, 27:99–108, 1975.

[58] B. Wahlberg. System identification using Laguerre models. *IEEE Trans. Aut. Control*, 36(5):551–562, May 1991.

[59] G. A. Dumont and Y. Fu. Non-linear adaptive control via Laguerre expansion of Volterra kernels. *Int. J. Adaptive Control Sig. Proc.*, 7:367–382, 1993.

[60] V. Z. Marmarelis. Identification of Nonlinear Biological Systems Using Laguerre Expansions of Kernels. *Annals of Biomedical Engineering*, 21:573–589, 1993.

[61] M. J. Korenberg and I. W. Hunter. The identification of nonlinear biological systems: Volterra kernel approaches. *Annals of Biomedical Engineering*, 24:250–268, 1996.

[62] K. Alataris, T. W. Berger, and V. Z. Marmarelis. A novel network for nonlinear modeling of neural systems with arbitrary point-process inputs. *Neural Networks*, 13:255–266, 2000.

[63] G. D. Mitsis and V. Z. Marmarelis. Modeling of nonlinear physiological systems with fast and slow dynamics. I. Methodology. *Annals of Biomedical Engineering*, 30:272–281, 2002.

[64] Q. Zheng and E. Zafiriou. Nonlinear system identification for control using Volterra-Laguerre expansion. In *Proc. American Control Conf.*, pages 2195–2199, Seattle, WA, 1995.

[65] C. Seretis and E. Zafiriou. Nonlinear dynamical system identification using reduced Volterra models with generalized orthonormal basis functions. In *Proc. American Control Conf.*, pages 3042–3046, Albuquerque, NM, 1997. IEEE, Piscataway, NJ.

[66] S. J. Qin and T. A. Badgwell. An overview of industrial model predictive control technology. In *Proceedings of the Fifth International Conference on Chemical Process Control*, Tahoe City, CA, 1996.

[67] J. H. Lee. Modeling and identification for nonlinear predictive control: Requirements, current status and future research needs. In F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control: Assessment and Future Directions*. Birkhäuser, 1999.

[68] F. Allgöwer, T. A. Badgwell, J. S. Qin, J. B. Rawlings, and S. J. Wright. *Nonlinear Predictive Control and Moving Horizon Estimation – An Introductory Overview*, pages 391–449. Advances in Control – Highlights of ECC '99. Springer, London, 1999.

[69] M. Morari and J. H. Lee. Model predictive control: Past, present and future. In *PSE ESCAPE-7 Symposium*, Trondheim, Norway, 1997.

[70] J. B. Rawlings, E. S. Meadows, and K. R. Muske. Nonlinear model predictive control: A tutorial and survey. In *IFAC Symposium on Advanced Control of Chemical Processes*, pages 203–214, Kyoto, Japan, 1994.

[71] D. B. Patience, J. B. Rawlings, and H. A. Mohameed. Crystallization of para-xylene in scraped-surface crystallizers. *AICHE J.*, 47:2441–2451, 2001.

[72] R. D. Braatz and S. Hasebe. Particle size and shape control in crystallization processes. In *Proceedings of CPC VI*, AIChE Symposia Series. CACHE Corporation, 2002.

[73] Q. Zheng and E. Zafiriou. Control-relevant identification of Volterra series models. In *Proc. American Control Conf.*, pages 2050–2054, Baltimore, MD, 1994.

[74] R. K. Mutha, W. R. Cluett, and A. Penlidis. Modifying the prediction equation for nonlinear model–based predictive control. *Automatica*, 34:1283–1287, 1998.

[75] J. P. Congalidis, J. R. Richards, and W. H. Ray. Feedforward and feedback control of a solution copolymerization reactor. *AIChE J.*, 35(6):891–907, 1989.

[76] J. A. Florian, Jr. and R. S. Parker. A nonlinear data–driven approach to type I diabetic patient modeling. 15th IFAC World Congress on Automatic Control, Barcelona, Spain, 2002.

[77] B. Wahlberg and P. M. Mäkilä. On approximation of stable linear dynamical systems using Laguerre and Kautz functions. *Automatica*, 32(5):693–708, 1996.

[78] J. Kurth and H. Rake. Identification of nonlinear systems with reduced Volterra series. In *Preprints of the 10th IFAC Symposium on System Identification SYSID '94*, pages 143–150, Copenhagen, Denmark, 1991.

[79] E. K. P. Chong and S. H. Żak. *An Introduction To Optimization.* Wiley Interscience, New York, 1996.

[80] Y. Fu and G. A. Dumont. An optimum time scale for discrete Laguerre network. *IEEE Trans. Aut. Control*, 38:934–938, 1993.

[81] A. M. Sabatini. A hybrid genetic algorithm for estimating the optimal time scale of linear systems approximations using Laguerre models. *IEEE Trans. Automat. Contr.*, 45(5):1007–1011, 2000.

[82] R. Malti, S. B. Ekongolo, and J. Ragot. Dynamic SISO and MISO system approximations based on optimal Laguerre models. *IEEE Trans. Automat. Contr.*, 43(8):1318–1323, 1998.

[83] R. J. G. B. Campello, G. Favier, and W. C. Amaral. Optimal expansions of discrete-time Volterra models using Laguerre functions. *Automatica*, 40:815–822, 2004.

[84] B. R. Maner. *Polymerization Reactor Control Using Computationally Tractable Input-Output Models*. PhD thesis, School of Chemical Engineering, Purdue University, 1996.

[85] H. Akaike. A new look at the statistical model identification. *IEEE Trans. Aut. Control*, AC-19(6):716–723, December 1974.

[86] J. A. Florian, Jr. and R. S. Parker. Empirical modeling for glucose control in diabetes and critical care. *Eur. J. Control*, 11:(to appear), 2005.

[87] J. W. Eaton and J. B. Rawlings. Model predictive control of chemical processes. *Chem. Eng. Sci.*, 47(4):705–720, 1992.

[88] B. A. Ogunnaike and W.H. Ray. *Process Dynamics, Modeling, and Control*. Oxford University Press, New York, NY, 1994.

[89] L. T. Biegler and J. B. Rawlings. Optimization approaches to nonlinear model predictive control. In Y. Arkun and W.H. Ray, editors, *Proceedings of the Fourth International Conference on Chemical Process Control*, pages 543–571, Padre Island, TX, 1991.

[90] J. A. Florian, Jr., J. L. Eiseman, and R. S. Parker. A nonlinear model predictive control algorithm for breast cancer treatment. In *Proc. DYCOPS 7*, Boston, MA, 2004.

[91] M. J. Tenny and J. B. Rawlings. Closed-loop behavior of nonlinear model predictive control. *Process Systems Engineering*, 50(9):2142–54, 2004.

[92] R. Hovorka, V. Canonico, L. J. Chassin, U. Haueter, M. Massi-Benedetti, M. O. Federici, T. R. Pieber, H. C. Schaller, L. Schaupp, T. Vering, and M. E. Wilinska. Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiological Measurement*, 25:905–920, 2004.

[93] R. S. Parker, E. P. Gatzke, R. Mahadevan, E. S. Meadows, and F. J. Doyle III. Nonlinear model predictive control: Issues and applications. In B. Kouvaritakis and M. Cannon, editors, *Advances in Non-Linear Model Predictive Control*, Control Series. IEE Press; UK, 2001.

[94] L. T. Biegler. Efficient solution of dynamic optimization and NMPC problems. In *International Symposium on Nonlinear Model Predictive Control: Assessment and Future Directions*, pages 46–63, Ascona, Switzerland, 1998.

[95] J. W. Eaton and J. B. Rawlings. Feedback control of chemical processes using on-line optimization techniques. *Comput. Chem. Eng.*, 14(4/5):469–479, 1990.

[96] G. A. Dumont, Y. Fu, and G. Lu. Nonlinear adaptive generalized predictive control and applications. In D. Clarke, editor, *Advances in Model–based Predictive Control*, pages 498–515. Oxford University Press, 1994.

[97] R. S. Parker. Nonlinear model predictive control of a continuous bioreactor using approximate data–driven models. In *Proc. American Control Conf.*, pages 2885–2890, Anchorage, AK, 2002.

[98] M. Korenberg, S. A. Billings, Y. P. Liu, and P. J. McIlroy. Orthogonal parameter estimation algorithm for non-linear stochastic systems. *Int. J. Control*, 48(1):193–210, 1988.

[99] Q. Zheng and E. Zafiriou. Identification of MIMO Volterra series and application to FCC unit. In *Proceedings of the IFAC' 96 World Congress*, 1996.

[100] U. Forssell and L. Ljung. Closed-loop identification revisited. *Automatica*, 35:1215–1241, 1999.

[101] H. Genceli and M. Nikolaou. New approach to constrained predictive control with simultaneous model identification. *AIChE J.*, 42:2857–2868, 1996.

[102] R. S. Parker and F. J. Doyle III. Nonlinear model predictive control of a continuous bioreactor at near–optimum conditions. In *Proc. American Control Conf.*, pages 2549–2553, Philadelphia, PA, 1998. New York, NY.

[103] C. E. García. Quadratic dynamic matrix control of nonlinear processes. An application to a batch reaction process. In *Proc. AIChE Annual Meeting*, San Francisco, CA, 1984.

[104] A. Zheng. A computationally efficient nonlinear MPC algorithm. In *Proc. American Control Conf.*, pages 1623–1627, Albuquerque, NM, 1997.

[105] D. A. Cox. Introduction to Gröbner bases. In D.A. Cox and B. Sturmfels, editors, *Applications of Computational Algebraic Geometry*, volume 53 of *Proceedings of Symposia in Applied Mathematics*, pages 1–24, Providence, RI, 1998. American Mathematical Society.

[106] E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, New York, NY, 8th edition, 1999.