METHODS FOR THE IDENTIFICATION OF MULTIPLE-INPUT NONLINEAR SYSTEMS

David Thomas Westwick

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Department of Electrical Engineering
and
Biomedical Engineering Department
McGill University, Montréal

June 1995

ISBN 0-612-08168-0

Abstract

In this thesis, we developed techniques which are capable of identifying single and multiple-input, linear and nonlinear systems. They were derived within an analytical framework which imposes few restrictions on the nature of the input signals, and includes the possibility of measurement noise. Extensive simulations demonstrated that these methods are robust in the presence of measurement noise, and that they can be used with highly coloured test inputs. A series of experiments were performed on a known, physical, nonlinear system to validate the simulation results. Finally, an investigation of the stretch reflex electromyogram was used to demonstrate the applicability of these methods to a physiological system.

Résumé

Dans cette thèse, nous avons développé des techniques nous permettant d'identifier des systèmes linéaires et non-linéaires à entrées simples et multiples. Ces techniques sont dérivées d'une structure analytique qui n'impose que peu de restrictions sur la nature des signaux d'entrée. De plus, les signaux mesurés peuvent contenir un certain degré de bruit. Les résultats des simulations appronfondies ont servi à démontrer que les techniques présentées ici sont robustes en presence de bruit et peuvent être utilisées avec des signaux colorés. Des simulations ont été faites sur un système physique, non-linéaire et connu afin de démontrer la validité des résultats obtenus. Finalement, ces techniques ont été appliquées à un électromyogramme représentant une réponse réflexe de déplacement afin de démontrer la possibilité d'application de ces méthodes à des systèmes physiologiques.

Acknowledgements

I would like to acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada, the Medical Research Council of Canada, and the McGill Faculty of Medicine.

I owe a debt of gratitude to my thesis supervisor, Dr. Robert Kearney, for his advice and encouragement during my stay at McGill, and for financial support through his research grants.

I would like to thank Dr. ir. Michel Verhaegen, for rekindling my interest in linear algebra, and Dr. V.Z. Marmarelis for the many insightful comments that he gave me at several conferences over the years. In addition, I have had many fruitful discussions with fellow researchers in the Neuromuscular Control Lab. In particular, I would like to mention Jan Duha, Bob Kirsch, Bart MacNeil, Eric Perreault, and Jim Trainor.

Finally, I would like to thank my wife, Patricia, for her patience, and for not laughing too loudly when I would say "just another couple of months...".

Contents

List of Figures

List of Tables

List of Symbols

Symbol	Definition	Page
A, B, C, D	matrices used in state-space realization of a linear system	14
$E[x]$	expected value of x	
$G_n[K_n, x(t, \alpha)]$	n'th order Wiener functional, applied to $x(t, \alpha)$ using the kernel K_n	19
$h_n(\tau)$	in parallel Wiener cascades, the impulse response of the n'th cascade path.	33
$h_n(\tau_1, \ldots \tau_n)$	n'th order Volterra kernel	17
$h_{un}(\tau)$	for multiple input Wiener structures, the impulse response that is applied to the n'th input	71
$K_n(\tau_1, \ldots \tau_n)$	n'th order Wiener kernel	19
$k_{n,m}$	correction term in n'th order Wiener functional, that ensures that it is orthogonal to all m'th order functionals applied to the same input	19
$m(\cdot)$	single-input static nonlinearity. Given an input signal, $z(t)$, its output is $m(z(t))$	33
N	number of data points available	16
T	memory length of a dynamic system	15
$u(t)$	noise-free system input	6
$\vec{u}_n$	The n'th left singular vector of a matrix.	80
$\vec{v}_n$	The n'th right singular vector of a matrix.	80

Symbol	Definition	Page
$v_u(t)$ $v_z(t)$	Measurement noise applied to the input and output signals, respectively	6
$v_j(t)$	Residuals after j iterations of the parallel cascade method	33
$w_1(t)$ $w_2(t)$	Disturbances on the output and input signals, respectively	6
$x_n(t)$	In parallel cascade systems, the output of the initial linear subsystem in the n'th path.	33
$x(t, \alpha)$	Brownian motion input used in Wiener's original orthogonalization of the Volterra series	19
$y(t)$	noise-free system output	6
$z(t)$	measured system output (may contain noise)	6
σ_x	standard deviation of the random variable x	
ϕ_{uy}	first-order cross-correlation between $u(t)$ and $y(t)$	
$\mu(t)$	white-noise input sequence. filtered to produce $u(t)$, where $u(t)$ is non-white	71
τ	time-lag in convolution sums and integrals	15

Chapter 1

Introduction

In the introduction to treatises on the subject of system identification, it is not uncommon to find the authors drawing comparisons between the construction of mathematical models and the pursuit of knowledge itself [11, 65]. Further musings invariably extend the analogy, casting system identification in the role of the scientific method. To further extend this allegory, the development of new algorithms for system identification can be seen as the creation of new tools for the scientist. In making this observation, an important point is raised. The development of new algorithms should not be an end in itself; they are tools, and nothing more. Foremost in the mind of any tool builder should be the applications, for without these, of what use is a tool?

In this thesis, we have attempted to further the discipline of nonlinear system identification by developing tools which may be applied under fairly liberal conditions. Traditionally, the application of nonlinear system identification has been limited by the need for rather stringent assumptions about the properties of the data. We have eased some of these restrictions, allowing wider use of these techniques.

We have designed our tools considering the requirements imposed by the study of joint dynamics, where the systems being examined can be highly nonlinear and involve the interaction of several inputs. Furthermore, the nature of the experimental apparatus imposes severe constraints on the "richness" of the input stimulus. Finally, measurements of the system inputs and outputs are often corrupted by measurement noise. Although we have developed these tools for this particular application, we believe they will be usable in a large number of disciplines.

1.1 Overview

Chapter 2 presents a general introduction to the topic of system identification together with a discussion of the requirements and constraints inherent in the study of human joint dynamics. This is followed with a detailed review of the recent system identification literature, paying particular attention to methods that may be applicable to the study of biomedical systems, and in particular, human joint dynamics.

Chapter 3 presents a theoretical examination of an established algorithm which is used widely for the identification of linear systems. Based on this analysis, we develop a new algorithm that yields estimates with dramatically lower variances than existing methods. This procedure will be used in the algorithms developed in later chapters.

Chapter 4 introduces an important block structured nonlinear system: the multi-input Wiener structure. After a theoretical investigation of this nonlinear system, we develop algorithms for its identification. In later chapters, we will construct more general nonlinear systems using both single- and multiple-input Wiener structures as "building blocks".

Chapter 5 presents algorithms for the identification of general nonlinear systems. We start with the parallel cascade method, recently proposed by Korenberg [56], which models an unknown system as a sum of Wiener models. Our contribution is a new procedure that finds the "best" possible Wiener cascade at each stage. This results in faster convergence, simpler models, and better noise performance than the original method. Furthermore, we will show that this expansion depends only on the statistics of the input and is therefore unique. We will present algorithms for both single- and multiple-input systems.

In Chapter 6, we present two applications of the techniques developed in this thesis. First, we construct a known, physical, nonlinear system, using several linear filters, and a four quadrant analogue multiplier. Using this system, we validate the identification methods under experimental conditions. Secondly, we investigate the dynamics of the stretch reflex electromyogram (EMG), using the methods developed in Chapters 3 and 5. The analysis of this system is used as an example, illustrating how and when the various algorithms developed in this document can be applied.

Finally, in Chapter 7, we summarize the contributions made in this thesis, and offer suggestions for further developments and improvements. We finish the chapter by discussing further applications for these techniques.

Chapter 2

Literature Review

2.1 Overview

In this review, our primary objective will be to describe techniques that are suitable for building models of the human peripheral neuromuscular system. Our description of this system will reveal multiple inputs and outputs, nested feedback loops, and the presence of several nonlinearities. A survey of the techniques used to model systems of this complexity reveals two broad classes: *a priori* or morphological modelling, and *a posteriori* or black-box modelling, which is also known as system identification. We begin our discussion by describing these two families of techniques in general terms.

To place this discussion in context, we will consider the problems associated with the modelling of human joint dynamics. With reference to the recent literature, we will describe the system and discuss the types of experiments that can be performed, each of which will limit our choice of modelling techniques in different ways. Thus, a minimum level of model complexity will be mandated by the nature of the system itself, whereas limitations in the data that can be obtained experimentally will dictate an upper bound on the model complexity that can be justified. Given this perspective, we will discuss the merits and pitfalls of both *a priori* and "black-box" modelling as applied to the study of the peripheral neuromuscular control system.

2.1.1 Morphological vs. Black-Box Models

A priori modelling uses analysis based on first principles, knowledge of the system structure, and the function of its subsystems, to create models of an entire system. These models are frequently referred to as morphological models since the individual elements and their interconnections are often related directly to the structure of the system being modelled.

Typically, such models incorporate a large number of parameters that must be determined experimentally. For examples of *a priori* models of the peripheral neuromuscular system see the series of review articles by Agarwal and Gottlieb [1, 2, 3].

In contrast, the *a posteriori* approach attempts to model the system without making assumptions about its structure. This approach is sometimes referred to as black-box modelling, since the resulting model is simply a "black box" whose behaviour mimics that of the system. This class of models describes the relationship between the system inputs and outputs, but may provide little structural or functional information about the system or its components.

In general, there are two uses for the models produced by system identification: control and understanding. In the design of control systems, particularly predictive control systems, models are needed to predict the plant's response to its input in order to design an effective controller. It is usually desirable to have the simplest possible model that describes the dynamics of the plant to be controlled.

The other broad application of system identification, pursued here, is to gain insight into the operation of a system. In this case, we want to extract the maximum amount of information from the input/output data. If one is willing to believe that a more complicated model will lead to better understanding of the system, it can be argued that our objective should be to create the most complex model that can be justified by the data. In any case, models identified for insight are often much more complex than those used for control.

Let us consider the general identification problem, as posed in Verhaegen and Dewilde [99], and illustrated in Figure 2.1. We will use this figure to establish the

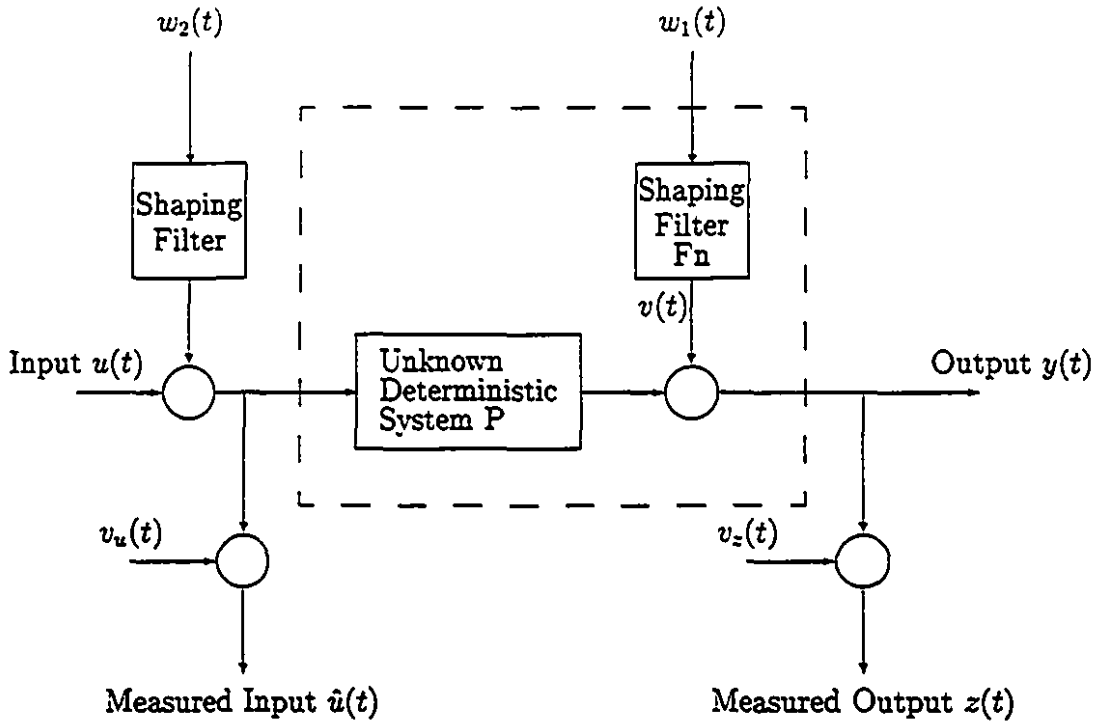


Figure 2.1: Block diagram representation of a generalized identification problem. Redrawn from [99]

notation used in the rest of this discussion.

We will define the "system" to consist of everything within the dotted box in Figure 2.1. It consists of two parts: one stochastic, and one deterministic. The stochastic part of the system is driven by a white-noise process, $w_1(t)$, which is not available to the experimenter. The deterministic part is driven by the sum of a controlled input, $u(t)$, and a filtered version of an inaccessible white noise process, $w_2(t)$. In addition to having control over $u(t)$, we will assume that the experimenter has access to a noise corrupted version of the complete input signal, $\hat{u}(t)$.

The noise-free output, $y(t)$, is the sum of the outputs of the deterministic and stochastic parts of the system. The experimenter, however, usually only has access to a corrupted version of the output signal, $z(t)$.

Given this structure, several problems can be addressed.

- Identification of the deterministic model, P. This consists of finding a relationship between $u(t)$ and $y(t)$, assuming that the process noise, $w_1(t)$, is zero. Note that both the input and output may still be corrupted by observation noise, $v_u(t)$ and $v_z(t)$, respectively. The identification of deterministic systems is generally pursued when the objective is to gain insight into the functioning of a system. This is the problem that is pursued in the balance of this thesis.

- Identification of the noise model, Fn. We are interested here in the relationship between $w_1(t)$ and $y(t)$, given observations of only the system output, $y(t)$. Usually, the input signal, $u(t)$, is assumed to be zero or constant. This type of identification problem is used in applications such as the study of economic systems, where the inputs are not available to the experimenter, or where it is unclear which signals are inputs, and which are outputs.

- Identification of the complete model. Given both the input and the output, estimate both P and Fn, the deterministic and noise models. This problem formulation is used when accurate predictions are desired, such as in the design of model based control systems.

2.1.2 Modelling of Joint Dynamics

In this section, we will consider how the problems of modelling joint dynamics relate to the *a priori* modelling and system identification approaches outlined above. We will start by defining the problem, and then examine the strengths and weaknesses of each approach.

Figure 2.2, redrawn from [38] and modified to include the myoelectric signal output (EMG), shows a simplified block diagram of information flow in the peripheral nervous system. Several important simplifications have been made in this diagram:

- The actions of agonist and antagonist muscle groups have been lumped into a single block (the dashed box).

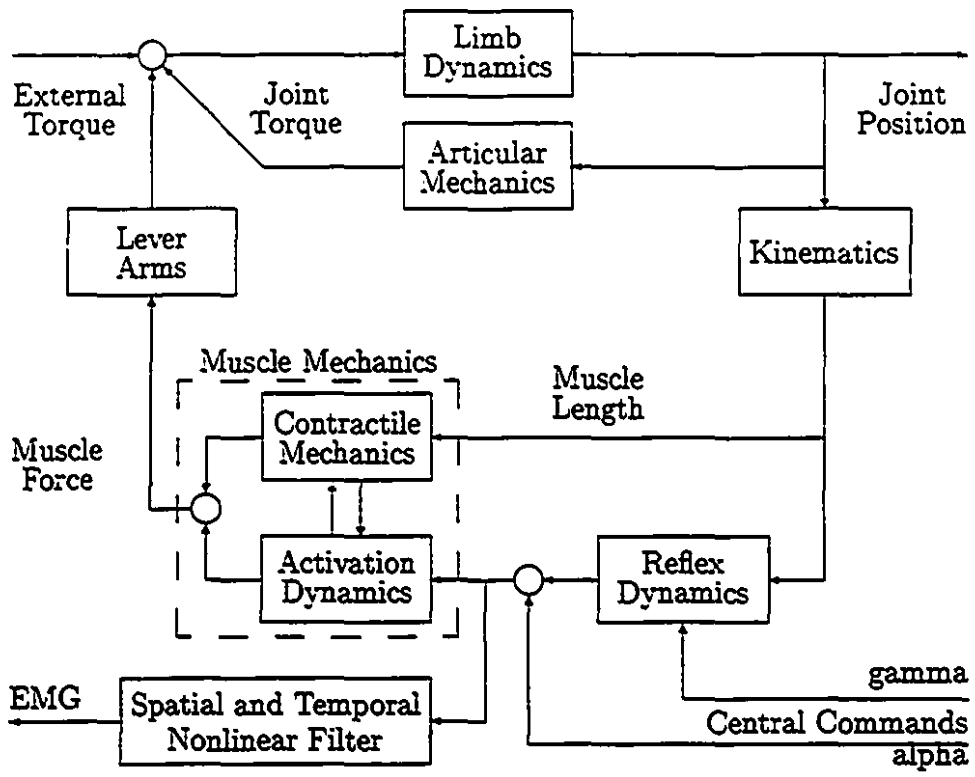


Figure 2.2: Information Flow Diagram for the Peripheral Neuromuscular System

- The myriad descending commands have been lumped into two inputs: one to the alpha motoneuron pool, which drives the muscle through the "activation dynamics", and one to the "reflex dynamics".

- The gain associated with the descending "alpha" command channel, as well as that of the reflex dynamics, can be modulated via interneurons [33]. This command channel is not represented in the diagram, in any form.

- Inputs from the receptors associated with other muscles, as well as those from receptors other than the muscle spindles, are entirely absent from this diagram.

Despite these simplifications, Figure 2.2 still represents a complex system, which will be difficult to model accurately, whatever approach is employed. Analyses based on either morphological modelling or system identification each have distinct disadvantages, which will be dealt with in turn.

Constructing an *a priori* model of the dynamics of a single joint is an extremely difficult undertaking. Even the simplified schematic shown in Figure 2.2 contains several subsystems and many interconnections. Treated in isolation, each block shown in Figure 2.2 has been the subject of extensive modelling efforts [1, 2, 3]. An overall model must include each of these subsystems and account for interactions between them.

Validation of such a complex model poses its own problems. Assuming a model of the whole system could be postulated, how does one prove that it mimics the original system at all levels of detail included in the model? Without such a validation, of what significance is the model? While the detailed structure of the model may suggest explanations of how the system functions, without adequate model validation such explanations remain speculative.

On the other hand, the chief disadvantage of black box models is that they provide no direct functional or structural information. They can, however, act as a reference against which morphological models can be validated. Thus, black box modelling may provide the means to validate morphological models, and so provide the functional insight which is our final objective.

We will now focus on the system identification approach. Our first step will be to relate the system shown in Figure 2.2 to the general identification problem outlined by Figure 2.1. As we can see, the system has one input, the external torque, that can be directly manipulated by the experimenter, as well as two inputs from the central nervous system, which are inaccessible. The outputs from the system are the joint position and the myoelectric signal.

Consider first the problems associated with applying the inputs. A mechanical actuator [39], using appropriate feedback, can be configured as either a torque or position servo. Figure 2.3 shows how an actuator can be used to generate a controllable torque input. In such an experiment, the position could be measured and fed back to the subject, who would be instructed to maintain a constant position. An alternative experiment is illustrated in Figure 2.4. In this case, the actuator is used to make the joint track a given position command input. The subject would be instructed to

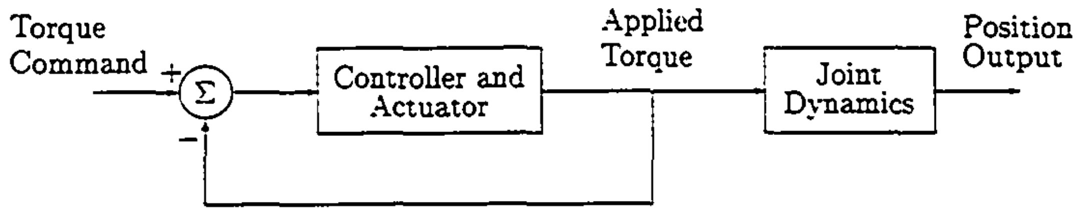


Figure 2.3: Actuator configured as a torque servo

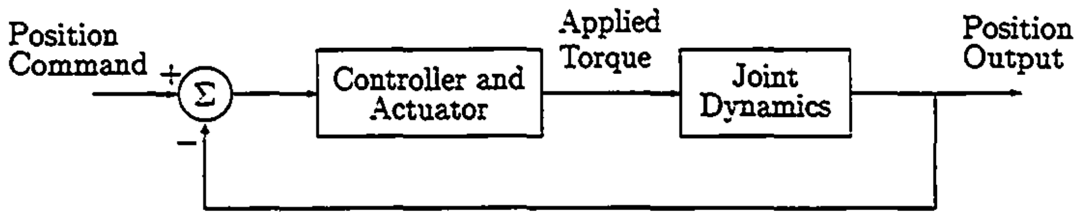


Figure 2.4: Actuator configured as a position servo

generate a prescribed muscle force, given the measured reaction torque as feedback.

Kearney and Hunter discussed the merits and pitfalls of both experimental approaches in a review of the joint dynamics literature [38]. They determined that generating a position input is more technically demanding, as a faster, more powerful actuator is required than for the torque input case. Their analysis concluded that, all things being equal, a position control experiment should yield better estimates at relatively high frequencies, whereas torque control experiments should produce better estimates of the low frequency response.

However the actuator is configured, the signal applied to the physiological system will be a torque whose spectrum is dependent both on the spectrum of the input signal and the dynamics of the actuator. In general, the actuator will act as a low-pass filter. For example, the electro-hydraulic actuator described by Kearney *et. al.* [39] had a gain which was flat to 25 Hz, followed by a third-order (60 dB per decade) roll-off. A more recent design [117] provides a flat gain to 100 Hz, which then rolls off at 60 dB per decade. Thus, the spectrum of the test input will depend on the dynamics of the actuator. Any system identification method must take this non-white input spectrum into account.

We must also concern ourselves with measurement noise. Sources include nonlinearities in the position and torque transducers, electrical noise in the signal conditioning apparatus, environmental noise, and quantization noise produced by the analogue to digital converters.

If the system is likely to remain constant over extended periods of time, noise effects can be reduced by collecting long data records. This option is not available in studying joint dynamics, as long experiments are likely to induce fatigue in the subject. While the extent of the changes induced by fatigue is not clear, for example compare [29] with [46], it is clear that we must minimize the effects of noise by using robust techniques rather than by relying on the averaging properties of long data records.

Finally, we must acknowledge that the system contains nonlinearities. For example, there is evidence [37] to suggest that the "Reflex Dynamics" block in Figure 2.2 responds primarily to the velocity of muscle stretching, implying the presence of something like a half-wave rectifier.

Further evidence supporting the existence of nonlinearities is provided by the quasi-linear analysis performed by Weiss *et. al.* [104, 105, 106, 107, 108]. A second-order, linear model described the relationship between torque and position well, provided that the position input was limited to small perturbations around a fixed mean position. The parameters of the second-order model varied strongly with the mean position, the level of background contraction, and the size of the perturbations. The changes in the linearized model strongly suggest the presence of underlying nonlinear behaviour.

Therefore, we need system identification methods that are capable of identifying nonlinear systems using non-white test inputs, and that are robust in the presence of measurement noise in the input, the output, or preferably both. Although we can only manipulate one of the inputs directly, we can modulate the descending inputs either electrically, [92], or by asking the subject to track a moving target signal [45, 67]. Because these limited two-input experiments are possible, we will also examine methods for identifying multiple-input systems.

The balance of this review is organized as follows:

- Section 2.2 will explore the various representations used to describe linear systems and the methods used to identify them from measurements of input-output data.
- Section 2.3 will consider several descriptions of nonlinear systems and the techniques used in their identification. This treatment will be limited to single-input systems.
- Section 2.4 will describe how some of the methods reviewed in Sections 2.2 and 2.3 have been extended to deal with multiple-input systems.
- Section 2.5 summarizes the methods discussed in this review, concentrating on their potential application to the study of joint dynamics.

2.2 Linear System Identification

A linear system obeys two properties: superposition and scaling. Hence, if F is a linear operator:

$$\begin{array}{l} y_1(t) = F(u_1(t)) \\ y_2(t) = F(u_2(t)) \end{array} \quad \Rightarrow \quad F(k_1u_1(t) + k_2u_2(t)) = k_1y_1(t) + k_2y_2(t)$$

2.2.1 Parametric Representations

A parametric model consists of a set of differential or difference equations which describe the system dynamics. Such equations usually contain a small number of parameters, which can be varied to alter the behaviour of the equations. The identification of an unknown system comprises two stages. First the structure of the parametric model is chosen, then the parameters themselves are estimated. What follows is a brief description of some important parametric linear system structures. For a comprehensive review of the techniques used to identify them, see Caines [11], or Ljung [65].

2.2.1.1 Linear Difference Equations

We can write the relationship between the input, output, and noise as a linear difference equation. Following Ljung [65], we write:

$$\begin{aligned} y(t) + a_1 y(t-1) + \ldots + a_{n_a} y(t-n_a) = & \\ & b_1 u(t-1) + b_2 u(t-2) + \ldots + b_{n_b} u(t-n_b) + \\ & e(t) + c_1 e(t-1) + \ldots + c_{n_c} e(t-n_c) \end{aligned} \tag{2.1}$$

which can be written more compactly:

$$A(q)y(t) = B(q)u(t) + C(q)e(t) \tag{2.2}$$

where $A(q) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a}$ and q^{-1} is the backward shift operator. This is the auto-regressive, moving average exogenous (ARMAX) model. The current output, $y(t)$, depends on an exogenous input, $u(t)$, an innovations process, $e(t)$, and the past values of the output. With respect to Figure 2.1, the polynomials $(A(q), B(q))$ correspond to the deterministic model, P, whereas $(A(q), C(q))$ represent the stochastic system, Fn. This model has several special cases, the first of which is the autoregressive (AR) model:

$$A(q)y(t) = e(t)$$

in which the output depends on the current disturbance, as well as the n_a previous values of the output.

Another special case is the moving average (MA) model:

$$y(t) = C(q)e(t)$$

in which the output depends on the previous values of the disturbance, $e(t)$.

Combining these two, we get the autoregressive moving average (ARMA) model:

$$A(q)y(t) = C(q)e(t)$$

If we add an accessible input, $u(t)$, to the AR model, the result is an auto-regressive exogenous input (ARX) model:

$$A(q)y(t) = B(q)u(t) + e(t)$$

A special case of the ARX structure, in which there is no disturbance input, is the finite impulse response (FIR) model:

$$y(t) = B(q)u(t)$$

In this case, the output depends solely on the previous values of the exogenous input. This structure forms the basis of many so-called non-parametric identification schemes.

Once a candidate model structure and order have been chosen, the model representation can be reduced to a parameter vector, $\Theta = [A(q)B(q)C(q)]$. The identification problem, then, is to find the optimal vector in parameter space, given a particular cost function.

2.2.1.2 State Space Models

Another parametric system representation is the state space model. In this case, we consider a set of equations of the form:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{2.3}$$

where the sequences u_k, y_k, and x_k represent the system's input, output, and state respectively. The classical method for identifying these models is the Ho-Kalman [28, 118] realization scheme. The impulse response (Markov parameters) of the system is first identified from input-output data, and then used to compute the system matrices, A,B,C, and D .

Recently, Verhaegen and co-workers [96, 97, 98, 99] proposed a class of subspace identification methods which estimate the system matrices directly, to within a similarity transform, from the input/output data. In this approach, the input and output data sequences are entered into a Hankel matrix, which is then compressed by an RQ factorization [20]. By partitioning R, it is possible to recover a matrix with the same column space as that of the extended observability matrix. Applying a singular value decomposition [20] to this partition, and retaining only the singular vectors that correspond to significant singular values, recovers the observability matrix, to within a similarity transformation. The number of singular values retained determines the rank of the observability matrix, and hence the order of the system. As a result, the task of model order selection is performed explicitly, in contrast with other parametric identification methods that generally rely on a separate model order test, such as one of the many variants of the Akaike information criterion (AIC) [4, 11].

Similar methods have been proposed by Van Overschee and de Moor [94, 95], and Moonen and Ramos [74]. Recently, subspace fitting techniques, originally developed for array signal processing [91, 93, 102], have been used to increase the accuracy of the estimates of the system matrices [76].

2.2.2 Nonparametric Representations

A linear system can be represented by its impulse response. In continuous time, we can compute the output via the convolution integral:

$$y(t) = \int_0^T h(\tau)u(t-\tau)d\tau \tag{2.4}$$

where T is the memory length of the system, and $h(\tau)$ is the impulse response. In this case, as the lower bound of the integration is 0, the system is causal.

Given that the analysis will be performed using sampled data on a digital computer, we will require a discrete time formulation. One benefit gained by restricting ourselves to discrete time is that it avoids the mathematical difficulties associated with a continuous-time white-noise signal. In continuous time, a white noise signal has infinite bandwidth and hence infinite power. In discrete time, however, it is simply a sequence of independently distributed random variables. In discrete time, the convolution integral becomes the summation:

$$y(t) = \Delta t \cdot \sum_{\tau=0}^{T-1} h(\tau)u(t-\tau) \tag{2.5}$$

Here the memory length, T, and the lag, τ, are integers. If the system is noncausal, then the lower limit of the summation will be negative. The sampling increment is Δt; for notational simplicity, we will assume that the sampling increment is 1, so that it can be dropped.

If the input process is white, it can be shown [68] that the impulse response can be recovered from the input/output cross-correlation function. Given N data points, a biased estimate of the cross-correlation [5] can be obtained as:

$$\hat{\phi}_{uy}(\tau) = \frac{1}{N} \sum_{t=\tau+1}^{N} u(t-\tau) y(t) \tag{2.6}$$

Substituting (2.5) into (2.6)

$$\begin{aligned} \hat{\phi}_{uy}(\tau) &= \frac{1}{N} \sum_{t=\tau+1}^{N} u(t-\tau) \sum_{j=0}^{T-1} h(j) u(t-j) \\ &= \sum_{j=0}^{T-1} h(j) \left\{ \frac{1}{N} \sum_{t=\tau+1}^{N} u(t-\tau) u(t-j) \right\} \\ &= \sum_{j=0}^{T-1} h(j) \hat{\phi}_{xx}(\tau - j) \end{aligned} \tag{2.7}$$

Hence, the input-output cross-correlation is equal to the convolution of the impulse response with the input auto-correlation function. If the input is white, the auto-correlation function is an impulse, and the cross-correlation and impulse response are equal. If the input is non-white, the input auto-correlation function must be deconvolved, somehow, from the cross-correlation estimate.

Ljung [65] approached this problem by modelling the observed input as a white-noise process filtered by an autoregressive filter. This filter can be estimated, and its inverse (a moving average filter) applied to both the input and output signals. The cross-correlation between the filtered input and filtered output is then estimated. Since the filtered input signal is effectively white, the cross-correlation estimate provides an estimate of the impulse response.

Hunter and Kearney [30] used a different approach. The input auto-correlation was estimated, and the convolution between the input auto-correlation and the impulse response written in matrix form:

$$\begin{bmatrix} \hat{\phi}_{uy}(0) \\ \hat{\phi}_{uy}(1) \\ \vdots \\ \hat{\phi}_{uy}(T-1) \end{bmatrix} = \begin{bmatrix} \hat{\phi}_{uu}(0) & \hat{\phi}_{uu}(1) & \dots & \hat{\phi}_{uu}(T-1) \\ \hat{\phi}_{uu}(1) & \hat{\phi}_{uu}(0) & \dots & \hat{\phi}_{uu}(T-2) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\phi}_{uu}(T-1) & \hat{\phi}_{uu}(T-2) & \dots & \hat{\phi}_{uu}(0) \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(T-1) \end{bmatrix} \quad (2.8)$$

which may be written compactly as:

$$\hat{\phi}_{uy} = \hat{\Phi}_{uu} h \quad (2.9)$$

This equation can be solved efficiently, using Levinson's algorithm [20], since $\hat{\Phi}_{uu}$, the matrix derived from the input auto-correlation function, has a symmetric Toeplitz structure. As we shall see in Chapter 3, this procedure has substantial numerical advantages over the scheme proposed by Ljung [65].

2.3 Nonlinear System Identification

2.3.1 Functional Expansion Methods

As was shown in Equation (2.5), a linear system can be represented by its impulse response; superposition guarantees that this fully characterizes the system. Volterra [103] developed a generalization of this representation for nonlinear systems in which the single impulse response is replaced with a series of integration kernels. This generalization of the impulse response, usually called the Volterra series, can be used to approximate a wide variety of systems. Indeed, Boyd and Chua [9] showed that a finite Volterra series can be used to approximate any time invariant operator which has fading memory. In the general case, the system output is generated by a series of generalized convolutions:

$$y(t) = \sum_{n=0}^{\infty} \int_0^T \cdots \int h_n(\tau_1, \dots, \tau_n) u(t-\tau_1) \dots u(t-\tau_n) d\tau_1 \dots d\tau_n \quad (2.10)$$

The zero-order term, h_0, is a constant, and is independent of any input to the system. Clearly, for linear systems this term will be zero.

The first order kernel is similar to the linear impulse response. Indeed, for linear systems the Volterra series collapses into its first order term, which is then precisely equal to the impulse response. In any case, the first-order term represents that part of the system response to an impulsive input that scales linearly with the weight of the impulse. Its output is computed using a convolution integral:

$$\int_0^T h_1(\tau)u(t-\tau)d\tau \tag{2.11}$$

The second-order Volterra kernel is used to compute the system response due to the interaction of pairs of impulsive inputs. The second-order response is computed via a generalized convolution integral:

$$\int_0^T \int_0^T h_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)d\tau_1 d\tau_2 \tag{2.12}$$

For this purpose, a single impulse can be thought of as a pair of coincident impulses. Thus, the diagonal elements of the second-order kernel give rise to that part of the impulse response which scales with the square of the weight of the impulsive input. Similarly, the n'th order Volterra kernel can be used to compute the system response to n impulsive inputs, and its diagonal values will correspond to that component of the impulse response which scales with the n'th power of the weight of the impulsive input.

While this representation is useful in computing the system response to a given input, the terms in the series are not orthogonal, and therefore must be identified all at once. A least squares solution to this problem, demonstrated by Doukoglou and Hunter [17, 32], is computationally intensive, even for low-order systems with comparatively short memory lengths.

Wiener [113] proposed a solution to this problem in which the Volterra series is orthogonalized using a Gramm-Schmidt orthogonalization [20], assuming that the input is a one-dimensional Brownian motion, $x(t,\alpha)$. Wiener [113] defined Brownian motion as the motion of a particle, in one dimension, such that given a reference time, t_1, the departure at any time, t_2, from the original reference position has a Gaussian distribution. In addition, the distributions taken over two non-overlapping time increments must be independent.

The parameter α, in the Brownian motion $x(t,\alpha)$, ranges continuously from 0 to 1, and determines its path. Each path that the Brownian motion could follow is associated with a unique value of α. Thus, integrating over α from 0 to 1 is equivalent to integrating over the distribution of all possible Brownian motions.

Other investigators [62] recast Wiener's continuous time formulation in discrete time, replacing the Brownian motion input with a "white" Gaussian signal with variance σ_u^2. Each sample of this discrete time white noise sequence is the time integral of a one-dimensional Brownian motion over the corresponding sample period. Thus, this signal becomes a sequence of independent Gaussian random variables.

In both cases, the terms in the Wiener series are orthogonal, and can be estimated individually. Following the developments given by Wiener [113], the system output can be written:

$$y(t) = \sum_{n=0}^{\infty} G_n[K_n, x(t, \alpha)] \tag{2.13}$$

K_n is referred to as the n'th order Wiener kernel, and the functionals, $G_n[K_n, x(t, \alpha)]$, will be chosen to be orthogonal, given a Brownian motion input, $x(t, \alpha)$. Wiener's approach involves constructing each successive functional, G_n, such that it is orthogonal to any homogeneous functional of lower order. The first such functional, K_0, is of zero order, and is a constant. A general form for the first order functional is:

$$G_1[K_1, x(t, \alpha)] = \int K_1(\tau) dx(\tau, \alpha) + k_{1,0}$$

To make this orthogonal to any zero-order functional, we must solve:

$$E\left[G_1[K_1, x(t, \alpha)] F_0[x(t, \alpha)]\right] = 0$$

for any zero order functional, $F_0[x(t, \alpha)]$. As $x(t, \alpha)$ is a Brownian motion, setting $k_{1,0}$ equal to zero solves this and orthogonalizes the first two functionals.

Similarly, starting with a general, second-order functional:

$$G_2[K_2, x(t, \alpha)] = \int\int K_2(\tau_1, \tau_2)\, dx(\tau_1, \alpha)\, dx(\tau_2, \alpha) \; + \int k_{2,1}(\tau) dx(\tau, \alpha) + k_{2,0}$$

and orthogonalizing it with respect to all zero and first-order functionals yields:

$$G_2[K_2, x(t, \alpha)] = \int\int K_2(\tau_1, \tau_2) dx(\tau_1, \alpha) dx(\tau_2, \alpha) - \int K_2(\tau, \tau) d\tau$$

Note that in the preceding equations, the functionals have been defined in terms of "Stieltjes" integrals, where the integration takes place along the trajectory of the Brownian motion. Wiener [113] demonstrated that this could be transformed into a time integral by integrating by parts:

$$\int_0^1 f(t)dx(t,\alpha) = f(1)x(1,\alpha) - \int_0^1 f'(t)x(t,\alpha)dt$$

provided that the derivative of $f(t)$ exists, and is bounded. The boundedness of the derivative of the kernels is one of the conditions that restricts the class of functions that can be represented by the Wiener series.

Lee and Schetzen [62] reformulated the Wiener series in discrete time using a Gaussian "white noise" input signal. In this formulation, the Stieltjes integrals of Wiener's [113] Brownian motion formulation are replaced with generalized convolutions. Palm and Poggio [78] investigated the mathematical implications of this change in the Wiener series formulation. In particular, they distinguished between the "Stieltjes kernels" employed by Wiener [113] and the "symbolic kernels" used by Lee and Schetzen [62]. They concluded that the validity of the Lee-Schetzen method is restricted to the class of systems whose derivatives belong to the original Wiener class of systems, and that this new class of systems is smaller than the Wiener class.

In the discrete-time framework, the first three symbolic Wiener functionals become:

$$\begin{aligned}
G_0[K_0, u(t)] &= K_0 \\
G_1[K_1, u(t)] &= \sum_{i=0}^{R-1} K_1(i)u(t-i) \\
G_2[K_2, u(t)] &= \sum_{i,j=0}^{R-1} K_2(i,j)u(t-i)u(t-j) - \sigma_u^2 \sum_{i=0}^{R-1} K_2(i,i)
\end{aligned}$$

Similarly, the n'th order discrete Wiener functional can be written:

$$G_n[K_n, u(t)] = \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \sum_{j_1=0}^{R-1} \cdots \sum_{j_{n-2i}=0}^{R-1} \sum_{k_1=0}^{R-1} \cdots \sum_{k_i=0}^{R-1} \frac{(-1)^i n! \, \sigma_u^{2i}}{2^i (n-2i)! \, i!} \cdot$$

$$K_n(j_1, \ldots, j_{n-2i}, k_1, k_1, \ldots, k_i, k_i) u(t-j_1) \ldots u(t-j_{n-2i})$$

where $\lfloor \frac{n}{2} \rfloor$ refers to the largest integer less than or equal to $\frac{n}{2}$.

Marmarelis and Marmarelis [68] and Rugh [84] give equations relating the Wiener and Volterra kernels of nonlinear systems. Given the Volterra kernels, h_n, the Wiener kernels are found to be:

$$K_n(\tau_1, \ldots, \tau_n) = \sum_{j=0}^{\infty} \frac{(n+2j)! \sigma_u^{2j}}{n! j! 2^j}$$
$$\int_{-\infty}^{\infty} h_{(n+2j)}(\tau_1, \ldots, \tau_n, \sigma_1, \sigma_1, \ldots, \sigma_j, \sigma_j) d\sigma_1, \ldots, d\sigma_j$$

Similarly, given the Wiener kernels, the Volterra kernels can be computed as:

$$h_n(\tau_1, \ldots, \tau_n) = \sum_{j=0}^{\infty} \frac{(-1)^j (n+2j)! \sigma_u^{2j}}{n! j! 2^j}$$
$$\int_{-\infty}^{\infty} K_{(n+2j)}(\tau_1, \ldots, \tau_n, \sigma_1, \sigma_1, \ldots, \sigma_j, \sigma_j) d\sigma_1, \ldots, d\sigma_j$$

As each functional, $G_n[K_n, u(t)]$, is orthogonal to all functionals not of the same order, Wiener [113] observed that "instrumental" kernels could be used to isolate an individual kernel. If $y(t)$ is given by Equation (2.13), and Q_m is an m'th order functional, $Q_m[q_m, u(t)]$, then:

$$E[y(t)Q_m(t)] = \sum_{n=0}^{\infty} E[G_n(t)Q_m(t)] \tag{2.14}$$

However, due to the orthogonality of the Wiener functionals:

$$E[G_n(t)Q_m(t)] \equiv 0 \qquad \text{for } m \neq n$$

Hence, the only non-zero term in (2.14) is:

$$E[G_m(t)Q_m(t)] = m! \sigma_u^{2m} \int \ldots \int_0^{\infty} h_m(\tau_1, \ldots, \tau_m) q_m(\tau_1, \ldots, \tau_m) d\tau_1, \ldots d\tau_m$$

Furthermore, if the m'th order system kernel is to be expanded in terms of an orthogonal basis, then m'th order instrumental kernels can be constructed from those basis functions. Applying these to the input, multiplying the result by the observed output, and taking the time average results in the coefficients of the basis functions and, eventually, an estimate of the m'th order kernel.

Wiener [113] suggested the use of Laguerre functions as a basis for the kernels. He showed how these could be generated using simple RC ladder networks, and detailed apparatus for performing both the multiplication and time averaging operations.

Lee and Schetzen [62] used products of delta functions as the instrumental kernels. As such, the coefficient computation was implemented as a series of cross-correlation calculations. The zero order Wiener kernel is equal to the output mean. Once this has been computed, it is subtracted from the output.

$$\begin{aligned} \hat{h}_0 &= \frac{1}{N}\sum_{t=1}^{N} y(t) && (2.15) \\ y_0(t) &= y(t) - \hat{h}_0 && (2.16) \end{aligned}$$

The first-order Wiener kernel is then estimated by computing the cross-correlation between the input and the output residuals.

$$\hat{h}_1(\tau) = \hat{\phi}_{uy_0}(\tau) = \frac{1}{N}\sum_{t=\tau+1}^{N} u(t-\tau)y_0(t) \qquad (2.17)$$

The output due to the first-order kernel can then be evaluated using a discrete convolution, and subtracted from the remaining system output:

$$y_1(t) = y_0(t) - \sum_{\tau=0}^{T-1} \hat{h}_1(\tau)u(t-\tau) \qquad (2.18)$$

The second-order kernel is then estimated from the second-order cross-correlation between the input and the residuals.

$$\hat{h}_2(\tau_1,\tau_2) = \hat{\phi}_{uuy_1}(\tau_1,\tau_2) = \frac{1}{N}\sum_{t=1+\max(\tau_1,\tau_2)}^{N} u(t-\tau_1)u(t-\tau_2)y_1(t) \qquad (2.19)$$

Marmarelis and Marmarelis [68] considered the computation of Wiener kernels via the Lee and Schetzen [62] approach using Gaussian white noise inputs, as well as several families of pseudo-random sequences.

The Lee and Schetzen implementation of the Wiener series has two distinct disadvantages. First, it requires a white noise input which is often impossible to generate. Secondly, the shape of the kernels is a function of the input power level, σ_u^2. French [19] and Korenberg and Hunter [61] considered how to estimate Wiener kernels using non-white inputs. French calculated the Wiener kernels in the frequency domain using the following relations for the zero through second-order Wiener kernels:

$$H_0 = Y(0) \tag{2.20}$$

$$H_1(\omega) = \frac{< Y(\omega)U^*(\omega) >}{< U(\omega)U^*(\omega) >} \tag{2.21}$$

$$H_2(\omega, n) = \frac{< Y(\omega + n)U^*(\omega)U^*(n) >}{2 < U(\omega)U^*(\omega) >< U(n)U^*(n) >} \quad \omega + n \neq 0 \tag{2.22}$$

where $U(\omega)$ is the Fourier transform of $u(t)$, $U^*(\omega)$ is its complex conjugate, and $<>$ represents an ensemble average.

Korenberg and Hunter [61] computed kernels in the time domain and compensated for non-white input spectra by deconvolving the input auto-correlation from the high order input-output cross-correlations. Specifically, the first order kernel was estimated using Equation (2.9). To estimate the second-order kernel, $h_2(i_1, i_2)$, observe that:

$$\phi_{uuy}(j_1, j_2) = 2 \sum_{i_1, i_2 = 0}^{I} h_2(i_1, i_2)\phi_{uu}(j_1 - i_1)\phi_{uu}(j_2 - i_2) \tag{2.23}$$

Define:

$$g(j_1, i_2) = \sum_{i_1 = 0}^{I} h_2(i_1, i_2)\phi_{uu}(j_1 - i_1) \qquad j_1, i_2 = 0, \ldots, I \tag{2.24}$$

We can then rewrite Equation (2.23) as:

$$\phi_{uuy}(j_1, j_2) = 2 \sum_{i_2 = 0}^{I} g(j_1, j_2)\phi_{uu}(j_2 - i_2) \tag{2.25}$$

which can be used to solve for each of the columns of g. Equation (2.24) can then be solved, row by row, to yield an estimate of $h_2(i_1, i_2)$. Similar procedures can be used to correct higher-order Wiener kernels for imperfections in the input spectrum. Notice that the estimation of an n'th order kernel requires each one-dimensional slice of the n'th order cross-correlation, taken parallel to each of its n axes, to be multiplied by the inverse of the Toeplitz structured auto-correlation matrix. This is equivalent to dividing the n'th order cross-spectrum by the n'th power of the input auto-spectrum in the frequency domain.

The time and frequency domain techniques are equivalent [109], and can compensate for minor imperfections in the input spectrum. However, in estimating second and higher-order kernels, both techniques will break down if the input spectrum has any significant "holes". For the first-order kernel, we can see from Equation (2.21), that for any frequencies, ω, where $U(\omega)$ is near zero any noise present in the estimate of $Y(\omega)$ will be amplified by $\frac{1}{|U(\omega)|}$. For the second-order kernel, Equation (2.22), shows that estimation noise will be amplified by $\frac{1}{|U(\omega)|^2}$. Thus, the sensitivity to defects in the input spectrum increases dramatically with the kernel order.

These kernel estimation methods are based on cross-correlations or cross-spectra, of various orders, between the system input and output. Korenberg *et al.* [54, 55, 59] developed a technique in which a series of signals are derived from the measured input and then orthogonalized with respect to each other. A least squares fit with the system output is then used to assign optimal weights to each of these orthogonal basis functions. In its original form [59], the basis functions were created and orthogonalized explicitly, before performing the regression. A subsequent "fast-orthogonal" algorithm [54, 55] functions similarly, but the orthogonal functionals are never computed explicitly, resulting in a large saving of computation time.

Goussard *et al.* [21] proposed a kernel estimation technique based on stochastic approximation. As in Korenberg's orthogonal methods, Goussard's technique attempts to solve the minimum mean square error problem. However, it uses an iterative stochastic approximation technique to obtain the solution, rather than the exact minimization used by Korenberg.

Marmarelis [71] recently proposed a method, closely related to Wiener's [113] original proposal, in which the kernels are expanded using a basis of Laguerre polynomial filters and synthesized using a multiple-input Hermite polynomial nonlinearity. Unlike Wiener's proposal, all of these operations are accomplished digitally. Furthermore, the projections onto the instrumental kernels are accomplished using a least squares error regression, rather than a weighted time average. This approach is implemented by filtering the system input with a bank of Laguerre filters, and using a least squares regression to fit a multiple-input Hermite polynomial between the outputs of the filter bank and the observed output. In addition, because the polynomial coefficients are evaluated in a single regression, rather than with individual time-averages as in Wiener's [113] original method, the white-noise requirement can be relaxed somewhat. This method yields very accurate estimates, from relatively short data records, and requires relatively little computational effort.

The only potential shortcoming of this method lies in the size of the least squares regression. If m basis functions are required and k is the maximum kernel order, the regression matrix will have $\frac{(m+k)!}{m!k!}$ columns [71] and N rows, where N is the number of data points. While this approach works excellently with relatively few basis functions, the size and complexity of the regression increase rapidly with both the number of basis functions and the kernel order. This leads to a relatively large number of model parameters, and hence to relatively poor noise performance.

One approach used to overcome this difficulty involves the identification of principal dynamic modes [72, 73]. This technique identifies so-called principal dynamic modes of a system by performing an eigen-decomposition on a matrix consisting of the first and second-order coefficients of the static nonlinearity. The principal eigenvectors of this matrix correspond to the impulse responses of the principal dynamic modes. The new nonlinearity is then fitted between the outputs of the principal dynamic modes and the observed output. If the system has a small number of these modes, this intermediate step can result in a dramatic reduction in the number of parameters required, and hence an increase in robustness. Note however, that the use of an eigen-decomposition limits this method to extracting the dynamic modes from the first and second-order Wiener kernels.

2.3.2 Block Structured Methods

In the block-structured approach systems are represented by an interconnection of linear dynamic and static nonlinear elements. A common block structure is the LNL or sandwich model illustrated in Figure 2.5. This model consists of a linear dynamic element, $h(\tau)$, whose output, $x(t)$, is transformed by a static nonlinearity, $m(\cdot)$. The output of the nonlinearity, $w(t) = m(x(t))$, is processed by a second linear system, $g(\tau)$. Methods for the identification of these systems were first proposed by Korenberg [48, 49, 50]. Subsequently, several other methods have been proposed [6, 60, 85, 90].

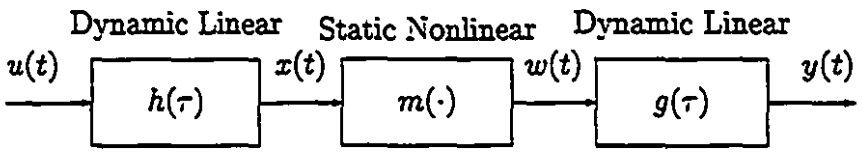


Figure 2.5: Block diagram of an LNL sandwich system

The LNL cascade has two special cases, the Hammerstein system (NL) and the Wiener system (LN), shown in Figures 2.6 and 2.7, respectively. Hunter and Korenberg [31] developed methods for the identification of the LNL cascade, and these two special cases, based on an application of the following theorem, which is originally due to Bussgang [10].

> *Let $u(t)$ and $y(t)$ be two zero-mean Gaussian signals, and let $m(\cdot)$ be a continuous, zero-memory, nonlinear transformation. If $z(t) = m(y(t))$, then:*
>
> $$\phi_{uz}(\tau) = K\phi_{uy}(\tau)$$
>
> *where K is a constant which depends on the variance of $y(t)$ and on the shape of the nonlinearity.*

Hence, the linear element in a Wiener or Hammerstein system can be estimated from the first-order cross-correlation between the system input and output. By Bussgang's theorem, this will be proportional to the cross-correlation measured across the linear element. The impulse response can then be estimated by deconvolving the input auto-correlation function using Equation(2.9).

If the Taylor series for the nonlinearity contains no significant odd terms, then the constant of proportionality in Bussgang's theorem will be zero, and identification based on the first-order input-output cross-correlation will not be possible. However, Korenberg and Hunter [31, 60] demonstrated that any non-zero slice of the second-order input-output cross-correlation will be proportional to the cross-correlation across the linear element. For the second-order cross-correlation to be non-zero, the Taylor series for the nonlinearity must contain at least one significant even term. Thus, if identification based on the first-order correlation fails, a single slice of the second-order correlation can be used instead.

The identification of an LNL cascade model depends on the fact that its Wiener and Volterra kernels are proportional to each other [48, 49, 50, 60], a corollary of Bussgang's theorem [10]. Hence, for a white Gaussian input signal:

$$\begin{aligned} \phi_{uy}(\tau) &= k_1 \int_0^T g(\sigma)h(\tau-\sigma)d\sigma && (2.26) \\ \phi_{uuy}(\tau_1,\tau_2) &= k_2 \int_0^T g(\sigma)h(\tau_1-\sigma)h(\tau_2-\sigma)d\sigma && (2.27) \end{aligned}$$

where k_1 and k_2 are constants of proportionality which depend on the shape of the nonlinearity $m(\cdot)$. If the time integral of $h(\tau)$ is non-zero, i.e. the first linear element is not high-pass, Equation (2.27) alone will be sufficient to identify both linear elements [60], provided that k_2 is non-zero.

2.3.2.1 Hammerstein Systems

Iterative methods have been proposed for the identification of Wiener and Hammerstein systems [31]. The first step in the identification of a Hammerstein system (NL, see Figure 2.6), is to fit a linear system between the output, $y(t)$, and the input, $u(t)$. This results in $\hat{h}^{-1}(\tau)$, an estimate of the *inverse* of the linear element. Convolving this with $y(t)$ produces an initial estimate, $\hat{x}(t)$, of the intermediate signal,

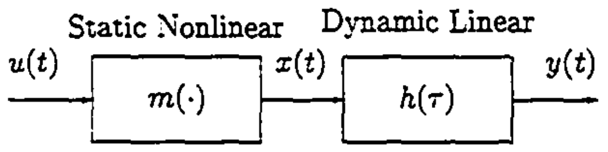


Figure 2.6: Block diagram of a Hammerstein system

$x(t)$. We can then approximate the static nonlinearity, $m(\cdot)$, by fitting a high-order polynomial between the input, and $\hat{x}(t)$. Applying this polynomial to the input, $u(t)$, produces an updated estimate of the intermediate signal, $x(t)$. The inverse of the linear element, $\hat{h}^{-1}(\tau)$, can then be updated by fitting a linear system between the output and the updated estimate of the intermediate signal. This process is repeated until it converges. Note that the inverse filter may be non-causal, and the static nonlinearity never needs to be inverted.

2.3.2.2 Wiener Systems

A similar method was proposed [31] for Wiener systems (LN, see Figure 2.7). A linear filter, $\hat{h}(\tau)$, is estimated between the input, $u(t)$, and the output, $y(t)$, and its output, $\hat{x}(t)$, is generated by convolution with $u(t)$. A static nonlinearity is then fitted between $y(t)$ and $\hat{x}(t)$; this provides an estimate of the *inverse* of the static nonlinearity in the original system. The output, $y(t)$, is then transformed by this inverse estimate, producing an updated estimate, $\hat{x}(t)$, of the intermediate signal, $x(t)$. The linear element is then re-estimated, this time between the input and $\hat{x}(t)$. The iteration continues until it converges.

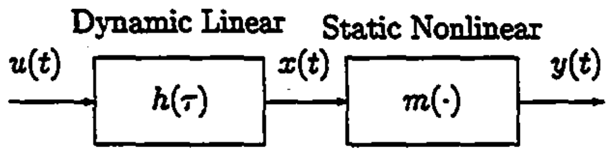


Figure 2.7: Block diagram of a Wiener system

The major difficulty with this approach lies in the estimation of the inverse of the static nonlinearity. If the static nonlinearity is not a one-to-one function over the range probed by the identification experiment, its inverse will not exist since there will be information present in the nonlinearity's input, $x(t)$, that cannot be recovered from its output, $y(t)$.

Wigren [114] considered the estimation of Wiener systems under the assumption that the static nonlinearity was a one-to-one function and exactly known. The output was assumed to contain additive noise. Recursive output-error methods were developed which identified the linear system from the input-output data, and compared with "conventional linearizing inversion", in which the inverse of the static nonlinearity is applied to the output signal, yielding an estimate of the output of the linear element. The signal to noise ratio (SNR) of the measured output was shown to be higher than that of the "linearized" signal. It was suggested that if the noise entered the system before the static nonlinearity, then the linearizing inversion would yield better results than methods which use the input and output measurements directly.

2.3.2.3 LNL Systems

Korenberg and Hunter [60] described an iterative procedure for LNL identification. Using Equation (2.26), the convolution of the two linear elements is estimated from the first-order input-output cross-correlation. A first-order unity-gain filter is constructed with a time constant that best fits this correlation, and is used as an initial estimate for the first linear element, $\hat{h}(\tau)$. Its output, $\hat{x}(t)$, is then generated by convolving $\hat{h}(t)$ with $u(t)$. A Hammerstein system is then estimated between $\hat{x}(t)$ and the output, $y(t)$, using the iterative technique described above. A relaxation technique is then used to modify the estimate of the initial linear system, and the process repeated. As in the Hammerstein case, the nonlinearity is never inverted, thus avoiding the difficulties posed by non-invertible non-linearities.

Correlation-based methods for the identification of Wiener, Hammerstein, and LNL block cascades all assume that the system actually has the appropriate structure. Korenberg [57] developed a least squares method which estimates the best Hammerstein system between a given input and output without assuming any particular structure for the system. It calculates the Hammerstein system that minimizes the mean square error between the system and model outputs. Although this algorithm is limited to white input signals, it requires no assumptions about the amplitude distribution.

2.3.2.4 Estimation of the Nonlinearity

The methods discussed above concentrate on the identification of the dynamic linear elements of block-structured models. Relatively little attention has been given to the static nonlinearity, which is usually modelled as a polynomial function.

However, Greblicki and Pawlak [22, 23, 24, 25, 26, 27, 80, 81] developed methods in which the static nonlinear elements of both Wiener and Hammerstein systems are represented by nonparametric functions. The identification of the linear subsystems was not discussed, but they were assumed to be represented by a set of linear state equations (2.3) that had to be at least asymptotically stable.

Their technique for Wiener systems [23] requires the estimation of the inverse of the static nonlinearity. As a result, the nonlinearity is restricted to the class of strictly monotonic Borel functions having bounded derivatives. The restriction to strictly monotonic nonlinearities is shared by the algorithm described by Hunter and Korenberg [31].

For a Hammerstein system, such as in Figure 2.6, where the linear subsystem is represented by a state space model such as in Equations (2.3), the method proceeds as follows. The output sequence is:

$$y_k = Cx_k + Dm(u_k)$$

where $m(\cdot)$ is the static nonlinearity. If the samples of u_k are independent, then the static nonlinearity output, $m(u)$, will also be white. Furthermore, the current value of the state, x_k, will be independent of the current input, u_k. Hence:

$$\begin{aligned} E[y_k|u_k = u] &= CE[x_k|u_k = u] + DE[m(u_k)|u_k = u] \\ &= CE[x_k] + Dm(u) \\ &= \beta + \alpha m(u) \end{aligned}$$

Therefore, β depends on the expected value of the state, x_k, as well as any constant offset present in the static nonlinearity. If the nonlinear characteristic, $m(\cdot)$, is assumed to be odd and the input distribution is assumed to be symmetric about zero, $\beta = 0$. The value of α can be assumed to be 1, as any change in the scaling of the nonlinearity output will be absorbed in the estimate of the linear element.

Greblicki and Pawlak also considered the case where $m(\cdot)$ was not odd, or the input distribution was not symmetric. They concluded that under those conditions, it was possible to recover only $\alpha m(\cdot) + \beta$, with both α and β unknown. However, if $m(0)$ was known to be zero, they observed that they could take $\hat{m}(u) - \hat{m}(0)$ as an estimate of $\alpha m(\cdot)$. Thus, although a parametric representation of the nonlinearity is never required, it is still severely restricted in form. It must either be odd and have a symmetric input, or $m(0)$ must be 0. Given these restrictions, it is unclear how applicable these techniques, at least in their present form, would be to the study of physical systems.

2.3.3 Parallel Cascades

Palm [77] showed that any finite dimension, finite order, finite memory Volterra system can be represented exactly by a finite sum of LNL models, as illustrated in Figure 2.8. More recently, Korenberg [56] showed that this was true for Wiener cascade elements as well. This was a significant advancement, since the identification algorithms for Wiener models are much simpler than those for LNL cascades (See Section 2.3.2). Thus, practical methods for the identification of parallel cascade models were made possible.

In general, the parallel cascade method [51, 52, 56, 58, 77] consists of first fitting a block-structured, nonlinear system between the input and the output. The output of this first system is computed, and subtracted from the measured output. A second block structured model is then fitted between the input and the output residuals. This process is repeated until the variance of the output residuals is reduced to the point where no additional significant paths can be added to the model. The estimation of the first two paths is illustrated in Figure 2.9.

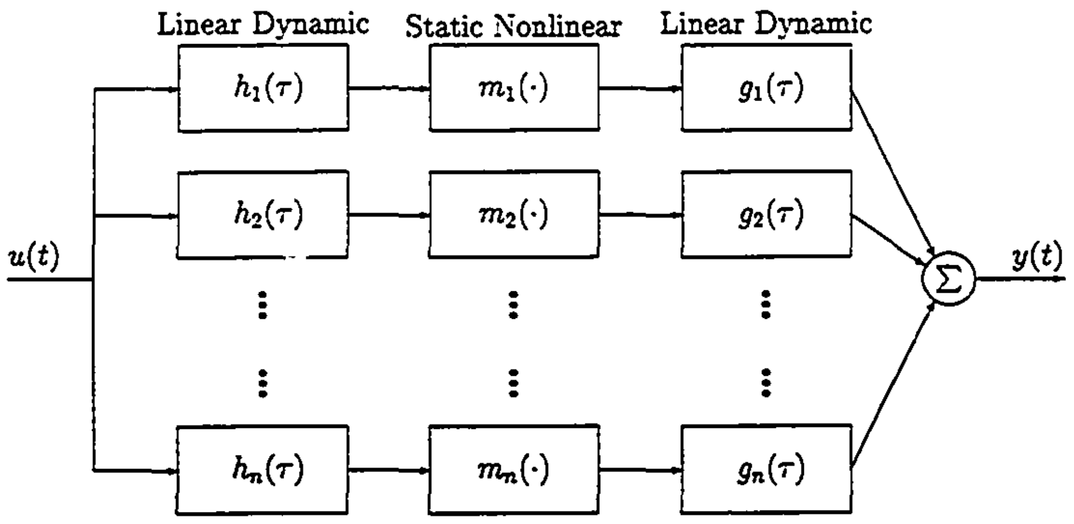


Figure 2.8: A parallel cascade model made up of LNL paths

With reference to Figure 2.9, consider the identification of the j'th path. In this case, a Wiener cascade will be fitted between the input, $u(t)$, and the residuals remaining after the outputs of the first $j-1$ paths have been removed, $v_{j-1}(t)$. The output of this new cascade will be labelled $\hat{y}_j(t)$.

The key to the parallel cascade method's success is the estimation of the linear parts of the cascade paths. Palm [77] was not aware of any method for the identification of LNL cascades. Korenberg [56] made a number of suggestions as to how Wiener or LNL paths might be constructed. The only method developed in detail involved using slices of input/output cross-correlation functions, of various orders, as estimates of the linear subsystems of Wiener cascades. The impulse response of the linear part of the first Wiener cascade was estimated from the first-order input-output cross-correlation. Subsequent paths used single slices, selected at random, of the second-order cross-correlation function between the input and the output residuals. Furthermore, randomly weighted impulses were added to the diagonal elements of the slices. For example, if the j'th path was based on the i'th slice of the second-order correlation, the linear subsystem would have:

$$h_j(\tau) = \frac{1}{N} \sum_{t=max(i,k)}^{T-1} u(t-\tau)u(t-i)v_{j-1}(t) + c_j\delta_{i\tau} \tag{2.28}$$

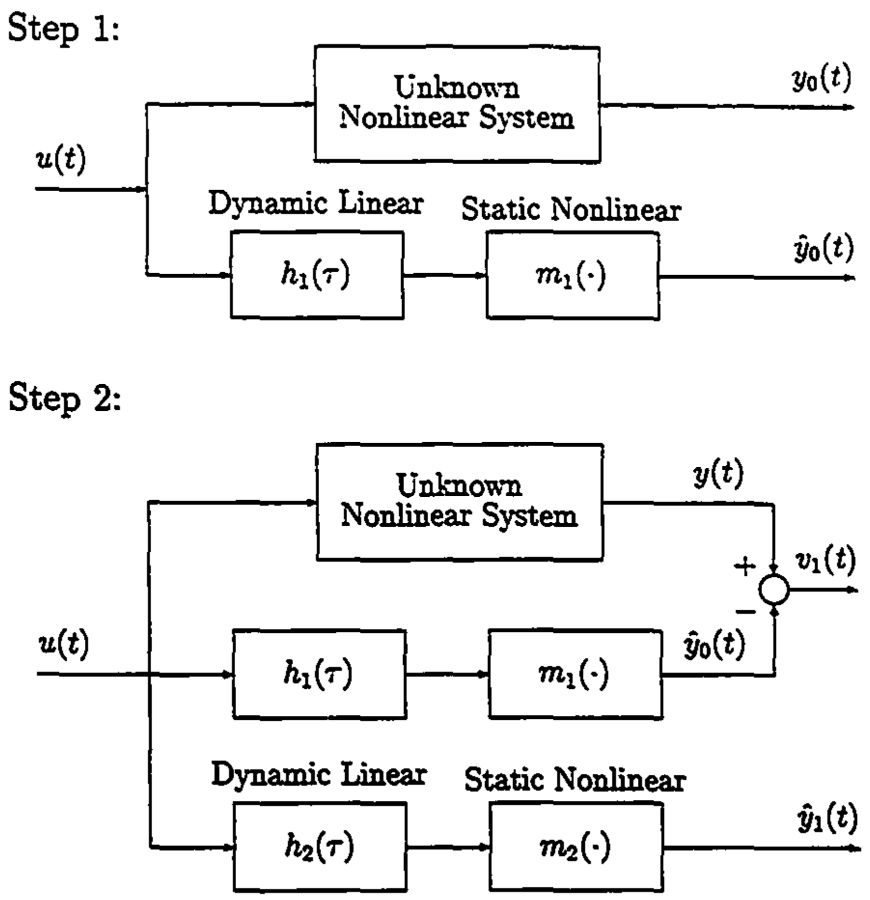


Figure 2.9: The parallel cascade method for nonlinear system identification.

1. **Fit a Wiener cascade between the input and output of the nonlinear system.**
2. **Subtract the output of the first cascade from that of the unknown system, generating the output residuals. Fit a Wiener cascade between the input, and the output residuals.**

as its impulse response, where c_j is a random weight, chosen such that the sequence of weights, c_j, vanishes as j goes to infinity, and $\delta_{i\tau}$ is the Kroneker delta. The vanishing sequence of randomly weighted impulses were required to prove convergence.

Given the random selection of paths, it is likely that many insignificant paths would be selected. Korenberg proposed a correlation based test to determine whether or not a given cascade path models a significant portion of the remaining dynamics. Specifically, he observed that if a cascade path were fitted between two independent, zero-mean Gaussian sequences, then:

$$\frac{\overline{y_{i+1}^2(t)}}{\overline{v_i^2(t)}} < \frac{4}{N - T + 1} \tag{2.29}$$

with a probability of about 0.95. In keeping with our notation, $y_{i+1}(n)$ is the output of the $i+1$'th path, $v_i(n)$ is the i'th residual, N is the length of the data records and T is the memory length of the cascade path. The overbar indicates a time average.

Thus, when a new cascade path is identified, the ratio given on the left hand side of (2.29) is formed. If this ratio exceeds the threshold given on the right, then the new pathway is likely to contain useful information about the system dynamics, and is added to the model. If the ratio is less than the threshold, the pathway is probably modelling noise, and is rejected.

2.3.4 Parametric Methods

Billings and Leontarities [63, 64] proposed a general parametric structure for the analysis of nonlinear systems. This so-called NARMAX structure can be used for the identification of both the stochastic and deterministic components of a system. If we let F be a nonlinear mapping, and $\epsilon(n)$ be the disturbance, or innovations, process, then the model output can be written:

$$y(n) = F[y(n-1), \ldots, y(n-k), u(n), \ldots, u(n-p), \epsilon(n-1), \ldots, \epsilon(n-p)] + \epsilon(n)$$

In general, extended least squares methods [7, 8] are used to identify the parameters of this class of models. Korenberg's [54] fast orthogonal algorithm is well suited to this class of nonlinear difference equations. Recently, neural networks have been used to select the optimal parameters for these models [14].

2.4 Multiple Input Systems

In this section, we will consider the identification of multiple-input single-output (MISO) systems. In one sense, the extension to multiple-input multiple-output (MIMO) systems is trivial, as one MISO system can be identified for each of the outputs. This approach, which was used by Korenberg and Hunter [31, 53] in their treatment of MIMO LNL systems, has the disadvantage that any dynamics that are common to all outputs must be modelled separately in each subsystem.

One of the convenient features of state space models (See Equation 2.3) is that they are readily generalizable to the MIMO case. All that changes are the dimensions of the system matrices A, B, C and D.

2.4.1 Linear Systems

Given that one of the defining properties of a linear system is superposition [5], there is little difference between single-input and multiple-input linear system identification. Furthermore, a multiple-input system is fully characterized by the impulse responses associated with each of its inputs. Provided that the test signals are uncorrelated with each other, performing single input identifications between each input and the output will yield the impulse responses, and give a complete description of the system.

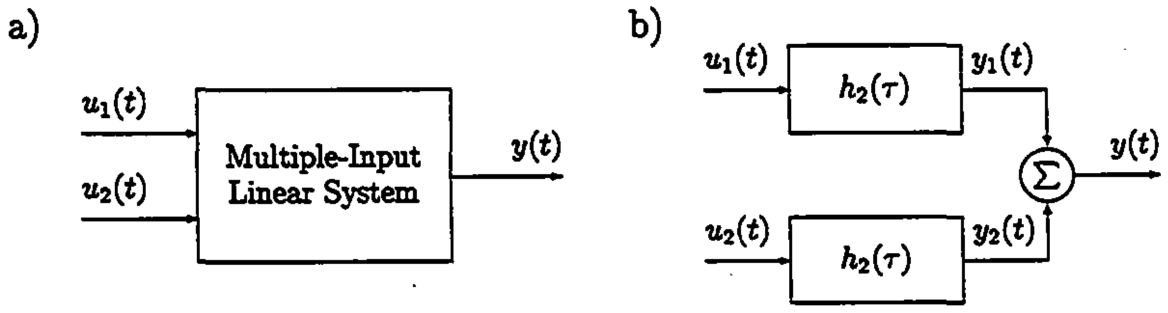


Figure 2.10: Using superposition, a two-input linear system (a) can be decomposed into two single-input linear systems (b)

Consider a two-input system, shown in Figure 2.10, driven by two independent inputs, $u_1(t)$ and $u_2(t)$. Let the output due to $u_1(t)$ acting alone be $y_1(t)$, and the output due to $u_2(t)$ be $y_2(t)$. Then the output observed when the the first input is driven by u_1 and the second input is driven by u_2 is:

$$y(t) = y_1(t) + y_2(t)$$

Given that the inputs are uncorrelated, y_1 and y_2 will also be uncorrelated. Hence, if one were to attempt to fit a linear system between u_1 and y, the signal y_2 would appear to be output noise. Similarly, y_1 would act like output noise in an attempted identification between u_2 and y. As a result, it is often better to attempt to identify the whole system in one operation. This is the approach used by MIMO state-space methods such as the MOESP schemes [96, 98, 99].

2.4.2 Quasi-Linear and Time-Varying Systems

A classical method for analyzing nonlinear systems is to linearize them over a narrow range around an operating point. For a single-input system, the operating point is defined in terms of the statistics of the input signal, usually its mean. For a multiple-input system, the operating point may depend on the values of the other inputs. In identifying such a system, all of the parameters that define the operating point are fixed. A linear identification is then performed between the remaining input and the output. The operating point is changed, and the experiment repeated. Once a family of linear descriptions has been identified, a regression may be performed between the parameters of the operating point and some features of the linear responses.

This approach was used by Weiss *et. al.* [104, 105, 106, 107, 108], in the study of the dynamics of the human ankle. A second-order linear model was shown to be adequate to explain the dynamics of the human ankle, but the parameters of the model depended on the operating point. In analyzing the results, Weiss *et. al.* used the IBK form of the second-order linear model, hence:

$$T(t) = I\ddot{\Theta}(t) + B\dot{\Theta}(t) + K\Theta(t)$$

where T is the torque, Θ is the angular position, and I,B and K are the elastic, viscous and inertial parameters of the model. The inertial parameter, I,was shown to be constant over the full range of motion, and at all levels of background contraction. The spring constant, K, was shown to vary linearly with the mean background contraction, although the slope was higher for increasing contraction in the plantarflexing muscles than in the dorsiflexing muscles. The viscous parameter, B, varied in such a way as to keep the damping of the system relatively constant, despite large variations in the background contraction level.

While this quasi-linear approach can be used to produce models of the system under a wide variety of conditions, it is limited to describing the system at a particular operating point. It provides no information about how the system behaves while the operating point changes.

If the operating point follows a particular time trajectory, the system can be linearized about that trajectory, and represented as a time-varying system [65]. Such a system can be described using a time-varying convolution [67].

$$y(t) = \sum_{j=0}^{T-1} h(t,j)x(t-j) \tag{2.30}$$

where $h(t,j)$ describes the time-varying linear system. In this formulation, the function $h(t,j)$, evaluated at time t, computes the current output, $y(t)$, from previous $T-1$ samples of the input. Hence this description corresponds to a time-varying weighting function.

There is an equally valid formulation where the system is described using a time-varying impulse response:

$$y(t) = \sum_{j=t-T}^{t} h(t,j)x(j) \tag{2.31}$$

In this case, $h(t,j)$ indicates the value of the impulse response that is associated with the input that occurred at time j. The two descriptions are equivalent. MacNeil *et. al.* [67] describe the transformation between the two formulations.

Note that this description depends on the particular time course, or trajectory, followed by the operating point. No information is provided about the system during trajectories other than that used to create the model. In order to achieve that goal, a more general multiple-input nonlinear system description is needed.

Recently, this time-varying technique was used to study changes in the linearized dynamics of the human ankle during step changes in the background contraction level [67], as well as during an electrically evoked muscle twitch [92]. They have also been used to examine changes in the dynamics of the stretch reflex EMG during rapid isometric contractions [45] and rapid imposed movements [44]. In all of these experiments, the quasi-linear second-order models proposed by Weiss *et. al.* [104, 105, 106, 107, 108] were shown to apply before and well after the change in operating point. Had there been no dynamics associated with the operating point, the time-varying model would have shifted through a series of second-order models whose parameters varied as they did in the quasi-static experiments. However, in all cases, the second-order models broke down during the rapid shift in the operating point. This suggests that there is a dynamic, nonlinear relationship between the parameters that define the operating point in each of these experiments, and the position perturbation. Hence, these experiments have taken a dynamic, multiple-input, nonlinear system, and allowed it to be linearized about a particular trajectory. The time-varying, quasi-linear analysis yields a description of the system, but only for points on the trajectory imposed by the experiment. To fully explore the multiple-input nonlinear relationship, a more general model, and tools for its identification, are needed.

2.4.3 Functional Expansions

The Volterra series representation of a nonlinear system has been extended to cover multiple-input systems [69]. Two types of kernels are required: self-kernels, each driven by a single input, and cross-kernels, which have multiple inputs. Thus, for a two-input system terms of the form:

$$\int\!\!\int_{\tau_1,\tau_2=0}^{T} h_{2,u_1,u_2}(\tau_1,\tau_2)u_1(t-\tau_1)u_2(t-\tau_2)d\tau_1 d\tau_2 \tag{2.32}$$

are added to the Volterra series output given in Equation (2.10). Here, $h_{2,u_1,u_2}(\tau_1,\tau_2)$ is a second-order cross-kernel which is first-order in each of two inputs, u_1 and u_2.

Similar expressions can be written for higher-order cross-kernels, and for cross-kernels that involve more than two inputs. The lowest order cross-kernel that can be associated with k inputs is first-order in all k inputs, or k'th order overall.

As in the single-input case, the terms in the multiple-input Volterra series are not orthogonal. As a result, methods for the identification of multiple-input functional expansions must be based on the Wiener series, and to date have required either white, or nearly white, Gaussian inputs. These methods have been used primarily in vision research [69, 115, 116], where the generation of such inputs does not pose any special problems.

A particular focus in this review has been towards methods suited for the study of human joint dynamics. As stated previously, one of the key requirements is the use of significantly non-white test signals. Hence, the functional-expansion based methods, which require white inputs, are likely to be of little use.

2.4.4 Block Structures

Multiple input block structures seem to hold more promise. Korenberg [53] briefly considered the extension of single input LNL cascades to a MIMO case, and showed that single-input identification methods could be used to identify a multiple-input LNL cascade. The only extension required was the estimation of a multi-dimensional nonlinearity.

Chen *et. al.* [12, 13] studied several multiple-input block structures, and derived characteristic relationships between their low order self- and cross-kernels. These relationships provide necessary, but not sufficient, conditions. If a system has a particular structure, the kernels must obey these relationships. However, kernels which obey these relationships do not necessarily arise from systems with that structure.

Thus, if anatomical considerations were to rule out all but a small number of possible structures, kernel tests could be used to determine the most appropriate structure, and the method specialized to that structure used to identify it. Chen [12] developed several techniques, each specialized to a particular structure, which could be used to compute the IRFs of the linear elements, given estimates of the first-

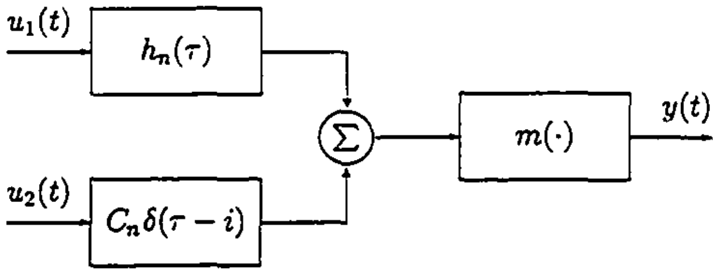


Figure 2.11: Multiple-input Wiener cascade path proposed by Korenberg

through third-order Volterra kernels. The accuracy of these techniques, however, is limited by the accuracy of the initial kernel estimates. Unfortunately, there are, as yet, few methods specialized to particular multiple-input block structures, which estimate their dynamics directly from the input-output data.

2.4.5 Parallel Cascades

Korenberg [56] considered the extension of the parallel cascade method to the multiple-input case. The dynamics of the self-kernels could be modelled using single-input cascade paths, as in the single-input version of the algorithm, described in Section 2.3.3. To capture the dynamics of the cross-kernels, Korenberg proposed incorporating paths where:

$$h_n(\tau) = \frac{1}{N} \sum_{t=max(i,\tau)}^{T-1} u_1(t-\tau)u_2(t-i)y(t) \tag{2.33}$$

and where the output is computed as:

$$w_n(t) = \sum_{\tau=0}^{T-1} h(\tau)u_1(t-\tau) + C_n u_2(t-i)$$

where C_n is a randomly chosen weight. A block diagram of a single path is shown in Figure 2.11. In principle, a parallel cascade assembled from elements such as these is capable of representing any system which has a multiple-input, finite Volterra series expansion. However, convergence is likely to be slow, as the slices used in Equation (2.33) are selected randomly. Furthermore, the only nonlinear interaction between inputs is provided by the single-input nonlinearity acting on the sum of the output of the linear filter (the top pathway in Figure 2.11) and the delayed and (randomly) scaled version of the other input (the lower pathway).

2.5 Summary

In this review, we considered methods that might be used to construct mathematical models of the human peripheral neuromuscular system. We considered the strengths and weaknesses of both morphological modelling and system identification. We concluded that system identification techniques should be used to create a black-box model, which could then be used to validate any morphological models that may be postulated.

Given the structure and complexity of the peripheral neuromuscular system, methods must be capable of identifying multiple-input nonlinear systems. Due to the constraints imposed by the experimental apparatus, these methods must be resistant to the effects of noise, and should place minimal restrictions on the test input. The balance of the review considered various system identification techniques, with reference to these requirements. Our findings may be summarized:

- The parallel cascade [56] and the Laguerre expansion [71] methods both hold promise for the identification of single-input nonlinear systems, especially when there is no *a priori* structural information available. If the system is known to have a simple block structure, such as a Wiener, Hammerstein or LNL, a method specialized to that structure should be used.

- When a block-structured method is used, the only dynamic elements that must be estimated are linear systems. Hence, correction for the input spectrum involves only a single division in the frequency domain (Equation 2.21), or solution of Toeplitz matrix equation in the time domain (Equation 2.9). As a result, block-structured methods, and by extension, the parallel cascade method, should be relatively insensitive to defects in the input spectrum.

- No suitable method for the identification of multiple-input nonlinear systems is evident. Given the relative insensitivity of block-structured methods to defects in the input spectrum, the extension of either specific block-structured methods or the parallel cascade method to the multiple-input case would seem to hold promise. Given its modest computational requirements and relative insensitivity to the input spectrum, an extension of the Laguerre expansion method to the multiple-input case may also be applicable.

In the subsequent three chapters, we will explore some of these possibilities.

Chapter 3

Nonparametric Identification of Linear Systems

In this chapter, we will analyze the performance of the nonparametric FIR filter estimate provided by the inversion of the Toeplitz matrix in Equation (2.9). Aside from its use in the identification of linear systems, this algorithm forms the basis for the ensemble method used to identify time-varying systems [44, 45, 67, 92], as well as the block-structured [31, 60] and parallel cascade [56] methods. Furthermore, it will play an important role in several new algorithms developed in this thesis.

3.1 Preliminaries

Throughout this and the subsequent two chapters we will make extensive use of the following result, which can be found, for example, in Bendat and Piersol [5]. Given n zero-mean, jointly Gaussian random variables x_1 through x_n, the product:

$$E[x_1 \cdot x_2 \cdot \ldots \cdot x_n]$$

is zero for odd values of n, and for even n, the expected value is equal to the sum, over all possible permutations of the product of the expected values of products of pairs of random variables. For example, when n is 4:

$$E[x_1x_2x_3x_4] = E[x_1x_2]E[x_3x_4] + E[x_1x_3]E[x_2x_4] + E[x_1x_4]E[x_2x_3]$$

and, in the special case where all the x_n are identically distributed:

$$\begin{aligned} E[x_1 \cdot x_2 \cdot \ldots \cdot x_n] &= (1 \cdot 3 \cdot 5 \cdot \ldots \cdot n-1) E[x_i x_j]^{\frac{n}{2}} \\ &= \frac{(n-1)!}{2^{n/2}(n/2)!} E[x_i x_j]^{\frac{n}{2}} \qquad i \neq j \end{aligned} \tag{3.1}$$

The other result which we will need concerns the estimation of correlation functions, and can also be found in Bendat and Piersol [5]. If we use Equation (2.6) to estimate the cross-correlation between two signals, $u(t)$ and $y(t)$, the variance of that estimate is:

$$\begin{aligned} \mathrm{Var}\left[\hat{\phi}_{uy}(\tau)\right] &= \frac{1}{N}\int_{-N}^{N}\left(1-\frac{|\xi|}{N}\right)\Big(\phi_{uu}(\xi)\phi_{yy}(\xi) + \\ &\qquad \phi_{uy}(\xi+\tau)\phi_{yu}(\xi-\tau)\Big)\, d\xi \end{aligned} \tag{3.2}$$

In general, the length of the data records, N, will be much greater than the support of the correlation functions, hence:

$$\mathrm{Var}\left[\hat{\phi}_{uy}(\tau)\right] \approx \frac{1}{N}\int_{-N}^{N}\Big(\phi_{uu}(\xi)\phi_{yy}(\xi) + \phi_{uy}(\xi+\tau)\phi_{yu}(\xi-\tau)\Big)\, d\xi \tag{3.3}$$

Similarly, the variance of the auto-correlation estimate may be approximated by:

$$\mathrm{Var}\left[\hat{\phi}_{uu}(\tau)\right] \approx \frac{1}{N}\int_{-N}^{N}\Big(\phi_{uu}^2(\xi) + \phi_{uu}(\xi+\tau)\phi_{uu}(\xi-\tau)\Big)\, d\xi \tag{3.4}$$

Throughout the rest of this thesis, we will need to evaluate the accuracy of a model, or perhaps that of its output. Let $\hat{y}$ be an estimate of y, and define the "percent variance accounted for" (%VAF) by this estimate as:

$$\%\ \mathrm{VAF} = 100 \times \frac{\mathrm{var}(y-\hat{y})}{\mathrm{var}(y)} \tag{3.5}$$

where $\mathrm{var}(x)$ is the variance of the random variable x. In practice these quantities will be estimated from time averages.

3.2 Linear System Identification

Assume that $u(t)$ and $y(t)$ are the input and output of a linear system, which will be represented by its impulse response, $h(\tau)$. Thus:

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

Given the input-output cross-correlation,

$$\phi_{uy}(\tau) = E[u(t-\tau)y(t)]$$

and a Toeplitz structured matrix generated by the input auto-correlation,

$$\Phi_{uu}(i,j) = E[u(t-i)u(t-j)]$$

Equation (2.9) can be used to compute the impulse response of the linear system.

$$h = \Phi_{uu}^{-1}\phi_{uy}$$

However, since we do not have access to the actual correlation functions, we must use estimates obtained from finite segments of data. Furthermore, instead of the output signal, $y(t)$, we have access to a noise corrupted measurement,

$$z(t) = y(t) + v_z(t)$$

where $v_z(t)$ is assumed to be a zero-mean sequence which is independent of the input, $u(t)$. We will write our correlation estimates as the sum of the actual correlation functions and the associated estimation errors. Thus:

$$\begin{aligned}
\hat{\phi}_{uu}(\tau) &= \phi_{uu}(\tau) + \tilde{\phi}_{uu}(\tau) \\
\hat{\phi}_{uy}(\tau) &= \phi_{uy}(\tau) + \tilde{\phi}_{uy}(\tau) \\
\hat{\phi}_{uz}(\tau) &= \phi_{uy}(\tau) + \tilde{\phi}_{uy}(\tau) + \tilde{\phi}_{uv_z}(\tau)
\end{aligned}$$

where $\hat{\phi}$ is an estimate of ϕ, and $\tilde{\phi}$ is the estimation error.

We use a first-order perturbation expansion [88] for the matrix inversion in Equation (2.9), to estimate the effects of these estimation errors on the impulse response estimate. Hence:

$$\hat{h} \cong \left(\Phi_{uu}^{-1} - \Phi_{uu}^{-1}\tilde{\Phi}_{uu}\Phi_{uu}^{-1}\right)\left(\phi_{uy} + \tilde{\phi}_{uy} + \tilde{\phi}_{uv_z}\right) \tag{3.6}$$

Expanding:

$$\begin{aligned}\hat{h} \;&\cong\; h + \Phi_{uu}^{-1}\tilde{\phi}_{uy} - \Phi_{uu}^{-1}\tilde{\Phi}_{uu}h + \Phi_{uu}^{-1}\tilde{\phi}_{uv_z} \\ &\qquad\qquad - \Phi_{uu}^{-1}\tilde{\Phi}_{uu}\Phi_{uu}^{-1}\tilde{\phi}_{uy} - \Phi_{uu}^{-1}\tilde{\Phi}_{uu}\Phi_{uu}^{-1}\tilde{\phi}_{uv_z}\end{aligned} \tag{3.7}$$

In keeping with the matrix perturbation expansion used to write Equation (3.6), we will restrict ourselves to a first-order analysis. Hence, we can discard the last two terms in (3.7). Note the following:

$$\begin{aligned}\tilde{\Phi}_{uu}h &= \sum_{i=0}^{T-1}\tilde{\Phi}_{uu}(\tau,i)h(i) \\ &= \sum_{i=0}^{T-1}\tilde{\phi}_{uu}(\tau-i)h(i) \\ &= \sum_{i=0}^{T-1}\left(\phi_{uu}(\tau-i) - \frac{1}{N}\sum_{t=1}^{N}u(t-\tau)u(t-i)\right)h(i) \\ &= \phi_{uy} - \frac{1}{N}\sum_{t=1}^{N}u(t-\tau)\sum_{i=0}^{T-1}h(i)u(t-i)\end{aligned}$$

As the record length is assumed to be much greater than the system memory, we may ignore initial conditions, $\sum_{i=0}^{T-1}h(i)u(t-i) = y(t)$. Hence:

$$\begin{aligned}\tilde{\Phi}_{uu}h &= \phi_{uy} - \frac{1}{N}\sum_{t=1}^{N}u(t-\tau)y(t) \\ &= \phi_{uy} - \hat{\phi}_{uy} \\ &= \tilde{\phi}_{uy}\end{aligned}$$

the second and third terms on the right of (3.7) cancel, and it reduces to:

$$\hat{h} \cong h + \Phi_{uu}^{-1}\tilde{\phi}_{uv_z} \tag{3.8}$$

Therefore, when Equation (2.9) is used to estimate the impulse response of a linear system, the only terms of first-order magnitude in the estimation error are due to the output noise. To a first-order approximation, under noise-free conditions this algorithm completely corrects for any statistical fluctuations in the cross-correlation estimates.

Next, let us calculate the statistics of this one remaining first-order term. First consider its expected value:

$$E[\tilde{h}(\tau)] = \Phi_{uu}^{-1}E[\tilde{\phi}_{uv_z}] = 0 \tag{3.9}$$

The error variance is:

$$\begin{aligned}
\mathrm{Var}(\tilde{h}(\tau)) &= E\left[\left(\sum_{i=1}^{T}\Phi_{uu}^{-1}(\tau,i)\tilde{\phi}_{uv_z}(i)\right)^2\right] && (3.10)\\
&= \sum_{ij=1}^{T}\Phi_{uu}^{-1}(\tau,i)\Phi_{uu}^{-1}(\tau,j)E\left[\tilde{\phi}_{uv_z}(i)\tilde{\phi}_{uv_z}(j)\right] && (3.11)
\end{aligned}$$

To proceed further, we must evaluate the expectation operation on on the right hand side of (3.11). Hence:

$$\begin{aligned}
E\left[\tilde{\phi}_{uv_z}(i)\tilde{\phi}_{uv_z}(j)\right] &= \frac{1}{N^2}\sum_{t_1t_2=1}^{N}E\left[u(t_1-i)v_z(t_1)u(t_2-j)v_z(t_2)\right]\\
&= \frac{1}{N^2}\sum_{t_1t_2=1}^{N}\Big\{\phi_{uv_z}(i)\phi_{uv_z}(j)+\\
&\qquad \phi_{uu}(t_1-t_2-i+j)\phi_{v_zv_z}(t_1-t_2)+\\
&\qquad\quad \phi_{uv_z}(t_1-t_2-i)\phi_{uv_z}(t_2-t_1-j)\Big\}\\
&= \frac{1}{N^2}\sum_{t_1t_2=1}^{N}\phi_{uu}(t_1-t_2-i+j)\phi_{v_zv_z}(t_1-t_2)
\end{aligned}$$

Making a change of variables, such that $t = t_1 - t_2$, this becomes:

$$E\left[\tilde{\phi}_{uv_z}(i)\tilde{\phi}_{uv_z}(j)\right] = \frac{1}{N^2}\sum_{t=-T}^{T}(N - |t|)\phi_{v_z v_z}(t)\phi_{uu}(j-i-t) \tag{3.12}$$

In all applications, the number of data points, N, will be much greater than the number of lags in the estimated filter, T. Thus, $N - |t| \approx N$, and (3.12) becomes:

$$E\left[\tilde{\phi}_{uv_z}(i)\tilde{\phi}_{uv_z}(j)\right] \approx \Xi_{uv_z}(i,j) \doteq \frac{1}{N}\sum_{t=-T}^{T}\phi_{v_z v_z}(t)\phi_{uu}(j-i-t) \tag{3.13}$$

The covariance matrix of the input-noise cross-correlation estimate, $\Xi_{uv_z}(i,j)$, is a Toeplitz matrix. Its entries can be computed by filtering the input auto-correlation function with the auto-correlation of the measurement noise, provided both functions are represented as two-sided filters.

Substituting (3.13) into (3.11), we see that the variance of the estimation error is:

$$\text{Var}(\tilde{h}(\tau)) = \sum_{ij=1}^{T}\Phi_{uu}^{-1}(\tau,i)\Xi_{uv_z}(i,j)\Phi_{uu}^{-1}(j,\tau) \tag{3.14}$$

Writing this more compactly, we see that the error variance is equal to the diagonal of the matrix:

$$\Phi_{uu}^{-1}\Xi_{uv_z}\Phi_{uu}^{-1}$$

Whether or not the noise is white, we can see that the the estimation variance depends only on the statistics of the input and measurement noise, and not on the value of the impulse response. If the measurement noise is white, then Ξ_{uv} will be proportional to Φ_{uu}, and the error variance becomes:

$$\text{Var}(\tilde{h}(\tau)) = \frac{\sigma_{v_z}^2}{N}\Phi_{uu}^{-1}(\tau,\tau) \tag{3.15}$$

where $\sigma_{v_z}^2$ is the variance of the noise sequence v_z. Finally, if the input is also white, this reduces to:

$$\text{Var}(\tilde{h}(\tau)) = \frac{\sigma_{v_z}^2}{N\sigma_u^2}$$

3.2.1 Use of a Pseudoinverse

Rewriting Equation (3.8), we can see that, to a first-order approximation, the impulse response estimate is:

$$\hat{h} = \hat{\Phi}_{uu}^{-1}\hat{\Phi}_{uu}h + \Phi_{uu}^{-1}\tilde{\phi}_{uv_z}$$

The first term is the contribution from the system, and will be exactly h, provided the auto-correlation matrix, $\hat{\Phi}_{uu}$, is nonsingular.

Consider the singular value decomposition (SVD) [20] of the input auto-correlation matrix, $\hat{\Phi}_{uu}$:

$$\hat{\Phi}_{uu} = USV^T \tag{3.16}$$

where U and V are unitary matrices, and S is diagonal with positive real entries $s_1 \geq s_2 \geq \ldots \geq s_n > 0$. To proceed further, we must show that $\hat{\Phi}_{uu}$ is a positive definite matrix, in which case U will be equal to V. Thus, we need the following lemma:

Lemma 1 *Let h be the impulse response of a linear FIR filter, and u be a finite segment, of length N, of a zero-mean stochastic process. Let $\hat{\phi}_{uu}(\tau)$ be a biased estimate of the autocorrelation of u, generated using Equation (2.6), and $\hat{\Phi}_{uu}(i,j)$ be a Toeplitz matrix, generated from $\hat{\phi}_{uu}(\tau)$. If y is the output of the filter h, when driven by u and starting from zero initial conditions, then the mean square value of y is given by:*

$$E[y^2] = h^T\hat{\Phi}_{uu}h$$

Proof

$$\begin{aligned}
E[y^2] &= \frac{1}{N}\left(\sum_{n=1}^{N}\sum_{i=0}^{T-1} h(i)u(n-i)\right)^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{i,j=0}^{T-1} h(i)h(j)u(n-i)u(n-i) \\
&= \sum_{i,j=0}^{T-1} h(i)h(j)\left(\frac{1}{N}\sum_{n=1}^{N} u(n-i)u(n-j)\right) \\
&= h^T\hat{\Phi}_{uu}h
\end{aligned}$$

□

As a direct result of Lemma 1, $\hat{\Phi}_{uu}$ must be positive definite, $U \equiv V$, and the auto-correlation inverse becomes:

$$\hat{\Phi}_{uu}^{-1} = VS^{-1}V^T \tag{3.17}$$

where $S^{-1} = \operatorname{diag}(1/s_1, \ldots, 1/s_n)$. Now, let $\zeta = V^T h$ be the projection of h onto the columns of the unitary matrix V. Furthermore, let us define the projection of the noise correlation as: $\nu = V^T \hat{\phi}_{uv_z}$. Substitute these projections, along with Equation (3.17) into (2.9):

$$\begin{aligned} \hat{h} &= VS^{-1}V^T(VS\zeta + V\nu) \\ &= V\zeta + VS^{-1}\nu \\ &= \sum_{i=1}^{T}(\zeta_i + \frac{\nu_i}{s_i})v_i \end{aligned}$$

where $V = [v_1; v_2; \ldots v_T]$, and ζ_i and ν_i are the i'th entries of ζ and ν respectively. Clearly, terms for which $\frac{|\nu_i|}{s_i} > |\zeta_i|$ will add more "noise" than "signal" to the impulse response estimate $\hat{h}$. Eliminating terms which are dominated by noise should lead to an improvement in the estimate of the impulse response. In this way, we make use of a *pseudoinverse* of $\hat{\Phi}_{uu}$ [20]. Let us partition the SVD of $\hat{\Phi}_{uu}$ as follows:

$$\hat{\Phi}_{uu} = \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \tag{3.18}$$

where the subscript "1" refers to the terms which will be retained, and "2" denotes terms which will be dropped. The corresponding pseudoinverse is then:

$$\hat{\Phi}_{uu}^{\dagger} = V_1 S_1^{-1} V_1^T$$

In constructing the pseudoinverse, we must decide which terms to retain, and which to drop. To aid in this decision, let us examine the first two moments of the noise coefficients, ν_i. From Equations (3.12) through (3.14), replacing Φ_{uu} with its SVD, given in (3.16), we see that ν_i is zero in expectation, and its variance is:

$$\sigma_{\nu_i}^2 = E[\nu_i^2] = v_i^T \Xi_{uv} v_i \tag{3.19}$$

where Ξ_{uv} is defined in (3.13). Let us define a threshold, which depends on i:

$$Th_i = \frac{2\sigma_{\nu_i}}{s_i} \tag{3.20}$$

If $|\hat{\zeta}_i|$ exceeds this threshold, we can be 90% certain that it is not noise, because the coefficients are normally distributed, with 0 mean and variance given by Equation (3.19). Of course, constants other than 2 can be used, resulting in different confidence levels. V_1 in Equation (3.18) will include all terms for which the coefficients, $\zeta_i + \nu_i/s_i$ exceed the threshold Th_i. Using this pseudoinverse, the IRF estimate becomes:

$$\hat{h} = \hat{\Phi}^{\ddagger}_{uu}\hat{\phi}_{uz} \tag{3.21}$$

and the variance of the estimation error will be the diagonal of:

$$\Phi^{\ddagger}_{uu}\Xi_{uv}\Phi^{\ddagger}_{uu} \tag{3.22}$$

Using the pseudoinverse should result in better conditioned estimates of the impulse response (3.21) and the estimation variance (3.22). In many cases, it may be possible to derive a meaningful estimate of the estimation variance due to the output noise by simply applying Equations (3.22) and (3.13) to the input signal and the residuals. This improvement will be at the cost of introducing a bias into the estimate. The expected value of the IRF estimate becomes:

$$E[\hat{h}] = \hat{\Phi}^{\ddagger}_{uu}\hat{\Phi}_{uu}h \tag{3.23}$$

Furthermore, estimation of the induced bias is not straightforward. An obvious procedure would be to compare $\hat{h}$ and $\hat{\Phi}^{\ddagger}_{uu}\hat{\Phi}_{uu}\hat{h}$. This will not work, however, because $\hat{\Phi}^{\ddagger}_{uu}\hat{\Phi}_{uu}$ is a projection operation. Thus:

$$\hat{\Phi}^{\ddagger}_{uu}\hat{\Phi}_{uu} \equiv (\hat{\Phi}^{\ddagger}_{uu}\hat{\Phi}_{uu})^2$$

As a result, $\hat{h}$ and $\hat{\Phi}^{\ddagger}_{uu}\hat{\Phi}_{uu}\hat{h}$ will be identical, to within machine accuracy.

While we cannot estimate the point by point magnitude of the bias, we can estimate its norm over the length of the IRF. From Equation (2.21), we see that the deconvolution operation is equivalent to dividing the input-output cross-spectrum by the input auto-spectrum. If it is assumed that the noise in the correlation estimate is approximately white, then the noise in the IRF estimate will be concentrated at frequencies where there is little input power. Thus, there will be little overlap between the spectra of the input signal and that of the noise in the IRF estimate, and this estimation noise should have little effect on the output prediction. Any change in the predicted output can therefore be attributed to bias introduced into the IRF estimate.

Let $\hat{h}_0$ and $\hat{h}_k$ be the IRFs estimated using the full inverse, and the k'th order pseudoinverse, respectively, and let the variances of their outputs be $\sigma^2_{\hat{y}_0}$ and $\sigma^2_{\hat{y}_k}$. If we assume that any change in the output variance is the result of bias errors introduced into the IRF estimate, we can approximate the 2-norm of the bias as:

$$\| \Delta h \|_2^2 = \| \hat{h} \|_2^2 \frac{\sigma^2_{\hat{y}_k}}{\sigma^2_{\hat{y}_0}} \tag{3.24}$$

where Δh is the bias error. The 2-norm is an upper bound on the infinity norm [20], which is the absolute value of the largest element. Hence, (3.24) is an estimate of an upper bound on the largest bias error in the IRF estimate.

3.2.1.1 Algorithmic Summary

The following steps are required to estimate a linear impulse response between an input u and a possibly noise corrupted output, z. The variance of the first order noise term and the norm of the bias introduced by the use of the pseudoinverse are also estimated.

1. Use Equation (2.6) to estimate the input auto-correlation, $\hat{\phi}_{uu}$, and the input output cross-correlation $\hat{\phi}_{uz}$.

2. Create a Toeplitz matrix from the estimated auto-correlation:
   $\hat{\Phi}_{uu}(i,j) = \hat{\phi}_{uu}(| i - j |)$.

3. Form an initial estimate of the impulse response, $\hat{h}_0$ using Equation (2.9)

4. Compute the following SVD: $\hat{\Phi}_{uu} = USV^T$

5. Project $\hat{h}_0$ onto the columns of the unitary matrix V.

6. Use Equations (3.13), (3.19) and (3.20) to compute decision levels Th_i, using the output residuals from $\hat{h}_0$ as an estimate of the noise process v_z.

7. Sum the terms whose coefficients exceed the threshold function, to create an enhanced estimate of the IRF, $\hat{h}_e$.

8. Generate a new set of output residuals, and use these, together with the pseudoinverse to compute the estimation variance, using Equations (3.13 and 3.22).

9. Use Equation (3.24) to compute the 2-norm of the bias error.

3.2.2 Validity of the First-Order Approximation

The magnitude of the second-order terms will depend on the square of the record length, N, in general. For the rest of this discussion, we will restrict ourselves to the two second-order terms dropped from Equation (3.7). The first of these is:

$$\Phi_{uu}^{-1}\tilde{\Phi}_{uu}\Phi_{uu}^{-1}\tilde{\phi}_{uy} = \Phi_{uu}^{-1}\tilde{\Phi}_{uu}\Phi_{uu}^{-1}\tilde{\Phi}_{uu}h \tag{3.25}$$

which depends on the square of the error in the estimate of the input auto-correlation. Thus, its variance will depend on the square of the record length (see Equation 3.4).

Since this term also depends on the square of Φ_{uu}^{-1}, it may become significant if this matrix is particularly ill-conditioned, which will be the case if the input is highly coloured. Use of the pseudoinverse deconvolution, however, effectively improves the condition of the auto-correlation matrix. Therefore, this term should remain insignificant if the pseudoinverse is used in the deconvolution.

The second term which was dropped from (3.7) is:

$$\Phi_{uu}^{-1}\tilde{\Phi}_{uu}\Phi_{uu}^{-1}\tilde{\phi}_{uv_z}$$

This term is not likely to be significant, as it depends on the magnitude of the output noise. The first-order noise term will dominate in all cases.

In cases where the record lengths are short, and the signal to noise ratio is high, the term in (3.25) could become significant, with respect to the first-order noise term.

3.2.3 Simulations Involving Linear Systems

We performed several Monte Carlo simulations, to validate the theoretical results obtained in the previous section. In these simulations we wished to address three points.

1. In the noise free case, do the first-order terms in the estimation error cancel, as suggested by Equation (3.8)?
2. Can a pseudo inverse of the auto-correlation matrix be used to increase the robustness of the IRF estimate?
3. How reliable are the estimates of the error variance and bias?

3.2.3.1 First-Order Terms

The system simulated was a fourth-order Butterworth low-pass filter, with a normalized cut-off frequency of 0.4. Its impulse response is shown in Figure 3.1. We performed a Monte Carlo simulation involving 10,000 repetitions. During each trial, the filter was driven by a different 1000 point sequence of white Gaussian noise. We then calculated the cross-correlation between the input and output, using Equation (2.6), as well as the impulse response, using Equation (2.9). As the input was zero-mean white Gaussian noise with unit variance, both the cross-correlation and the IRF estimate should have been equal to the IRF of the simulated filter. Thus, the estimation error in the cross-correlation was taken to be the difference between the cross-correlation estimate, and the simulated IRF. Once all the trials had been performed, we calculated the mean and variance of the cross-correlation and impulse response errors as functions of lag over the ensemble.

Figure 3.2 shows the variances of the impulse response and cross-correlation estimates, plotted as a function of the lag, τ. The variance of the IRF estimate is two to three orders of magnitude smaller than that of the cross-correlation. Nevertheless, comparing these variances with the impulse response in Figure 3.1, makes it clear that

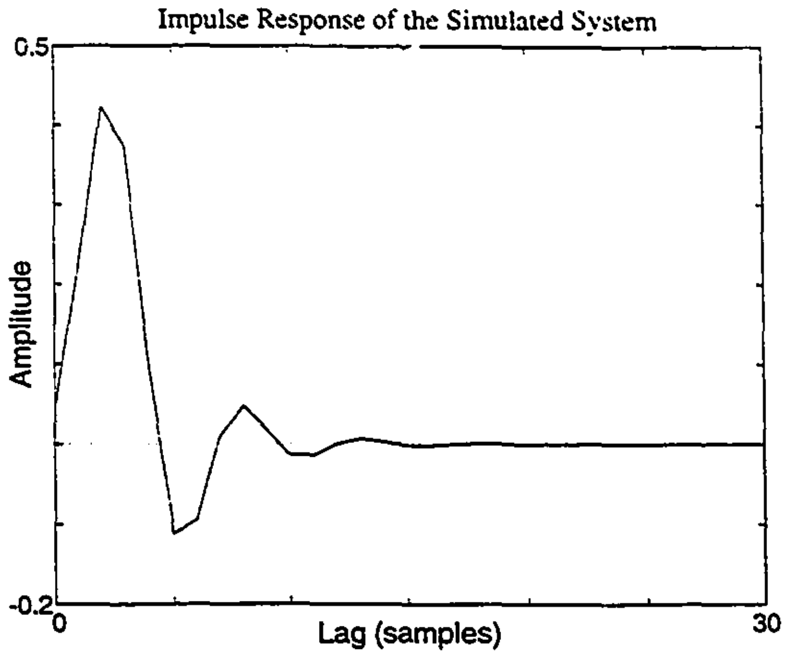


Figure 3.1: Impulse response of the fourth-order Butterworth low-pass filter used in the Monte Carlo simulations.

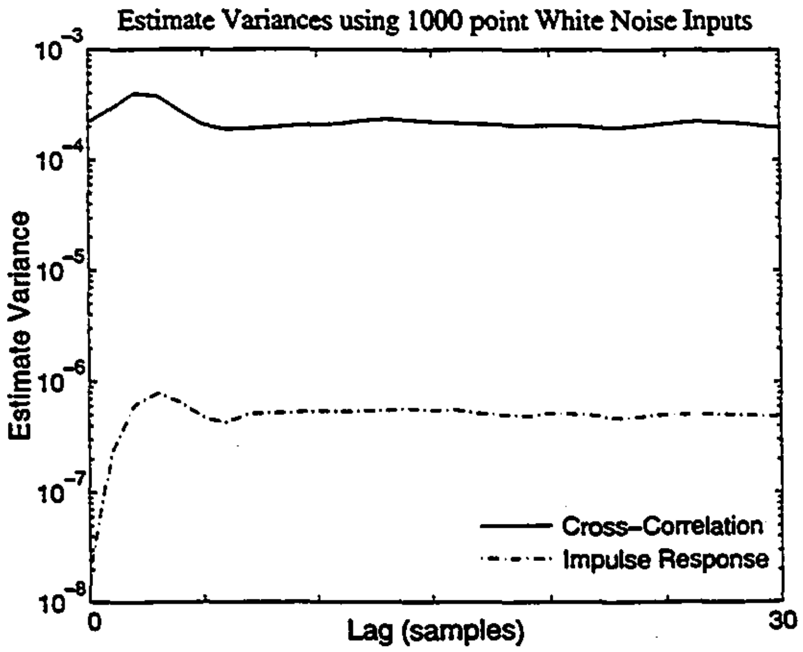


Figure 3.2: Variance in the estimates of the cross-correlation and the impulse response of a linear system. The inputs were 1000 point sequences of white Gaussian noise. Estimates were computed from noiseless records.

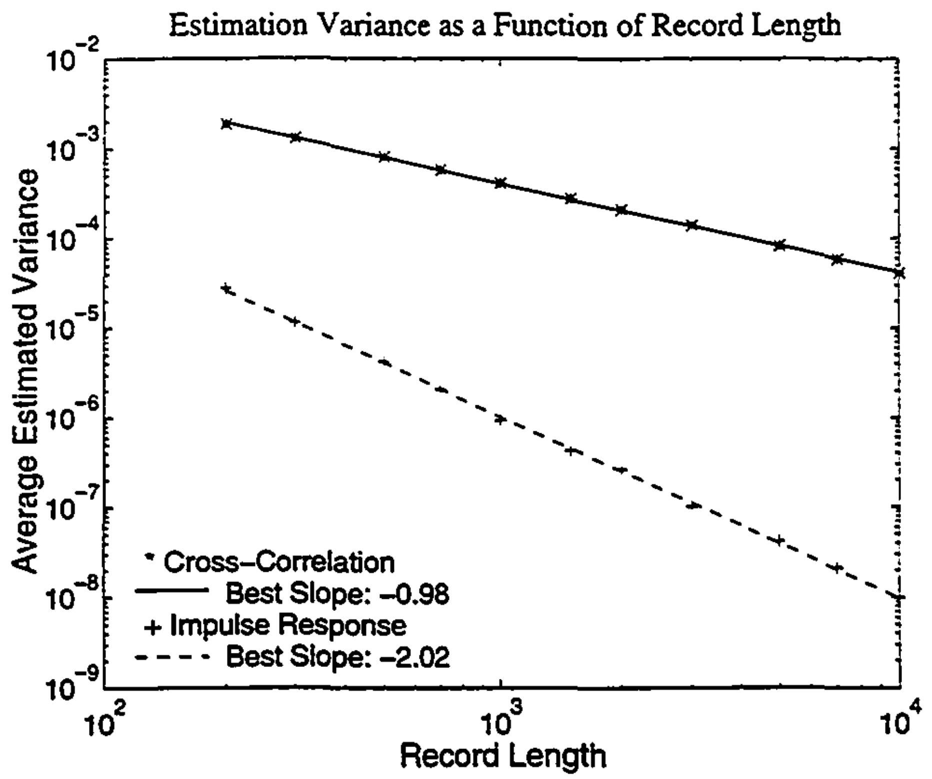


Figure 3.3: Estimation variance, averaged over the impulse response length, as a function of the length of the data records. Note that the impulse response variance decreases with the square of the record length, whereas the cross-correlation variance depends linearly on the record length.

both the correlation and IRF estimates are excellent, as would be expected given the length of the data records and the absence of noise.

Next, we examined the effect of varying the record length. The Monte Carlo simulation was repeated, with record lengths from 200 to 10,000 points. Figure 3.3 shows the estimation variance, averaged over the length of the IRF, as a function of the record length. On this log-log plot, we can see that the variance of the IRF estimate decreases with the square of the record length, while that of the cross-correlation estimate decreases linearly with the record length. This, together with the previous figure, supports our claim, based on Equation (3.8), that the Toeplitz matrix inversion procedure corrects for all first-order error terms in the cross-correlation estimate.

Finally, we added noise to the filter output, at a signal to noise ratio of 10 dB, and compared the variance of the impulse response estimate with that of the cross-

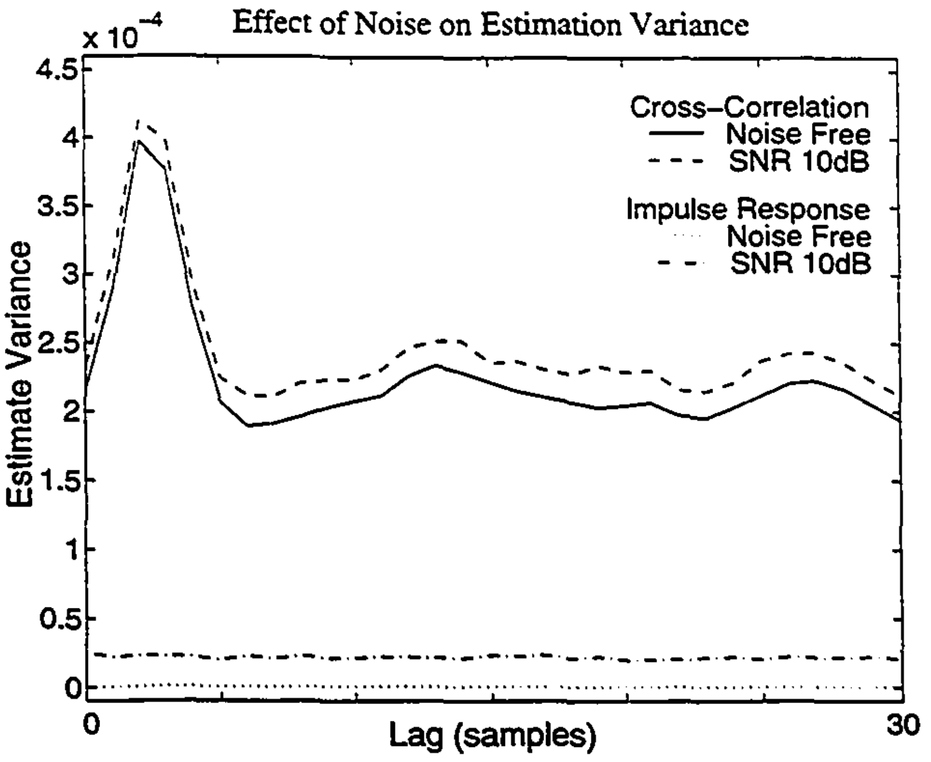


Figure 3.4: Effect of output noise on the variance of cross-correlation and impulse response estimates. The variance of the IRF estimated when the SNR was 10 dB is approximately equal to the difference between the variances of the cross-correlation estimated under 10 dB noise, and under noise free conditions.

correlation estimate. This time, both estimates were dominated by terms whose variance scaled linearly with the record length. As can be be seen in Figure 3.4, the variance of the IRF error was approximately equal to the difference between that of the cross-correlation function estimated with and without the noise. Hence, we concluded that when the input signal is a finite segment of a realization of a white process, the impulse response and cross-correlation estimates both contain the same noise term. The impulse response estimate, however, is not influenced by statistical fluctuations in the sognal components of the correlation estimates caused by the finite record length.

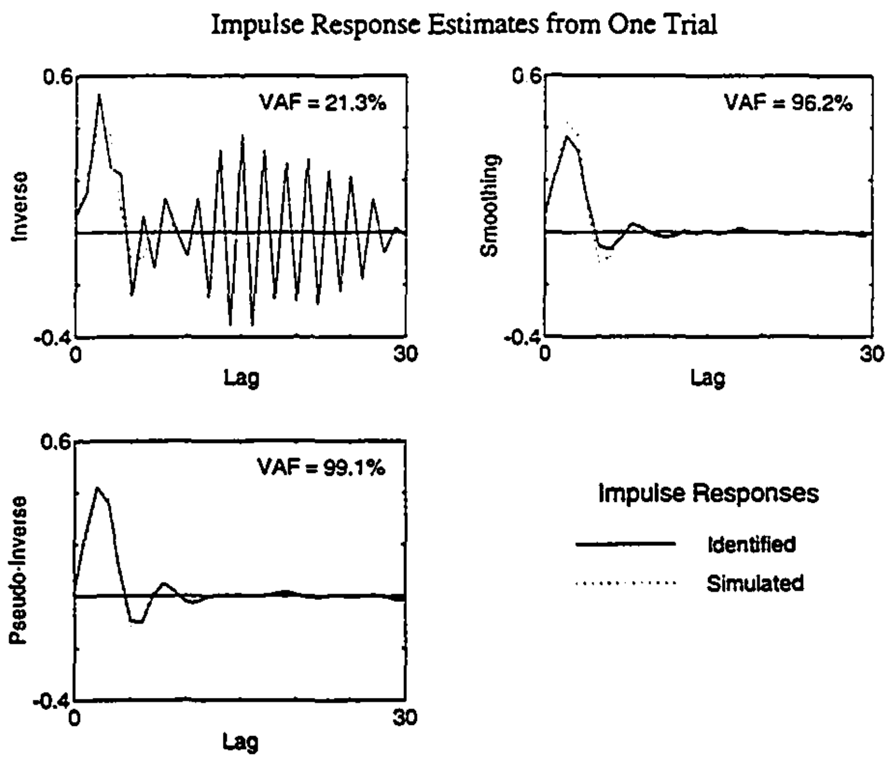


Figure 3.5: Results from a single trial of the Monte Carlo simulation. Impulse response estimates are plotted with that of the simulated system.

3.2.3.2 Examination of the Pseudoinverse

We used another Monte Carlo simulation to compare the performance of the pseudoinverse based IRF estimate with that provided by Equation (2.9). We examined the effects of using the pseudoinverse on the the estimation variance, and the bias error. These changes were compared to those resulting from the application of a simple three point smoothing filter to the initial IRF estimate. We chose to use the 3 point smoother for comparison purposes, as it is commonly used to suppress high frequency noise in IRF estimates [66, 92].

As in the previous simulations, the system was a fourth-order Butterworth low-pass filter, with a normalized cut-off of 0.4. In this case, the input was a coloured Gaussian sequence, generated by filtering a white Gaussian sequence with a second-order Butterworth filter with a normalized cut-off of 0.4. Ten thousand trials were performed using 1000 point data records, with the SNR set to 10 dB.

During each trial, we computed three filter estimates: the exact deconvolution, the smoothed deconvolution, and the pseudoinverse. We compared the impulse response estimates directly with the true impulse response, and compared the outputs predicted by the estimated filters with both the clean and noise corrupted outputs of the original filter. In all cases, we assessed the accuracy as the percentage of the signal variance accounted for by the model (3.5).

Figure 3.5 shows typical IRF estimates from a single trial of the simulation. The first panel shows the IRF identified using the exact inverse of the Toeplitz autocorrelation matrix. This IRF estimate contains noise concentrated near the Nyquist frequency, whose amplitude is comparable to that of the actual IRF.

The IRF shown in the second panel was computed by applying a 3-point smoother to the exact inverse solution. While the smoothing filter eliminated almost all of the high frequency noise, (visible in the second half of the estimate, where the simulated IRF is zero), it attenuated the peaks in the IRF.

The third panel shows the IRF estimated using the pseudoinverse based input deconvolution. Like the smoothed IRF estimate, there is virtually no high frequency ringing in the tail of the impulse response. In contrst with the smoothed IRF, the pseudoinvserse did not attenuate the the peaks in the IRF.

Figure 3.6 shows the distributions of the IRF and prediction accuracies for the three methods. The first panel, in the upper left corner, shows the estimated probability density functions of the impulse response accuracy. Note that the results from from the inverse solution (Equation 2.9) are absent from this panel, as that distribution fell well below the other two (mean 50.5% standard deviation 24.3%). From this panel, we see that the pseudoinverse produces significantly better estimates of the IRF shape than does smoothing of the inverse solution.

The second panel shows the prediction accuracy, with respect to the uncorrupted output, $y(t)$. Both the exact inverse and pseudoinverse solutions yielded significantly better predictions than did the smoothed solution. The pseudoinverse solution yielded a slightly better output prediction than the exact inverse solution. On average, the prediction error due to the pseudoinverse method was 38.6% less than that due to the

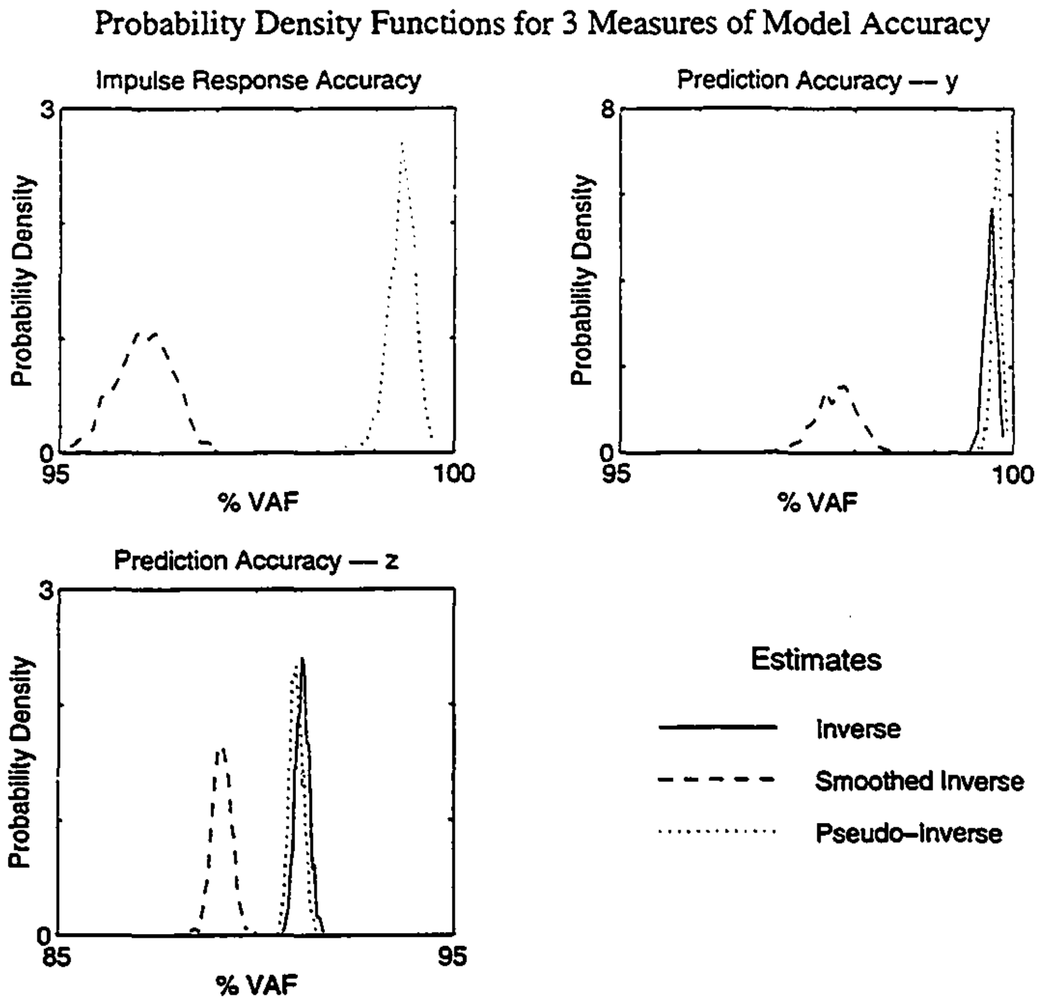


Figure 3.6: Probability density functions estimated from the Monte Carlo simulation. The first panel shows the estimated PDF of the accuracy of the IRF estimates. In the second panel and third panels, the PDFs of the prediction accuracy for the uncorrupted output and noise-corrupted output are shown.

exact inverse solution. Due to the excellent predictions produced by both methods, this only corresponded to an average 0.077% increase in the VAF, with a standard deviation of 0.049%.

The third panel in Figure 3.6 shows the distributions of the prediction accuracies, with respect to the noise corrupted signal, $z(t)$. Again, both the exact inverse and the pseudoinverse produce significantly better predictions than the smoothed solution. Here, however, the exact inverse predicts slightly more variance (mean 0.15%, standard deviation 0.06%) than does the pseudoinverse techniques. This observa-

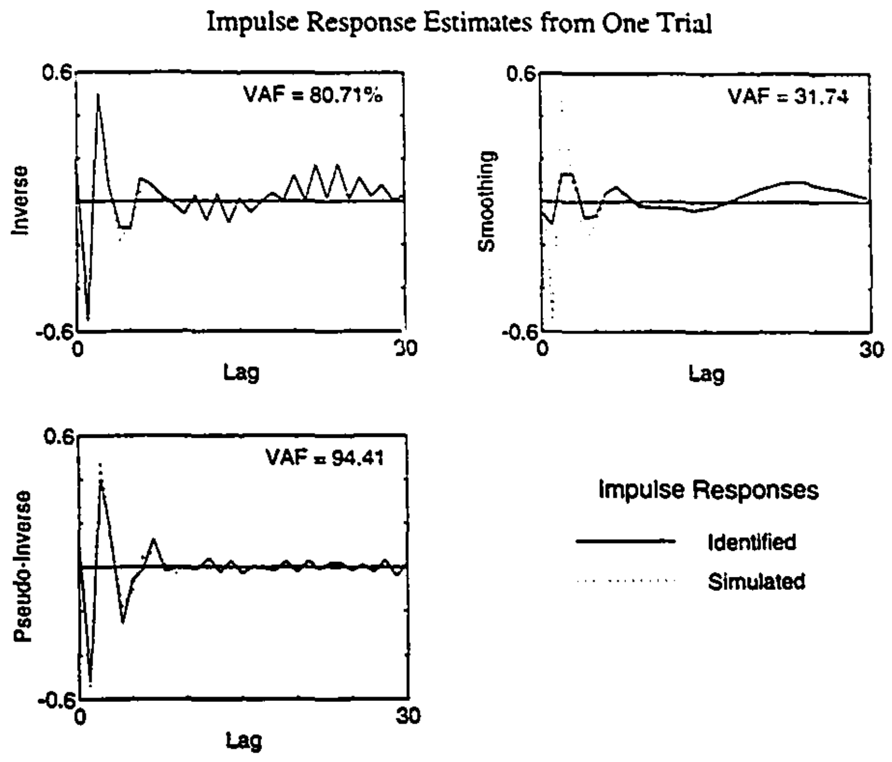


Figure 3.7: Estimates of the impulse response of a high-pass system. The first panel shows the result of using the full inverse of the autocorrelation matrix in the deconvolution. The second panel was produced by smoothing the inverse solution using a three-point, zero-phase smoother. The third panel shows the result of using the pseudoinverse based deconvolution technique developed in this chapter.

tion, taken together with the results from the previous panel, suggests that the exact solution models slightly more of the noise than does the pseudoinverse solution.

In this simulation, the pseudoinverse technique produced better estimates of the both the IRF shape and output than either the exact inverse solution, or the smoothed IRF.

Finally, we repeated the experiment, using a fourth-order Butterworth high-pass filter as the system, to demonstrate that the pseudoinverse method is not limited to low-pass systems. The input signal was white Gaussian noise, filtered by a fourth-order band-pass filter. White noise was added to the output.

Figure 3.7 shows the results of a typical single identification using the three methods. The first panel shows the estimate produced using the exact inverse (Equation

2.9) solution; it is contaminated with substantial noise at both low and high frequency. The second panel shows the result of smoothing this estimate using one pass of a three point smoothing filter. The high frequency ripple has been eliminated, but the peaks in the IRF have been greatly attenuated. Furthermore, the slow drift is still present. The final panel shows the IRF returned by the pseudoinverse method. Most of the high frequency noise, as well as all of the low frequency drift has been eliminated. As a result, this estimate is dramatically better than the other two.

3.2.3.3 Study of the Estimation Variance Estimate

Given the Toeplitz structured input auto-correlation matrix, the pseudoinverse used in the IRF estimation, and the IRF of the simulated system, we computed the bias error as follows:

$$\epsilon_{\text{bias}} = h - \Phi_{uu}^{\ddagger}\Phi_{uu}h \tag{3.26}$$

We could also calculate the random error:

$$\epsilon_{\text{rand}} = \Phi_{uu}^{\ddagger}\Phi_{uu}h - \hat{h} \tag{3.27}$$

Thus, for each trial in the Monte Carlo simulation, we calculated the random and bias components of the error in the impulse response estimate.

During each trial, we calculated the residuals, and used Equations (3.13) and (3.22) to predict the standard deviation of the IRF estimate, as a function of the lag. We then computed the ratio of this theoretical standard deviation to the random component of the measured error. The distribution functions for these ratios (one per point in the IRF) are plotted in Figure 3.8. We found that the distribution of the ratio of the random error to the theoretical standard deviation was well described (VAF > 97% in all cases) by a zero-mean Gaussian distribution, with unit variance. Thus, we conclude that Equations (3.13) and (3.22) produce accurate estimates of the variance of the random component of the estimation error.

Unfortunately, we do not have a point by point estimate of the bias error, and must limit ourselves to predicting its 2-norm. For each run in the Monte Carlo simulation,

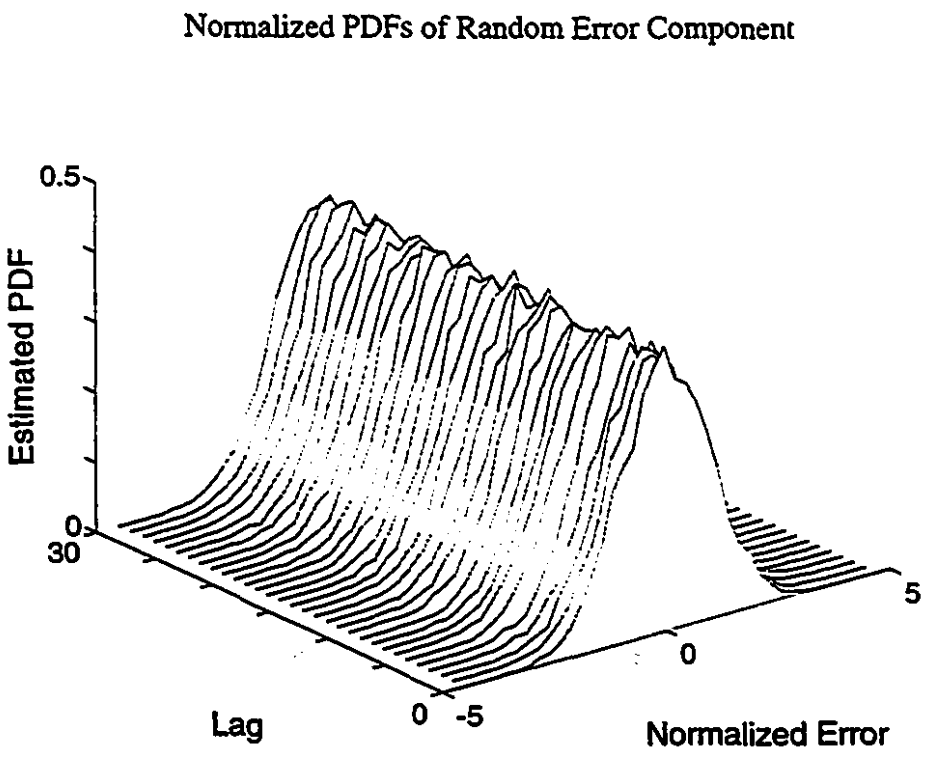


Figure 3.8: Estimated probability density functions of the ratio of the measured random error to the square root of the variance predicted by Equation (Estimation-Variance). Probability densities were estimated from a 10,000 trial Monte Carlo simulation.

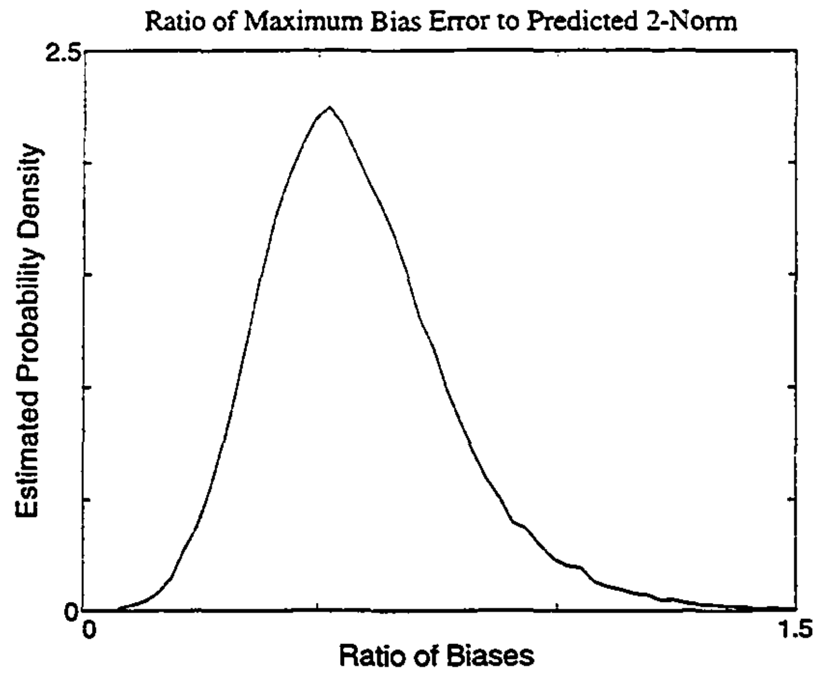


Figure 3.9: Estimated probability density function of the ratio of the maximum absolute bias error to the prediction of the 2-norm produced by Equation. (3.24). Probability densities were estimated from a 10000 trial Monte Carlo simulation

we computed the 2-norm of the bias component of the estimation error (3.26), and compared it to the estimate provided by Equation (3.24). The measured bias norm was always between half and double that of the estimate. In establishing a confidence interval for the IRF estimate, the 2-norm of the bias is of little use, as it provides no information about the bias at any one point.

However, given the an estimate its 2-norm, it is possible to construct confidence bounds on the bias component of the error, since the 2-norm of a vector is an upper bound on its infinity norm, [20]. We compared the estimate of the 2-norm of bias, to the maximum absolute value, the infinity norm in other words, of the measured bias. The probability density of this ratio is shown in Figure 3.9. Numerically integrating this PDF, we see that the 95% confidence bound on the maximum bias error is approximately 0.94 times the 2-norm estimate provided by Equation (3.24).

Similarly, we can construct confidence intervals due to the random component in the error. Because its distribution is Gaussian, twice the square-root of the variance

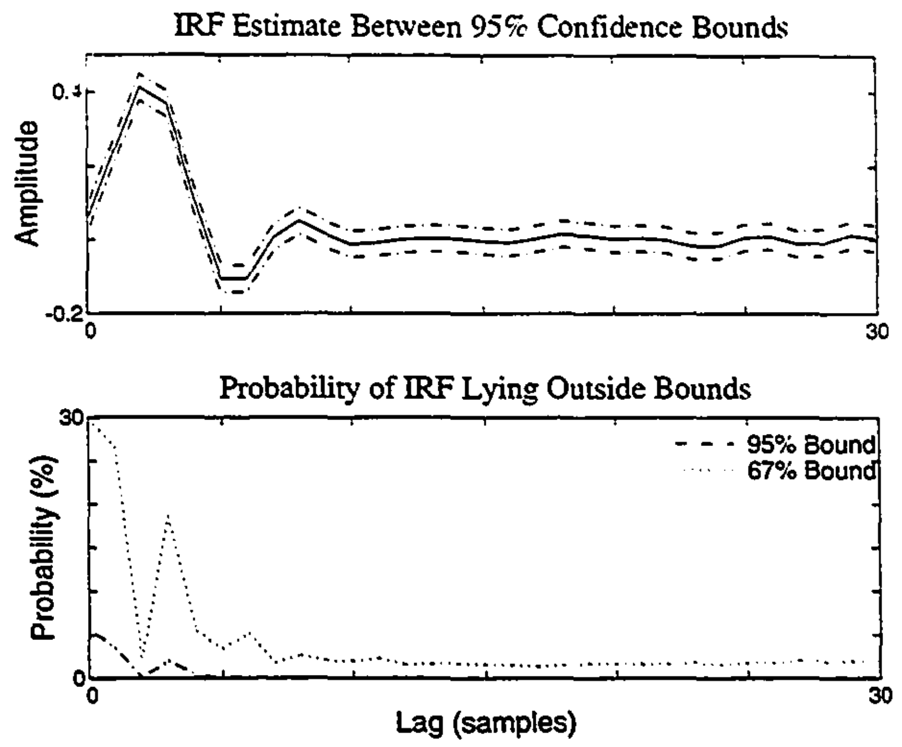


Figure 3.10: The upper panel shows an estimated IRF, between 95% confidence estimates. The lower panel shows the estimated probability, point by point, of the IRF lying outside of the 67% and 95% confidence bounds

predicted by Equation (3.14) is a 95% bound on the random component of the error. If we assume that the two error components are independent, the total error variance will be the sum of the variances of the two components. The confidence limits may be estimated as the square-root of the sum of the squared confidence bounds.

This confidence bound was tested on the results of the Monte Carlo simulation. The first panel of Figure 3.10 shows the results from a typical single trial, with the IRF estimate plotted between the estimated 95% confidence bounds. In the lower panel, we summarize the results from the whole simulation. The traces show the probability that each point in the true will IRF lie outside either the 67% or the 95% confidence bounds. We can see that the two bounds are only appropriate for one point, the first point in the IRF. All other points are well inside the confidence bounds. This is because the bias component could only be estimated globally. Thus, the maximum bias error must be assumed for every point in the IRF, which leads to a very conservative error bound in most places.

3.3 Multiple-Input Linear Systems

Consider a multiple-input linear system, such as that shown in Figure 2.10. We will assume that the two inputs, $u_1(t)$ and $u_2(t)$ are independent, and therefore, so are $y_1(t)$ and $y_2(t)$. Then:

$$\hat{\phi}_{u_1 y}(\tau) = \phi_{u_1 y_1}(\tau) + \tilde{\phi}_{u_1 v_1}(\tau) + \tilde{\phi}_{u_1 y_2}(\tau)$$

If we follow the development surrounding Equations (3.6) through (3.8), we can see that the only first-order error terms in the impulse response estimate will be:

$$\Phi_{u_1 u_1}^{-1} \tilde{\phi}_{u_1 v_z} + \Phi_{v_1 u_1}^{-1} \tilde{\phi}_{u_1 y_2}$$

Therefore, in the estimation of $h_1(\tau)$, $y_2(t)$ acts exactly like a second source of observation noise. Thus, Equation (3.14) may be used to predict the variance of $\hat{h}_1(\tau)$, provided the measurement noise, $v_z(t)$ in (3.14) is replaced with an "equivalent noise signal", $v_{ne}(t)$, which is the sum:

$$v_{ne}(t) = v_z(t) + y_2(t)$$

Similarly, $y_1(t)$ acts as a second source of observation noise in the estimate of $h_2(\tau)$. Thus, to estimate the variance in the estimate of $h_2(\tau)$, let:

$$v_{ne}(t) = v_z(t) + y_1(t)$$

and apply Equation (3.14), replacing $v_z(t)$ with $v_{ne}(t)$.

It is also evident that the pseudoinverse may be exploited exactly as with single input systems. It is likely that the spectra of the linear systems will overlap significantly, so the pseudoinverse alone will not remove all of the noise terms. In the next chapter, we will consider how to do so in a more general case: the multiple-input Wiener structure. The multiple-input linear system is a special case of this, and the techniques developed in Chapter 4 will apply.

3.4 Linear Elements of Wiener Systems

Here, we will consider the estimation of a Wiener (LN) system, as illustrated in Figure 2.7, when the nonlinearity has at least one significant odd term in its Taylor series. We will show how the expressions developed above can be used to estimate the variance of the impulse response estimate.

In keeping with the notation established by Figure 2.7, let $u(t)$, $x(t)$ and $y(t)$ be the input signal, the output of the linear element, and the output of the Wiener system, respectively. We will represent the static nonlinearity with a power series:

$$y(t) = \sum_{i=0}^{\infty} c_i x^i(t)$$

As in our general identification framework, $v_z(t)$ is an additive noise sequence. Thus, the observed output is:

$$z(t) = y(t) + v_z(t)$$

Consider the cross-correlation between $u(t)$ and $z(t)$:

$$\begin{aligned}
\phi_{uz}(\tau) &= E\left[u(t-\tau)\cdot\left\{\sum_{i=0}^{\infty} c_i x^i(t) \;+\; v_z(t)\right\}\right] \\
&= \sum_{i=0}^{\infty} c_i E[u(t-\tau)x^i(t)] \\
&= \sum_{i=0}^{\infty} c_{2i+1}\frac{(2i+1)!\sigma_x^{2i}}{2^{i+1}(i+1)!}\,\phi_{ux}(\tau)
\end{aligned}$$

This last step is obtained using Equation (3.1), which allows us to discard the terms which contain the product of an odd number of zero-mean Gaussian random variables (i.e. even powers of x). This result is simply an illustration of Bussgang's theorem [10]: If a linear IRF is fitted between $u(t)$ and $z(t)$ using cross-correlation and Toeplitz matrix inversion (Equation 2.9), its expected value will be:

$$E[\hat{h}(\tau)] = k_m h(\tau)$$

where the constant k_m is:

$$\sum_{i=0}^{\infty} c_{2i+1}\frac{(2i+1)!\sigma_x^{2i}}{2^{i+1}(i+1)!}$$

Let us subdivide the output signal $y(t)$ into two components, $y_o(t)$ and $y_e(t)$, which we shall call the "odd" and "even" components of $y(t)$. As the names suggest, the odd component, will contain all terms that include an odd power of $x(t)$, and the even component will contain all even powers of $x(t)$. Then, the input-output cross-correlation estimate can be written:

$$\hat{\phi}_{uz}(\tau) = k_m\phi_{ux}(\tau) + \tilde{\phi}_{uy_o}(\tau) + \tilde{\phi}_{uy_e}(\tau) + \tilde{\phi}_{uv_z}(\tau)$$

Applying the first-order perturbation estimate for the matrix inversion, and discarding higher-order terms results in:

$$\hat{h} = k_m h + \Phi_{uu}^{-1}\tilde{\phi}_{uy_o} - \Phi_{uu}^{-1}\tilde{\Phi}_{uu}\Phi_{uu}^{-1}\phi_{uy} + \Phi_{uu}^{-1}\tilde{\phi}_{uv_z} + \Phi_{uu}^{-1}\tilde{\phi}_{uy_e} \tag{3.28}$$

Furthermore, since:

$$\begin{aligned}
\tilde{\Phi}_{uu}\Phi_{uu}^{-1}\phi_{uy} &= k_m\tilde{\Phi}_{uu}h \\
&= k_m\sum_{i=0}^{T-1}\tilde{\Phi}_{uu}(\tau,i)h(i) \\
&= k_m\sum_{i=0}^{T-1}\left(\phi_{uu}(\tau-i) - \frac{1}{N}\sum_{t=1}^{N}u(t-\tau)u(t-i)\right)h(i) \\
&= k_m\phi_{ux} - \frac{k_m}{N}\sum_{t=1}^{N}u(t-\tau)\sum_{i=0}^{T-1}h(i)u(t-i) \\
&= \tilde{\phi}_{uy_o}
\end{aligned}$$

the second and third terms on the right hand side of Equation (3.28) cancel, and the only first order terms remaining in the estimation error are due to the observation noise and the even terms in the static nonlinearity.

The error term due to the even powers in the static nonlinearity enters Equation (3.28) exactly like the output noise term. Therefore, we can treat the even-power error terms as if they were simply other sources of observation noise. Define the following "equivalent noise" signal:

$$v_{ne}(t) = v_z(t) + y_e(t)$$

Equations (3.12) and (3.13) may now be used to predict the estimation variance, if we use $v_{ne}(t)$ in place of the measurement noise term, $v_z(t)$. Similarly, the pseudoinverse algorithm, described in Section 3.2.1.1, may be applied to Wiener systems whose nonlinearities include at least one significant odd term.

3.5 Summary

In this chapter, we have developed expressions for the variance of the impulse response estimate provided by the Toeplitz matrix inversion procedure [30] detailed in Equation (2.9). Analysis of the variance in the impulse response estimate suggested that it could be reduced, often dramatically, by replacing the inverse of the Toeplitz structured auto-correlation matrix with a suitably chosen pseudoinverse. This improvement in the estimation variance was realized at the cost of introducing a bias into the impulse response estimate. We developed procedures for choosing the pseudoinverse which sought a compromise between the reduction in the variance of the random error and the magnitude of the resulting bias.

We demonstrated that this procedure can be used for both low-pass and high-pass systems. It can remove both low frequency drift and high frequency noise from the impulse response estimate, and introduces comparatively little bias. This is in marked contrast to the use of a three point smoother, which cannot remove low frequency noise, and introduces considerable bias, especially to high-pass systems.

Use of the pseudoinverse also improved the condition of the estimate of the estimation variance. Simulations demonstrated that the estimation variance at each point in the IRF estimate could be estimated accurately using only measured quantities.

Estimation of the bias error proved to be more difficult. We were unable to develop any sort of point by point estimate or bound for the bias error. Hence, we attempted to estimate the mean square value of the bias. Simulations demonstrated how this estimate of the mean square bias could be used to bound the bias error over the whole IRF estimate.

Finally, we illustrated how this analysis can be applied to both multiple-input linear systems and single-input Wiener systems. The estimation of multiple-input Wiener systems is the topic of the next chapter.

Chapter 4

Multiple-Input Wiener Systems

4.1 Preliminaries and Notation

In this chapter, we will develop algorithms for the identification of a particularly important block-structure: the multiple-input Wiener system. We will deal first with two-input systems and then illustrate how our algorithms may be extended to deal with n-input systems.

A single-input Wiener system, as illustrated in Figure 2.7, consists of a dynamic linear element followed by a zero-memory nonlinearity [31]. Two possible multiple-input Wiener system structures, termed the b- and 1c-structures by Chen [13], are shown, for two-inputs, in Figure 4.1. In both cases, the inputs, (u_1, u_2), are processed by separate linear dynamic systems, (h_{u1}, h_{u2}). In the 1c-structure (Fig. 4.1a) the linear element outputs, (x_1, x_2) are transformed by a multiple-input static nonlinearity, $(m(\cdot,\cdot))$. In the b-structure (Fig. 4.1b) the linear system outputs are summed, and then transformed by a single-input static nonlinearity. Clearly, the b-structure is a special case of the 1-c structure.

Given either structure, let $u_1(t)$ through $u_n(t)$ represent the n system inputs, and let $y(t)$ represent its output. As stated previously, we will concentrate our discussion on the two-input case, where the inputs are $u_1(t)$ and $u_2(t)$. We will assume that the system is time-invariant and that the inputs, $u_1(t)$ and $u_2(t)$, are independent, stationary, zero-mean Gaussian signals, which need not be white. Let N represent

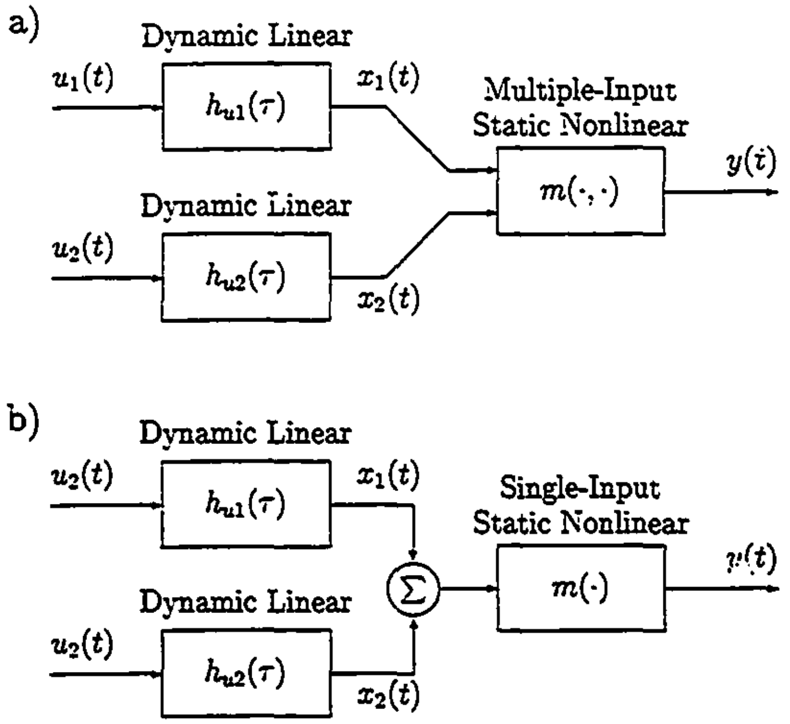


Figure 4.1: Two two-input Wiener system structures. The inputs $u_1(t)$ and $u_2(t)$ are processed by the dynamic linear systems $h_{u1}(\tau)$ and $h_{u2}(\tau)$. In the first case (a), the outputs of the linear system are transformed by a multiple input nonlinearity, $m(\cdot,\cdot)$. A special case of this is (b), where the linear system outputs are summed to from $x(t)$, which is then transformed by a single-input static nonlinearity $m(\cdot)$.

the number of input/output points available in the data records, and let T represent the memory length of the linear elements.

We will also assume that the non-linearity can be represented by a power series; this forces the static nonlinearity to be continuously differentiable with respect to its input, and is required for the following mathematical development. Thus for the more general 1c-structure:

$$m(x_1, x_2) = \sum_{i,j=1}^{\infty} c_{[i,j]} x_1^i x_2^j \tag{4.1}$$

and the output can be written:

$$y(t) = \sum_{i,j=1}^{\infty} c_{[i,j]} x_1^i(t) x_2^j(t) \tag{4.2}$$

For the simpler b-structure, shown in Fig. 4.1b:

$$m(x) = \sum_{i=1}^{\infty} c_i x^i \tag{4.3}$$

the output becomes:

$$\begin{aligned} y(t) &= \sum_{i=1}^{\infty} c_i (x_1(t) + x_2(t))^i \\ &= \sum_{i=1}^{\infty} c_i \sum_{j=0}^{i} \begin{pmatrix} i \\ j \end{pmatrix} x_1^{(i-j)}(t) x_2^j(t) \end{aligned} \tag{4.4}$$

where $\begin{pmatrix} i \\ j \end{pmatrix}$ is the binomial coefficient: $\frac{i!}{(i-j)!j!}$.

4.2 Identification of the b-Structure

Due to its simpler structure, we will consider the b-structure first, and then show how the methods developed for it can be applied to the more general 1c-structure. For this simpler system, there are two cases which must be considered separately: the first case occurs when the polynomial representation of the nonlinearity contains at least one significant odd term, in the second case, the nonlinearity contains at least one significant even term. Clearly, these two are not mutually exclusive.

4.2.1 Case 1 - Nonlinearities Containing Odd Terms

If the polynomial representation of $m(\cdot)$ contains significant odd terms, the linear subsystems can be estimated from first-order cross-covariances, as shown in Korenberg [53]. To see how this is achieved, examine the cross-covariance between one input (u_1) and the output, (y).

$$\phi_{u_1y}(\tau) = E[u_1(t)(y(t+\tau) - E[y(t)])] \tag{4.5}$$

where $E[x]$ refers to the expected value of x. Substituting (4.4), we get:

$$\phi_{u_1y}(\tau) = \sum_{i=1}^{\infty} c_i \sum_{j=0}^{i} \begin{pmatrix} i \\ j \end{pmatrix} \cdot E[u_1(t-\tau)x_1^{(i-j)}(t)] \cdot E[x_2^j] \tag{4.6}$$

This is a weighted sum of terms of the form:

$$E[u_1(t-\tau)x_1^{(i-j)}(t)] \cdot E[x_2^j(t)] \tag{4.7}$$

The terms in (4.7) are expected values of products of jointly Gaussian random variables. From the discussion proceeding Equation (3.1), we know that this will be zero, for odd numbers of terms. Hence, if j is odd, the second term in (4.7) will vanish. If j is even, the second term will be non-zero, but an even value of i will cause the first term to contain an odd number of terms, and therefore vanish. Therefore, unless j is even and i is odd, (4.7) yields zero. For even j and odd i, both expected values are non-zero, and using (3.1), we can see that (4.7) becomes:

$$(1 \cdot 3 \ldots \cdot (i-j))(1 \cdot 3 \ldots \cdot (j-1))\sigma_{x_1}^{(i-j-1)}\sigma_{x_2}^{j}\phi_{u_1x_1}(\tau) \tag{4.8}$$

where σ_{x_1} is the standard deviation of the linear subsystem output $x_1(t)$. Therefore, each term in (4.6) is either equal to zero, or is proportional to the cross-covariance between $u_1(t)$ and $x_1(t)$. Hence, as can be seen in (21) of Korenberg [53], the weighted sum is also proportional to the cross-covariance taken across the linear sub-system.

$$\phi_{x_1y}(\tau) = K\phi_{u_1x_1}(\tau) \tag{4.9}$$

where K is given by:

$$K = \sum_{i=0}^{\infty}\sum_{j=0}^{i} c_{2i+1} \begin{pmatrix} 2i+1 \\ 2j \end{pmatrix} (1 \cdot 3 \cdot \ldots \cdot (2i-2j+1)) \cdot (1 \cdot 3 \cdot \ldots \cdot (2j-1)) \sigma_{x_1}^{(2i-2j)} \sigma_{x_2}^{2j} \tag{4.10}$$

It is evident that K depends on the shape of the nonlinearity, through the dependence on the polynomial coefficients c_i, and the variances of the outputs of the two linear subsystems. It might appear that the constant of proportionality would depend on which linear subsystem was being identified. However, by rewriting the expression for K in terms of the factorial operation, we get:

$$\begin{aligned} K &= \sum_{i=0}^{\infty} c_{2i+1} \sum_{j=0}^{i} \begin{pmatrix} i \\ j \end{pmatrix} \frac{(2i+1)!}{2^i i!} \sigma_{x_1}^{2(i-j)} \sigma_{x_2}^{2j} \\ &= \sum_{i=0}^{\infty} c_{2i+1} \frac{(2i+1)!}{2^i i!} (\sigma_{x_1}^2 + \sigma_{x_2}^2)^i \end{aligned} \tag{4.11}$$

Hence K is a function of the shape of the nonlinearity and the total power at its input, but does not depend on the input being used. Therefore, using input-output cross-covariances to estimate the cross-covariances across the linear elements introduces no relative scaling; each subsystem has its cross-covariance scaled by the same amount. Hence, by deconvolving the input autocorrelation, as in Equation (2.9), we can recover the impulse responses of the linear systems to within a single scaling factor.

4.2.1.1 Improved Covariance Estimates

Covariances estimated using finite duration time averages will contain noise, whose amplitude will decrease as the length of the time average is increased, (see Equations 3.2 through 3.4) . Terms in (4.6) which vanish in expectation, will nonetheless contribute noise to finite-length covariance estimates. Eliminating those terms should improve the covariance estimate by reducing the number of terms in the sum which contribute only noise. This may be seen by considering the cross-covariance between a zero-mean Gaussian signal, $w(t)$, (possibly correlated with either $u_1(t)$, $u_2(t)$, or both), and the output, $y(t)$.

$$\phi_{wy}(\mu) = \sum_{i=1}^{\infty} c_i \sum_{j=0}^{\infty} \begin{pmatrix} i \\ j \end{pmatrix} E[w(t-\mu)x_1^{(i-j)}(t)x_2^{i}(t)] \tag{4.12}$$

Because $u_1(t)$ and $u_2(t)$ are independent, it is possible to decompose $w(t)$ into two orthogonal components $w_1(t)$, and $w_2(t)$ such that :

$$w(t) = w_1(t) + w_2(t)$$

and

$$\begin{matrix} E[u_1(t-\mu)w_2(t)] \equiv 0 \\ E[u_2(t-\mu)w_1(t)] \equiv 0 \end{matrix} \ \forall\mu \tag{4.13}$$

Now, using this orthogonal construction, we can rewrite (4.12):

$$\begin{aligned} \phi_{wy}(\mu) = \ & \sum_{i=1}^{\infty} c_i \sum_{j=0}^{\infty} \begin{pmatrix} i \\ j \end{pmatrix} \ \Big\{E[w_1(t-\mu)x_1^{(i-j)}(t)]E[x_2^{j}(t)] \\ & \qquad + E[x_1^{(i-j)}(t)]E[w_2(t-\mu)x_2^{j}(t)]\Big\} \end{aligned} \tag{4.14}$$

Using the arguments surrounding (4.6) through (4.10), we can rewrite (4.14):

$$\phi_{wy}(\mu) = K\phi_{wx_1}(\mu) + K\phi_{wx_2}(\mu) \tag{4.15}$$

Assume that we have formed an initial estimate of the linear subsystem, $h_{u2}(\tau)$. Let us see how we can use this to improve our estimate of the other linear system. The initial subsystem estimate is:

$$\hat{h}_{u2}(\tau) = Kh_{u2}(\tau) + n_2(\tau) \tag{4.16}$$

where $n_2(\tau)$ is an additive, zero mean noise process. Estimate the output due to $\hat{h}_{u2}(\tau)$ by convolution:

$$\begin{aligned} \hat{x}_2(t) &= \hat{h}_{u2}(\tau) * u_2(t) \\ &= Kh_{u2}(\tau) * u_2(t) + n_2(\tau) * u_2(t) \end{aligned} \tag{4.17}$$

The cross-covariance between the input signal $u_1(t)$, and $\hat{x}_2(t)$ is:

$$\hat{\phi}_{u_1\hat{x}_2}(\mu) = K\hat{\phi}_{u_1x_2}(\mu) + n_2(\tau) * \hat{\phi}_{u_1u_2}(\mu) \tag{4.18}$$

In the expected value sense, or equivalently if time averages are performed using infinitely long records, all the terms on both sides of (4.18) will go to zero. However, estimates of these terms produced from finite duration time averages may be non zero.

Now, consider the cross-covariance between the input $u_1(t)$, and the system output. Using (4.15), replacing $w(t)$ with the input signal $u_1(t)$, we get:

$$\hat{\phi}_{u_1y}(\mu) = K\hat{\phi}_{u_1x_1}(\mu) + K\hat{\phi}_{u_1x_2}(\mu) \tag{4.19}$$

As $u_1(t)$ and $u_2(t)$ are independent, $\phi_{u_1x_2}(\mu)$ is identically equal to zero. However, $\hat{\phi}_{u_1x_2}(\mu)$ is an estimate obtained from finite data records, which may be non-zero and so contribute noise to the cross-covariance estimate. To form an improved estimate for the cross-covariance across the first subsystem, subtract (4.18) from (4.19).

$$\hat{\phi}_{u_1(y-\hat{x}_2)}(\mu) = K\hat{\phi}_{u_1x_1}(\mu) - n_2(\tau) * \hat{\phi}_{u_1u_2}(\mu) \tag{4.20}$$

Comparing equations (4.19) and (4.20), it is clear that an improvement in the cross-covariance estimate will result if:

$$\mid n_2(\tau) * \hat{\phi}_{u_1u_2}(\mu) \mid < \mid K\hat{\phi}_{u_1x_2}(\mu) \mid = \mid Kh_{u2}(\tau) * \hat{\phi}_{u_1u_2}(\mu) \mid \tag{4.21}$$

To interpret this condition, we must know the statistical properties of $\hat{\phi}_{u_1u_2}(\mu)$. Its expected values is zero, as $u_1(t)$ and $u_2(t)$ are independent. Its autocorrelation can be computed:

$$E[\hat{\phi}_{u_1u_2}(\mu-\tau)\hat{\phi}_{u_1u_2}(\mu)] = \frac{1}{N^2}\sum_{t_1,t_2=1}^{N} \phi_{u_1u_1}(t_1 - t_2 + \tau)\phi_{u_2u_2}(t_1 - t_2) \tag{4.22}$$

Making a change of variables, and assuming that the correlation lengths are much less than the data length, this becomes:

$$\frac{1}{N}\sum_{t=-N}^{N} \phi_{u_1u_1}(t+\tau)\phi_{u_2u_2}(t)$$

hence, the auto-correlation of $\hat{\phi}_{u_1u_2}$ is equal to the convolution of the autocorrelations of the two input signals, and its spectrum is equal to the product of the spectra of the two inputs. Therefore, given (4.21), improvement in the estimate of h_{u1} will result if the power of the noise component in (4.16) is less than power in the signal component over the bandwidth of $\hat{\phi}_{u_1u_2}$ (which is less than the bandwidth of either of the two input signals).

Subtracting its output from the system output and computing the cross-covariance between u_1 and this new signal will lead to an improved estimate of the first subsystem. This process can be applied iteratively, first updating one half of the system, and then the other. Note that while this iterative process can theoretically eliminate the noise arising from the vanishing terms in (4.6) it will have no effect on the noise in the non-vanishing terms.

4.2.1.2 Algorithm for Non-Even Systems

The overall algorithm for identifying a two-input Wiener system (b-structure) with a non-even nonlinearity is:

1. Estimate a linear IRF between u_1 and y, $\hat{h}_{u1}(\tau)$
2. Generate $\hat{x}_1(t)$, the convolution of $\hat{h}_{u1}(\tau)$ and u_1.
3. Generate $\tilde{y}_1(t) = y(t) - \hat{x}_1(t)$
4. Estimate a linear IRF between u_2 and $\tilde{y}_1(t)$ ($\hat{h}_{u2}(\tau)$)
5. Generate $\hat{x}_2(t)$, the convolution of $\hat{h}_{u2}(\tau)$ and u_2.
6. Generate $\tilde{y}_2(t) = y(t) - \hat{x}_2(t)$
7. Identify a linear response between u_1 and $\tilde{y}_2(t)$ ($\hat{h}_{u1}(\tau)$)
8. If a significant change in the IRFs has occurred, go to step 2. Otherwise, continue.
9. Generate $\hat{x}(t) = \hat{x}_1(t) + \hat{x}_2(t)$

10. Fit a polynomial between $\hat{x}(t)$ and $y(t)$

For systems having more than two inputs, the algorithm must be modified as follows.

- Repeat steps 4, 5 and 6 for every input.
- In step 6, if $h_n(\tau)$ is to be estimated in step 7, then the estimated outputs of all of the other linear subsystems must be subtracted from $y(t)$.

4.2.2 Case 2 - Even Nonlinearities

If the static nonlinearity contains no significant odd terms the first order input-output cross-covariances will vanish, and the method outlined above will fail. In this case, an approach employing either the second-order input-output cross-covariances, as suggested by Korenberg [53] or the second-order, cross-cross-covariance function, as suggested by Chen *et al.*, [13], may be used. Consider the second-order cross-cross-covariance function:

$$\begin{aligned}\phi_{u_1u_2y}(\tau_1,\tau_2) &= E[u_1(t-\tau_1)u_2(t-\tau_2)(y(t)-E[y(t)])] \qquad (4.23)\\ &= \sum_{i=0}^{\infty} c_{2i}\sum_{j=0}^{2i}\begin{pmatrix}2i\\ j\end{pmatrix} E[u_1(t-\tau_1)x_1^{(2i-j)}(t)]\,\cdot\\ &\qquad E[u_2(t-\tau_2)x_2^{j}(t)]\end{aligned}$$

Proceeding as before, we may eliminate all terms for which the expected value is zero. In this case, terms for which j is even will have expected values of zero since they contain products of an odd number of zero-mean Gaussian random variables. So,

$$\phi_{u_1u_2y}(\tau_1,\tau_2) = \sum_{i=1}^{\infty} c_{2i}\sum_{j=1}^{i}\begin{pmatrix}2i\\ 2j-1\end{pmatrix} E[u_1(t-\tau_1)x_1^{2(i-j)+1}(t)]\,\cdot \qquad (4.24)$$
$$E[u_2(t-\tau_2)x_2^{2j-1}(t)]$$

Using (3.1) to evaluate the expected values and then simplifying, we can write:

$$\phi_{u_1u_2y}(\tau_1, \tau_2) = K_2 \cdot \phi_{u_1x_1}(\tau_1)\phi_{u_2x_2}(\tau_2) \tag{4.25}$$

where:

$$K_2 = \sum_{i=1}^{\infty} \frac{c_{2i}(2i)!}{2^{i-1}(i-1)!}(\sigma_{x_1}^2 + \sigma_{x_2}^2)^{i-1} \tag{4.26}$$

Equation (4.25) provides the means to estimate the cross-covariances for the two linear subsystems, as every column of $\phi_{u_1u_2y}$ is proportional to $\phi_{u_1x_1}$. Similarly, every row of $\phi_{u_1u_2y}$ is proportional to $\phi_{u_2x_2}$. However, instead of the exact cross-covariance, we must use an estimate derived from finite length records. This will surely contain some estimation noise. Given estimates of the linear subsystem cross-covariances, we can use (4.25) to estimate the second-order cross-cross-covariance function. Estimates of $\phi_{u_1x_1}$ and $\phi_{u_2x_2}$ should be chosen such that the mean square error (MSE) between the measured and computed values of $\hat{\phi}_{u_1u_2y}$ is minimized. Thus:

$$\frac{1}{T^2}\sum_{i=1}^{T}\sum_{j=1}^{T}\left\{\hat{\phi}_{u_1u_2y}(i,j) - \hat{\phi}_{u_1x_1}(i)\hat{\phi}_{u_2x_2}(j)\right\}^2 \tag{4.27}$$

should be minimized. Minimizing the MSE over the entire cross-covariance function will use all of the information available in that function to estimate the first-order cross-covariances. These optimal estimates of the first order-covariance functions can be obtained using the singular value decomposition (SVD) [20] as follows. Let the SVD of $\hat{\phi}_{u_1u_2y}$ be:

$$\hat{\phi}_{u_1u_2y} = USV^T \tag{4.28}$$

where U and V are orthogonal matrices, and S is a diagonal matrix:

$$S = \text{diag}(s_1, s_2, \ldots, s_T)$$

with positive real entries, ordered such that $s_1 \geq s_2 \geq \ldots \geq s_T$. We will use an overbar to denote vectors, and to avoid a notational collision with the input signals $u_1(t)$ and $u_2(t)$. Thus, we write the columns of U and V as:

$$
\begin{aligned}
U &= \begin{bmatrix} \overline{u}_1 & \overline{u}_2 & \dots & \overline{u}_T \end{bmatrix} \\
V &= \begin{bmatrix} \overline{v}_1 & \overline{v}_2 & \dots & \overline{v}_T \end{bmatrix}
\end{aligned}
$$

Then, we rewrite the SVD of $\hat{\phi}_{u_1u_2y}$ as [20]:

$$\hat{\phi}_{u_1u_2y} = \sum_{i=1}^{T} \overline{u}_i s_i \overline{v}_i{}^T$$

In expectation, the cross-cross-correlation is:

$$\phi_{u_1u_2y} = \phi_{u_1x_1}\phi^T_{u_2x_2}$$

Since the U and V are orthogonal matrices, the optimal choices for $\phi_{u_1x_1}$ and $\phi_{u_2x_2}$ are the left and right singular vectors, $\overline{u}_1$ and $\overline{v}_1$, which are associated with the largest singular value, s_1.

4.2.2.1 Improved Estimates

The resulting cross-covariance estimates will be scaled by an unknown constant and contain noise arising from the estimation of both vanishing and non-vanishing terms. In the odd case, described above, we proposed an iterative process to reduce the noise due to the estimation of terms which vanish in expectation. This process was derived by evaluating the cross-covariance between a generic signal, $w(t)$ and the system output. Separating $w(t)$ into components $w_1(t)$ and $w_2(t)$ allowed us to analyze all the terms in the cross-covariance and so find alternate ways of estimating the vanishing terms. We will use a similar procedure for the even case; this time, an additional generic signal will be required since it is the second-order, cross-cross-covariance function that must be estimated. Denote this new signal by $z(t)$ and define $z_1(t)$ and $z_2(t)$ as in (4.13). The second-order cross-cross-covariance between w, z, and y can be written:

$$\phi_{wzy}(\tau_1, \tau_2) = E[w(t-\tau_1)z(t-\tau_2)\{y(t) - E[y(t)]\}] \quad (4.29)$$

Note that the expected value of the output is subtracted prior to the evaluation of the covariance function, just as in (4.23). Using procedures similar to those outlined in (4.23) through (4.26), it can be shown that:

$$\begin{aligned}\phi_{wzy}(\tau_1,\tau_2) = \; K_2 \cdot [\; & \phi_{w_1x_1}(\tau_1)\phi_{z_1x_1}(\tau_2) + \phi_{w_1x_1}(\tau_1)\phi_{z_2x_2}(\tau_2) \\ & + \phi_{w_2x_2}(\tau_1)\phi_{z_1x_1}(\tau_2) + \phi_{w_2x_2}(\tau_1)\phi_{z_2x_2}(\tau_2)]\end{aligned} \tag{4.30}$$

Now, replace the signals w and z with the inputs u_1 and u_2 respectively. Thus w_1, the component of w which is parallel to u_1, is simply w_1. Similarly, $z_2 = u_2$. Thus, the second term on the right side of Equation (4.30) corresponds to the ideal result in Equation (4.25).

Clearly, w_2, which is the component of w which is orthogonal to u_1, and z_1, both will be zero, in expectation. Therefore, the first, third and fourth terms in Equation (4.30) will have zero expected value. However, for finite record lengths, these estimates are unlikely to be exactly zero, so removing them from (4.30) should improve the estimates of the cross-cross-covariance function. The problem then is to obtain independent estimates of these terms; this is possible for the first and fourth terms as follows.

The first term in (4.30) may be estimated by computing the second-order, cross-cross-covariance between the two inputs, $u_1(t)$ and $u_2(t)$, and $x_1^2(t)$, multiplied by an appropriate scale factor. This scale factor may be determined by forming a least squares estimate for the gain between the squared linear system output, $x_1^2(t)$, and the actual system output, $y(t)$.

$$\kappa(x_1^2, y) = \frac{E[x_1^2 y] - E[x_1^2]E[y]}{E[x_1^4] - (E[x_1^2])^2} \tag{4.31}$$

Expanding the above, noting that x_1 is zero mean and Gaussian, yields:

$$\kappa(x_1^2, y) = \frac{1}{2\sigma_{x_1}^4} \cdot \sum_{i=1}^{\infty} c_{2i} \sum_{j=0}^{i} \begin{pmatrix} 2i \\ 2j \end{pmatrix} E[x_2^{2j}] \left\{ E[x_1^{2(i-j+1)}] - \sigma_{x_1}^2 \cdot E[x_1^{2(i-j)}] \right\}$$

Expanding the expected values and gathering similar terms:

$$\kappa(x_1^2, y) = \frac{1}{2} \sum_{i=1}^{\infty} \frac{c_{2i}(2i)!}{2^{i-1}(i-1)!} (\sigma_{x_1}^2 + \sigma_{x_2}^2)^{i-1} = \frac{K_2}{2} \tag{4.32}$$

But, the cross-cross-covariance between w_1, z_1 and $\kappa(x_1^2, y)(x_1^2 - E[x_1^2])$ is:

$$\frac{K_2}{2} \cdot 2\phi_{w_1 x_1}(\tau_1)\phi_{z_1 x_1}(\tau_2) \tag{4.33}$$

Hence, we can estimate the term $\kappa(x_1^2, y)x_1^2(t)$ and subtract it from the output, $y(t)$. Substituting this signal for $y(t)$ in(4.29) will eliminate the first term in (4.30).

The fourth term in (4.30) can be eliminated by repeating the above process, using the output of the second subsystem. Eliminating both the first and fourth terms in (4.30) should improve the estimate of the cross-cross-covariance, and lead to better estimates of the linear subsystem dynamics. This iterative process can be repeated until it converges. As before, it will be limited by the noise inherent in the estimates of non-vanishing terms in (4.23).

4.2.2.2 Scaling Issues:

As the matrices returned by the SVD, U and V, are unitary, the cross-covariance estimates will be scaled to have unit norms. This will result in linear subsystem estimates that are scaled by different amounts. To proceed further, we must estimate the degree of relative scaling, and correct for it. (Note that in the non-even case, scaling was not an issue since the scale factor introduced by the estimation procedure was the same for both linear elements.)

The previous section provides a mechanism to estimate the degree of relative scaling. Assume that $\hat{x}_1(t) = k_1 x_1(t)$ and $\hat{x}_2(t) = k_2 x_2(t)$. Then:

$$E\left[\frac{\kappa(\hat{x}_1^2, y)}{\kappa(\hat{x}_2^2, y)}\right] = \left(\frac{k_2}{k_1}\right)^2 \tag{4.34}$$

Hence, we can estimate the relative gain applied to the linear subsystems. Once the linear subsystem outputs have been rescaled, they can be summed, and a static function fitted between this and the system output.

4.2.2.3 Algorithm for Even Wiener Systems

The complete algorithm for even systems is similar to that for non-even systems except that it is based on the second-order cross-cross-covariance. It proceeds as follows:

1. Estimate $\hat{\phi}_{u_1 u_2 y}(\tau_1, \tau_2)$ from the input/output data.
2. Apply the singular value decomposition to $\hat{\phi}_{u_1 u_2 y}(\tau_1, \tau_2)$. Use the columns of U and V associated with the largest singular value as $\hat{\phi}_{u_1 x_1}(\tau_1)$ and $\hat{\phi}_{u_2 x_2}(\tau_2)$.
3. Estimate $h_{u1}(\tau)$ from $\hat{\phi}_{u_1 x_1}(\tau_1)$.
4. Estimate $h_{u2}(\tau)$ from $\hat{\phi}_{u_2 x_2}(\tau_2)$.
5. Calculate the outputs due to the linear subsystems by convolution, and square them.
6. Use (4.31) to scale the squared linear subsystem outputs.
7. Estimate the variance accounted for by the squared linear subsystem outputs. If this has not improved since the last iteration, go to step 10.
8. Subtract the scaled squared outputs (step 6) from $y(t)$, producing $\tilde{y}(t)$.
9. Estimate $\hat{\phi}_{u_1 u_2 \tilde{y}}(\tau_1, \tau_2)$, and go to step 2.
10. Use (4.34) to estimate the relative gain. Correct for this and generate $\hat{x}(t)$, a scaled estimate of $x(t)$.

11. Fit a static nonlinearity between $\hat{x}(t)$ and $y(t)$.

If the system being identified has more than two inputs, the following modifications to the algorithm are necessary:

- In steps 1 and 9, a second-order cross-cross-covariance function must be estimated for every pair of inputs (i.e. a 3 or 4 input system will require two such functions).
- Steps 2 and 3 must be repeated for every estimated linear subsystem.

4.2.3 Simulations

The performance of the method was assessed using data obtained from simulations of the two-input Wiener system shown in Figure 4.2. The linear subsystems were represented as low-pass filters, similar to those encountered in neuromuscular research. A half-wave rectifier was used for the static nonlinearity because it is neither an even nor an odd function. Furthermore, it is not continuously differentiable, hence its polynomial representation requires many high-order terms. The simulation and the identification procedures were carried out using MATLAB[1], a commercial software package for scientific and engineering numeric computation. In all cases, the identification accuracy was assessed in terms of prediction accuracy evaluated using white Gaussian inputs.

4.2.3.1 Odd Method

The first set of simulations was carried out using white Gaussian inputs, with no measurement noise. For input signals containing 10,000 points, the algorithm converged quickly producing the estimates shown in Figure 4.3 which accounted for 98.93% of the output variance of the data set used in the identification procedure. More importantly, the estimates accounted for 98.59% of the output variance when tested with a independent set of white Gaussian inputs. Notice that although the amplitude scale

[1]The MathWorks Inc., Natick, Mass

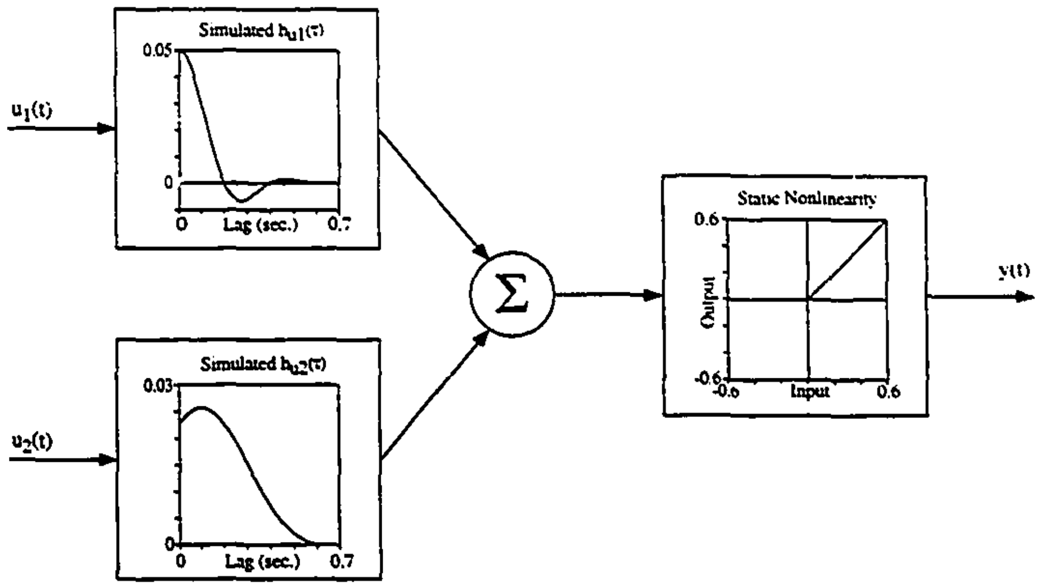


Figure 4.2: Block diagram of the simulated multiple-input Wiener system.

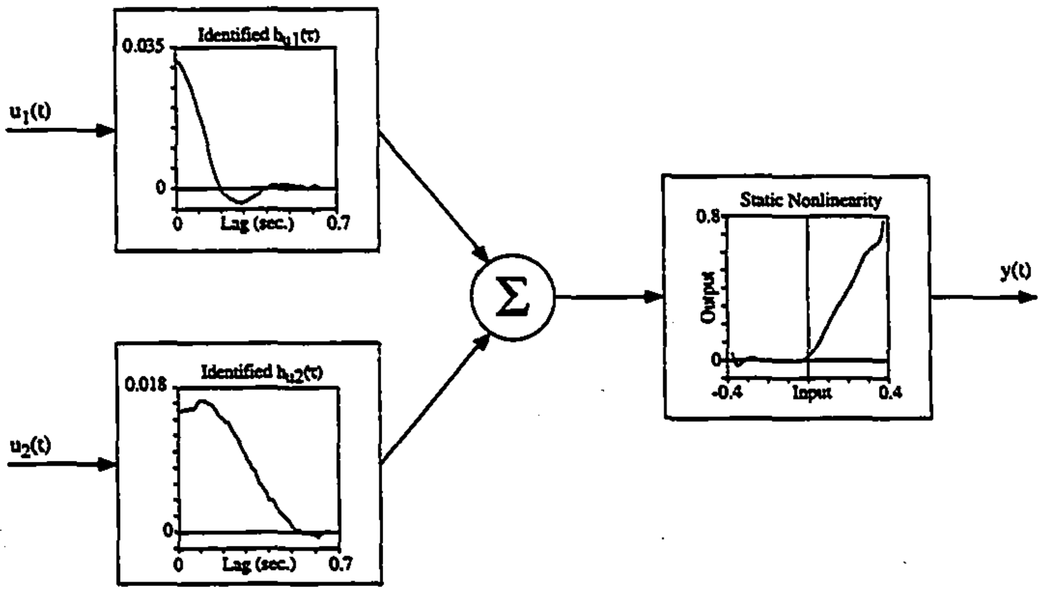


Figure 4.3: Estimated elements for the multiple-input Wiener system in Figure 4.2 For this simulation, the system was driven by two independent 10,000 point white Gaussian signals. The identified model accounted for 98.9% of the output variance. Note the different divisions of the overall system gain between this and Figure 4.2.

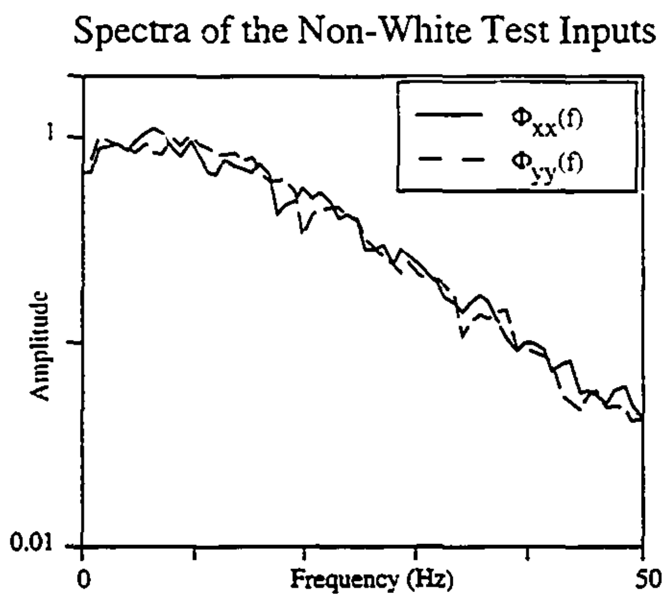


Figure 4.4: Power spectra of the low-pass filtered PRBS inputs which were used in the noise performance experiments.

of the IRFs in Figure 4.3 was about twice that of those in Figure 4.2 there was an equal reduction in the input scale for the static nonlinearity so the overall system gain was unchanged.

The algorithm was then evaluated under more realistic conditions:

- Shorter record lengths, 5,000 and 2,500 points, were used to assess the effect of record length on the identification accuracy.
- Measurement noise was simulated by adding white Gaussian noise, having a variance of 10 to 100% of the model output variance, to the output signal.
- Low-pass filtered pseudo-random-binary-sequences (PRBS) were used as inputs. The spectra of these non-Gaussian inputs, shown in Figure 4.4, were significantly non-white.

In all cases the iteration procedure converged rapidly; the most significant improvements were achieved in the first few iterations while subsequent iterations had little effect. This is illustrated in Table 4.1 which shows prediction accuracy of the linear subsystem estimates for the first five iterations in the noise free case. The iteration procedure contributed relatively little (1-2%) when long records were used since the estimates obtained with the first iteration were very good. However, when shorter records were used, the initial estimate was not as good and the iterative procedure brought about more substantial improvements (4-7%). The last line of Table 4.1 presents these results in terms of the reduction in residual variance, instead of the gain in variance accounted for. Here the value of the iterative enhancement becomes clear, as it produced between a 3 and 9 fold reduction in the variance of the residuals.

Iteration	10,000 point records		5,000 point records		2,000 point records	
	$\hat{h}_{u_1}(\tau)$	$\hat{h}_{u_2}(\tau)$	$\hat{h}_{u_1}(\tau)$	$\hat{h}_{u_2}(\tau)$	$\hat{h}_{u_1}(\tau)$	$\hat{h}_{u_2}(\tau)$
1	99.57	99.32	98.35	97.20	96.66	96.86
2	99.89	99.84	99.61	99.64	99.32	99.12
3	99.90	99.85	99.67	99.71	99.49	99.10
4	99.90	99.85	99.67	99.69	99.45	99.13
5	99.90	99.85	99.70	99.69	99.45	99.12
Ratio of Residual Variances	4.30	4.53	5.5	9.03	6.07	3.57

Table 4.1: The accuracy of the linear subsystem estimates is shown at each iteration of the non-even system. (Fig. 4.3) Accuracy is measured as percent variance accounted for (%VAF) The last line in the table shows the reduction in the the residual variances. It is the ratio of the initial and final residual variances.

Figure 4.5 summarizes the results of the evaluation procedure in terms of the model accuracy as a function of output noise level for each condition. It is evident that the algorithm is robust in the presence of output noise. Excellent estimates of the linear and nonlinear subsystems were obtained even with large amounts of output noise. Thus, even when the variance of the noise was equal to that of the system output, the models identified with 10,000 point records predicted almost 95% of the noise-free output variance. As would be expected, performance of the method in the presence of noise improved with increasing record length. Thus, the prediction accuracy decreased with increasing noise levels for all record lengths; but the rate of decrease became larger as the record length was decreased.

Figure 4.5 also demonstrates that there was no significant change in the predi-

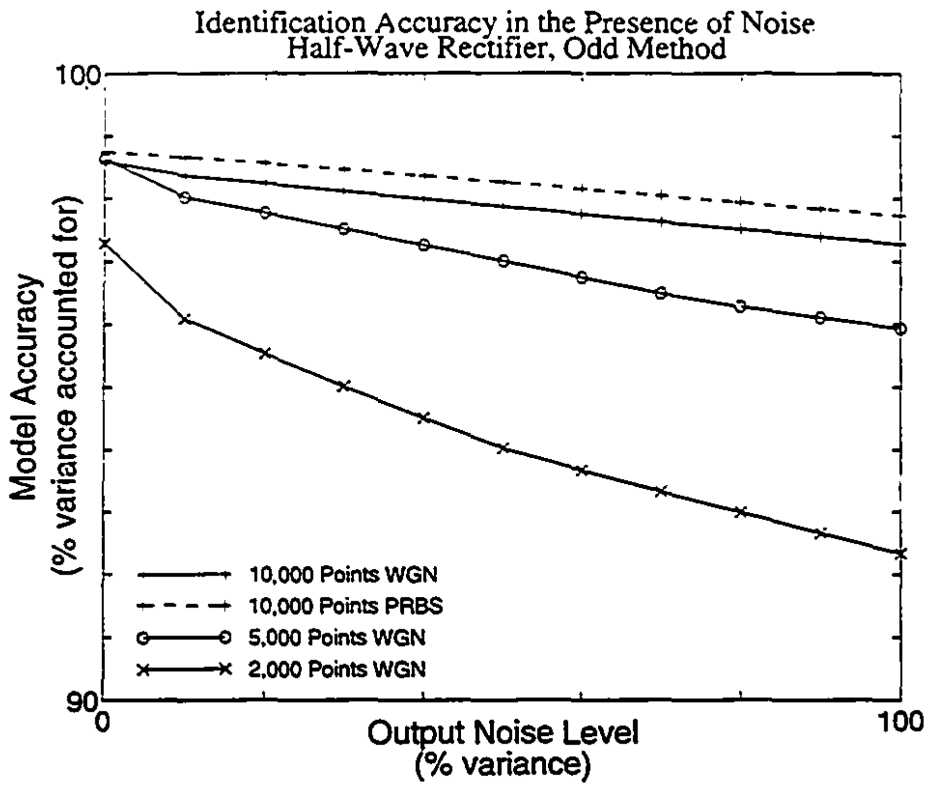


Figure 4.5: Noise performance of the odd algorithm applied to the system shown in Figure 4.2. Four different inputs were used: Gaussian White Noise records with lengths of 10,000, 5,000 and 2,500 points, and 10,000 point, low-pass filtered pseudo-random binary sequences.

cation accuracy when the low-pass filtered PRBS inputs were used rather than the Gaussian white noise. This indicates that the method is likely to work with the types of input signals that can be applied under realistic experimental conditions. However, the results obtained with the non-white inputs must be interpreted with caution. Initial estimates of the linear IRFs obtained with non-white inputs were contaminated with high frequency noise. We were able to reduce this contamination by using the pseudoinverse based techniques developed in the previous chapter. Provided that the inputs had significant power over the bandwidths of the linear systems that they excited, the pseudoinverse technique was able to separate the estimates of the impulse responses from much of the random estimation error. We must stress, however, that using the pseudoinverse technique introduced bias into the impulse response estimates, and that the magnitude of this bias was not easily estimated.

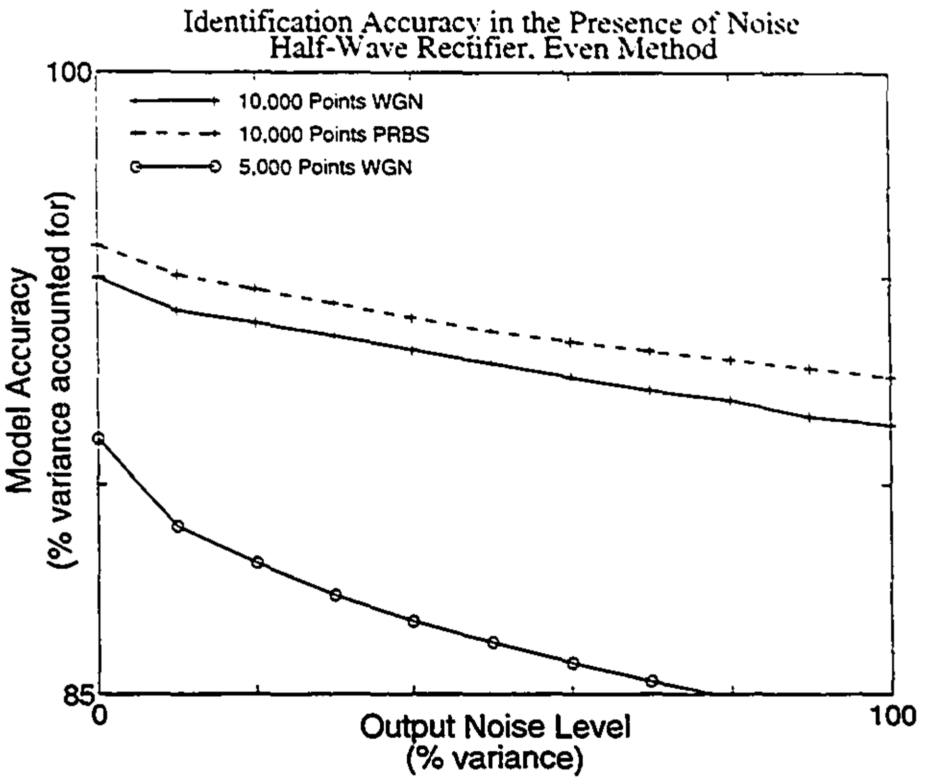


Figure 4.6: Noise performance of the even algorithm applied to the system shown in Figure 4.2.

4.2.3.2 Even Method

We repeated the simulations discussed above, using the even method to identify the system. Once more, the algorithm converged rapidly in all cases and gave good estimates of the system. The results of these simulations are summarized in Figure 4.6. The behaviour in the presence of noise, the sensitivity to record length, and performance with non-white, non-Gaussian inputs were generally similar to that of the non-even algorithm. The major difference between the two algorithms is that the accuracy of the even algorithm was generally somewhat lower than for the "odd" algorithm under equivalent conditions.

4.2.3.3 Alternate Methods for Even Systems

Korenberg [53] and Korenberg and Hunter [60] suggested using a single slice of the second-order cross-covariance function to estimate the linear dynamics of a single-

	Estimates using the cross-cross-covariance.			Estimates using the the second-order cross-covariance.		
	SVD	Average	Slice	SVD	Average	Slice
Initial Estimate						
$\hat{h}_{u_1}(\tau)$	96.91	95.24	64.47	94.44	79.36	83.07
$\hat{h}_{u_2}(\tau)$	91.68	82.07	69.35	91.92	87.21	27.79
Final Estimate						
$\hat{h}_{u_1}(\tau)$	98.45	98.44	93.00	98.08	96.97	97.01
$\hat{h}_{u_2}(\tau)$	96.91	96.94	93.27	99.02	98.44	80.81

Table 4.2: Accuracy of the linear system estimates based on the second-order cross-covariance functions. Accuracy is expressed in terms of the percent of the variance accounted for by the estimated system

input even Wiener system. This is also possible in the multiple input case, and would greatly reduce the computational burden associated with estimating the linear subsystem dynamics. Korenberg and Hunter [60] also suggest using the average of the second-order cross-correlation. We examined these possibilities using the half-wave rectifier simulation under noiseless conditions. Table 4.2 shows the prediction accuracy of initial linear system estimates obtained from the SVD, the best single rows and columns, and the average over the rows and columns of the second-order cross-cross-covariance function. The second row in this table shows the accuracy of the IRF estimates resulting from the iteration process. The IRF estimates obtained from the best single slices were always inferior to the those produced using the principal singular vectors. Averaging of the covariance function produced estimates comparable to the SVD, but only after convergence of the iteration had dramatically reduced the estimation noise in the cross-covariance function. In producing linear system estimates from the noisy initial estimate of the cross-covariance function, the SVD was vastly superior to averaging.

4.3 Identification of the 1-c Structure

Thus far, we have restricted ourselves to the b-structure illustrated in Fig. 4.1b. We will now consider the identification of systems having the more general 1c-structure shown in Fig. 4.1a. The output of a general two-input Wiener system was given by Equation (4.2):

$$y(t) = \sum_{i,j=1}^{\infty} c_{[i,j]} x_1^i(t) x_2^j(t)$$

4.3.1 Estimation of the Linear Dynamic Elements

The symmetries present in (4.4) are not present in (4.2). For the b-structure, we developed methods for systems which were members of two (not mutually exclusive) classes: those with odd terms in their nonlinearities, and those with even terms. An algorithm was developed for each class; systems which fell into both classes could be identified using either algorithm. For the more general 1c-structure three classes of systems can be described. Any system whose nonlinearity includes terms in which both $x_1(t)$ and $x_2(t)$ are raised to odd powers will be called an odd/odd system. Similarly, we can define even/even and even/odd systems.

4.3.1.1 Odd/Odd Systems

If the output of a generalized multiple-input Wiener system contains significant odd/odd terms, the dynamics of both linear subsystems may be estimated from the second-order cross-cross-correlation function. In fact, the first 9 steps of our algorithm for even b-structures may be applied exactly as presented. Once the linear elements have been estimated, a two-input nonlinearity is fitted between the linear subsystem outputs and the measured system output.

4.3.1.2 Even/Even Systems

If the output contains significant even/even terms, initial estimates of the linear subsystem dynamics may be inferred from the second-order cross-covariances between each of the inputs and the output, as suggested by Korenberg [53]. While in principle a single slice of this function may be used, we suggest using the SVD over the whole second-order cross-covariance function.

The development of section 4.2.2 may be modified to produce better estimates of the second-order, single-input cross-covariance. Replacing $z(t)$ with $w(t)$ in (4.29) and (4.30) makes these equations apply to the single input covariance function, instead of the dual input function. Following the rest of the development, we can compute:

$$\tilde{y}_1(t) = y(t) - \kappa(\hat{x}_2^2, y)\hat{x}_2^2(t) - \kappa(\hat{x}_1\hat{x}_2, y)\hat{x}_1(t)\hat{x}_2(t) \tag{4.35}$$

where $\kappa(x, y)$ is defined in (4.31). Calculating the second order cross-covariance between the input $u_1(t)$, and this new signal, will result in an improved estimate of $h_{u1}(\tau)$, provided reasonably good estimates of the dynamics of both linear systems are available (see 4.21).

4.3.1.3 Odd/Even Systems

The presence of odd/even terms in (4.2) allows us to estimate the dynamics of $h_{u1}(\tau)$ from the first order cross-covariance between $u_1(t)$ and $y(t)$. Similarly, even/odd terms allow the estimation of $h_{u2}(\tau)$ from a first order covariance function. If both odd/even and even/odd terms are present, our odd-system algorithm may be used as presented to produce estimates of the linear subsystem dynamics. If, however, the system contains only odd/even terms, a third-order cross-covariance function (first order in $u_1(t)$ and second order in $u_2(t)$) will be required to gain an estimate of $h_{u2}(\tau)$. In Section 5.2.3 we will develop a gradient search procedure to extract an impulse response function from a third-order cross-correlation.

4.3.2 Estimation of the Nonlinearity

In all cases where the 1-c structure is used, a multiple-input polynomial surface must be estimated, which leads to a more demanding least squares problem than in the single-input case. For example, consider a two input system, where $x_1(t)$ and $x_2(t)$ are the outputs of the linear elements. If we let R be the maximum polynomial order fitted, then we must choose the coefficients $c_{i,j}$ to minimize:

$$\epsilon = \frac{1}{N}\sum_{t=1}^{N}\left(y(t) - \sum_{i,j=0}^{R} c_{i,j}T(x_1(t), i)T(x_2(t), j)\right)^2$$

where $T(x(t), i)$ is the i'th order Tchebyshev polynomial applied to $x(t)$. Although this is a somewhat more difficult least squares problem than that for the simplified system, it does eliminate the need for the relative gain calculation (4.34), since any relative scaling introduced by the IRF estimation will be compensated for by the scales of the domains of the polynomial surface.

The difficulties associated with estimating this nonlinear surface have received little, if any, attention in the literature on multiple-input block-structures. Korenberg and Hunter [31, 53] consider a MIMO extension of the LNL cascade which includes this type of nonlinearity, and provide references for techniques to estimate them. However, these papers [83, 112] deal primarily with single input functions, treating the multiple-input case as a simple generalization. They do, however, deal with the very interesting problems associated with performing least-squares fitting when both the input and the output contain measurement errors. This is the "Total Least Squares" problem, and is particularly relevant to cascade identification, as the input(s) to the static nonlinearity will certainly contain noise due to errors in the estimates of the linear system(s).

Apart from the added complexity of the coefficient estimation, using a multiple-input nonlinearity involves some additional difficulties in dealing with the domain of the polynomial. To illustrate, first consider the estimation of a single-input polynomial, $m(\cdot)$, given records of its input and output, $x(t)$ and $y(t)$ respectively. Clearly, the largest domain over which we may reliably characterize the polynomial is the interval from the minimum to the maximum of the input record. Hence:

$$\mathcal{D}\text{om}(\hat{m}(\cdot)) = \{x : \min(x(t)) \leq x \leq \max(x(t))\}$$

If the distribution of the input is heavily weighted towards its mean value, as in a Gaussian distribution, our estimate of the polynomial will be much less reliable near the edges of its domain, where there are comparatively few data points to influence the cost function, than in the middle.

If we use $\hat{m}(\cdot)$ to predict the output due to an arbitrary input, $z(t)$, we must treat any points outside of the domain with care. At the very least, any points in $z(t)$ that lie outside of the domain of definition for $\hat{m}$ should be tagged as unreliable estimates. Alternatively, the input signal could be limited to the domain of definition of the polynomial. In that case, we compute $\hat{m}(\tilde{z}(t))$, where:

$$\begin{aligned}
\tilde{z}(t) &= \min(x(t)) & z(t) \le \min(x(t)) \\
\tilde{z}(t) &= z(t) & \min(x(t)) < z(t) < \max(x(t)) \\
\tilde{z}(t) &= \max(x(t)) & z(t) \ge \max(x(t))
\end{aligned}$$

Let us consider how this procedure may be applied to an estimate of a two-input nonlinearity. Given $\hat{m}(\cdot,\cdot)$, an estimate of the polynomial obtained from inputs, $x_1(t)$ and $x_2(t)$, and output $y(t)$, we might be tempted to define its domain as follows:

$$\mathcal{D}\text{om}(\hat{m}(\cdot,\cdot)) = \left\{ [x_1, x_2] : \begin{array}{c} \min(x_1(t)) \le x_1 \le \max(x_1(t)) \\ \min(x_2(t)) \le x_2 \le \max(x_2(t)) \end{array} \right\} \tag{4.36}$$

This definition of the domain, however, is not restrictive enough. Consider the following simulation. Two independent, Gaussian signals, $x_1(t)$ and $x_2(t)$, were generated, and plotted as ordered pairs in an x-y plot (Figure 4.7). The dashed line is the boundary of the domain defined in (4.36). It is evident from this figure that there are large regions in the $[x_1,\ x_2]$ plane that are within the simple rectangular domain, that are not near any of the data-points. Clearly, if a polynomial is fitted between the input sequences, $x_1(t)$ and $x_2(t)$, and the output sequence $y(t)$, there are no data-points in the corners of this domain to influence the cost function. Hence, the polynomial value in these regions will not necessarily reflect the behaviour of the actual system.

A somewhat better description of the domain of support is given by the convex hull [15, 34] of the points $(x_1(t), x_2(t))$. This is defined as the smallest convex polygon that includes all of the points in $(x_1(t), x_2(t))$, and is marked by the solid line in Figure 4.7. Although it gives a much better characterization of the support of the two sequences than is provided by the simple rectangular domain, there are still poorly defined regions within the convex hull. This is similar to the one dimensional case, where there can be relatively few points influencing the polynomial fit near the edges of the domain.

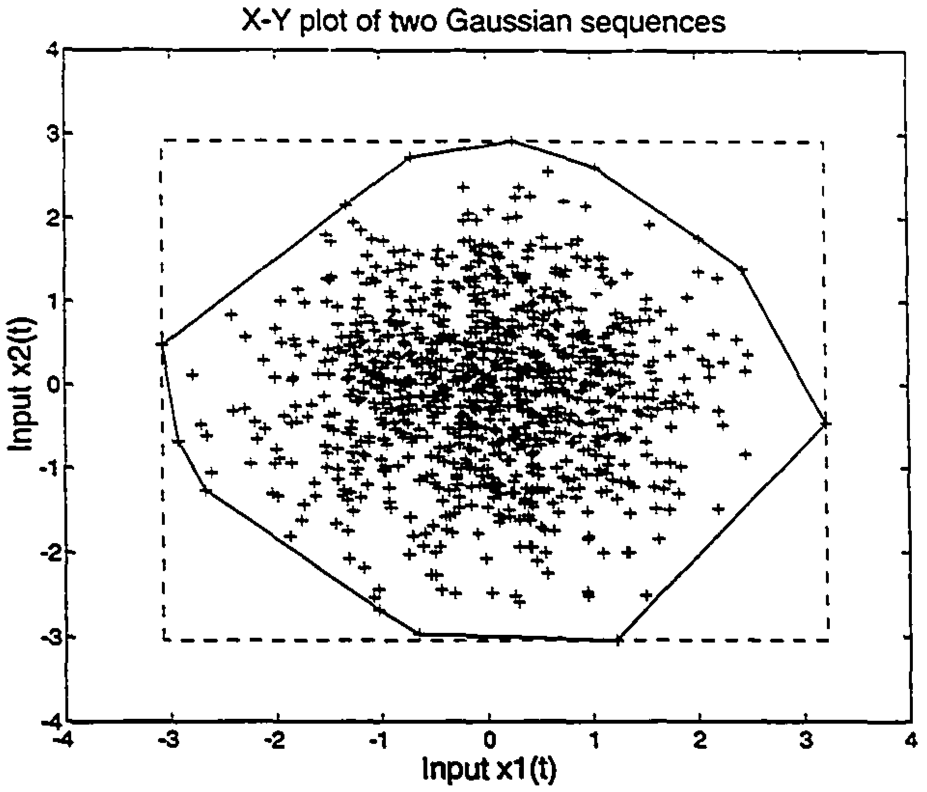


Figure 4.7: X-Y plot of two Gaussian sequences. The dashed line is the rectangular domain defined by the two sequences. The solid line is their convex hull.

When performing output prediction in the single-input case, the input signal was limited to the domain of support of the input signal that was used to estimate the polynomial. In the two-input case, we must limit the input pairs to the domain defined either by the simple rectangular domain, or to the convex hull. Restriction to the rectangular domain is easily achieved, as the inputs can be limited independently. With the convex hull, we must project any exterior points onto the closest edge.

For more than two inputs, the problem of describing the support of the polynomial becomes very complex. In three dimensions, one could construct a convex hull, from triangles whose vertices are all data-points. In constructing the hull, the triangular elements would share each of their sides with a neighbouring element. A four input system would require constructing a boundary in four-dimensional space, where the hull would be made up of tetrahedrons, each joined to four others by a common surface.

Construction of a convex hull is relatively straight forward. We will summarize it below, using the three dimensional case, with co-ordinates x, y and z as an example.

1. Select the points with the maximum and minimum values for each co-ordinate. These points are surely on the boundary

2. Construct a surface of triangular elements, whose vertices are currently selected boundary points

3. For each triangular surface element, calculate the perpendicular distance from each point to the plane defined by that surface, defining the direction of the outer surface normal as being positive. Find the point (if any) with the largest positive distance from the plane, and add it to the list of vertices. If, after all of the triangular surface elements have been checked, no new vertices have been added to the list, exit. Otherwise, go to step 2

This algorithm generalizes to higher dimensional spaces in a straightforward manner. For the two-dimensional case, there are more elegant solutions, such as those presented in [15].

4.3.3 Simulations

A third set of simulations was performed, this time using a general, second-order, two-input nonlinearity:

$$y(t) = c_{11}x_1(t) + c_{12}x_2(t) + c_{21}x_1^2(t) + c_{2c}x_1(t)x_2(t) + c_{22}x_2^2(t) \tag{4.37}$$

This nonlinearity was chosen because its output includes all three types of terms: odd/odd, odd/even and even/even, allowing each of the methods for identifying a general multiple-input Wiener system to be tested. The coefficients, c_{11} through c_{22}, were adjusted such that each of the five terms on the right of (4.37) had equal variance. Thus, we compared the three algorithms under similar conditions.

The linear elements were the same as in the previous simulations. They were both driven by 10,000 point sequences of white Gaussian noise. Noise was added to

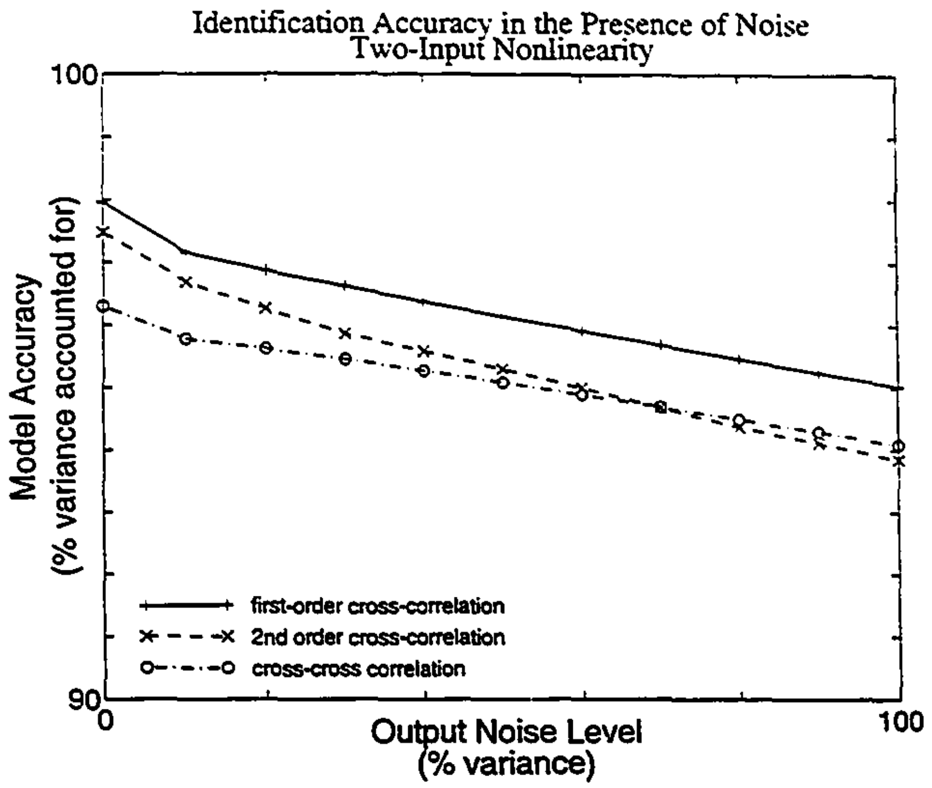


Figure 4.8: Performance of the three algorithms for generalized systems. The inputs were 10,000 point records of Gaussian White Noise.

the output at levels between 0 and 100% of the output variance, producing signal to noise ratios between infinity, and 0dB. Figure 4.8 shows the noise performance of each method. As in the previous simulation, the model accuracy is reported as the percentage of the output variance accounted for by the model, using an independent set of input-output data for the validation.

In this experiment, the performance of the three algorithms was similar, although the odd/even method, which is based on first-order input-output cross-correlations, performed slightly better than the other two methods, which are based on second-order correlations, at all noise levels. Furthermore, from Figure 4.8, it appears that the odd/even method is slightly more noise resistant than either the even/even or the odd/odd method.

Clearly, the performance of the various methods depends on the relative contributions made by the different terms in the nonlinearity. Hence, for an unknown system, one of the methods could yield substantially better results than the other two, simply because of the type of nonlinearity in the system.

4.4 Summary

In this chapter, we have developed a set of algorithms which, taken together, are capable of identifying any multiple-input Wiener structured system. Given its superior noise performance, and more modest computational requirements, we suggest using the "odd" algorithm first. Should this fail, or should the nonlinearity be dominated by even terms, one of the "even" algorithms should be used to identify the system.

Using simulations, we have demonstrated that the these methods are robust in the presence of output noise, and that they do not require a white input, although the input must contain significant power over the bandwidth of the linear elements. Furthermore, we have shown that these techniques can be applied when the input is a low-pass filtered PRBS signal, which is neither Gaussian nor white.

The algorithms developed in this chapter are all included in the MATLAB toolbox for nonlinear identification, described in Appendix A.

Chapter 5

Optimized Identification of Parallel Cascade Models

5.1 Overview

In this chapter, we develop an algorithm that produces a parallel Wiener cascade model of a nonlinear system. We start with single-input systems, and show how the algorithm proposed by Korenberg [56] can be optimized, resulting in a system description which is unique, given the statistics of the input sequence. We demonstrate that this optimization results in simpler models, faster convergence, and better noise performance than the original single-slice version of the algorithm. Furthermore, we demonstrate a practical method for modelling the dynamics present in the third (and potentially higher) order Volterra kernels.

In the multiple-input case, we show how multiple-input Wiener cascades can be used to form a parallel cascade expansion. We also show that using the techniques developed in the previous chapter results in an optimization analogous to that developed for single input systems.

Digital simulations are used to demonstrate the capabilities of these new algorithms. Applications to physical systems will be shown in Chapter 6.

5.2 Single Input Systems

In this section, we will consider the estimation of a single-input parallel cascade model, as proposed by Korenberg [56] and reviewed in Section 2.3.3. Figure 5.1, a repeat of Figure 2.9, illustrates the central idea behind the parallel cascade method. A Wiener system is fit between the input and output of a nonlinear system, and the prediction error, also called the output residuals, is calculated. The input and residuals are then considered to be the input and output of a new nonlinear system, which is then approximated by a second Wiener cascade. In this way, a sum of Wiener cascades, whose input/output behaviour converges to that of the unknown system, is assembled.

Clearly, the rate of convergence of this expansion, as well as whether or not it converges at all, depends entirely on the methods used to fit the Wiener cascades. Korenberg [56] proved convergence for the parallel Wiener cascade and used the proof, which is constructive in nature, as the basis for the single-slice implementation of the parallel cascade method.

The first Wiener cascade had linear dynamics which were estimated from the first-order cross-correlation between the input and output. Korenberg [51, 56] proved that this pathway reduced the first-order correlation between the input and the residuals to zero.

Korenberg [56] then showed that the second-order input-residual cross-correlation could be driven to zero using a finite number of Wiener cascades. The linear element of each of these cascades had an IRF which was equal to a randomly chosen single slice of the second-order input-residual cross-correlation, with a randomly weighted impulse added at the position corresponding to the diagonal in the whole cross-correlation function (see Equation 2.28). To prove convergence, the randomly chosen weights were required to tend to zero with the variance of the residuals.

This procedure was then generalized for higher-order correlations. Paths based on single-slices of the third-order correlation required two randomly weighted impulses: one on each diagonal (i.e. each point in the correlation function for which at least 2

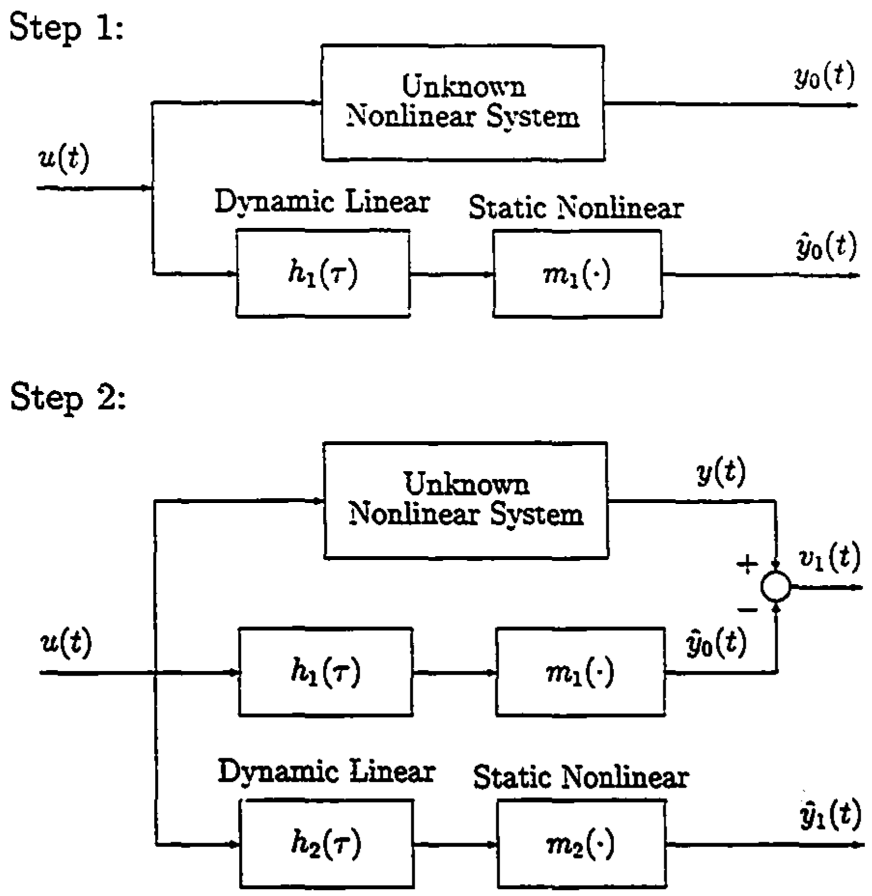


Figure 5.1: The parallel cascade method for nonlinear system identification.

1. Fit a Wiener cascade between the input and output of the nonlinear system.
2. Subtract the output of the first cascade from that of the unknown system, generating the output residuals. Fit a Wiener cascade between the input, and the output residuals.

of the 3 indices are equal). A finite number of these paths could then be used to drive the third-order cross-correlation to zero. Similarly, paths based on the n'th order correlation required $n-1$ such impulses.

In this chapter, we will derive an optimized variant of the parallel cascade method. As in the original single slice method [56], correlations between the input and the residuals will be reduced to zero, starting with the first-order cross-correlation. Once it has been reduced to zero, the second-order cross-correlation, followed in turn by any higher-order correlation functions, will be driven to zero.

Our objective will be to find the path that is optimal, in some sense, at each stage in the iteration. First, we must define optimality in the context of a parallel cascade identification. At each stage in the analysis, the procedure will attempt to reduce the magnitude of the lowest-order non-zero input-residual cross-correlation. Therefore, at least in a local sense, the optimal path will be that which contributes the most to this correlation function, while introducing no lower order correlations between the input and residuals.

We will start by showing how a single Wiener cascade path can be used to reduce both the residual mean and the first-order cross-correlation between the input and the output residuals to zero. We shall then show that a maximum of T paths, where T is the memory length of the system, are necessary to reduce the second-order cross-correlation between the input and the output residuals to zero. Finally, we shall develop new methods for estimating additional paths based on the third- and, in principle, higher-order input-residual cross-correlation functions.

5.2.1 First Order Correlation Function

The linear element of the first Wiener cascade, h, may be calculated from the first-order cross-correlation between the input, $u(t)$, and the output, $y(t)$. If the input is non-white, the exact deconvolution presented in Equation (2.9), or the pseudo-inverse based deconvolution in Equation (3.21), may be used to compute the IRF from the cross-correlation. Then, the output of this linear element, $x(t)$, may be computed by convolving the IRF with the input. The nonlinearity may then be characterized by fitting a polynomial between $x(t)$ and $y(t)$, using a MMSE fitting procedure.

Consider the following lemma:

Lemma 2 *Let $u(t)$ and $y(t)$ be the sampled input and output of a finite dimensional, finite memory nonlinear system. Then, the variance of the residuals:*

$$v(t) = y(t) - \sum_{j=0}^{T-1} h(j)u(t-j) \tag{5.1}$$

where $h(j)$ is computed by solving (2.9), will always be less than or equal to the variance of $y(t)$, with equality only occurring when $\phi_{uy}(k) \equiv 0$.

Proof

Evaluate the variance of $v(t)$:

$$\begin{aligned}
E[v^2(t)] &= E\left[\left(y(t) - \sum_{j=0}^{T-1} h(j)u(t-j)\right)^2\right] \\
&= \sigma_y^2 - 2\sum_{j=0}^{T-1} h(j)E[u(t-j)y(t)] + \sum_{ij=0}^{T-1} h(i)h(j)E[u(t-i)u(t-j)] \\
&= \sigma_y^2 - 2\sum_{j=0}^{T-1} h(j)\phi_{uy}(j) + \sum_{ij=0}^{T-1} h(i)h(j)\phi_{uu}(i-j)
\end{aligned}$$

rewrite this as a matrix equation:

$$\begin{aligned}
E[v^2(t)] &= \sigma_y^2 - 2h^T\phi_{uy} + h^T\Phi_{uu}h \\
&= \sigma_y^2 - 2h^T\phi_{uy} + h^T\Phi_{uu}\Phi_{uu}^{-1}\phi_{uy} \\
&= \sigma_y^2 - h^T\Phi_{uu}h
\end{aligned}$$

Therefore the residual variance is reduced by an amount equal to $h^T\Phi_{uu}h$, which, by Lemma 1, is equal to the variance of the output of the linear filter $h(j)$, and is therefore nonnegative. Furthermore, it can only be zero if the impulse response $h(j)$, and therefore the cross-correlation $\phi_{uy}(j)$, is identically equal to 0 for all lags j. □

Corollary 1 *Thus, using Eq. (2.9) to determine the linear part of a Wiener cascade, and then estimating the static nonlinearity using orthogonal polynomials and a minimum mean squared error fitting technique, results in a cascade that reduces both the residual mean and the first-order cross-correlation between the input and the output residuals to zero.*

Proof

The polynomial representation of the nonlinearity includes a constant term which is orthbogonal to the rest of the nonlinearity. Thus, using a MMSE fitting technique will result in a zero-mean residual.

Since the cross-correlation across a compound system, made up of the difference of the original system and the Wiener cascade, must be proportional to the cross-correlation evaluated across the original system, any further path added to the model, which is based on the first-order input-residual correlation, will have a linear element that is proportional to that of the first path. Because the first nonlinearity was fitted using a MMSE technique and the outputs of the two linear systems are proportional to each other, no further reduction in the residual variance is possible. From Lemma 2, we can see that the only remaining solution is that the residual cross-correlation itself is zero. Clearly, if another path is added to the model, the resulting residuals must remain uncorrelated with the input. □

This result is originally due to Korenberg [51]. It has been included here for continuity, and to establish the form of the argument, which will be used in later proofs.

5.2.2 Second-Order Correlations

The residual mean and the first-order correlation between the input and the residuals will be zero after fitting the first pathway. Thus, any dynamics remaining in the system must be inferred from higher order correlation functions. In this section, we will consider how the second-order cross-correlation can be used for this. We will assume that $u(t)$ and $y(t)$ are the input and output of a nonlinear system, where the output, $y(t)$, is zero-mean and the first-order input-output cross-correlation, $\phi_{uy}(\tau)$, is zero.

Theorem 1 *Let $\phi_{uuy}(\tau_1, \tau_2)$ be the second-order cross-correlation between the input, $u(t)$, and the output, $y(t)$, of a nonlinear system. Assume that $u(t)$ is a Gaussian noise sequence of arbitrary colour, $y(t)$ is zero mean, and the first-order cross-correlation between $u(t)$ and $y(t)$ is zero. Let $v(t)$ be the residuals:*

$$v(t) = y(t) - k\left(\sum_{\tau=0}^{T-1} h(\tau)u(t-\tau)\right)^2 + kh^T\Phi_{uu}h \tag{5.2}$$

for some filter, h, and gain, k. Then, the variance of the residuals, $v(t)$, is minimized when h is the principal generalized eigenvector of the pencil (ϕ_{uuy}, Φ_{uu}).

proof

Let $x(t)$ be the output of $h(\tau)$. By Lemma 1, the variance of $x(t)$ is:

$$\sigma_x^2 = h^T\Phi_{uu}h$$

Thus, the residual, $v(t)$, is zero mean. Its variance can be written:

$$\begin{aligned}
\sigma_v^2 &= E[v^2(t)] \\
&= E\left[(y(t) - kx^2(t) + k\sigma_x^2)^2\right] \\
&= E[y^2] - 2kE[x^2y] + 2k\sigma_x^2E[y] + k^2E[x^4] - 2k^2E[x^2]\sigma_x^2 + k^2\sigma_x^4
\end{aligned}$$

Recall that y is zero mean, by assumption. Using Equation (3.1), $E[x^4] = 3\sigma_x^4$. Therefore:

$$\sigma_v^2 = \sigma_y^2 - 2kE[y(t)x^2(t)] + 2k^2\sigma_x^4$$

However, we may expand $E[y(t)x^2(t)]$ as follows:

$$\begin{aligned} E[y(t)x^2(t)] &= E\left[y(t)\sum_{\tau_1,\tau_2=0}^{T-1} h(\tau_1)h(\tau_2)u(t-\tau_1)u(t-\tau_2)\right] \\ &= \sum_{\tau_1,\tau_2=0}^{T-1} h(\tau_1)h(\tau_2)E[y(t)u(t-\tau_1)u(t-\tau_2)] \\ &= h^T\phi_{uuy}h \end{aligned}$$

Thus:

$$\sigma_v^2 = \sigma_y^2 - 2kh^T\phi_{uuy}h + 2k^2(h^T\Phi_{uu}h)^2 \tag{5.3}$$

Let (λ_j, g_j) be the generalized eigenvalues and eigenvectors of the matrix pencil (ϕ_{uuy}, Φ_{uu}). Since ϕ_{uuy} is a real symmetric matrix [68], and Φ_{uu} is positive definite (Lemma 1), the pencil is symmetric definite. Thus, for any two generalized eigenvectors, g_i and g_j:

$$g_i^T\Phi_{uu}g_j = \delta_{ij}$$

where δ_{ij} is the Kronecker delta.

Express h as a sum of the eigenvectors:

$$h = \sum_{j=1}^{T} a_j g_j$$

and, due to the presence of the gain k in (5.2), let:

$$\sum_{j=1}^{T} a_j^2 = 1$$

Differentiate (5.3) with respect to one of the weights, a_i:

$$\frac{\partial\sigma_v^2}{\partial a_i} = -4k(h^T\phi_{uuy}g_i) + 8k^2(h^T\Phi_{uu}h)(h^T\Phi_{uu}g_i)$$

But $\phi_{uuy}g_i = \lambda_i\Phi_{uu}g_i$, and $h^T\Phi_{uu}g_i = a_i$. Thus:

$$\frac{\partial\sigma_v^2}{\partial a_i} = -4ka_i(\lambda_i - 2k)$$

Optimizing, we see that either $k = 0$, $a_i = 0$, or $k = \frac{\lambda_i}{2}$. Thus, h must be one of the g_i, in which case the residual variance becomes:

$$\begin{aligned}\sigma_v^2 &= \sigma_y^2 - 2\left(\frac{\lambda_i}{2}\right) g_i^T \phi_{uuy} g_i + 2\left(\frac{\lambda_i}{2}\right)^2 \\ &= \sigma_y^2 - \frac{\lambda_i^2}{2}\end{aligned}$$

Thus, the variance is reduced by $\frac{\lambda_i^2}{2}$. The maximum reduction in the residual variance is achieved by choosing h to be the eigenvector associated with the largest generalized eigenvalue. □

Theorem 2 *Let $u(t)$ and $y(t)$ be the input and output of a nonlinear system, where $u(t)$ is an arbitrarily coloured, zero-mean Gaussian sequence, $y(t)$ is zero-mean, and the first-order cross-correlation between $u(t)$ and $y(t)$ is zero. Let (λ_i, g_i) be the generalized eigenvalues and eigenvectors of (ϕ_{uuy}, Φ_{uu}). Let $x(t)$ be the convolution:*

$$x(t) = \sum_{\tau=0}^{T-1} g_k(\tau) u(t-\tau)$$

for some generalized eigenvector g_k, and let $m(\cdot)$ be a polynomial fitted between $x(t)$ and $y(t)$, using a MMSE technique. Let $v(t) = y(t) - m(x(t))$.

Then, for $i \neq k$, (λ_i, g_i) are generalized eigenvalues and eigenvectors of (ϕ_{uuv}, Φ_{uu}), and g_k is in the null-space of ϕ_{uuv}.

Proof

First, use the MMSE second-order polynomial nonlinearity, $m(z) = \frac{\lambda_k}{2} z^2 - \frac{\lambda_k}{2}$, which was derived in Theorem 1. In that case, the second-order cross-correlation between the input and the residuals, $v(t)$, is:

$$\phi_{uuv} = \phi_{uuy} - \lambda_1 \Phi_{uu}(g_1 g_1^T)\Phi_{uu}$$

Multiplying by one of the generalized eigenvectors of ϕ_{uuy}:

$$\begin{aligned}\phi_{uuv} g_i &= \phi_{uuy} g_i - \lambda_k \Phi_{uu} g_k g_k^T \Phi_{uu} g_i \\ &= \lambda_i \Phi_{uu} g_i - \lambda_k \Phi_{uu} g_k \delta_{ki}\end{aligned}$$

where δ_{ki} is a Kronecker delta. Thus, the generalized eigenvectors of (ϕ_{uuv}, Φ_{uu}) are equal to those of (ϕ_{uuy}, Φ_{uu}), except that λ_k, the eigenvalue associated with the IRF of the linear part of the Wiener system, has been reduced to zero.

If we replace the second-order polynomial nonlinearity with a higher order polynomial, the second-order cross-correlation across the Wiener path will be scaled. Thus, if we recompute the residuals, $v(t)$, using a different nonlinearity, the g_i will continue to be the generalized eigenvectors of (ϕ_{uuy}, Φ_{uu}), and the only eigenvalue affected by the change will be λ_k.

If, however, the new nonlinearity is fitted using a MMSE technique, then the principal eigenvalue of (ϕ_{uuy}, Φ_{uu}) will be reduced to zero, just as it was by the second-order polynomial nonlinearity. If this were not the case, $\phi_{uuv} g_k = \lambda \Phi_{uu} g_k$, and Theorem 1 would guarantee a reduction in the residual variance of at least $\frac{\lambda^2}{2}$, which is clearly not possible, since the nonlinearity was fitted using a MMSE technique. Therefore g_k must be in the null-space of ϕ_{uuv}. $\square$

Note that if the input, $u(t)$, is white, then $\Phi_{uu} = \sigma_u^2 I$. Thus, the generalized eigenvalue problem is replaced with an ordinary eigenvalue problem. Using the principal eigenvalue of ϕ_{uuy} results in the greatest reduction possible in residual variance, when the nonlinearity is restricted to second-order polynomials, as in the non-white case. Furthermore, such a path results in the maximum reduction in the Frobenius norm [20] of the second-order cross-correlation between the input and the residuals.

5.2.2.1 Proposed Algorithm

We propose the following algorithm to build a parallel cascade model between $u(t)$ and $y(t)$. We assume that $y(t)$ is zero-mean, and that $\phi_{uy} \equiv 0$. This will be the case for any system once the output of a Wiener path based on the first-order cross-correlation between $u(t)$ and $y(t)$ has been removed from the output.

1. Compute Φ_{uu}, a symmetric Toeplitz matrix whose first row is an estimate of the input autocorrelation.

2. Initialize the residuals, $v_0(t) = y(t)$, and set $k = 1$.

3. Compute $\phi_{uuv_{k-1}}$, an estimate of the second-order cross-correlation between $u(t)$ and $v_{k-1}(t)$.

4. Solve the generalized eigenvalue problem: $\phi_{uuv_{k-1}} g = \lambda \Phi_{uu} g$, and let $h_k = g_1$, the generalized eigenvector associated with the largest eigenvalue (in absolute value).

5. Compute $x(t)$, the convolution of h_k with $u(t)$.

6. Fit a high-order polynomial, $m_k(\cdot)$, between the IRF output, $x(t)$, and the current residuals, $v_{k-1}(t)$.

7. Calculate the next set of residuals: $v_k(t) = v_{k-1}(t) - m_k(x(t))$

8. Either exit, or set $k = k + 1$ and return to step 3

Each iteration adds one vector to the null-space of the second-order correlation between the input and the residuals. Thus, after T iterations the second-order cross-correlation will be reduced to zero. Furthermore, the impulse responses returned by this procedure will be equal to the generalized eigenvectors of ϕ_{uuy}, and they will be returned in decreasing order of significance. Thus, for a given system, this algorithm will lead to a model which depends only on the input auto-correlation function.

5.2.2.2 Benefit of Iterative Computation

Given the discussion surrounding the algorithm given in Section 5.2.2.1, one must ask why we compute the paths from the second-order cross-correlation between the input and the current set of output residuals, rather than simply computing the cross-correlation once, and then basing the cascade paths on the eigenvectors of that cross-correlation matrix. To answer this question, we must consider the effects of estimation errors on the cross-correlation matrix and, in particular, on its eigenvalues and eigenvectors. To that end, we present the following lemma.

Lemma 3 *Let $u(t)$ and $y(t)$ be zero-mean signals, and let h be a generalized eigenvector of (ϕ_{uuy}, Φ_{uu}), with eigenvalue λ. If*

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

Then:

$$\phi_{uw} = \lambda \Phi_{uu} h$$

Proof

Because h is a generalized eigenvalue of (ϕ_{uuy}, Φ_{uu}):

$$\lambda \Phi_{uu} h = \phi_{uuy} h$$

Expanding the product $\phi_{uuy}h$:

$$\begin{aligned}
\sum_{\tau_2=1}^{T} \phi_{uuy}(\tau_1,\tau_2)h(\tau_2) &= \sum_{\tau_2=1}^{T} h(\tau_2)E[u(t-\tau_1)u(t-\tau_2)y(t)] \\
&= E\left[u(t-\tau_1)\sum_{\tau_2=1}^{T} h(\tau_2)u(t-\tau_2)y(t)\right] \\
&= E[u(t-\tau_1)w(t)] \\
&= \phi_{uw}(\tau_1) \qquad \square
\end{aligned}$$

Thus, instead of examining the effects of measurement noise on the estimate $\hat{\phi}_{uuy}$, we need only consider the estimation error in the first-order cross-correlation $\hat{\phi}_{uw}$. From Equation (3.3), we see that:

$$\text{Var}\left[\hat{\phi}_{uw}(\tau)\right] \approx \frac{1}{N}\int_{-N}^{N}\left(\phi_{uu}(\xi)\phi_{ww}(\xi) + \phi_{uw}(\xi+\tau)\phi_{wu}(\xi-\tau)\right)d\xi$$

The second term depends only on the value of the cross-correlation, ϕ_{uw}, and is not affected by the iterative computation. Consider ϕ_{ww}, which appears in the first term.

$$\phi_{ww}(\tau) = E[x(t)x(t-\tau)y(t)y(t-\tau)]$$

where $x(t)$ is the convolution of h and $u(t)$. We can see that the estimation error in the impulse response depends in part on the variance of the output, $y(t)$. Thus, by performing the calculation iteratively, we reduce the output variance as far as possible prior to the calculation of each path, and therefore get the best possible estimate of each path in the model.

5.2.2.3 Choosing the Final Path

At some point, the relative noise and signal powers present in the residuals will be such that subsequent paths will be dominated by estimation error, and will therefore corrupt the model. As the optimal path is selected at each stage, once this point is reached, no further paths can be based on the second-order cross-correlation. Consequently, we will develop a test to determine when this point has been reached.

To determine whether or not the current path is primarily due to signal or noise, we will examine its associated eigenvalue in the second-order cross-correlation matrix. If the magnitude of its eigenvalue is consistent with the hypothesis that the residuals contain only noise, we will reject the path, and conclude that no further paths can be based on the second-order cross-correlation.

First, consider the perturbation of the eigenvalues of a Hermetian matrix, which is perturbed by a Hermetian perturbation. From [89], we see that the maximum perturbation in the principal eigenvalue of $\hat{\phi}_{uuz} = \hat{\phi}_{uuy} + \hat{\phi}_{uun}$ is less than the 2-norm of $\hat{\phi}_{uun}$. Our task, then, is to estimate the 2-norm of the perturbation, $\hat{\phi}_{uun}$, using known quantities.

We will need to know the second-order statistics of the entries of the perturbation matrix, $\hat{\phi}_{uun}$. The expected value of the perturbation is zero, since u and n are independent. If n is white, the perturbation variance can be written:

$$\begin{aligned}
E[\hat{\phi}^2_{uun}(i,j)] &= E\left[\frac{1}{N^2}\sum_{s,t=1}^{N} u(t-i)u(t-j)n(t)u(s-i)s(s-j)n(s)\right] \\
&= \frac{1}{N^2}\sum_{s,t=1}^{N} E[u(t-i)u(t-j)u(s-i)u(s-j)]E[n(s)n(t)] \\
&= \frac{\sigma_n^2}{N}\left[2\phi^2_{uu}(i-j) + \phi^2_{uu}(0)\right]
\end{aligned}$$

Thus, aside from points near the diagonal, the variance of the perturbation elements is $\frac{\sigma_v^2 \sigma_u^4}{N}$.

From [47], we have the following matrix inequality:

$$\| A \|_2 \leq \sqrt{n} \max_j \| a_j \|_2 \tag{5.4}$$

where $A[a_1 \ a_2 \ \ldots \ a_n]$ is an m by n matrix. Due to length of the data records and the central limit theorem, the elements of the perturbation matrix must have nearly Gaussian elements. Therefore, the squared 2-norms of its columns will have chi-squared distributions, with T degrees of freedom.

Thus, to continue adding paths based on the second-order cross-correlation to the model, we will require the principal eigenvalue of the cross-correlation matrix to exceed the following threshold:

$$\hat{\lambda}_1^2 \geq \frac{\chi\sqrt{T}\hat{\sigma}_u^4 \hat{\sigma}_v^2}{N} \tag{5.5}$$

where χ is the "chi-squared" level for T degrees of freedom at a some chosen confidence level. Given that Equation (5.5) was derived using a fairly loose upper bound, it will be quite conservative, and may reject paths that are still dominated by the system dynamics. Simulations, presented in Section 5.2.4.5, will be used to demonstrate the efficacy of this technique, as well as to establish an appropriate value for the confidence level, which in turn will determine χ and the threshold.

Two comments are in order. Firstly, we do not have access to the noise signal, v, so the variance of the current residuals must be used in this test, making it even more conservative. Secondly, this threshold is not based on the residuals that result from the addition of the current path to the model, and therefore is not influenced by any errors in the nonlinearity estimation.

5.2.2.4 Use of a Pseudo-inverse

In Chapter 3, we used a pseudo-inverse based deconvolution operation to remove noise from IRFs estimated using coloured input signals. In this section, we will examine how this technique can be applied to IRF estimates obtained by solving a generalized eigenvalue problem involving the second-order input-output cross-correlation. ϕ_{uuy}, and the Toeplitz structured input auto-correlation matrix, Φ_{uu}.

To apply the pseudo-inverse based deconvolution, we replace h with $\Phi_{uu}^{\ddagger}\Phi_{uu}h$, where $\Phi_{uu}^{\ddagger}$ is a pseudo-inverse of Φ_{uu}. This is the projection of h onto the subspace spanned by V_1, the singular vectors of Φ_{uu} which correspond to significant singular values. To achieve this, we must choose the order of the pseudo-inverse, and hence the dimension of the space spanned by V_1, appropriately.

Let $x(t)$ be the convolution:

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

and $w(t)$ is the product $x(t)z(t)$. From Lemma 3, we see that if (h, λ) are a generalized eigenvector and eigenvalue of $(\hat{\phi}_{uuy}, \hat{\Phi}_{uu})$, then they are also solutions to:

$$\phi_{uw} = \lambda \Phi_{uu} h$$

which we shall call the "equivalent linear problem". In effect, in forming $w(t)$, we have increased the order of all of the nonlinearities in the system by one, transforming an even system into an odd system.

Once we have estimated h from the generalized eigenvalue problem, we compute $x(t)$ and $w(t)$. A polynomial, $\hat{m}_w(\cdot)$ is fitted between them and the residuals:

$$v_w(t) = w(t) - \hat{m}_w(x(t))$$

are computed. We now apply the algorithm described in Section 3.2.1.1, using ϕ_{uw} and $v_w(t)$ as the cross-correlation and noise, to determine the correct order for the pseudo-inverse, $\Phi_{uu}^{\ddagger}$. We then use $\Phi_{uu}^{\ddagger}\Phi_{uu}h$ as the IRF of the Wiener cascade.

5.2.3 Third-Order Correlations

Consider a nonlinear system, with input $u(t)$ and output $y(t)$. Assume that we have used the parallel cascade method to drive the first and second-order cross-correlations between the input and the output residuals to zero. Let $v(t)$ represent the current output residuals.

As there is no information left in the first and second-order correlations, we must estimate the linear element of the next cascade from the third-order correlation between the input and the output residuals. Given N data points, a biased estimate of the third-order cross-correlation between $x(t)$ and $v(t)$ is:

$$\hat{\phi}_{uuuv}(i,j,k) = \frac{1}{N^3}\sum_{n=1}^{N} u(n-i)u(n-j)u(n-k)v(n) \tag{5.6}$$

Let $h(i)$ be the impulse response of the next Wiener cascade. The static nonlinearity in this cascade will be chosen such that the first and second-order correlations between the cascade input and output are zero. In this case, the third-order cross-correlation function across this system can be written:

$$\phi_{uuuv}(i,j,k) = ch(i)h(j)h(k) \tag{5.7}$$

where c is the third-order term in the polynomial representation of the static nonlinearity.

We will choose $h(i)$ such that the mean squared error (MSE) between the third-order cross-correlations measured across the cascade (5.7) and estimated from the data (5.6) is minimized. The MSE is:

$$\frac{1}{T^3}\sum_{i,j,k=1}^{T}\left(\hat{\phi}_{uuuv}(i,j,k) - ch(i)h(j)h(k)\right)^2 \tag{5.8}$$

We will attempt to find $h(i)$ using a gradient search procedure. Assuming that we have an initial guess for $h(i)$, the gradient can be written:

$$\begin{aligned} g(i) &= \frac{\partial MSE}{\partial h(i)} \\ &= \frac{-6}{T^3}\left\{\sum_{j,k=1}^{T}\hat{\phi}_{uuuv}(i,j,k)h(i)h(j)h(k) - h(i)\left(\sum_{j=1}^{T}h^2(j)\right)\right\} \end{aligned}$$

We can also compute the best distance to go in the direction of the gradient. If we replace our guess $h(i)$ with $h(i)+\alpha g(i)$, and optimize with respect to α, we get a fifth degree polynomial equation:

$$A\alpha^5 + B\alpha^4 + C\alpha^3 + D\alpha^2 + E\alpha + F = 0 \tag{5.9}$$

with coefficients:

$$\begin{aligned}
A &= -3[g,g]^3 \\
B &= -15[g,g]^2[g,h] \\
C &= -24[g,g][g,h]^2 - 6[h,h][g,g]^2 \\
D &= 3\sum_{i,j,k=1}^{T} \hat{\phi}_{uuuv}(i,j,k)(g(i)g(j)g(k) - 18[h,h][h,g][g,g] - 12[h,g]^3 \\
E &= 6\sum_{i,j,k=1}^{T} \hat{\phi}_{uuuv}(i,j,k)g(i)g(j)h(k) - 12[h,h][g,h]^2 - 3[h,h]^2[g,g] \\
F &= 3\sum_{i,j,k=1}^{T} \hat{\phi}_{uuuv}(i,j,k)h(i)h(j)g(k) - 3[h,h]^2[h,g]
\end{aligned} \tag{5.10}$$

where $[h,g]$ is taken to be the inner product between the vectors $h(i)$ and $g(i)$. Equation (5.9) will either have 1,3 or 5 real solutions, which correspond to the local extrema of the MSE along the gradient vector. As $\alpha = \pm\infty$ correspond to maxima in the MSE, only the odd numbered roots need be considered. The best of these is used as the starting point for the next iteration.

This gradient iteration can be applied repeatedly, until no further reduction in the MSE is obtained. Furthermore, it can be shown that when the gradient vanishes, the vector $h(i)$ will have the following property, which is in some sense analogous to it being an eigenvector.

$$\sum_{i,j=1}^{T} \hat{\phi}_{uuuv}(i,j,k)h(i)h(j) = \lambda h(k) \tag{5.11}$$

This result can be used to test the convergence of the iteration, and determine when a (potentially local) minimum has been reached.

Clearly, a similar derivation could be used to develop a gradient search algorithm for fourth (or higher) order systems.

5.2.4 Single Input Simulations

We examined the behaviour of these algorithms by simulating a single-input, homogeneous second-order system. This system was completely described by the second-order

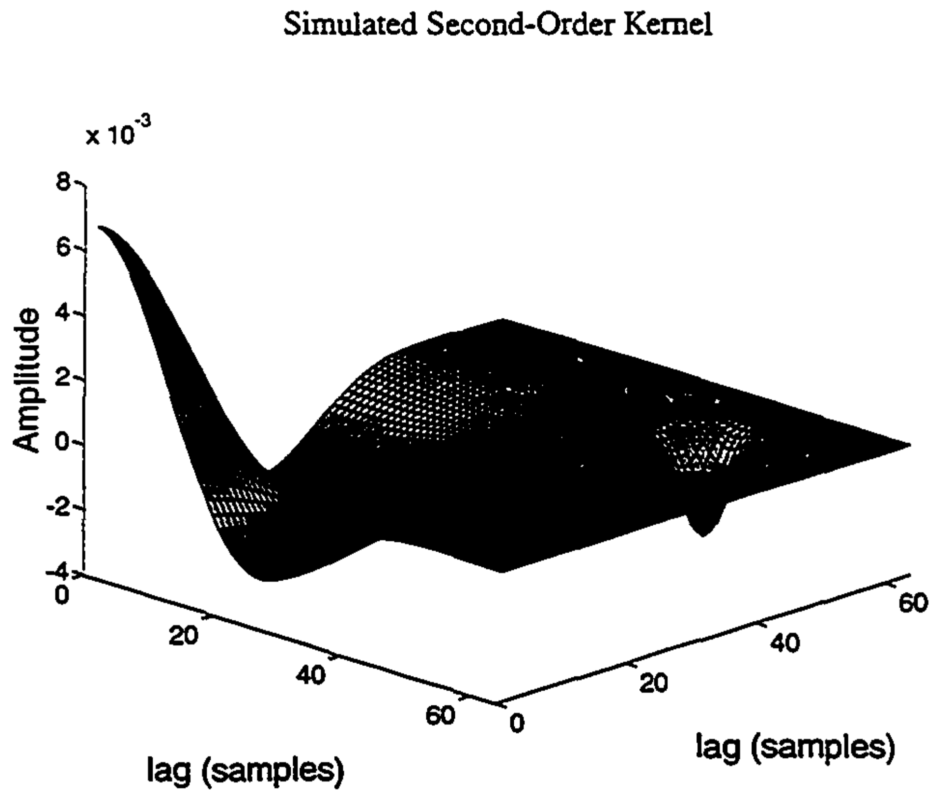
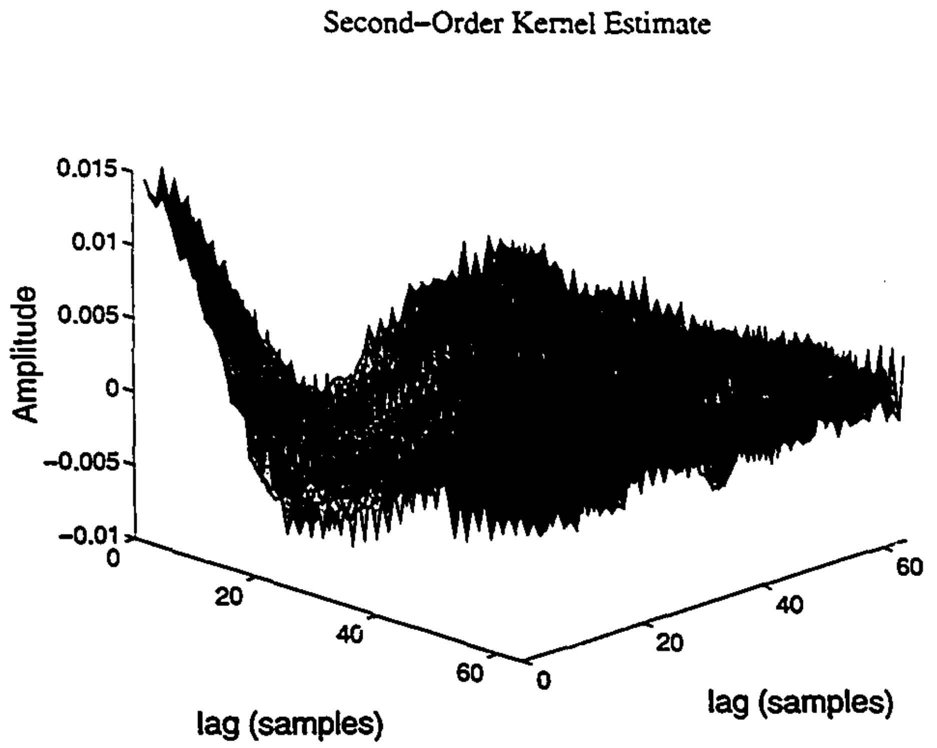


Figure 5.2: Second-order Volterra kernel of the system used in the single-input simulations.

Volterra kernel shown in Figure 5.2. This kernel was produced by placing 3 weighted impulses on a 64 by 64 point grid. "Mirror images" of the impulses were placed across the diagonal, and the resulting kernel smoothed.

This kernel was designed to have several different features of different sizes, so that the quality of the kernel estimates would be readily evident from a visual inspection. The largest feature was the peak at $(0, 0)$ lag. This, and the two negative peaks, one along each axis, were well modelled by all methods. The next smallest feature was the negative peak on the diagonal. Finally, the small ridge separating the two negative peaks near the origin provides a feature that is quite difficult to model.

In the first set of simulations, the input was a 10,000 point sequence of white Gaussian noise. The kernel output was generated using a second-order convolution (Equation 2.12).



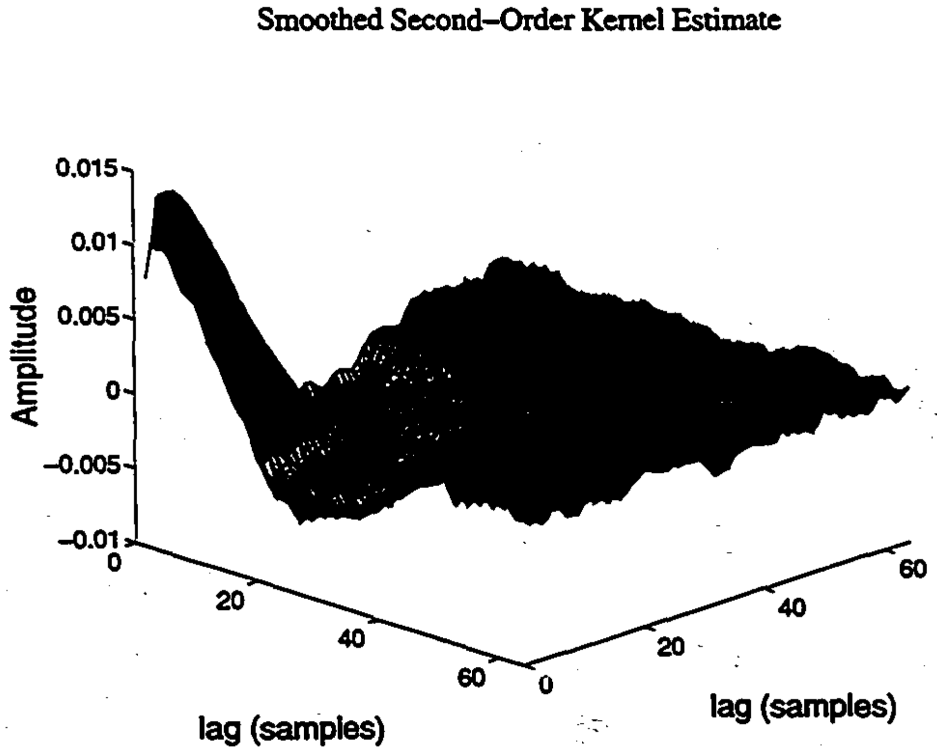


Figure 5.3: Second-order Wiener kernel estimated using the Lee-Schetzen cross-correlation method, before (upper) and after (lower) smoothing

5.2.4.1 Lee-Schetzen Cross-Correlation

First, for comparison purposes, we estimated the second-order Wiener kernel, shown in the upper panel of Figure 5.3, using the Lee and Schetzen cross-correlation method [62]. Even though the input was white Gaussian noise, the cross-correlation estimate was dominated by high frequency noise. This noise could be suppressed, somewhat, using a 9-point smoothing filter. The smoothed kernel estimate is shown in the lower panel of Figure 5.3. Although the noise is much reduced, only the largest of the kernel features are visible.

5.2.4.2 Parallel Cascade Methods

We examined two versions of the parallel cascade method. In the first case, the linear elements of the Wiener paths were based on randomly selected single slices of the second-order cross-correlation function. In the second case, the principal eigenvector of the whole second-order cross-correlation function was used to identify the linear elements.

We first examined convergence speed under noiseless conditions. After each path had been added to the model, the residual variance was computed. This was used to compute the percentage of the output variance that had been accounted for by the model (%VAF, see Equation 3.5). The number of floating point operations (flops) required in the computation of each path was also determined. Convergence speed was evaluated in terms of the number of flops required to reach a given level of model accuracy.

Figure 5.4 plots the model accuracy (%VAF) against the number of flops for the two methods. Initial convergence was faster when the linear dynamics were estimated from single slices of the second-order correlation function, as suggested by Korenberg [56]. This was due to the high cost associated with the computation of each path using the eigenvector method. Both methods required approximately the same number of computations to reach an accuracy of 90%VAF. For higher accuracies, the eigenvector method proved to be faster than the single slice implementation since many fewer

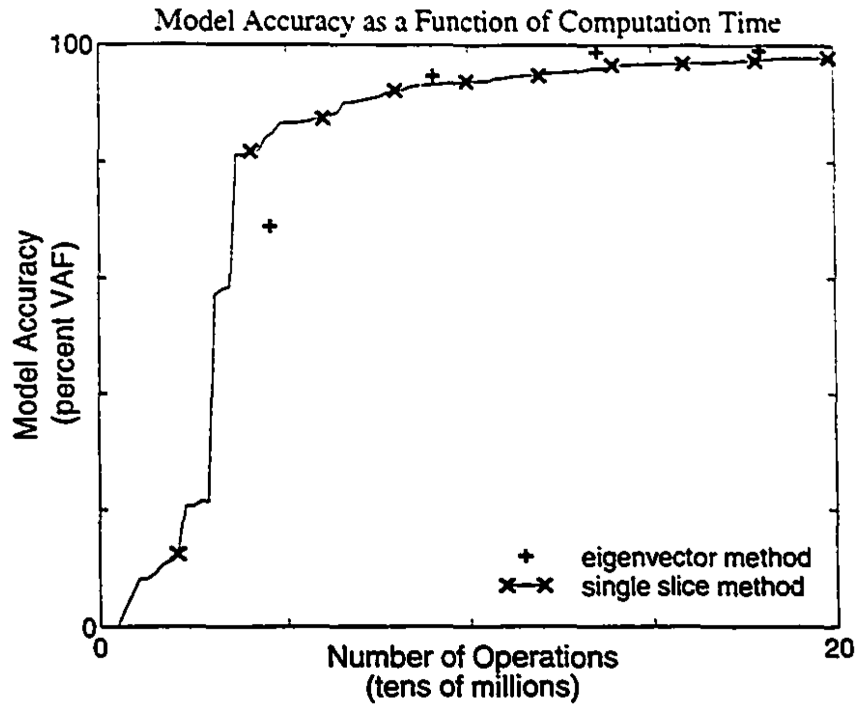


Figure 5.4: Convergence speed for the single slice and eigenvector versions of the parallel cascade method under noiseless conditions. The model accuracy, expressed as the percentage of the output variance accounted for by the model, is plotted as a function of the number of floating point operations (flops) required in the computations. Each + sign represents one pathway added by the eigenvector method. Each cross indicates the addition of 10 pathways by the single-slice algorithm

paths needed to be computed.

Figure 5.5 shows the kernel estimated by the original single slice variant of the parallel cascade method. This kernel accounted for 97.8% of the variance of the output, but contained high frequency estimation noise. Note that at this stage, the single-slice model comprised 90 Wiener cascade paths, and had more parameters than the original second order kernel.

Applying the eigenvector method and fitting 5 Wiener paths resulted in a model which accounted for 99.7% of the output variance. All the features present in the simulated kernel were clearly visible in the model's second-order kernel, shown in Fig. 5.6. Unlike that produced by the single slice implementation, this model had far fewer parameters than the simulated system.

Using the eigenvector method to add an additional 5 paths to the model, resulted

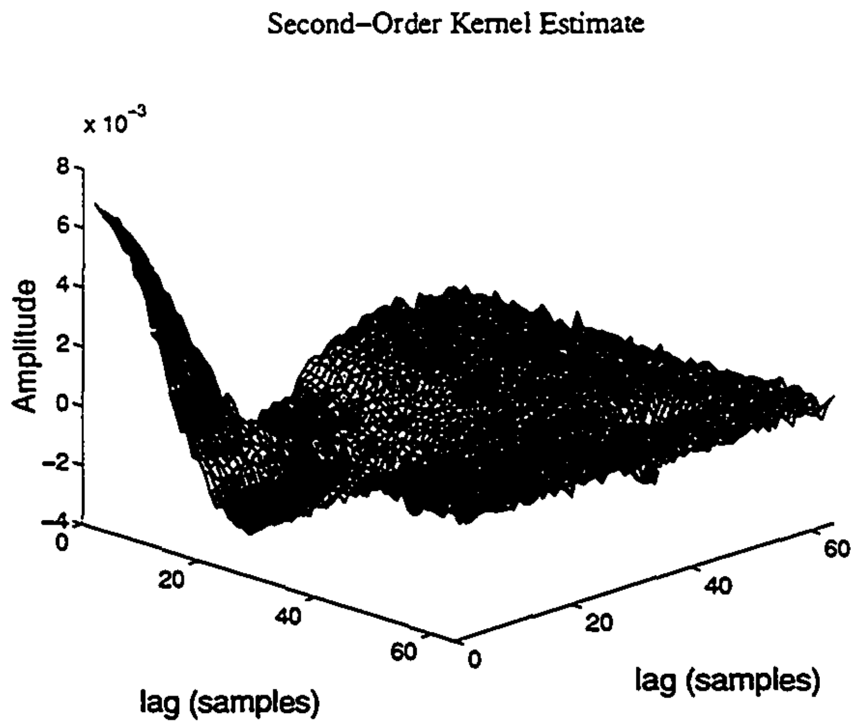


Figure 5.5: Second-order Volterra kernel computed from the first 90 paths identified by the single slice method. It accounts for 97.8% of the output variance.

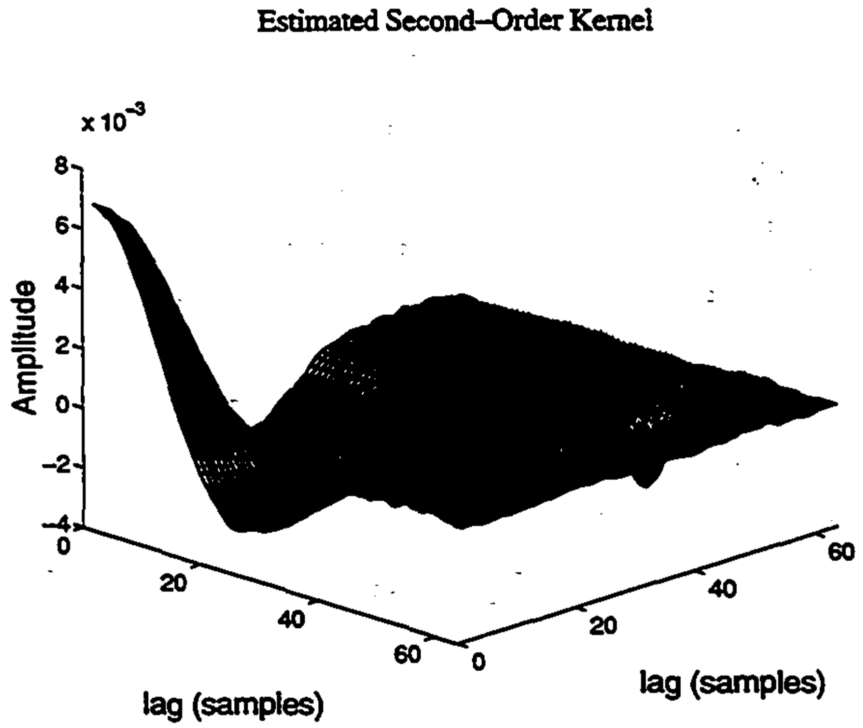


Figure 5.6: Second-order Volterra kernel computed from the first 5 paths identified by the eigenvector method. It accounts for 99.7% of the output variance.

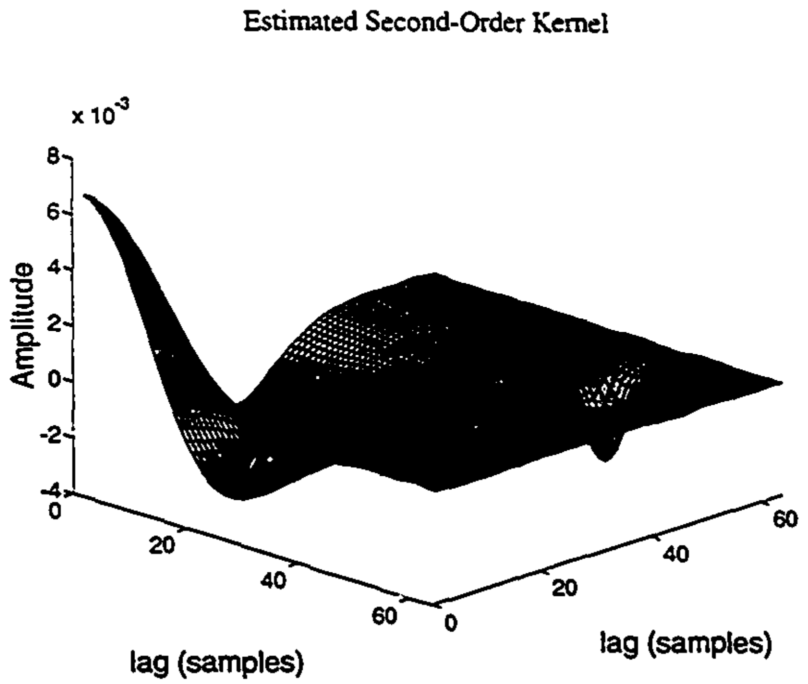


Figure 5.7: Second-order Volterra kernel computed from the first 10 paths identified by the eigenvector method under noiseless conditions. This kernel accounts for 99.97% of the output variance.

in a model that accounted for 99.97% of the output variance. At this stage, the kernel estimate, shown in Figure 5.7, is virtually indistinguishable from the true kernel.

5.2.4.3 Performance Under Noisy Conditions

We then examined the performance of these methods in the presence of output noise. White Gaussian noise was added to the system output, with variances ranging from 10 - 100% of the output signal variance. Both methods were used to identify the system at each noise level. Model accuracy was assessed by measuring the ability of the identified systems to predict the uncorrupted system output. In all cases, as paths were added to the model, the prediction accuracy increased initially, reached a maximum, and then decreased, as subsequent paths began to model the noise present in the residuals rather than the underlying dynamics. Fig. 5.8 shows the maximum model accuracy obtained by each method as a function of the output noise level. It is evident that the eigenvector method produced better estimates of the system

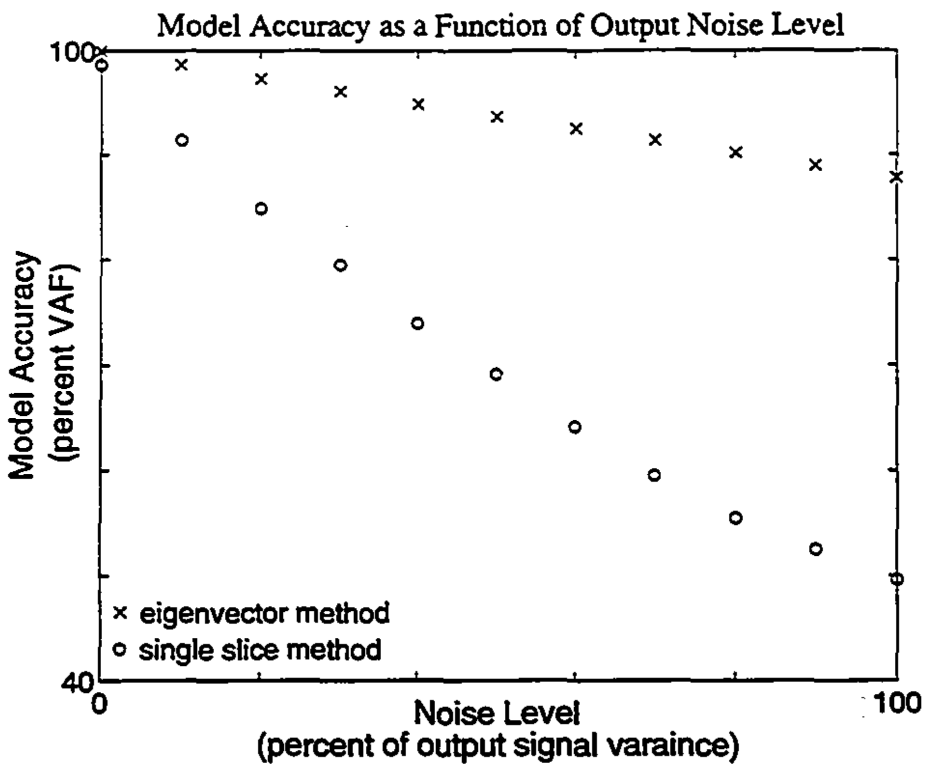


Figure 5.8: The maximum model accuracy obtained as a function of the output noise level. In all cases, model accuracy was assessed in terms of output error with respect to the uncorrupted output signal.

dynamics, at all noise levels. Furthermore, the eigenvector method performed much better at higher noise levels than did the single slice method.

5.2.4.4 Performance Using Band-Limited Inputs

In this section, we present a simulation which shows that when the test input is band-limited kernel estimates obtained from noisy data can be obscured by deconvolution noise, as suggested in Section 5.2.2.4. Furthermore, we will demonstrate the improvements in both the kernel estimates and the prediction accuracy that are realized when the pseudo-inverse deconvolution algorithm is used to suppress the deconvolution noise.

In this simulation, a band-limited limited input signal was generated by filtering a 10,000 point sequence of white Gaussian noise with a fourth order Butterworth low-pass filter with a normalized cut-off of 0.1. This band-limited Gaussian signal

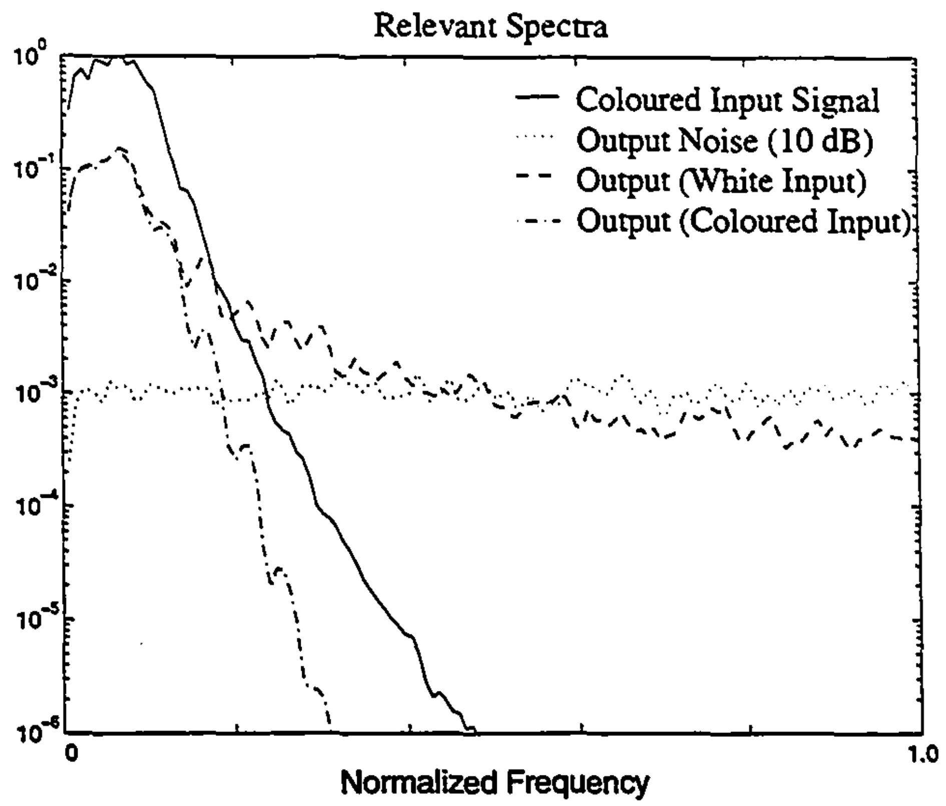


Figure 5.9: Spectra relevant to the band-limited input simulation. The solid line shows the power spectrum of the band-limited test input used during the identification experiment. The dashed and dash-dotted lines show the power spectra of the system output, when the system was driven with a white input and the band-limited test input, respectively. The measurement noise spectrum is plotted as a dotted line.

was then processed by the kernel shown in Figure 5.2. White Gaussian noise was added to the kernel output at a SNR of 10 dB.

Figure 5.9 shows four spectra which are relevant to the simulation. The input spectrum is plotted as a solid line. It is evident that at normalized frequencies of less than 0.15, the output spectrum evoked by a white input (dashed line) and that evoked by the band-limited test input (dash-dotted line) are identical. At higher frequencies the white input evokes more power than the band-limited input, however both soon drop below the noise spectrum (dotted line). Thus the band-limited input appears to excite those dynamic modes present in the system which are visible through the noise. Hence, this input should be adequate for system identification purposes under these conditions.

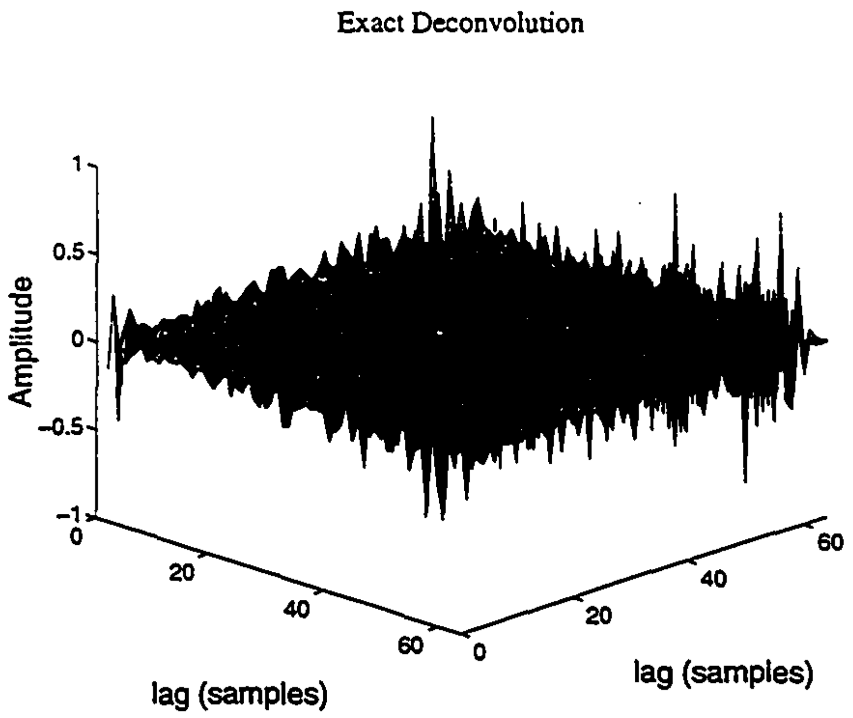


Figure 5.10: Second-order Volterra kernel estimated by applying the optimized parallel cascade method with exact deconvolution to noisy, band-limited data.

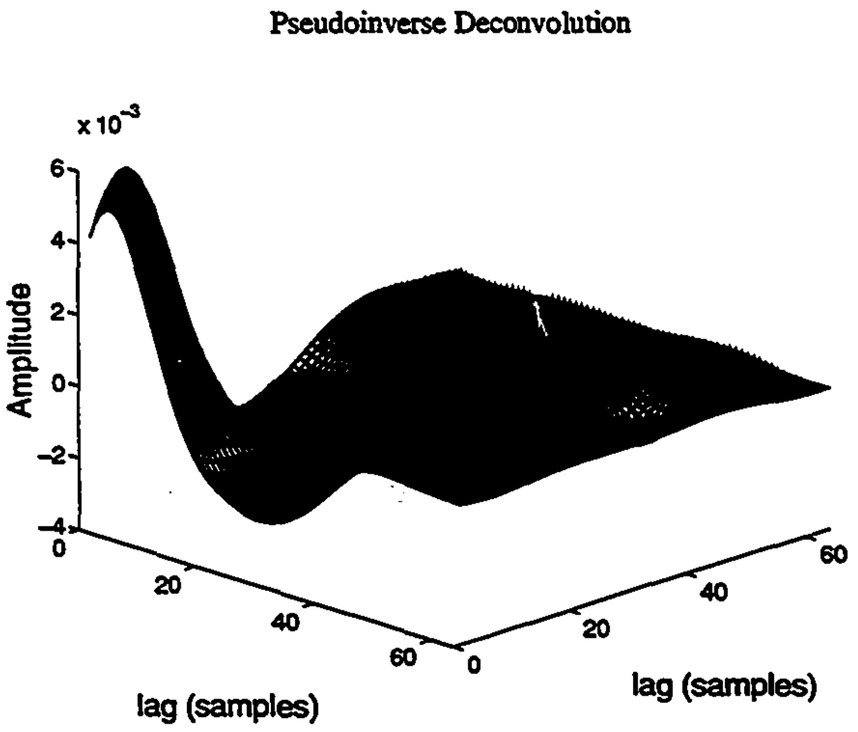


Figure 5.11: Kernel estimated from noisy band-limited data using the pseudo-inverse based deconvolution

As in previous simulations, a parallel Wiener cascade model was identified using the generalized eigenvector algorithm described in Section 5.2.2.1. The resulting kernel, shown in Figure 5.10, was dominated by large amplitude, high frequency noise, which is assumed to be due to the exact deconvolution performed implicitly by the generalized eigenvector algorithm. Despite the presence of this noise, which completely obscures the kernel, this model predicted 98.6% of the variance of the noise-free output.

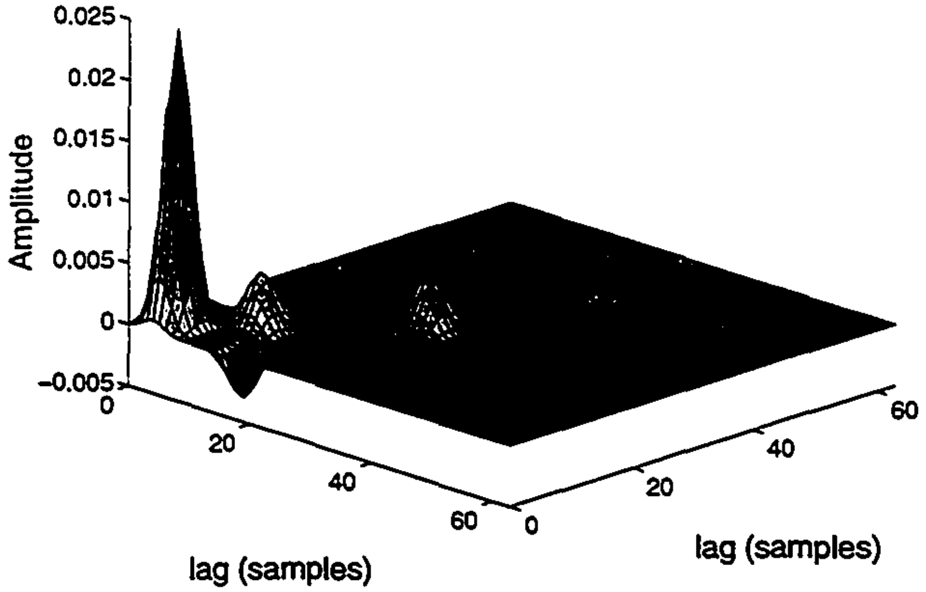
The kernel shown in Figure 5.11 was generated using the generalized eigenvector algorithm in conjunction with the pseudo-inverse based deconvolution method, described in Section 5.2.2.4. Here, the deconvolution noise has been effectively suppressed. This model predicts 99.8% of the noise-free output and 98.3% of the kernel variance, both substantial improvements over the model generated without the pseudo-inverse based deconvolution.

5.2.4.5 Choosing the Final Path

A series of Monte Carlo simulations was performed to determine how well the threshold proposed in Equation (5.5) discriminated paths that contributed information about the system from those that modelled mostly noise. During each run of the Monte Carlo simulation, new system input and output and noise signals were generated. A 20 path Wiener cascade model was then fitted between the input and the noise corrupted output. As each path was estimated, the ratio of the principal eigenvalue to the threshold given in (5.5) was computed, as well as the prediction accuracy of the model. Once all 20 paths had been fitted, the prediction accuracy was normalized with respect to the maximum obtained during that run. For each run, the normalized prediction accuracy was evaluated as a function of the threshold level.

This test was performed on three different systems, chosen such that the parallel cascade expansion would converge at different rates for each system. The first system was the homogeneous, second-order system, shown in Figure 5.2, used in the previous simulations. The second system had a kernel, shown in the upper panel of Figure

Second–Order Kernel for Second System



Second–Order Kernel for Third System

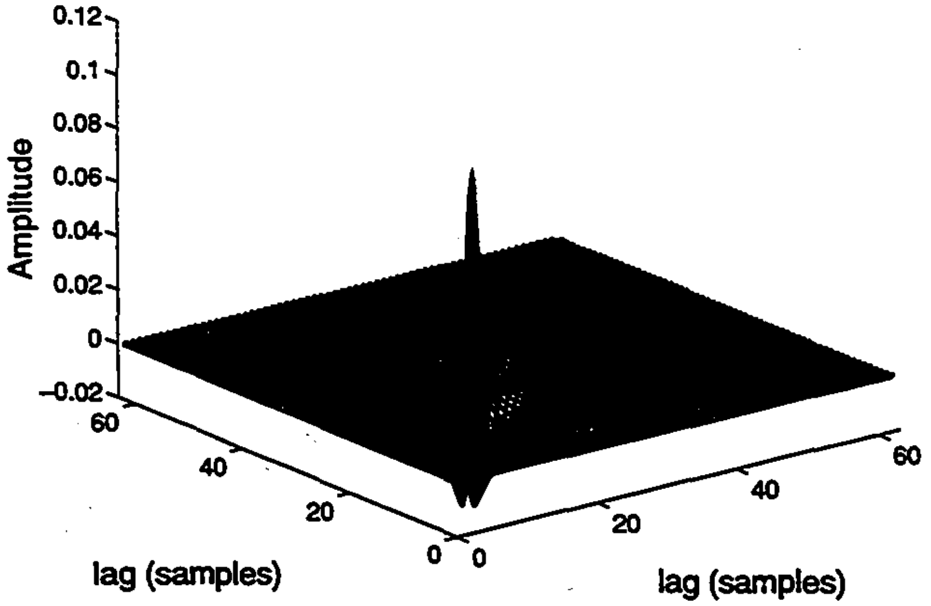


Figure 5.12: The top panel shows the second-order Volterra kernel of the second system used in the threshold tests. The kernel for the third test system is shown in the lower panel.

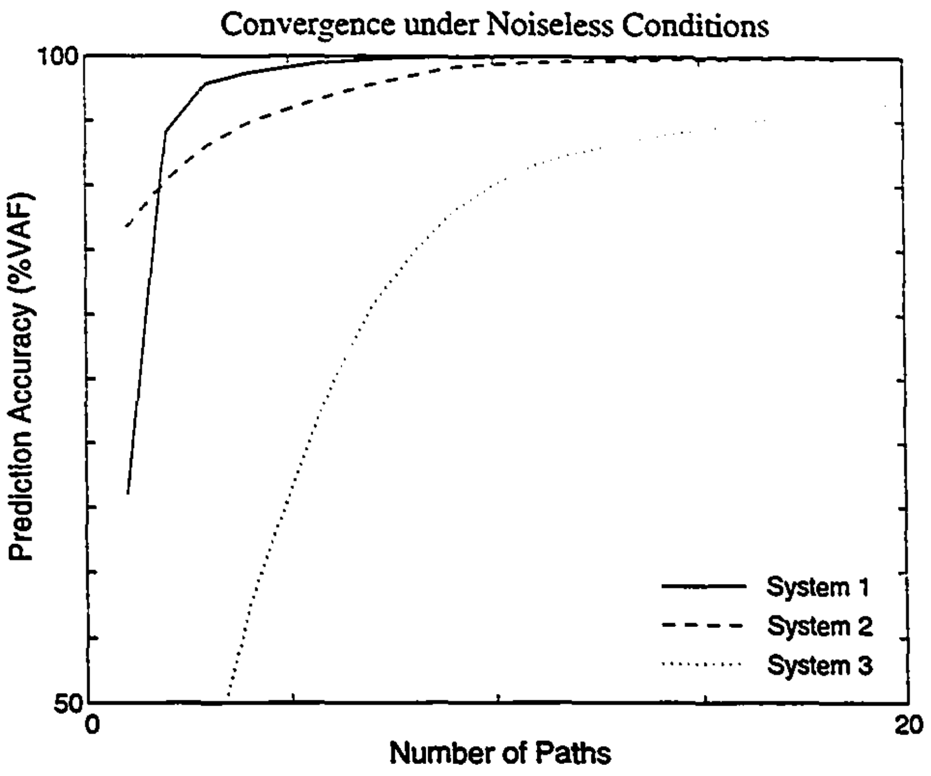


Figure 5.13: Convergence, under noise-free conditions, of the optimized parallel cascade method applied to the three systems used in the threshold simulations.

5.12, which was constructed in a similar manner to the first. However, the peaks in the kernel were concentrated near the diagonal to slow the convergence of the parallel cascade. The third system was an LNL cascade consisting of a fourth-order Butterworth high-pass filter, followed by a squarer, followed by a fourth-order Butterworth low-pass filter. Given the nearly diagonal shape of this kernel, shown in the lower panel of Figure 5.12, we would expect convergence to be slow indeed.

The three systems were first identified under noiseless conditions, to establish the speed at which the parallel cascade expansion converged for each system. Figure 5.13 shows the prediction accuracy as a function of the number of paths for models of the three systems. From this figure, it is evident that parallel cascade models of the three systems converge at very different rates. Taken together, simulations performed on these three systems should be representative of the behaviour expected with a wide variety of unknown systems.

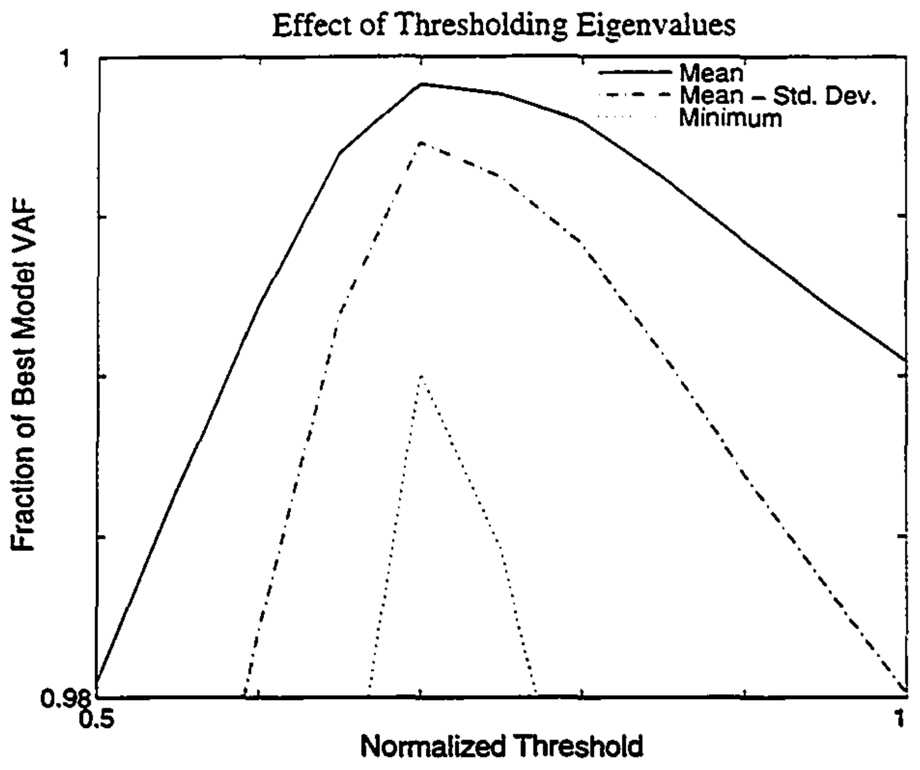


Figure 5.14: Normalized model accuracy as a function of the decision threshold. Normalized accuracy of 1 corresponds to the best model produced by a given parallel cascade identification.

Ten runs were performed on each system, at each of four SNRs: 0, 3, 6 and 10 dB. In each case, the confidence level, which determines the value of χ in Equation (5.5), was set to 50%.

The upper trace in Figure 5.14 plots the mean normalized accuracy, evaluated over all systems and SNRs, as a function of the decision threshold. This plot has a peak at a threshold of 0.7, which yields a normalized accuracy of 0.9992. Thus, on average, if a decision threshold of 0.7 is used, the resulting model would account for 99.92% of the variance accounted for by the best possible model estimated from the data.

The lower two traces in Figure 5.14 show the mean less one standard deviation and the minimum normalized prediction accuracies. These curves also reach a maximum for a threshold of 0.7. The minimum normalized accuracy was 0.9901. Thus, even in the worst case observed, using a decision threshold of 0.7 resulted in a model which accounted for 99.01% of the variance accounted for by the best parallel Wiener cascade model which could be obtained from the given data.

5.3 Multiple-Input Systems

As we discussed in Section 2.4, multiple-input systems can be represented by Wiener/ Volterra series that contain two types of kernels: self-kernels and cross-kernels.

The self-kernels are driven by single inputs; in general, a multiple-input system will have a series of self-kernels for each of its inputs. In principle, these may be identified in the same ways as the kernels of a single-input system. However, as was the case with the multiple-input Wiener structures considered in Chapter 4, the presence of multiple inputs can cause special interference problems.

The cross-kernels, on the other hand, receive excitation from several inputs. In principal, they may be estimated from multiple-input cross-correlation functions. In this case, the presence of any self-kernels, or of cross-kernels involving different sets of inputs, can lead to interference, and estimation error.

The lowest order cross-kernel is second-order (first-order in two inputs). This may be estimated much like the second-order self-kernel. We will now develop a method to find the two-input Wiener system, as illustrated in Fig. 5.15, that best approximates the cross-kernel.

Initially, assume that the inputs, $u_1(t)$ and $u_2(t)$, are white, so that the cross-correlations, $\phi_{u_1x_1}(\tau)$ and $\phi_{u_2x_2}(\tau)$, are equal to the impulse responses, $h_{u_1}(\tau)$ and $h_{u_2}(\tau)$, respectively. If the inputs are not white, the analysis presented in this section will estimate the correlations rather than the IRFs. The input auto-correlation functions will then have to be deconvolved, as suggested by Hunter and Kearney [30], and presented in Equation (2.9).

Calculate the second-order cross-cross-correlation across the candidate path:

$$\phi_{u_1u_2y}(\tau_1,\tau_2) = E[u_1(t-\tau_1)u_2(t-\tau_2)y(t)] = kh_{u_1}(\tau_1)h_{u_2}(\tau_2) \qquad (5.12)$$

and determine the mean-squared error between the cross-correlation observed in the

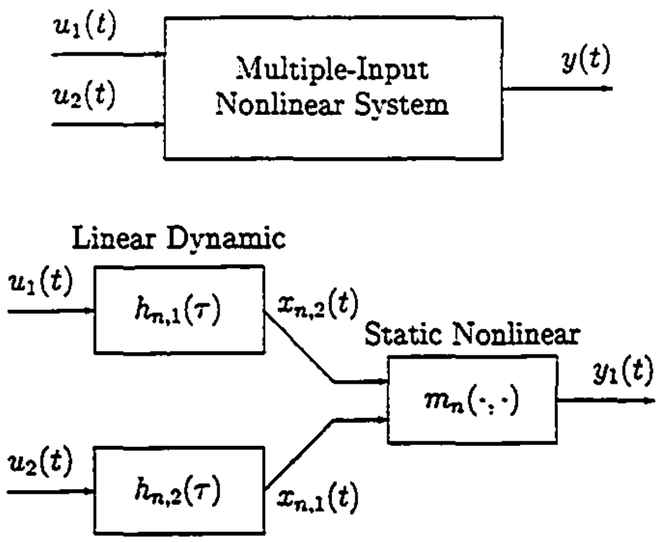


Figure 5.15: A multiple-input nonlinear system (top), and the multiple-input Wiener system (bottom) that is used to approximate it.

data, and that calculated for our candidate path. We will seek impulse responses, $h_{u_1}(\tau_1)$ and $h_{u_2}(\tau_2)$, that minimize this MSE, which can be written:

$$\text{MSE} = \frac{1}{T^2}\sum_{i,j=1}^{T}\left(\hat{\phi}_{u_1u_2y}(i,j) - kh_{u_1}(i)h_{u_2}(j)\right)^2 \tag{5.13}$$

Calculate the optimal value for the constant, k, given the two IRFs $h_{u_1}(\tau_1)$ and $h_{u_2}(\tau_2)$ by differentiating the MSE with respect to k, and setting the result to zero. Due to the scaling introduced by k, let $h_{u_1}(\tau_1)$ and $h_{u_2}(\tau_2)$ have unit norms.

$$k = \sum_{i,j=1}^{T}\hat{\phi}_{u_1u_2y}(i,j)h_{u_1}(i)h_{u_2}(j) \tag{5.14}$$

Given this value for k, expand Eq (5.13)

$$\text{MSE} = \frac{1}{T^2}\left\{\sum_{i,j=1}^{T}\hat{\phi}^2_{u_1u_2y}(i,j) - \left[\sum_{i,j=1}^{T}\hat{\phi}_{xyv}h_{u_1}(i)h_{u_2}(j)\right]^2\right\} \tag{5.15}$$

We must therefore maximize the second term in (5.15), subject to the constraint of unit norms for the IRFs. To achieve this, write the cross-cross-correlation matrix in terms of its singular value decomposition (SVD) [20]:

$$\hat{\phi}_{u_1u_2y} = USV^T \tag{5.16}$$

where U and V are orthonormal matrices with rows equal to the left and right singular vectors of the cross-correlation matrix. S is a diagonal matrix of the singular values, s_k. Now, write the IRFs $h_{u_1}(\tau)$ and $h_{u_2}(\tau)$ as weighted sums of the singular vectors $\overline{u}_k$ and $\overline{v}_k$:

$$h_{u_1} = \sum_{k=1}^{T} \alpha_k \overline{u}_k, \quad h_{u_2} = \sum_{k=1}^{T} \beta_k \overline{v}_k \tag{5.17}$$

We must now choose the α_k and β_k to minimize the MSE. Expanding (5.15) it can be shown that this is equivalent to maximizing:

$$\left\{ \sum_{k=1}^{T} s_k \alpha_k \beta_k \right\}^2 \tag{5.18}$$

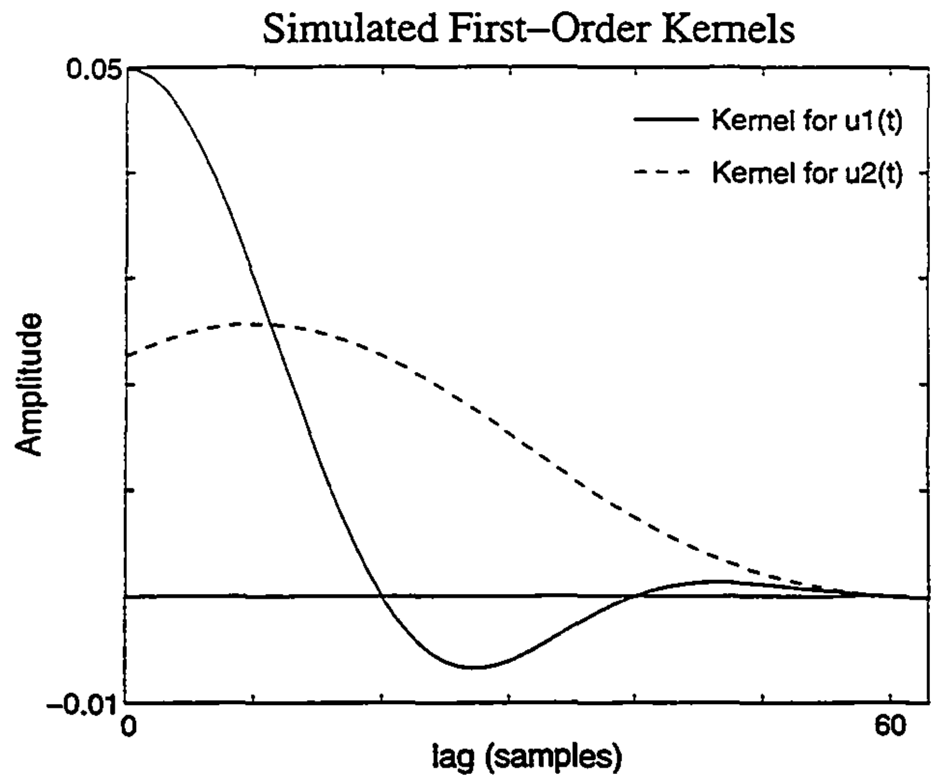
subject to the constraints:

$$\sum_{k=1}^{T} \alpha_k^2 = \sum_{k=1}^{T} \beta_k^2 = 1 \tag{5.19}$$

Solving this constrained optimization, we see that all of the α_k and β_k must be equal to zero, except those that are associated with the largest singular value, which must both be unity. Therefore choosing the vectors associated with the largest singular value gives us the optimal choice of the IRFs. As stated previously, if the inputs are non-white, the singular vectors will have to have the input auto- correlation functions deconvolved from them, as suggested in Hunter and Kearney [30].

5.3.1 Simulations Involving Multiple-Input Systems

For these simulations, we created a two-input, second-order system that had first and second-order self-kernels associated with each input, as well as a cross-kernel that processed both inputs. For the first-order kernels, shown in the upper panel of Figure 5.16, we used the IRFs of the two-input Wiener system simulated in Chapter 4. The second-order kernels were generated using techniques similar to that used in the single-input simulation, presented in Section 5.2.4, by placing impulses on a grid, and then smoothing the resulting kernel. These kernels, shown in Figures 5.16 and 5.17, had memory lengths of 64 points each, totalling 8256 independent kernel values.



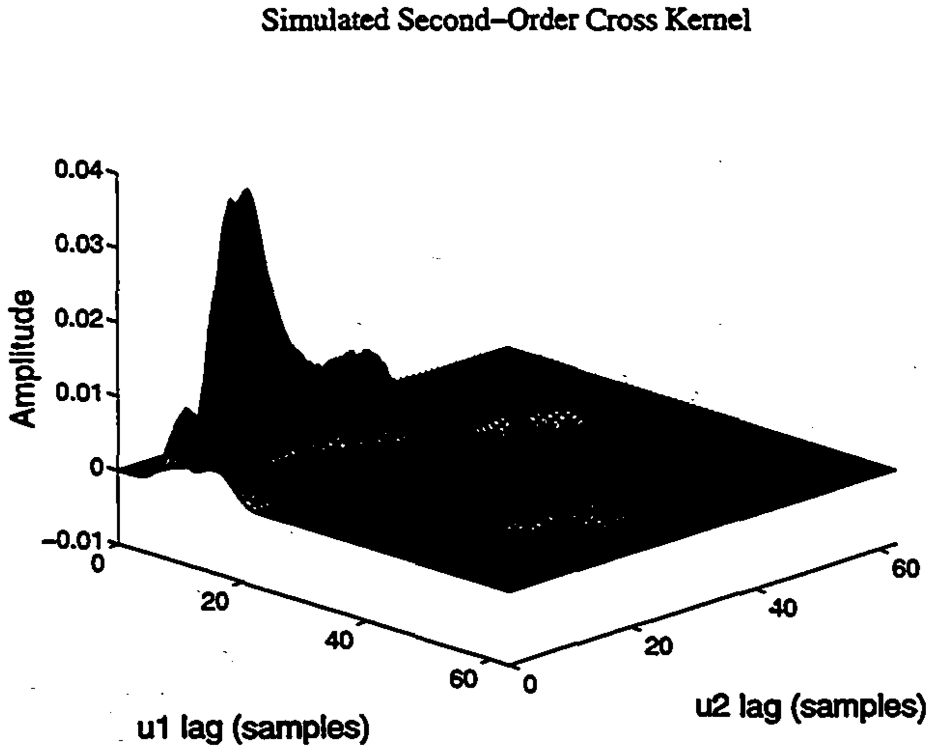
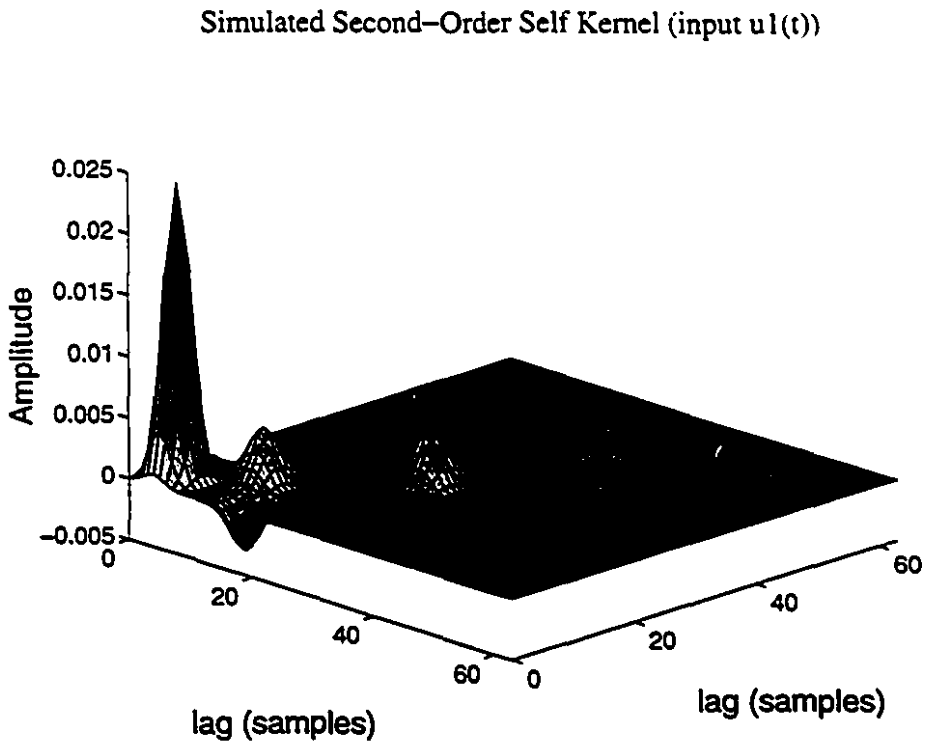


Figure 5.16: Kernels of the simulated two-input system. The upper panel shows two first-order self-kernels, one for each input. The second-order cross-kernel is shown in the lower panel



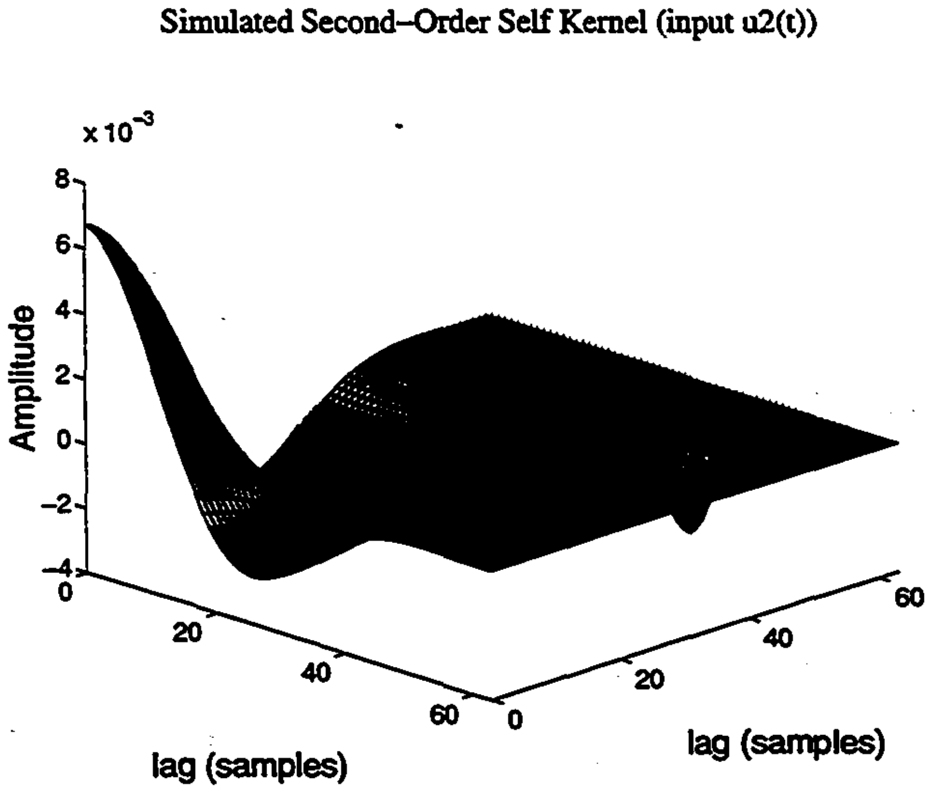


Figure 5.17: Second-order self-kernels of the simulated two-input system. The upper panel shows the kernel associated with $u_1(t)$. The kernel which was driven by $u_2(t)$ is shown in the lower panel

Two 25,000 point records of white Gaussian noise were used as inputs. Kernels were estimated using the methods presented in this paper, and by using single slices of the correlation functions, as suggested by Korenberg [56].

As in the previous simulations, we evaluated model accuracy as a function of the number of computations required, both under noiseless conditions, and with various levels of output noise. Figure 5.18 shows the rate of convergence under noiseless conditions for both methods. Even under noiseless conditions, the SVD method converged significantly faster than the single slice formulation.

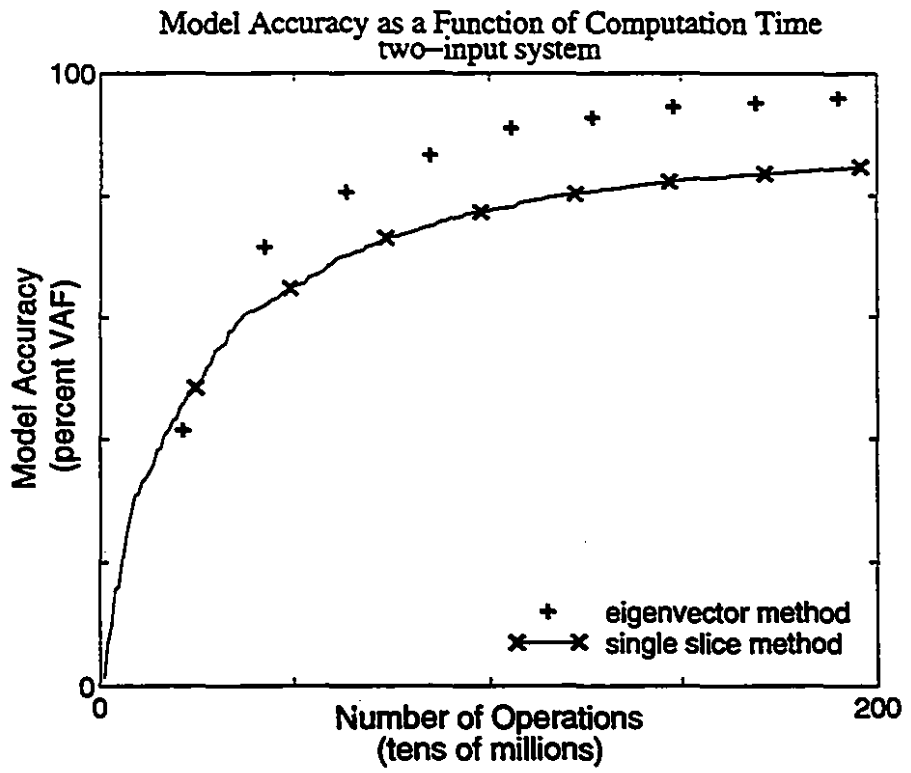


Figure 5.18: Convergence speed for the multiple-input implementation of the single-slice and eigenvector methods. Model accuracy is plotted as a function of the number of flops required in the computations. The + signs each represent one pathway being added by the eigenvector method. The crosses each represent 25 pathways being added by the single slice method

In Fig. 5.19, the maximum model accuracy obtained is plotted as a function of the output noise level. From this figure, it is evident that the SVD based method produced much better models than the single-slice method, at all noise levels.

Even under noiseless conditions the single slice method was only able to account

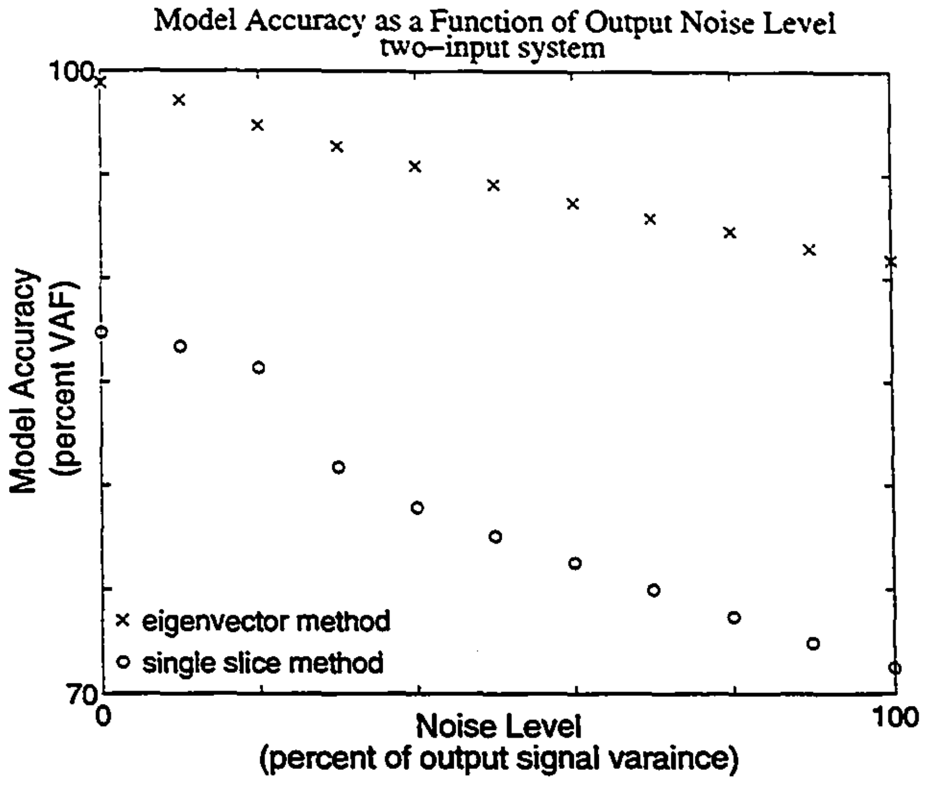


Figure 5.19: Noise performance of the multiple-input algorithms.

for 85% of the output variance. This poor performance is probably due to interference in the correlation estimates caused by nonlinear interactions among the input signals, as described in the previous chapter, which dealt with multiple-input Wiener systems.

This interference will be present, regardless of which method is used, and result in the degradation of correlation estimates. Provided an adequate initial estimate of the IRFs is available, the iterations proposed in Chapter 4 may be used to estimate and remove much of this interference. At some point, due to it's poor noise performance, the single slice method becomes unable to provide such an initial estimate, and the iteration fails.

However, due to its superior noise performance, the SVD method is less sensitive to this degradation, and therefore produces better estimates of the system. Furthermore, these estimates can then be used to initiate the interference suppression. Thus, the superior noise performance of the SVD based method provides two advantages: first, it yields better initial estimates than the single slice method, and second, it allows the iterative noise suppression techniques, developed in Chapter 4 to be used at lower SNRs, further increasing its advantage in model accuracy.

5.4 Summary

In this chapter, we considered how to model nonlinear systems using a parallel sum of Wiener systems. We developed procedures that estimate the optimal Wiener system to add to the array at each stage in the modelling. Furthermore, we demonstrated that our expansion is unique, given the input auto-correlation.

The development of these methods represents an important advance in the technology of nonlinear system identification. Because our expansion is unique, and each pathway is optimal, the results are repeatable, and require little, if any, user intervention. As a result, they should be suitable for use by a wide variety of researchers who have little background in the specifics of nonlinear system theory.

Our approach is to use a single Wiener cascade, whose linear element is derived from the first-order input-output cross-correlation, to drive the mean of the residuals and the first-order cross-correlation between the input and the residuals to zero.

The linear elements of subsequent paths are derived from the second-order cross-correlation between the input and the residuals. The procedure outlined in this chapter identifies the Wiener cascade that produces the greatest reduction in the variance of the residuals, subject to the constraint that the nonlinearity is represented by a second-order polynomial. When the input is white, this procedure also results in the greatest reduction in the F-norm of the second-order cross-correlation matrix.

Under noiseless conditions, this procedure should drive the second-order correlation to zero, using a number of paths which is at most equal to the memory length of the system. If noise is present, this point cannot be reached, as paths will eventually model the noise instead of remaining dynamics. We provide a criterion to help the investigator decide when to stop basing cascade paths on the second-order correlation function.

Finally, we developed a technique that can be used to find the Wiener path that produces the greatest reduction in the mean square value of the third-order cross-correlation between the input and the residuals. Because a closed form solution to this problem was not evident, we developed a gradient search scheme. In principal, this gradient search could also be used with higher order correlation functions.

We have shown how these methods may be applied to multiple input systems. The approach is essentially the same as that employed with single input systems, although we have more correlation functions at our disposal. Furthermore, the basic building block is a multiple-input Wiener cascade, Chen's [13] 1-c structure, instead of the single input systems used in the previous case.

Using numerical simulations, we demonstrated that these algorithms converge more quickly than the original single-slice based parallel cascade methods, and that this speed advantage increases with the desired level of model accuracy. Furthermore, the optimized parallel cascade methods were more robust against noise than the original implementations. In Chapter 6, we will demonstrate how these procedures work when they are applied to data from physical systems.

Chapter 6

Case Studies in Nonlinear Identification

While much can be learned in digital simulations, it is difficult to simulate many of the constraints facing experimenters. Some, such as the finite roll-off of both anti-aliasing and reconstruction filters, which limit the bandwidth of the measured input and output signals, and the finite resolution of analogue to digital converters (ADC), which generate quantization noise, are due to the data acquisition apparatus. Other constraints are due to the experiment itself. For example, the bandwidth of the test input may be restricted by the apparatus. In experimental studies of joint dynamics, perturbations are applied using either torque motors or hydraulic actuators, both of which act as low-pass filters and limit the perturbation bandwidth. A final constraint is the presence of measurement noise, which is often assumed to have a Gaussian distribution and white spectrum. Neither of these assumptions is necessarily valid.

This chapter describes two applications of the nonlinear system identification methods, developed in previous chapters, to real data from two physical systems. The first is an experiment conducted on an analogue nonlinear system constructed using a combination of linear filters and an analogue multiplier. We used this experiment to assess the applicability of our methods under laboratory conditions. The second application is the identification of a model of the relationship between the movement of a joint and the resulting myoelectric activity: the stretch-reflex EMG.

This was done to examine how our algorithms can be applied to data from an unknown, physical system and to elucidate some of the choices that must be made during an analysis.

6.1 Identification of an Analogue System

6.1.1 Methods

The test system was created using analogue filters and an analogue multiplier[1]. A block diagram of this system is shown in Fig. 6.1. Details regarding the construction of this system can be found in Appendix B.

If the linear elements in Figure 6.1 were all low-pass, it would be possible, at least theoretically, to map the static nonlinearity simply by applying constant inputs at various levels. Clearly, if one or more of the elements is high-pass, this approach will fail. Therefore, we examined how the identification algorithms performed when both low-pass and high-pass filters were used in various configurations in the test system. A summary of the configurations used in the experiments is presented in Table 6.1.

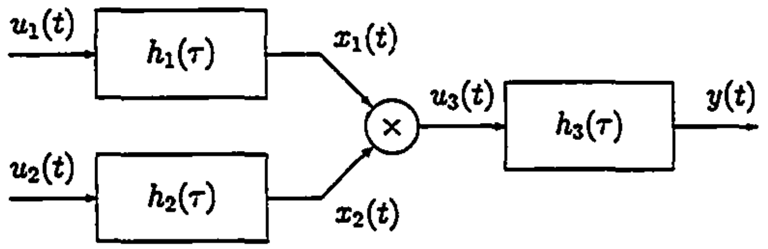


Figure 6.1: Block diagram of the electronic system used in the experimental verification of the eigenvector method. Note the internal signals, x_1,x_2, and u_3 were not measured, and are included for discussion purposes only.

In each experiment, the system was driven by two highly coloured inputs. These consisted of 6000 points of white Gaussian noise, filtered digitally by a fourth-order Butterworth low-pass filter with a normalized cut-off of 0.1. This stimulus, repeated

[1]MPY-100BG Burr Brown

Experiment	$h_1(\tau)$	$h_2(\tau)$	$h_3(\tau)$
1	Low-Pass	Low-Pass	Low-Pass
2	Low-Pass	Low-Pass	High-Pass
3	High-Pass	High-Pass	Low-Pass

Table 6.1: Summary of experimental configurations

5 times, was presented at 1000 Hz by a 16 bit digital to analogue converter, and low-pass filtered by a 100 Hz reconstruction filter. All signals were low-pass filtered at 100 Hz, and then sampled at 1000 Hz, using 16 bit analogue to digital converters.

Once a model had been identified, its accuracy was assessed in two ways. Firstly, the measured input signals were applied to the model, and its output computed. This was compared with the measured output to yield the "prediction accuracy" attained by the model. Secondly, we generated the second-order cross-kernel for the model, and compared it with a theoretical kernel, generated analytically using the impulse responses of the linear filters employed in the system. This produced the "kernel accuracy". In both cases, accuracy was reported as the percentage of the variance accounted for by the model (%VAF) (See Equation 3.5).

6.1.1.1 Choice of Sampling Rate

Ideally, the anti-aliasing filters should have constant gain and linear phase for frequencies between DC and the cut-off, followed by a rapid transition to zero gain. In practice, they have nearly constant gain and linear phase below the cut-off frequency, but the theoretical attenuation is only reached well above the cut-off frequency. To avoid distortion and aliasing, we used a sampling frequency approximately ten times higher than the expected Nyquist frequency [5].

Because the signals were over-sampled, it was necessary to decimate them prior to the analysis. Decimation in software has two advantages over using a lower initial sampling rate. Firstly, the raw data is low-pass filtered using a digital filter, whose characteristics are much closer to ideal than the analogue filters used in the initial sampling. Secondly, the spectra of the input and output signals can be examined prior to selecting the final sampling rate, greatly reducing the risk of inadvertently aliasing the data.

The choice of sampling rate is straightforward for a linear systems, since the presence of energy at a particular frequency in its output implies both that the frequency is in the pass-band of the system, and that there is power at that frequency in the input signal. Thus, the bandwidth of the output will be, at most, the smaller of the input bandwidth and the system bandwidth. If the sampling rate is sufficient to represent the input signal without aliasing, it must also be sufficient to represent the output.

The problem is more complex for nonlinear systems since there may be output power at frequencies different from those present in the input [110]. For example, consider a Wiener system consisting of a linear dynamic element followed by a squarer. If the input is $\sin(\omega t)$ then the output of the linear element will be $k\sin(\omega t + \phi)$, for some gain, k, and phase shift, ϕ. The output of the static nonlinearity, however, will be $\frac{k}{2}(1 - \cos(2\omega t + 2\phi))$, which contains one term at DC, and another at twice the frequency of the input. Thus, while a sampling rate between 2ω and 4ω is adequate to represent the input [5], the high frequency component in the output would either be aliased, or eliminated by anti-aliasing filters. In sampling such a system, care must be taken to ensure that neither the input nor the output is aliased.

These considerations did not pose problems in this case, as the input bandwidths were always wider than those of the outputs. Figure 6.2 shows the power spectra of the input and output signals measured during an experiment where $h_1(\tau)$ and $h_2(\tau)$ were low-pass, and $h_3(\tau)$ was a high-pass filter. This configuration resulted in the widest output bandwidth. Therefore, a sampling rate chosen for this configuration would be adequate for all of the experiments.

By examining the spectra in Figure 6.2, we can see that the "noise floor" is reached at about 200 Hz. Thus, resampling the data at 400 Hz will not result in the loss of any significant information. To achieve a decimation by a factor of 2.5, we interpolated the data by a factor of 2, and then decimated the result by 5, for a final sampling rate of 400 Hz.

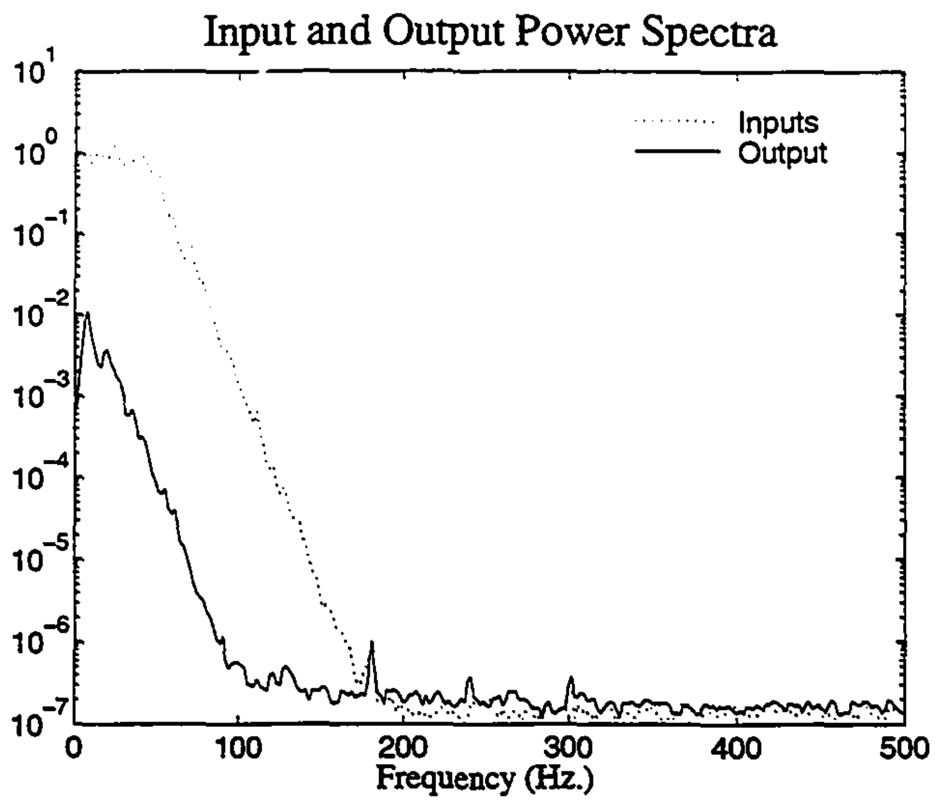


Figure 6.2: Power Spectra of the input and output signals measured during the low-pass/high-pass experiment.

6.1.1.2 Characterization of the Linear Filters

Nominal filter characteristics and impulse responses could have been calculated theoretically from the component values. However, given the tolerances associated with each component, there would be considerable uncertainty associated with each of the IRFs. Thus, we determined the filter IRFs experimentally with the same input sequences used to identify the nonlinear systems. Due to the low power in the input signal at high frequencies, the pseudoinverse based deconvolution technique, developed in Chapter 3, was used in the filter estimation. The impulse responses for the filters are shown in Figure 6.3. Design parameters for the filters, together with a circuit diagram, are presented in Appendix B.

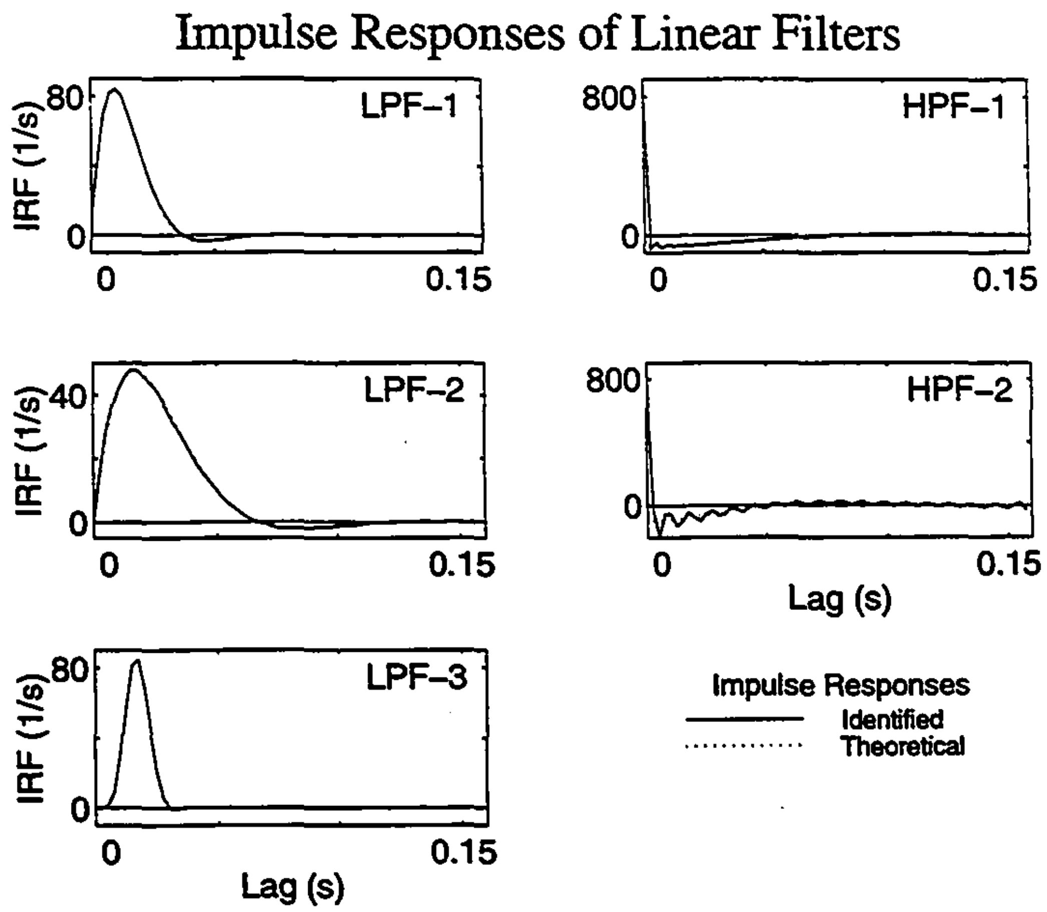


Figure 6.3: IRFs identified for the filters used in the analogue simulation

6.1.1.3 Characterization of the Measurement Noise

The inputs to the circuit were grounded and the output recorded for 30 seconds at a sampling frequency of 1,000 Hz. Its power spectrum was then estimated using 1 second segments, which corresponds to a frequency resolution of 1 Hz. The noise spectrum was white, except for two large peaks at 20 and 40 Hz, as shown in Figure 6.4. The average noise power was $2 \times 10^{-8} V^2$.

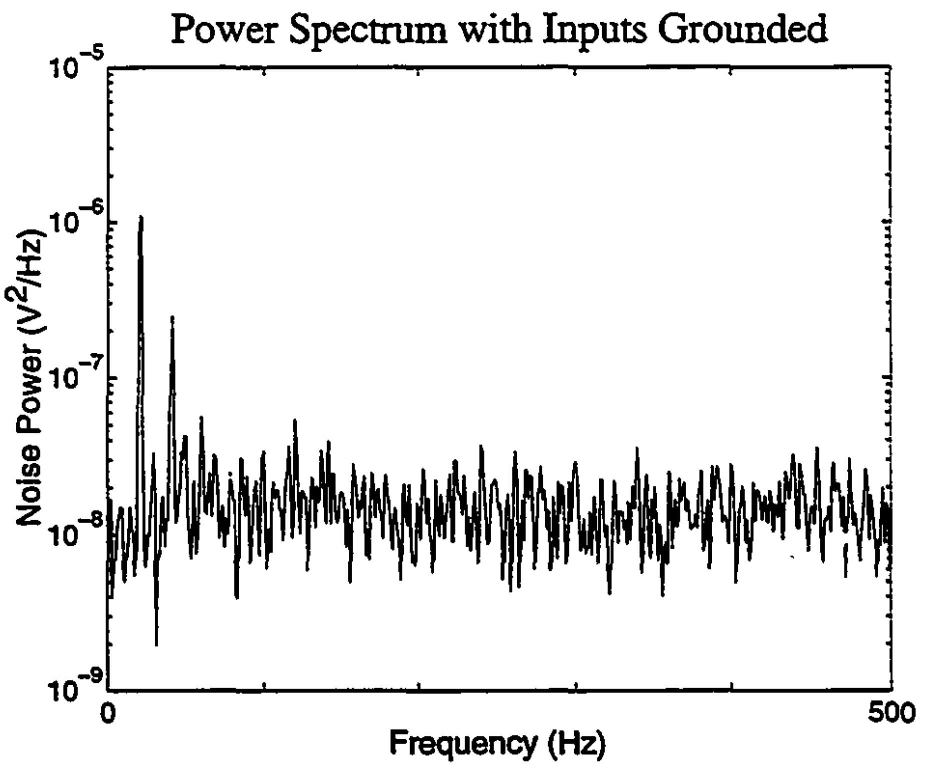


Figure 6.4: Output power spectrum measured when the inputs were grounded

6.1.1.4 Characterization of the Analogue Multiplier

The inputs to the multiplier were driven with two independent 30,000 point Gaussian noise sequences, each pre-filtered by an eighth order digital Butterworth filter with a cut-off of 25 Hz. This pre-filtering was used to ensure that both input signals would not be distorted by either the reconstruction or anti-aliasing filters. The product of the inputs was compared with the measured output. It accounted for 99.996% of the output variance, which is well within the 1.0% full scale error (0.01% error variance)

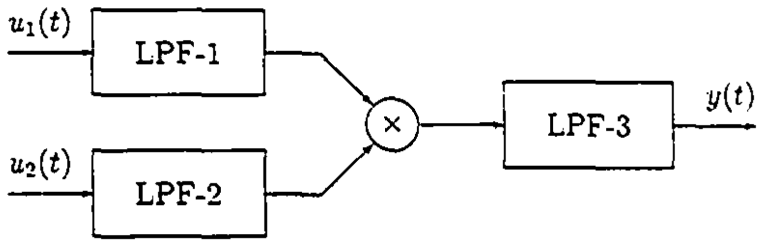


Figure 6.5: Block diagram of the configuration used in the low-pass/low-pass experiment. Filter characteristics can be found in Appendix B

specified for the multiplier.

6.1.2 Low-Pass/Low-Pass Case

We first examined the low-pass/low-pass configuration shown in Figure 6.5. This system is completely described by its second-order cross-kernel, which is given by:

$$h_{2u_1u_2}(\tau_1, \tau_2) = \sum_{i=0}^{T-1} h_3(i)h_1(\tau_1 - i)h_2(\tau_2 - i)$$

The filter impulse responses (shown in Figure 6.3), were used to generate the theoretical kernel for this system, which is shown in Figure 6.6.

The optimized parallel cascade method, developed in Chapter 5, was then used to estimate the second-order cross-kernel for the system. Since the system could be fully described by its second-order cross-kernel, the first-order input-output cross-correlation functions contained only noise. Similarly, there was no information present in the single-input second-order cross-correlations. Therefore, the linear dynamics of all paths were estimated from the principal left and right singular vectors of the second-order cross-cross-correlation, as described in Section 5.3.

Since the inputs were not white, the principal singular vectors were equal to the first order cross-correlations across the linear elements, rather than their IRFs. Thus, the input auto-correlations had to be deconvolved from the singular vectors to estimate the impulse responses of the linear elements. Deconvolving the input autocorrelation functions using the Toeplitz matrix inversion procedure, as suggested in [30] (see Equation 2.9), resulted in models that predicted the output well, but the kernels

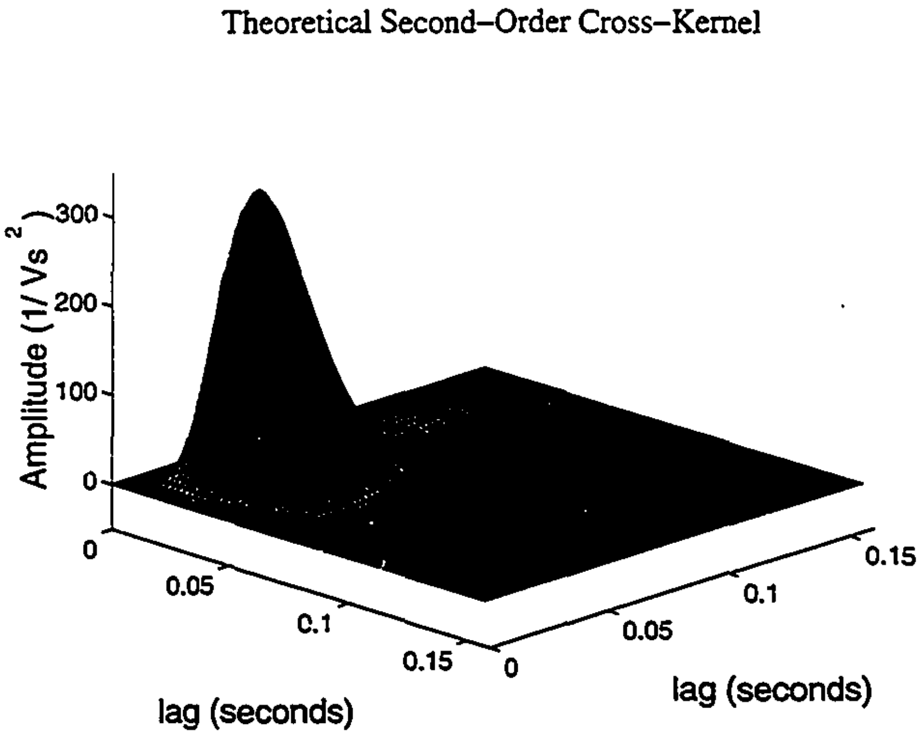
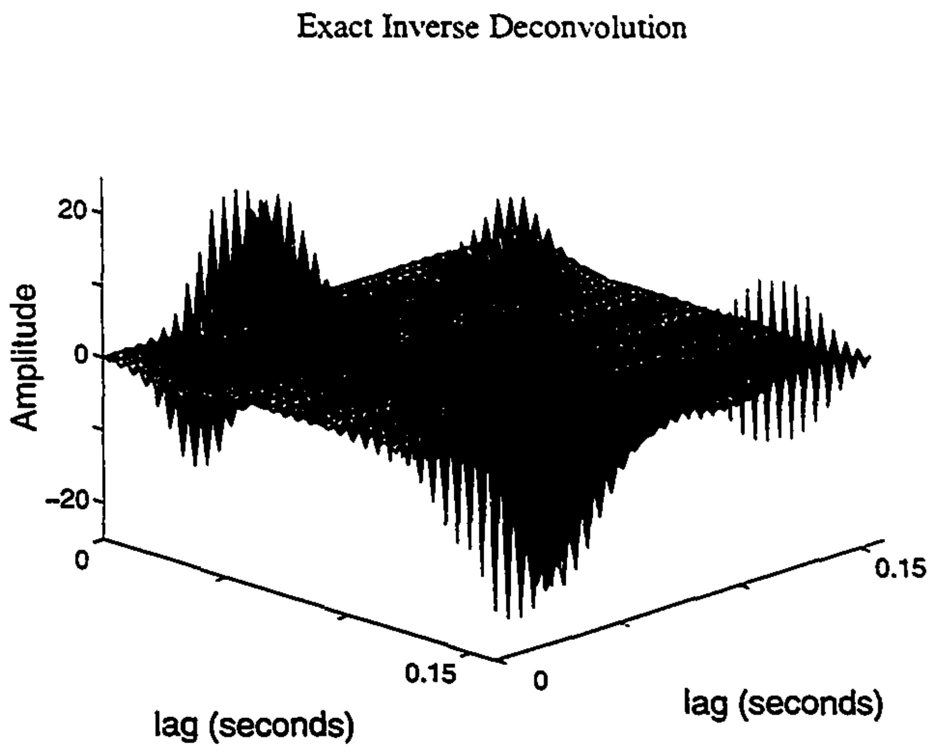


Figure 6.6: Second-order cross-kernel of the low-pass/low-pass system, generated from the IRFs of the filters used in the system.

were buried in large amplitude, high frequency noise, as shown in the upper panel of Figure 6.7. Given the similarity between this situation and the problems that arise when coloured inputs are used in the identification of linear systems, we attempted to use pseudoinverse based input deconvolution (Equation 3.21) to suppress this ringing.

Application of the pseudoinverse based deconvolution to this problem was not straightforward since an analytical method of selecting the correct pseudoinverse order had not been developed. For systems estimated from a single-input second-order cross-correlation, the solution of the generalized eigenvalue problem performs the deconvolution implicitly, and allows the formulation of an equivalent first-order problem, which may be used to determine the appropriate pseudoinverse order. In contrast, for the two-input problem, linear dynamics are estimated using a singular value decomposition followed by an explicit deconvolution: a formulation which does not lead to an equivalent first-order problem. Furthermore, the orders of the two pseudoinverses, one for each input, appear to be coupled. As a result, we have been unable to derive



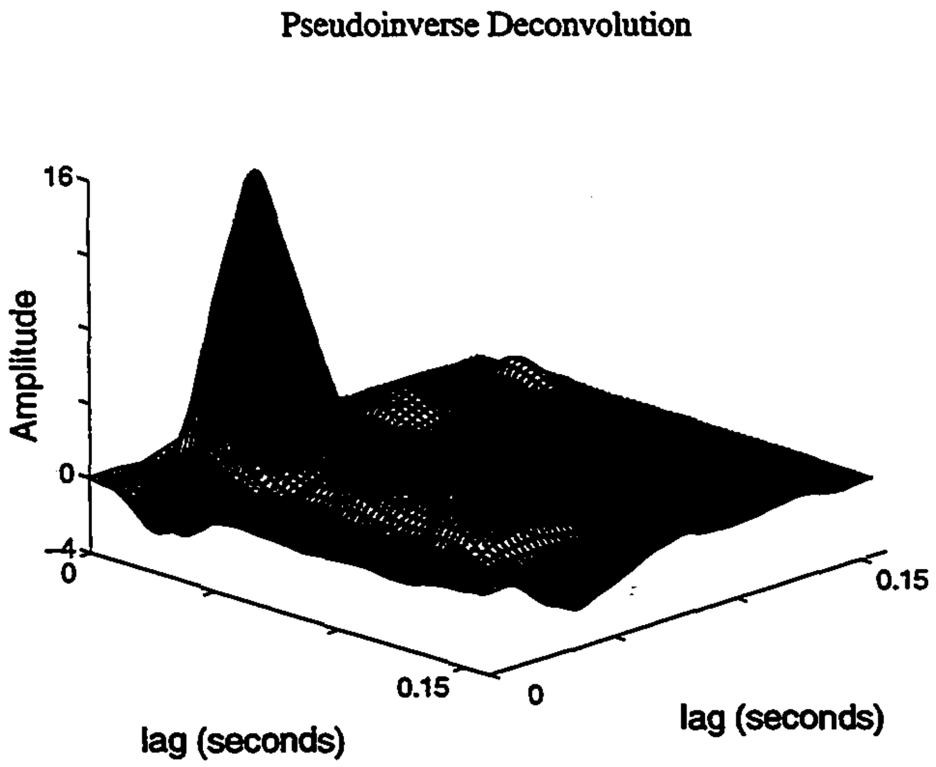


Figure 6.7: Effect of using pseudoinverse deconvolution on the initial estimate of the second-order cross-kernel. Note that the amplitude of the kernels is arbitrary, as these models do not include a static nonlinearity.

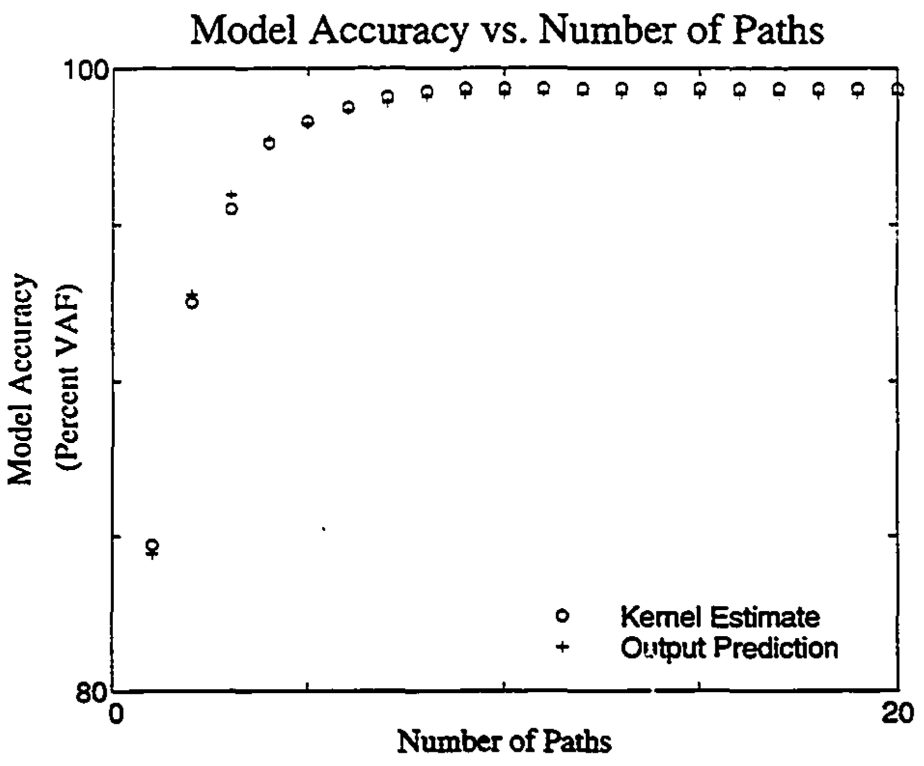


Figure 6.8: Accuracy of the kernel estimate and the output prediction as a function of the number of paths estimated in the low-pass/low-pass experiment.

an analytical solution for this problem. Consequently, we used an exhaustive search over all pairs of deconvolution orders, selecting the pair which minimized the residual variance.

Figure 6.7 shows the kernels of a single, two-input Wiener cascade, whose linear dynamics were estimated from the principal left and right singular vectors of the second-order cross-cross-correlation between the inputs and output. The upper panel shows the kernel of the initial estimate, when the exact inverses of the two input auto-correlation functions were used in the deconvolution. This kernel is dominated by high frequency ringing, particularly near its four corners. The lower panel shows the kernel obtained when pseudoinverses were employed in the deconvolution of the input auto-correlations. Here, the ringing has been suppressed, revealing the underlying kernel.

The accuracy of the kernel estimates and output predictions are plotted in Figure 6.8, as functions of the number of cascade paths used to estimate the model. Maximum kernel accuracy was achieved with 10 paths. Using more paths resulted in a

Second–Order Cross–Kernel Estimate

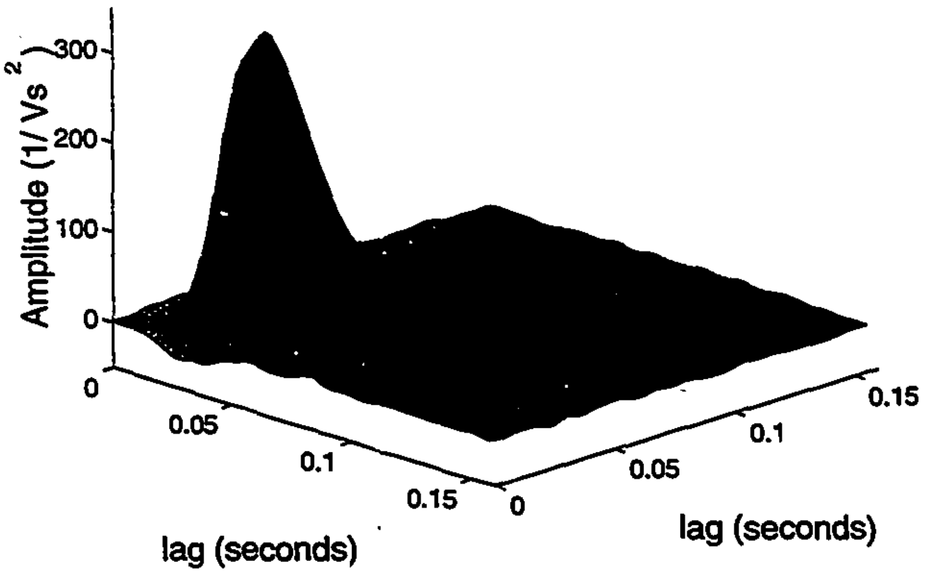


Figure 6.9: Cross-kernel estimated using 10 paths generated by the optimized parallel cascade method, using pseudoinverse based input deconvolution. This kernel accounts for 99.35% of the variance of the true kernel.

slight decrease in the kernel accuracy with no significant improvement in the output prediction. This suggests these additional paths were modelling noise rather than system dynamics.

The second-order cross-kernel based on the first 10 paths is shown in Figure 6.9. This kernel accounts for 99.4% of the variance of the analytically determined kernel and 99.2% of the output variance, demonstrating that an excellent system model has been identified. Note that the kernel shown in Figure 6.9 has not been smoothed.

The energy remaining in the output residuals was apparently dominated by errors introduced in estimating the nonlinearities. At each stage, a two-input, second-order nonlinearity was fitted between the filter outputs and the residuals. The second-order cross term generated the contribution of the current path to the estimate of the second-order cross-kernel. Given the characteristics of the analogue multiplier, coefficients other than that associated with the second-order cross-kernel should have been zero. However, estimates of these coefficients, although small, were non-zero in general. Indeed, forcing these terms to zero during the identification resulted in a model which accounted for 99.65% of the output variance and 99.70% of the kernel. Thus, using the correct nonlinearity structure halved the variance of the residuals, both in the output prediction and in the kernel.

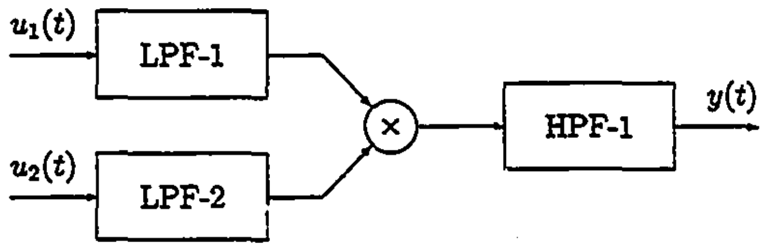
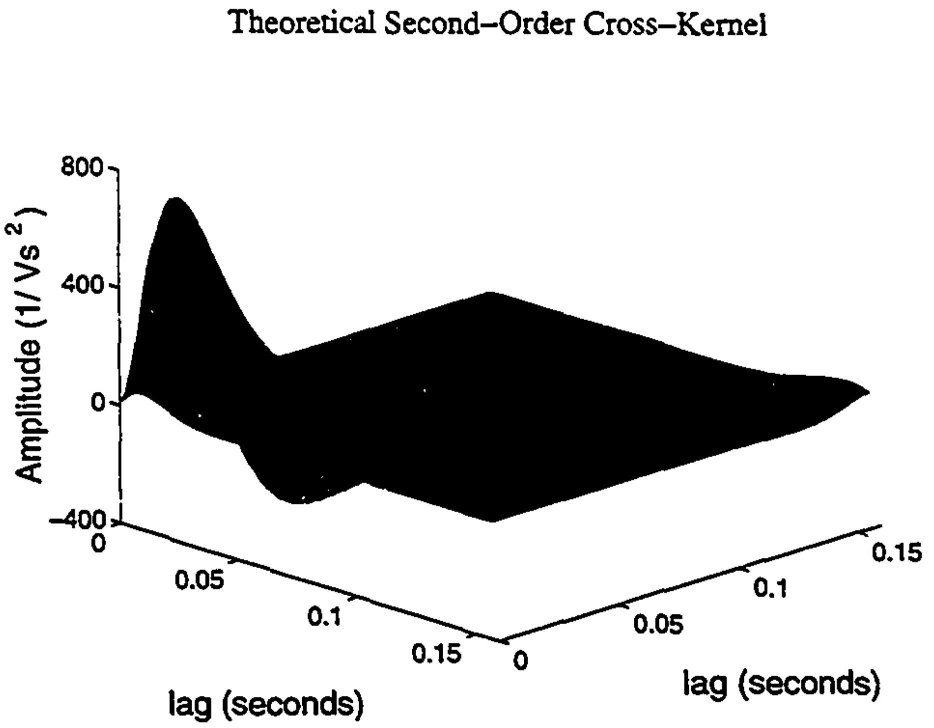


Figure 6.10: Block diagram of the configuration used in the low-pass/high-pass experiment. Filter characteristics can be found in Appendix B

6.1.3 Low-Pass/High-Pass Case

Next, we examined the low-pass/high-pass configuration, in in which the output filter, $h_3(\tau)$, was a 4.6 Hz Chebyshev type I high-pass filter (HPF-1 in Table B.1). A block



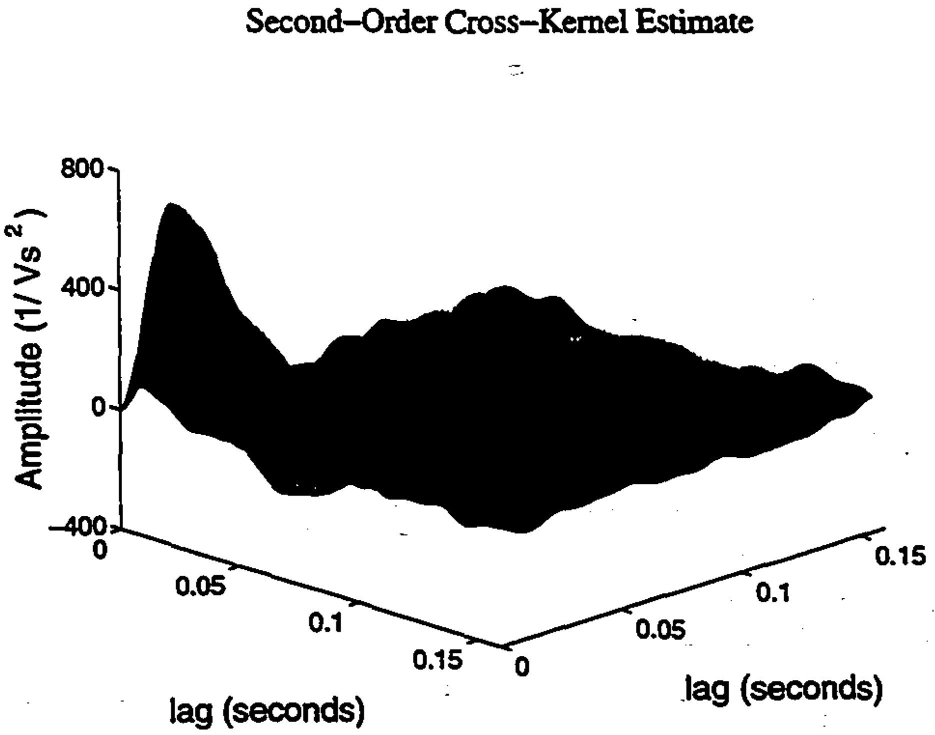


Figure 6.11: The upper panel shows the second-order cross-kernel for the low-pass /high-pass experiment. The lower panel shows the cross-kernel of the model estimated using 16 paths. It accounts for 97% of the variance of the theoretical kernel

diagram of this configuration is shown in Figure 6.10. The theoretical second-order cross-kernel for this system, shown in the upper panel of Figure 6.11, was generated using the impulse responses shown in Figure 6.3.

The kernel was then estimated from the input-output data using the optimized parallel cascade method. All paths were based on the second-order input-output cross-cross-correlation, using an exhaustive search over orders of both pseudoinverses to deconvolve the input auto-correlations from the impulse response estimates. Figure 6.12 shows the model accuracy as a function of the number of paths used to estimate the model. Maximum kernel accuracy was achieved with 16 paths. The corresponding kernel estimate is shown in the lower panel of Figure 6.11.

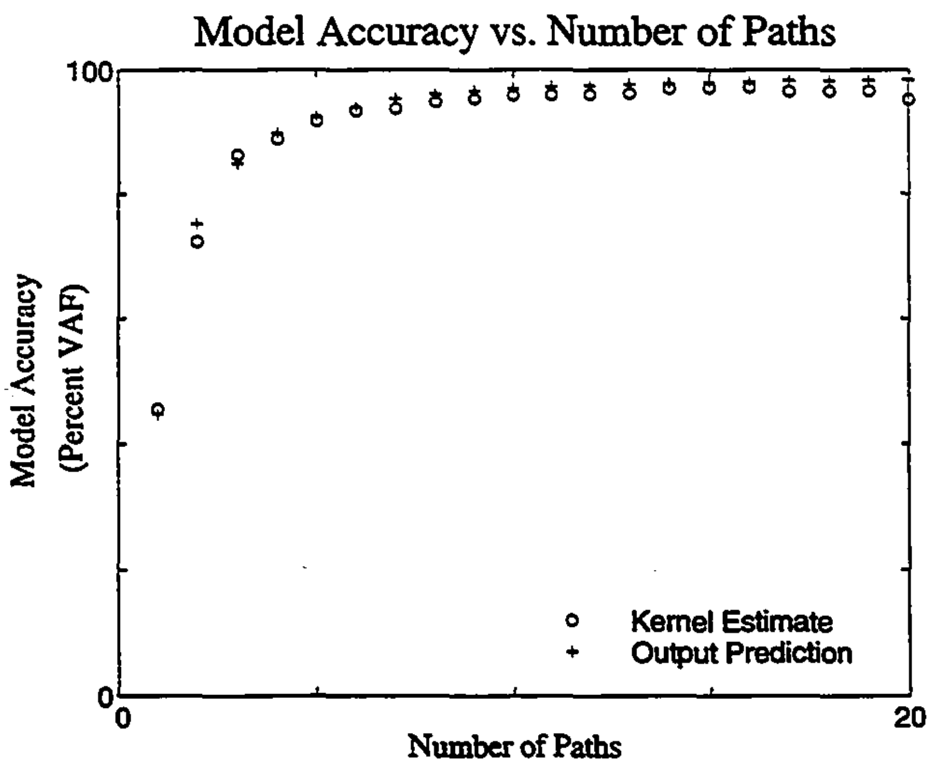


Figure 6.12: Accuracy of the kernel estimate and the output prediction as a function of the number of paths estimated in the low-pass/high-pass experiment.

Although the accuracy of the kernel estimate decreased after the first 16 paths had been identified, the prediction accuracy continued to increase slightly. This suggests that at this point, additional paths modelled measurement noise, and not system dynamics.

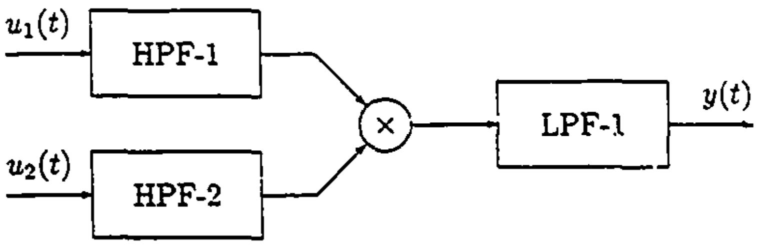


Figure 6.13: Block diagram of the configuration used in the high-pass/low-pass experiment. Filter characteristics can be found in Appendix B

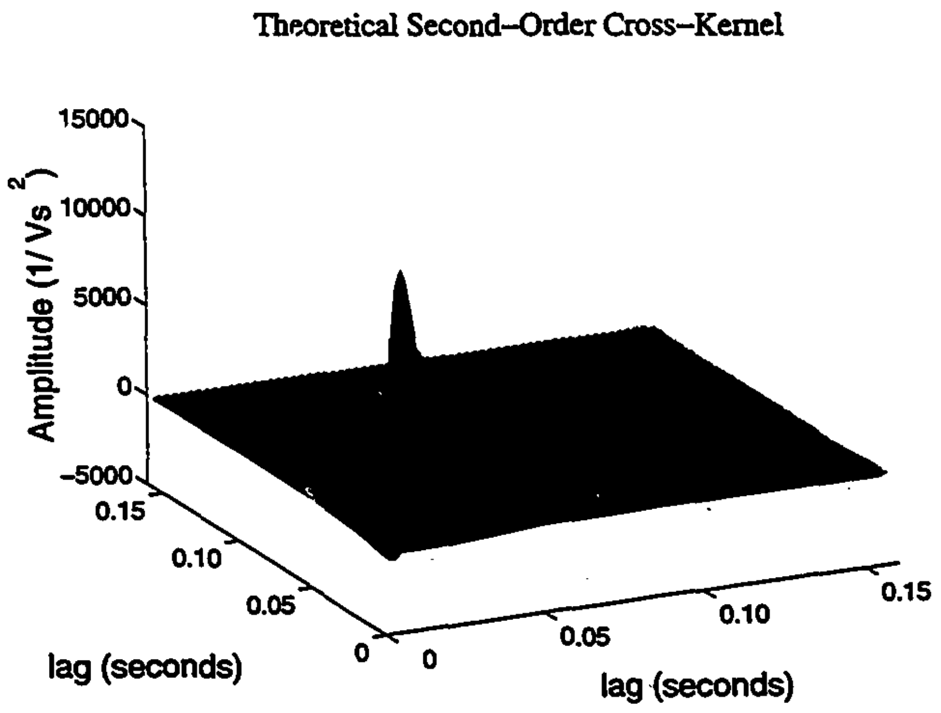


Figure 6.14: Theoretical high-pass/low-pass cross-kernel

6.1.4 High-Pass/Low-Pass System

Finally, we examined the high-pass/low-pass configuration illustrated in Figure 6.13. Two high-pass filters, HPF-1 and HPF-2 in Table B.1, were used as inputs to the multiplier, and a second-order low-pass Butterworth filter, LPF-1, was used to filter the multiplier output. The analytically generated kernel for this configuration is shown in Figure 6.14.

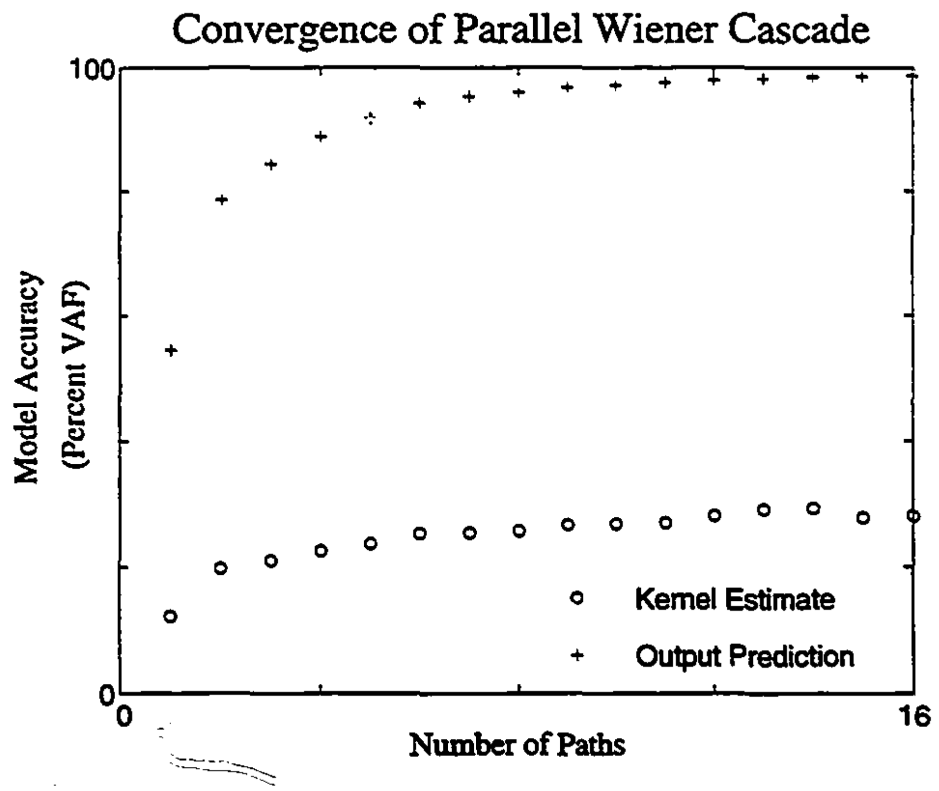


Figure 6.15: Accuracy of the kernel estimate and the output prediction as a function of the number of paths estimated in the high-pass/low-pass experiment.

Figure 6.15 shows both the prediction accuracy and the accuracy of the kernel estimate plotted as a function of the number of paths added to the model. It is evident that the combination of the optimized parallel cascade method and the pseudoinverse based input deconvolution technique produced a model which yielded excellent output predictions (98.5%VAF). Nevertheless, the model did not estimate the analytically derived kernel very well.

The best kernel model, shown in Figure 6.16 was generated using the first 14

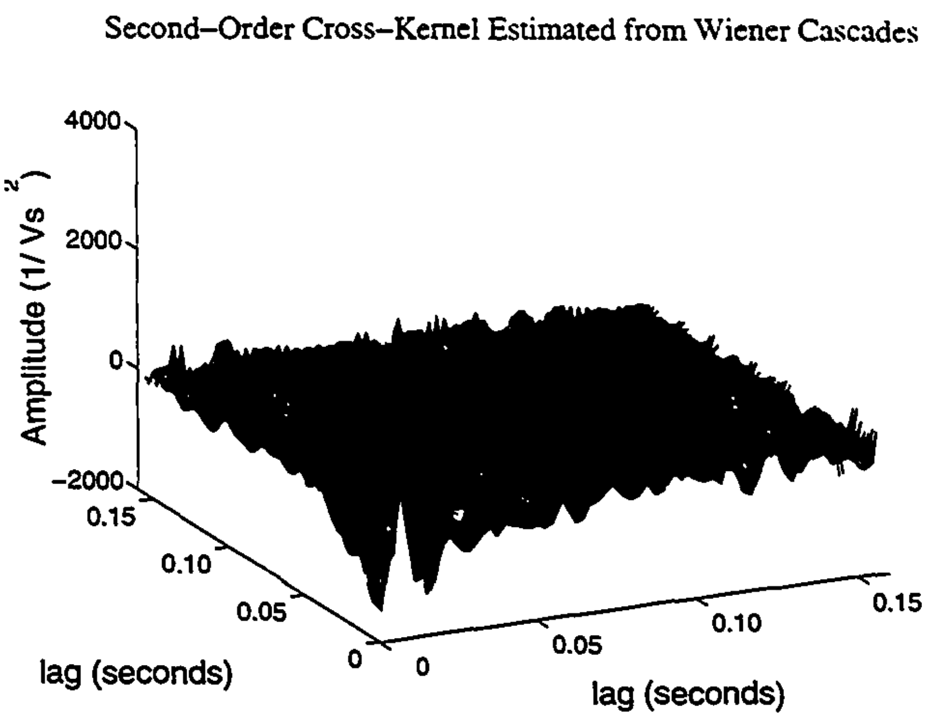


Figure 6.16: Cross-kernel estimated using 14 paths generated by the optimized parallel cascade method, using pseudoinverse based input deconvolution. This kernel accounts for 29.3% of the variance of the true kernel.

identified Wiener cascades, but accounted for only 29.3% of the variance of the true kernel. Comparing Figure 6.16 with 6.14, we can see that the estimated kernel appears to be a low-pass filtered version of the true kernel. This is not surprising, since the test inputs were highly filtered to begin with. In deconvolving the input spectra, the pseudoinverse technique traded this low-pass filtering for a dramatic reduction in estimation noise.

Depending on the application, this model may or may not be satisfactory. If the model is to be used for output prediction, then this model is likely to be adequate, provided that the desired inputs are not very different, statistically, from that used to identify the model. On the other hand, if inferences about the underlying system were to be drawn from the shape of the kernel, this model would not be satisfactory.

There are at least two possible reasons why the identified kernel differed so dramatically from the theoretical kernel. Either, the inputs were not "rich" enough to permit the system to be identified, or the expansion based on Wiener cascades, which is optimized to predict the output, failed to model the kernel structure.

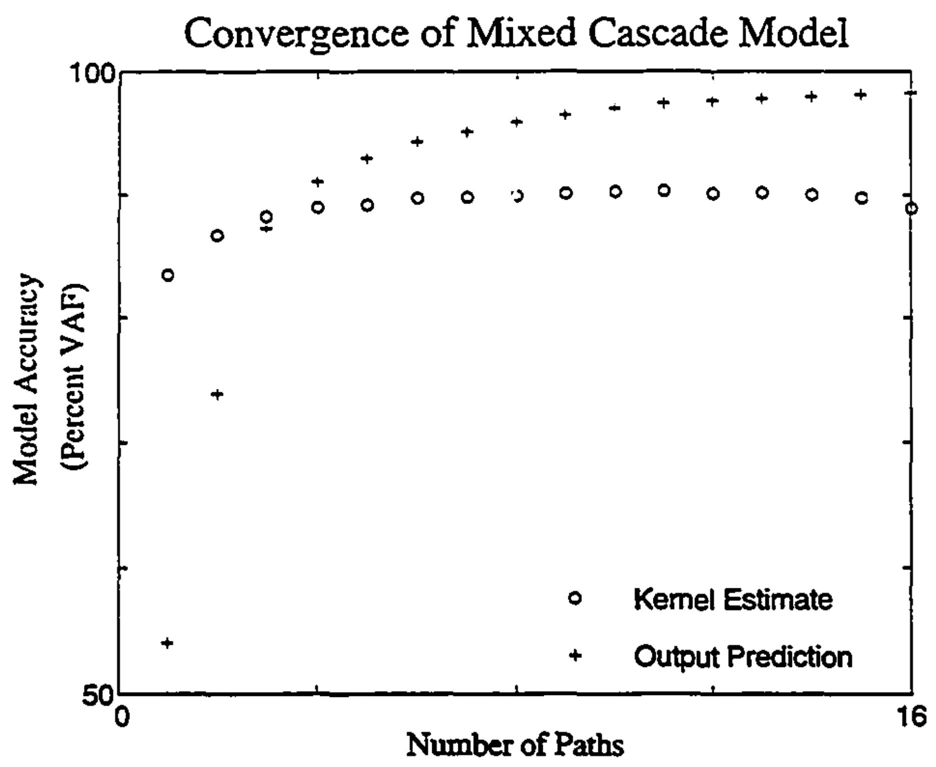


Figure 6.17: Accuracy of the kernel estimate and the output prediction as a function of the number of paths estimated in the high-pass / low-pass experiment. Here, the first path was a Hammerstein system.

If the inputs prevented the system from being identifiable, no identification method would succeed. On the other hand, using a more appropriate model structure may result in a model that predicts both the system output and the kernel shape.

6.1.4.1 Use of *a priori* information

The system kernel shown in Figure 6.14 has most of its energy concentrated near the diagonal, which is what would be expected for a Hammerstein structure. This is due to the high-pass filters that precede the analogue multiplier, whose impulse responses have very short memories.

The rationale behind the selection of the optimal Wiener cascade at each stage in the parallel cascade method, was to reduce the estimation noise by minimizing the number of paths required in the model. However, for a Hammerstein-like structure even the optimal Wiener system will be a poor choice as a cascade path, since it can only account for a small fraction of the kernel. Many such paths will be needed, each contributing estimation noise to the model. As a result, the overall level of estimation noise will be high.

Since the system appeared to be nearly Hammerstein and the identification based on a parallel cascade of Wiener systems failed to produce an adequate model, we attempted to fit the first path using a Hammerstein structure. The Hammerstein structure, by itself, cannot represent nonlinear systems whose Volterra kernels have non-zero off-diagonal values. It will, however, model much of the system dynamics which cannot be modelled efficiently by a sum of Wiener cascades. The remaining dynamics, due to off-diagonal elements in the second-order cross-kernel, could then be modelled using Wiener cascade paths.

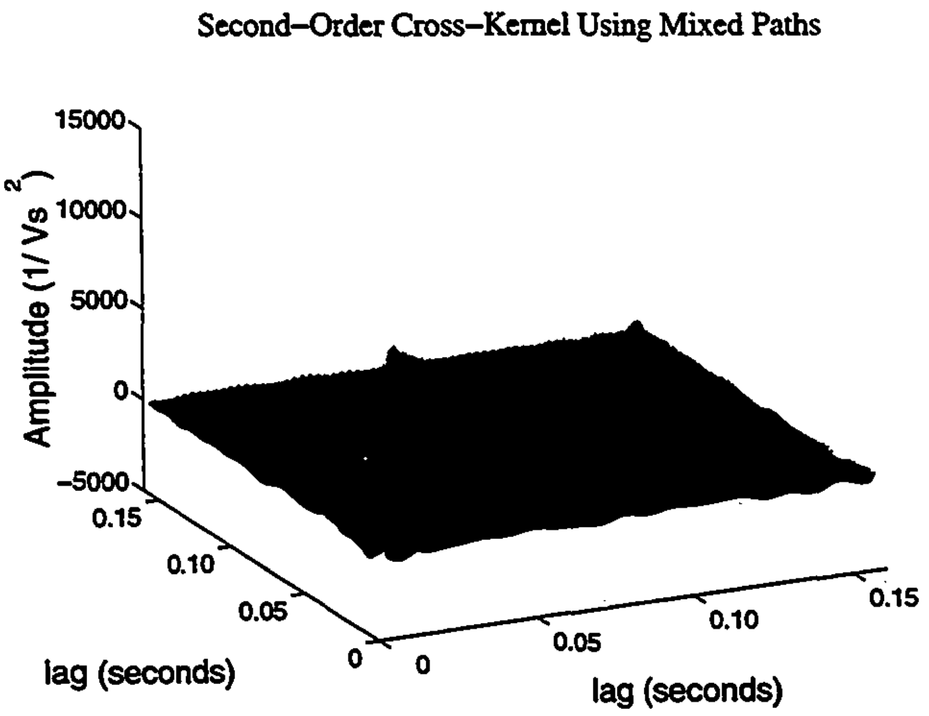


Figure 6.18: Cross-kernel estimated using one Hammerstein path and 10 Wiener paths. This kernel accounts for 90.28% of the variance of the true kernel.

As we were fitting a second-order cross-kernel, the input to the linear element was taken to be the product of the two input signals. We fitted a linear system between this product and the measured output, and computed the residuals remaining after the output of this path had been removed from the measured output. The remaining dynamics were modelled using a parallel Wiener cascade fitted between the inputs and these residuals.

The kernel and prediction accuracies obtained using this mixed structure model are plotted in Figure 6.17. The Hammerstein pathway accounted for 85% of the variance of the kernel, and 54% of the variance of the output. Adding 10 Wiener paths to the model increased the prediction accuracy to 97.42% and the kernel accuracy to 90.28%.

6.1.5 Summary

In this section, we identified three "real" two-input nonlinear systems using the methods developed in this thesis. System configurations were chosen to represent a wide variety of nonlinear systems, and to highlight practical difficulties which may arise in the identification of physical systems.

We showed that our algorithms estimate models which predict the system output very well. When the system kernels can be expanded efficiently as a sum of Wiener systems, the estimated models have kernels which are accurate estimates of the theoretical system kernels.

If the system kernels are concentrated near their diagonals, which will be the case for Hammerstein and nearly Hammerstein systems, an expansion based on Wiener systems is not efficient, and many pathways will be required to model the kernel. This can lead to unacceptable noise performance, and failure of the identification. In this case, Hammerstein paths can be added to the expansion.

In all cases, the pseudoinverse deconvolution algorithm was required to suppress high frequency ringing in the kernel estimates. The order selection criteria developed for SISO linear and nonlinear systems are not applicable to multiple-input Wiener cascades identified from multiple-input cross-correlation functions. In this case, the pseudoinverse orders appear to be co-dependent. Although computationally expensive, an exhaustive search can be used to select the pseudoinverse orders which result in the minimum residual variance.

6.2 Parallel Cascade Model of Reflex EMG

In this section, the methods developed in the previous 3 chapters will be applied to the identification of the relationship between the angular position of the ankle and the resulting EMG. This relationship, the stretch reflex, has been the subject of extensive study (for a review see [38]). It is known to be nonlinear, and the output (EMG) measurements are contaminated with high levels of noise.

This "noise" is dominated by the EMG of ongoing voluntary activity in the muscles, which is used to keep the muscles taut and the reflex active. There are likely to be nonlinear interactions between the background and reflex EMGs which are much more complex that the usual additive noise model assumed in the development of identification methods. In addition to the background EMG activity, there are other, more conventional, noise components, such as thermal noise in the electrodes and amplifiers, discretization noise, and the pick up of 60 Hz signals in the electrode leads.

One data set from a classical experimental paradigm will be analyzed in detail to illustrate how each algorithm may be applied. Based on the results from this analysis, we will identify potential shortcomings of the paradigm and suggest refinements which may address them.

6.2.1 Experimental Methods

Subjects lay supine, with their left foot attached to a rotary hydraulic actuator [39, 117] by means of a custom-fitted fibreglass boot [75]. The subject's ankle was constrained to rotate about its transverse axis, which was aligned with the rotational axis of the actuator. The torque produced by the subject was measured, low pass filtered, and fed back to an oscilloscope mounted above the subjects head. The subject was asked to make the torque feedback track a "command" signal displayed on the oscilloscope. With minimal training, subjects were capable of matching the two signals and producing a pre-determined torque level under the control of the experimenter.

Ankle position was measured via a precision potentiometer [2] on the axis of rotation. Torque was measured by a torque transducer [3] mounted on the shaft between the actuator and the subject, whose stiffness, 10^5Nm/rad, was much greater than that of the ankle. The maximum nonlinearities of the potentiometer and torque transducer were $\pm$ 0.2% and $\pm$ 0.1%, respectively.

EMGs fro_ the Triceps Surae (TS) and Tibialis Anterior (TA) were measured using bipolar Ag/AgCl electrodes [4] placed on the muscle bellies and aligned parallel to the muscle fibres. A reference electrode was placed on the patella. The EMGs were amplified using custom-built pre-amplifiers [82], high-pass filtered at 10 Hz [5], and full-wave rectified.

All signals were anti-alias filtered using 8-pole constant delay filters [6], and sampled at 1 Khz by 16-bit A/D converters [7].

6.2.2 Identification of Reflex Dynamics

The position of the actuator was servo-controlled to follow a random binary sequence input with a 100 ms switching time. This input was chosen because it tends to produce a large stretch reflex [40, 41]. More traditional inputs, such as white or coloured Gaussian noise, have been shown to suppress the stretch reflex [87].

The input signal was taken to be the angular velocity of the ankle, obtained by numerically differentiating the position signal. Ankle velocity and TS EMG signals were decimated 5 times, resulting in a final sampling rate of 200 Hz. The position, velocity and TS EMG are shown in Figure 6.19, along with an estimate of the probability density of the computed ankle velocity.

The input consisted of a 5 second sequence which was repeated 8 times, resulting in 40 seconds, or 8,000 samples, of data. Prior to the system identification analysis,

[2]Beckman 6273-R5K, Beckman Industrial, Fullerton, CA
[3]Lebow 2110-5K, Eaton Corp., Troy NY
[4]Jason, ElectroTrace ET301, Huntington Beach, CA
[5]Frequency Devices, 772BT-2
[6]Frequency Devices, 64PF
[7]IOTech ADC488

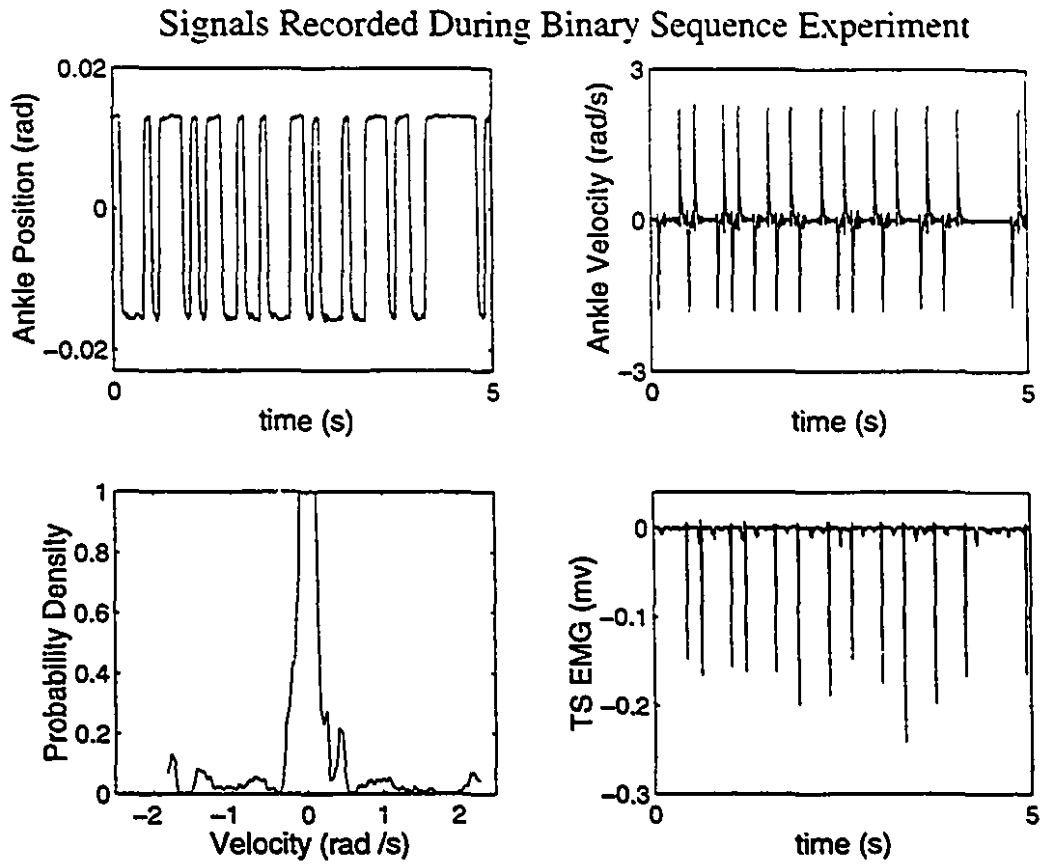


Figure 6.19: Signals from the binary sequence input experiment. Note that the vertical scale in the third panel was chosen to show details away from the centre. Thus, the central peak has been truncated.

the data was tested for stationarity. Since the input sequence was repeated each time, one would expect the output sequences to be identical, except for the effects of noise. We compared each pair of output segments by evaluating the the %VAF between them. These results are presented in Table 6.2.

If the system remained stationary throughout the experiment, we would expect all of off-diagonal entries in Table 6.2 to be approximately equal. From this table, we note that all comparisons between segments 6 – 8 yield greater than 90%VAF. Similarly, comparisons between segments 1 – 3 yield better than 89%VAF. However, comparisons between other pairs of segments are as low as 75%VAF. These results suggest that the system may have changed during the experiment.

One possibility is that the subject adjusted the level of background contraction,

	1	2	3	4	5	6	7	8
1	100							
2	92.5	100						
3	89.3	96.0	100					
4	82.4	82.3	79.2	100				
5	89.4	94.2	93.1	83.6	100			
6	89.3	92.0	89.7	80.6	91.6	100		
7	86.8	87.4	85.5	78.3	91.4	94.2	100	
8	82.1	86.1	81.8	75.7	89.0	93.7	95.3	1.00

Table 6.2: Results of the stationarity analysis applied to the binary sequence data. The (i, j)'th entry is the %VAF evaluated between the i'th and j'th segments of the output.

to better track the command signal. If this were the case, we would expect to see a change in the variance of the background EMG. Since the reflex contribution to the EMG consisted of a series of large spikes, it could be eliminated by a simple thresholding operation. We estimated the variance of the background component of the EMG in each of the 8 segments. The estimated background EMG variance is presented in Table 6.3.

Segment Number	1	2	3	4	5	6	7	8
Background EMG Variance (μV^2)	11.0	12.2	12.1	8.5	9.6	8.4	6.8	6.8

Table 6.3: Variance of the background EMG in each data segment.

This table suggests that the level of background torque, and hence the background EMG, underwent a substantial change between the first 3 segments and the last 2–3 segments. Thus, only the last 2,000 samples (10 seconds) of data were retained for further analysis. Of that, 1,000 points were used in the identification of the models while the remaining 1,000 points were set aside for model validation.

Several analyses were performed, which will be presented in the next sections. These results will be summarized in Table 6.4 on page 168.

6.2.2.1 Wiener Cascade Model

We used the optimized parallel cascade algorithm to identify a model of the stretch reflex dynamics. Thus, a parallel array of Wiener systems was fitted between the ankle velocity and the TS EMG. The first Wiener cascade was based on the first-order cross-correlation between the input (ankle angular velocity) and the output (TS EMG).

The linear part of this first path, shown in Figure 6.20 was identified by fitting a linear filter between the velocity and EMG signals. This linear model accounted for 55.01% of the variance of in the validation segment.

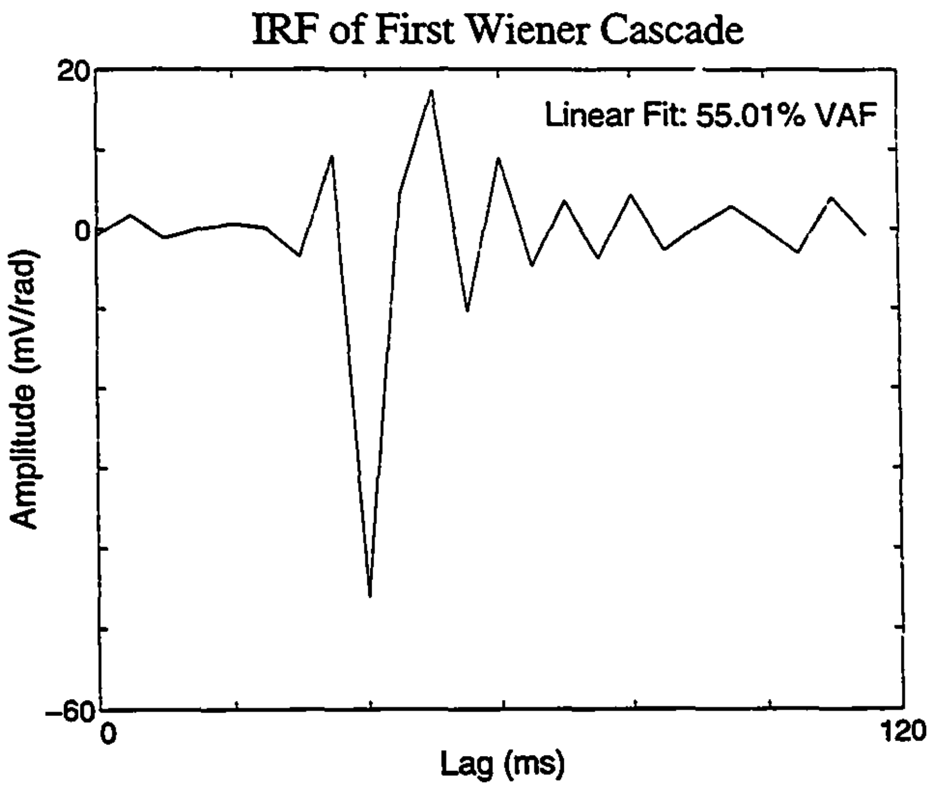


Figure 6.20: IRF for the first Wiener cascade identified between velocity and EMG.

A Tchebyshev polynomial was fitted between the output of the linear system and the measured EMG. Polynomials of orders 0 through 8 were fitted, and the %VAF between the EMG and the polynomial output computed. As no significant increase in prediction accuracy was noted beyond order 5, a fifth-order polynomial was used. The resulting Wiener system accounted for 94.64% of the measured EMG variance in the validation segment. The output of this Wiener cascade was computed, and subtracted from the measured EMG signal, resulting in the first set of output residuals.

We then applied the test described in Equation (5.5) to the second-order correlation matrix evaluated between the input and the residuals, and found that its principal eigenvalue was a factor of 6.05 times greater than the significance threshold. This suggested that there were significant dynamics present in the second-order cross-correlation between the input and the residuals. Therefore, the linear subsystem of the next path was based on the principal generalized eigenvector of the second-order cross-correlation between the input and the output residuals. This path accounted for 5.0% of the residual variance, increasing the model accuracy to 94.84%VAF.

The principal eigenvalue of the next second-order input-residual cross-correlation was a factor of 3.56 greater than the threshold, again suggesting that a significant pathway could be constructed. This pathway accounted for 2.01% of the residuals, raising the model accuracy to 94.91%.

Again, we tested the residuals for the possibility of adding a path based on the second-order cross-correlation. The test returned a value of 2.45. However, we were unable to construct a Wiener path which increased the accuracy of the prediction during the validation segment, although small (2-3% depending on the order of the static nonlinearity) increases were observed during the identification segment. Thus, the optimal Wiener pathway based on the second-order cross-correlation appeared to model noise despite the apparent significance of that correlation function, as suggested by Equation (5.5).

This observation was puzzling, so we examined the second-order cross-correlation function, which is plotted in Figure 6.21. This correlation function is concentrated near the diagonal, which suggests either a Hammerstein system or an LNL structure in which the first linear element has a very short memory. As noted previously, representing either of these structures by a parallel sum of Wiener cascades will be very inefficient. Thus, even though there were significant unmodelled dynamics remaining, no single Wiener path was able to model enough to dynamics to overcome the effects of noise. An alternative approach would be to use paths whose structure

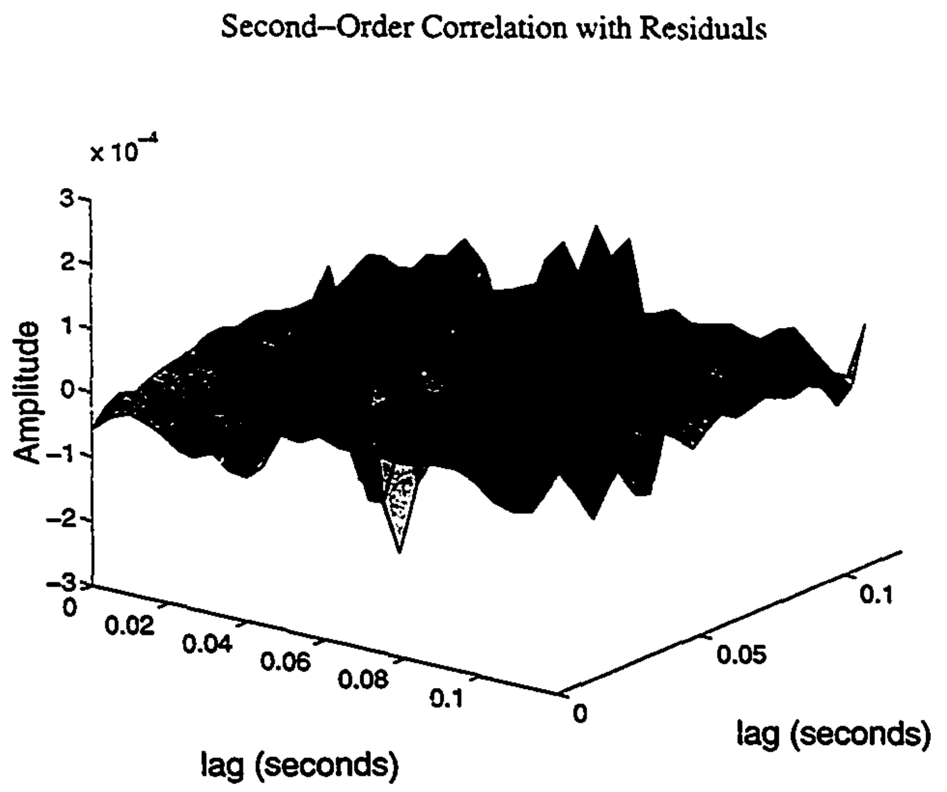


Figure 6.21: Second-order correlation between ankle velocity and EMG residuals

more closely approximates that of the underlying system.

6.2.2.2 Hammerstein Model

The stretch reflex has often been modelled as a Hammerstein system [36, 37, 44, 45] because of the observation that the EMG responds to stretches of the muscle, while rapid shortening of the muscle produces no change. Thus, the static nonlinearity is sometimes assumed to be a half-wave rectifier, which is similar to the shape obtained when the static nonlinearity is identified explicitly [37].

We used the iterative algorithm proposed by Hunter and Korenberg [31], and described in Section 2.3.2.1, to fit a Hammerstein model between the ankle velocity and the EMG. The only modification made was to use the pseudoinverse input deconvolution to estimate the linear subsystem. This system, whose elements are shown in Figure 6.22, accounted for 94.41% of the EMG variance in the identification segment and 95.07% in the validation segment, somewhat better than the results achieved

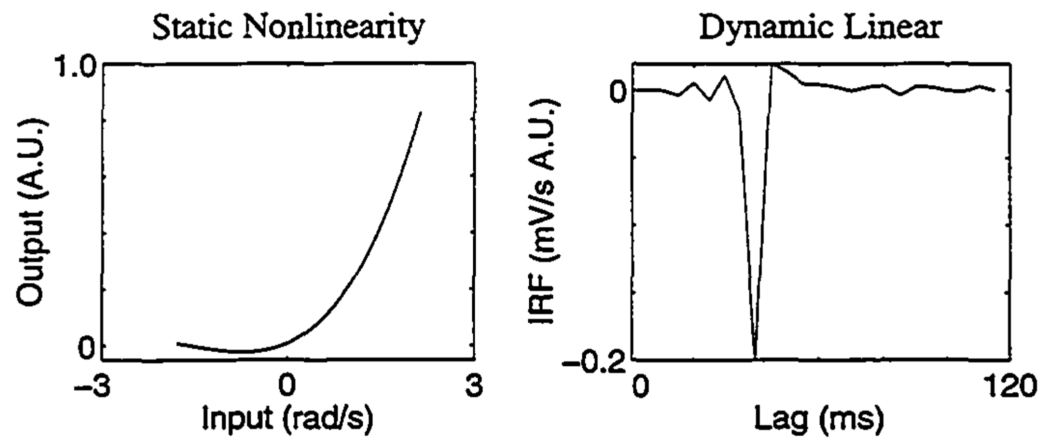


Figure 6.22: Static nonlinear element (left) and dynamic linear subsystem (right) of the Hammerstein model identified for the stretch reflex EMG. Note that the output of the static nonlinearity is given in arbitrary units (A.U.), as the gain associated with that signal is unknown.

with the Wiener cascade model.

6.2.2.3 Mixed Cascade Model

In Section 6.1.4.1, we noted that the Hammerstein structure is very efficient for modelling the diagonals of kernels but cannot account for off-diagonal terms. In that analysis, we had used an initial Hammerstein path to capture the kernel diagonal, and then added several Wiener cascades to model the off-diagonal kernel elements. We tried using a similar approach to model reflex dynamics. After computing the initial Hammerstein pathway, its output was subtracted from the measured EMG, and a second pathway was fit between the input and these residuals.

Unlike fitting a Wiener cascade, estimating the linear dynamics of a Hammerstein system from the first-order cross-correlation does not necessarily result in residuals which are uncorrelated with the input, since Lemma 2 and its corollary do not apply in this case. Thus, a Wiener cascade, whose linear dynamics were based on the first-order input-residual cross-correlation, was used to drive the first-order correlation to zero. It accounted for 2.25% of the residual variance, increasing the prediction accuracy to 95.24%.

We then attempted to fit paths based on the second-order cross-correlation. The

Wiener Cascade Model				
path	description	VAF(incr)	VAF(total)	VAF (valid.)
1	Wiener (odd)	93.5494	93.5494	94.6407
2	Wiener (even)	4.9974	93.8717	94.8405
3	Wiener (even)	2.0105	93.9949	94.9107

Hammerstein Model				
path	description	VAF(incr)	VAF(total)	VAF (valid.)
1	Hammerstein (odd)	94.4217	94.4217	95.0867

Mixed Wiener/Hammerstein Cascade Model				
path	description	VAF(incr)	VAF(total)	VAF (valid.)
1	Hammerstein (odd)	94.4217	94.4217	95.0867
2	Wiener (odd)	2.2493	94.5472	95.2435

Table 6.4: Summary of the stretch reflex models identified using a binary pulse sequence input. Model accuracy is reported as the percent variance accounted for (VAF). Incremental reports the fit of the current path to the residuals, total is the fit of the model to the identification data, and valid is the fit of the model to the validation set.

principal eigenvalue was a factor of 2.42 greater than the threshold given by (5.5), suggesting that there were significant dynamics present in the second-order cross-correlation. However, no significant Wiener cascade could be identified. As in the Wiener expansion, the second-order cross-correlation was concentrated near its diagonal, suggesting a Hammerstein-like structure.

The results of these experiments are summarized in Table 6.4. In all cases, results obtained with the validation data closely match those obtained with the identification segment, indicating that the models are unlikely to be fitting noise. This is an important observation regarding both the Wiener cascade and mixed cascade expansions, because many of the pathways accounted for very little of the residuals ($< 5\%$). Since these pathways resulted in similar improvements in both the identification and validation segments, we can conclude that they were in fact significant.

6.2.3 Discussion

These experiments raise several interesting questions about the identifiability of nonlinear systems, and how this relates to experimental constraints.

It is clear that the binary pulse sequence input is not well suited to the identification of Hammerstein models since these models include a static nonlinear transformation on the input. Clearly, the nonlinearity will only be well defined for values of the input which are commonly present in the identification input. From the third panel of Figure 6.19, it is evident that the nonlinearity will be well characterized near zero velocity, and perhaps at the extremal velocities. The region between these extremes, however, will not be probed by the input.

Furthermore, the velocity input consists of a train of narrow pulses. Thus, it is only able to excite the system kernels near their diagonals. For Hammerstein-like systems, this does not pose a serious problem since the kernels will be concentrated near their diagonals.

These problems could be addressed by using a Gaussian noise input of adequate bandwidth. The nonlinearities of any Hammerstein pathways would be well defined, since all intermediate input levels would be probed. Furthermore, the correlation structure of the input would ensure that all regions of the kernels were excited. However, the use of Gaussian noise inputs results in relatively weak reflex components, and therefore a low output SNR and poor prediction accuracy (≈50% VAF [111]). Experiments performed using Gaussian noise inputs support the assumption of a Hammerstein-like structure. However, since the resulting output SNR is so low, adequate models cannot be identified. Thus, an alternative input is required.

The design of an appropriate input signal is an important step in improving available models of the stretch reflex. The test signal must be rich enough, both in amplitude and spectral content, to excite all relevant modes in the system. However, it must also have characteristics which ensure a strong reflex and therefore high SNR. Currently, the effect of different perturbations on the reflex amplitude is being investigated [79, 86, 87]. The results of this investigation will be used to design an "optimal" input signal. This will certainly involve a compromise between the generality of the identified model, and the output SNR and resulting prediction accuracy.

Another important issue raised by these experiments is the selection of an appropriate model structure. In Section 6.1.4, where the identification of the high-pass/low-pass system was considered, we noted that Hammerstein-like systems cannot be efficiently represented by an expansion based on Wiener systems. This point was also raised in the stretch reflex identification. In both cases, this difficulty was overcome by using a mixed cascade model in which the first path was a Hammerstein system. While the Wiener cascade expansion is very general, there are cases where different expansions may be more efficient. It is important to realize this, and to try other system types if the parallel Wiener cascade should fail to produce an adequate model.

Chapter 7

Conclusions

In this thesis, we have developed practical methods for the identification of linear and nonlinear systems which are applicable under the relatively stringent constraints that exist in the experimental study of human joint dynamics. While these techniques were designed specifically with these constraints in mind, they should be applicable to experiments on a variety of systems in many different fields.

7.1 Statement of Original Contributions

The following is a list of the original contributions contained in this thesis.

1. *A new method for the nonparametric identification of linear systems was developed.* Matrix perturbation analysis was used to develop an improved technique for the deconvolution of the input auto-correlation from the input-output cross-correlation in the nonparametric identification of linear systems. This new technique dramatically reduces the variance of IRF estimates, and provides confidence limits on the resulting estimate.

2. *A new method for the identification of multiple-input Wiener systems was developed.* The interaction between inputs in a multiple-input nonlinear system were shown to produce interference in estimates of some parts of the system dynamics. An iterative technique was developed which predicted and then eliminated

much of this interference. thereby increasing the accuracy of the estimated system.

3. *An optimized variant of the parallel cascade method was developed.* The parallel cascade method [56] for the identification of nonlinear systems was modified such that each iteration added the optimal Wiener cascade (in an MMSE sense) to the model. This results in several advantages over the original algorithm:
   - Models have fewer paths, and hence fewer parameters.
   - Convergence is much faster.
   - The model is less sensitive to noise.
   - The model is unique (given the input auto-correlation).
   - Once an insignificant pathway is estimated, analysis can be stopped, as no further paths will be significant.

7.2 Suggestions for Further Research

Much work remains to be done, both in terms of algorithmic development, and in terms of the application of these techniques to real problems.

7.2.1 Further Investigation of Pseudoinverse Input Deconvolution

While the use of the a pseudoinverse in the deconvolution of the input autocorrelation from the impulse response estimate represents a major innovation in nonparametric system identification, many questions remain unanswered. In this section, we outline several possible avenues of research related to this topic.

7.2.1.1 Noise in the Input Measurement

The effects of input noise were ignored in the study of the deconvolution of the input auto-correlation from the impulse response estimate. If instead of using the noise-free input, $u(t)$, we let the measured input be:

$$\hat{u}(t) = u(t) + v_u(t)$$

using the notation established by Figure 2.1, then the input auto-correlation becomes:

$$\phi_{\hat{u}\hat{u}}(\tau) = \phi_{uu}(\tau) + \phi_{v_u v_u}(\tau)$$

provided the input, $u(t)$ and the input noise, $v_u(t)$ are independent. We will, of course, be working with a time-average evaluated from finite length records. Thus, our estimate of the input auto-correlation becomes:

$$\begin{aligned}
\hat{\phi}_{\hat{u}\hat{u}}(\tau) &= \phi_{\hat{u}\hat{u}}(\tau) + \tilde{\phi}_{\hat{u}\hat{u}} \\
&= \phi_{uu}(\tau) + \phi_{v_u v_u}(\tau) + \tilde{\phi}_{uu}(\tau) + \tilde{\phi}_{uv_u}(\tau) + \tilde{\phi}_{v_u u}(\tau) + \tilde{\phi}_{v_u v_u}(\tau)
\end{aligned}$$

Similarly, we must use a finite-length time average to estimate the input-output cross-correlation:

$$\hat{\phi}_{uy}(\tau) = \phi_{uy}(\tau) + \tilde{\phi}_{uy}(\tau) + \tilde{\phi}_{uv_z}(\tau) + \tilde{\phi}_{v_u y}(\tau) + \tilde{\phi}_{v_u v_z}(\tau)$$

In performing the perturbation expansion on the inverse of the auto-correlation matrix, we must decide what to treat as "signal", and what to treat as "noise". The obvious choice would be to treat ϕ_{uu} as "signal", and the rest as a perturbation. However, because of the size of the resulting perturbation, the first-order expansion is likely only to be valid for high input SNRs. Instead, we will partition the input-autocorrelation estimate as follows.

$$\begin{aligned}
\phi_{\hat{u}\hat{u}}(\tau) &= \phi_{uu}(\tau) + \phi_{v_u v_u}(\tau) \\
\tilde{\phi}_{\hat{u}\hat{u}} &= \tilde{\phi}_{uu}(\tau) + \tilde{\phi}_{uv_u}(\tau) + \tilde{\phi}_{v_u u}(\tau)\tilde{\phi}_{v_u v_u}(\tau)
\end{aligned}$$

Given that the terms in $\bar{\dot\phi}_{\dot u\dot u}$ are due to the approximation, from finite length time-averages, of the correlation between independent signals, they will be small compared to $\dot\phi_{\dot u\dot u}$, and should not compromise the validity of the first-order perturbation expansion on the matrix inversion. If we repeat the analysis that lead to Equation (3.8), we get:

$$\hat h = \dot\phi_{\dot u\dot u}^{-1}\dot\phi_{uu}h + \dot\phi_{\dot u\dot u}^{-1}\bar{\dot\phi}_{uv_z} - \dot\phi_{\dot u\dot u}^{-1}\bar{\dot\phi}_{uv_u}$$

The effect of the input noise is two-fold. First, there is an additive term of the same form as that due to output noise. Thus, the analysis developed in Chapter 3 should apply equally to this term. While the analysis of this term will be similar to that for the output noise term, it will affect the choice of the order of the pseudoinverse used in the deconvolution.

Secondly, the input noise induces a distortion in the "signal" term. In the previous case, we had an unbiased estimate, provided the exact inverse of the auto-correlation matrix was used. Now:

$$E[\hat h] = \phi_{\dot u\dot u}^{-1}\phi_{uu}h$$

Methods which estimate, and compensate for, this distortion need to be developed.

7.2.1.2 Application to Multiple-Input Nonlinear Systems

In the multiple-input case, the cross-correlations across a pair of linear systems are obtained from the principal left and right singular vectors of the second-order cross-cross-correlation. The input auto-correlation functions are then deconvolved explicitly. As might be expected, this deconvolution often leads to high frequency "deconvolution noise" in the IRF estimates. Use of appropriate pseudoinverses, one for each input, can reduce the variance of this noise. In this thesis, the pseudoinverse orders were chosen using an exhaustive search procedure.

In the single-input case, impulse responses were extracted from the second-order cross-correlation function by solving a generalized eigenvalue problem with the input auto-correlation matrix. Here, the deconvolution was performed implicitly, which lead to an equivalent linear problem. The solution of this equivalent problem, using the techniques developed for deconvolution from first-order cross-correlations, yielded the pseudoinverse order.

The restricted singular value decomposition (RSVD) [16] could be used to generate a pair of "equivalent linear" identification problems, and would accomplish the deconvolution implicitly, as the generalized eigenvalue solution does for single-input systems. Optimality of the RSVD solution should be relatively simple to prove.

Applying the pseudoinverse based deconvolution algorithm to this problem will yield two pseudoinverse orders, one for each input. It is likely that they will be interdependent, leading to coupled solutions.

All of this, however, is speculative, since algorithms for the computation of the RSVD are not yet available. When, or perhaps if, these algorithms are developed, research into this multiple-input method may proceed.

7.2.1.3 Estimation of the Bias Error

Ideally, we would like to be able to estimate the absolute value of the induced bias, at each point in the impulse response estimate. In this work, we succeeded in computing its infinity norm, which gives the maximum absolute value of the bias over the whole IRF. If the bias error were equally distributed over the IRF, this would be sufficient. Unfortunately, simulations show the bias error is often concentrated around the largest peaks in the impulse response. Thus, using the infinity norm overestimates the bias error for most of the impulse response, leading to very conservative confidence bounds on the IRF estimate. Further research is needed to improve the mathematical understanding of the bias induced by the use of the pseudoinverse, and to obtain a localized estimate of this error.

The solution presented in Chapter 3 uses the results from extensive Monte Carlo simulations to approximate the relationship between the decrease in the variance of the system output, and the infinity norm of the induced bias. It would be preferable to have an analytical expression, rather than the empirically derived distribution function.

7.2.2 Use of Generalized Wiener Models

We have shown that the parallel cascade method, as implemented in Chapter 5, can completely model the zero and first order Wiener kernels with a single path, and that only T paths are necessary to model all of the dynamics in the second-order Wiener kernel. Any remaining unmodelled dynamics must be estimated from the third-order cross-correlation. However, given that the IRFs themselves are only T points in length, it is clear that any new IRFs will be linear combinations of previously identified linear elements. Even though the linear elements in the cascades completely span the subspace occupied by the dynamics of the unknown system, additional paths may be needed because the single-input nonlinearities do not allow for cross-terms between the impulse responses which form a basis of that subspace.

An alternative approach would be to adopt a more generalized Wiener structure, such as that used by Wiener in his original monograph [113] and more recently by Marmarelis [71]. In this structure, the input signal will be processed by a bank of linear filters. The outputs of the filter bank could then be processed by a multiple-input nonlinearity, which would provide the cross-terms that are missing in the parallel cascade structure. However, we suggest using the linear elements identified by the optimized parallel cascade method, rather than the basis of Laguerre filters employed by Wiener [113] and Marmarelis [71]. Figure 7.1 illustrates the generalized Wiener model.

The biggest difficulty with this approach is likely to be the identification of the static nonlinearity. In Section 4.3.2, we described the difficulties encountered in the specification of the domain of definition for a two-input nonlinearity. The severity of these problems will likely increase with the number of inputs to the nonlinearity.

7.2.3 Use of Hammerstein and LNL Cascade Paths

Any time-invariant system which has a fading memory [9] can be represented by an expansion based on Wiener systems. However, even the optimal expansion of this type, which is identified by the algorithms developed in this thesis, is not necessar-

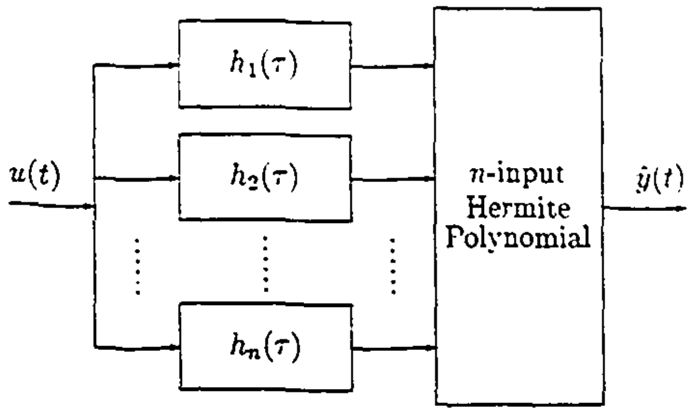


Figure 7.1: The generalized Wiener system structure

ily very efficient. For example, when the system in question has a "Hammerstein like" structure, a Hammerstein system optionally preceded by a short memory linear dynamic element, the Volterra kernels are concentrated near their diagonals. As a result, many linear elements are required to represent the space spanned by the system dynamics. Thus, any representation based solely on Wiener systems, whether a parallel cascade, or a generalized Wiener model, will require a large number of linear elements. Clearly, in these situations, methods are needed for creating Hammerstein or LNL paths which are somehow optimal. These methods must not assume that the system being identified has the particular structure in question.

For example, Korenberg [57] developed a method for the identification of the optimal Hammerstein system between a given input and output, which makes no assumptions about the structure of the true system. Unfortunately, this algorithm requires a white input, and is therefore impractical in many settings.

What is required, is an algorithm that finds the LNL system that generates the maximum reduction in the mean-squared size of the second (or higher) order cross-correlation function. This is the approach used to find the "optimal" Wiener pathway, where the IRF turned out to be the principal eigenvector of the second-order cross-correlation. The second-order kernel of an LNL cascade is:

$$k_2(\tau_1, \tau_2) = \sum_{\sigma=0}^{T} g(\sigma)h(\tau_1 - \sigma)h(\tau_2 - \sigma)$$

where $h(\tau)$ is the IRF of the first linear subsystem, and $g(\tau)$ is the IRF of the second linear element. Thus, we would search for IRFs g and h which minimize the error:

$$e(g, h) = \sum_{\tau_1, \tau 2=0}^{T} \left(\phi_{uuz}(\tau_1, \tau_2) - \sum_{\sigma=0}^{T} g(\sigma)h(\tau_1 - \sigma)h(\tau_2 - \sigma) \right)^2 \tag{7.1}$$

A brute force approach to this problem would be to minimize (7.1) using a numerical gradient descent algorithm.

Equation (7.1) assumes that the input signal is white. Therefore, as in the Wiener cascade case, stable and robust procedures must be developed for the deconvolution of the input autocorrelation from the resulting IRF estimates.

7.2.4 Use of Subspace System Identification Methods

Recently, the MOESP algorithm [96, 98, 99], which identifies discrete state space models of LTI systems in a subspace model identification context, has been applied to MIMO Hammerstein [100] and Wiener [101] systems. The extension to Wiener systems is particularly interesting, since these systems form the "building blocks" of the parallel cascade representation used throughout this thesis. A particularly exciting property of the state space models is that the extension from single-input single-output (SISO) systems to multiple-input multiple-output (MIMO) systems is trivial.

The nonlinear extensions of the subspace methods have been made, to date, under the assumption that the underlying system had a Wiener or Hammerstein structure. The application of the Wiener system extension of MOESP to parallel cascade identification will require dropping this assumption. The existing algorithm will have to be modified to find the "optimal" Wiener system between a given input and output, with no assumptions about the form of the actual system. Once this has been achieved, the general approach outlined in Chapter 5 can be used.

7.2.5 Applications to the study of joint dynamics

The techniques developed in this thesis, were designed with the constraints typical of joint dynamics experiments in mind. Here, we suggest applications in the field of joint dynamics, where these techniques may prove useful.

7.2.5.1 Use of Narrow-Band Perturbations

Wide-band perturbations, such as white noise (albeit filtered by an electro-hydraulic actuator), have been shown to suppress the stretch reflex in humans [40, 41]. Recent experiments have sought to characterize stretch reflex dynamics [42, 43], and determine the extent of their contribution to the overall dynamics of the ankle, using relatively narrow-band perturbations. These perturbations produce a strong stretch reflex, but exacerbate the problems caused by the input deconvolution. This seems to be a natural application for the pseudoinverse deconvolution technique, developed in Chapter 3, particularly, if it can be extended to account for input noise.

7.2.5.2 Multiple-Input Experiments

To date, multiple-input experiments performed on the human neuromuscular system have linearized the system about a time-varying trajectory [44, 45, 67, 92]. It would be interesting to repeat some of these experiments using a "richer" stimulus for the second input instead of the repeated steps or "fast ramps" which were used to modulate the "operating point" in the time-varying context. The multiple-input techniques developed in this document could then be used to produce a more complete description of the system. The time-varying linearized description could be derived analytically from the multiple-input nonlinear description, and compared to the results of previous experiments which used the linear time-varying system description.

In principal, a truly multiple-input system description should provide the investigator with more freedom in the design of the experimental protocol. For example, time-varying techniques require the operating point to traverse exactly (or almost exactly, in practice) the same trajectory during each trial. Much processing effort is required to reject trials in which the operating point trajectory deviates from the regime, and in aligning the trajectories of those trials which are retained for further analysis [66]. Using the algorithms presented herein would free the experimenter from these constraints.

Appendix A

A MATLAB toolbox for nonlinear system identification

A.1 Data Structures

A.1.1 Second-Order Wiener/Volterra Kernels

A second-order kernel or correlation function can be represented as a matrix. Hence, the MATLAB language is used to store these functions. In general, the first index is used to represent lags with respect to the first input signal. MATLAB itself, stores matrices in column major order, as is done in FORTRAN, as opposed to the row major ordering that is used by C.

A.1.2 Third-Order Wiener/Volterra Kernels

Third-order kernels are functions of three indices, $h(i,j,k)$. MATLAB itself only provides for matrices which have two indices. The functions in this toolbox represent third-order kernels and correlation functions as vectors. It is assumed that the third-order object is cubic. Therefore the range of the indices i,j and k are all equal to the cube-root of the length of the vector. If H_3 is an n by n by n kernel, and it is represented by the n^3 long vector h, the entry $H_3(i,j,k)$ is stored in $h(i*(n-1)^2+j*(n-1)+k)$. This conversion is accomplished by the c function *ma-*

trix_ref, which is linked to all of the mex-files that operate on third-order correlations and kernels.

A.1.3 Tchebyshev Polynomials

A crucial step in the estimation of a Tchebyshev polynomial is the scaling of the input to the range $[-1, 1]$. If $x(n)$ are the input samples, and a and b represent the minimum and maximum of the x, then this scaling is accomplished as:

$$\hat{x}(n) = \frac{b - 2x(n) + a}{a - b} \tag{A.1}$$

Hence, in order to completely specify the polynomial, both the limits a and b, and the polynomial coefficients themselves need to be specified. As a result, the Tchebyshev polynomials are stored as the column vector $[abc_0c_1 \ldots c_n]^T$. The interval $[ab]$ can also be used as the domain of definition of the Tchebyshev polynomial.

For two-input Tchebyshev polynomials, Equation (A.1) is used to scale each of the inputs. Hence, the first four elements of the polynomial representation represent the domain bounds. The remaining coefficients are stored in order of ascending degree. Terms of equal degree are stored starting with the term that has the highest degree in the first input.

Hence, a two input Tchebyshev polynomial would be stored as:

$$[a_xb_xa_yb_yc_0c_xc_yc_{x^2}c_{xy}c_{y^2}c_{x^3} \ldots c_{xy^{(n-1)}}c_{y^n}]^T$$

In the single input case, the interval $[ab]$ served as the domain of definition of the polynomial. In the two input case, things are not quite as simple. In principle, the rectangle bounded by (a_x, a_y), (a_x, b_y), (b_x, b_y) and (b_x, a_y) could be considered to be the domain of definition of the polynomial surface. However, it is highly probable that there will be large parts of this rectangle, where the polynomial estimate is not supported by any data. A somewhat more robust approach would be to measure, and store, the convex hull of all of the data points. This is the smallest closed polygon that enclosed all of the data points. It can be represented by the (x, y) co-ordinates of its vertices. To simplify computations involving the polygon, the first vertex is also stored as the last vertex. If this polygon is included, it is stored following the polynomial coefficients. First the coefficients are stored, followed by a NaN, followed by the x co-ordinates of the vertices, followed by another NaN, and the y co-ordinates of the vertices. If additional polygons are added, they are appended to the end of the structure, separated by NaNs. The maximum and minimum values of the two inputs remain in the first four places, just prior to the polynomial coefficients. While they could be obtained from the maximum and minimum values of the second and third columns, stating them explicitly with the polynomial coefficients allows a polygon other than the convex hull of the data points to be used as the domain boundary.

A.2 List of Functions

A.2.1 Conventions

A.2.1.1 m-files

The first lines of each m-file consist of comments describing the purpose of the routine, as well as the inputs that it requires and the outputs that it generates. All of these comments can be accessed within MATLAB by typing help "routine name". Also included in this documentation is a list of all of the other routines within the toolbox that it calls.

A.2.1.2 mex-files

For every mex-file, there is a parallel m-file. These contain comment lines, as described above, which allow the user to obtain online help using the MATLAB help utility. Furthermore, simply typing the routine name from within MATLAB generates a help message. If the mex-file has not been installed, MATLAB will invoke the m-file, which will print a message informing the user that the mex-file has not been installed. In some cases, m-file implementations have been provided, however, due to the nature of the computations, use of these routines is not recommended. The comments which appear at the head of the c-files are not accessible from within MATLAB.

convex_hull

Syntax: `bound = convex_hull(points)`

Inputs:

`points`	An $N \times 2$ array where each row contains the x and y co-ordinates of one of the N points

Outputs:

`bound`	An $K \times 2$ array, where each row contains the x and y co-ordinates of one of the K vertices of the convex hull around the N points passed in the array `points`

Creates a polygon that surrounds all of the x-y points passed in the array points. The routine starts at the rightmost of the N points, and proceeds counter-clockwise around the boundary. The first and last points in the boundary are equal, forming a closed figure.

This is intended primarily as a **service** routine.
It is called by: `tchebfit2d`

No Local Function Calls:

DTW June 1993

deadzone

Syntax: `v = deadzone (u,limit)`

This function applies a symmetric deadzone static nonlinearity to the signal u. The width of the deadzone is `limit`.

If	u > limit,	v = u - limit
	u < -limit	v = u + limit
	-limit < u < limit	v = 0

No Local Function Calls:

DTW 1992

fil2

Syntax: `[nfil,vaf,bounds]= fil2(u,y,numlags,numsides,level,mode)`

Inputs:

`u`	input signal
`y`	output signal
`numlags`	memory length of the identified IRF
`numsides`	1 for causal systems, 2 for mixed causal/anti-causal systems
`level`	confidence level to be calculated (1 - 100)
`mode`	used to set the pseudoinverse order selection mode, unless the current default is desired (see `toep_man`)

Outputs:

`nfil`	estimated IRF
`vaf`	output variance accounted for by IRF output
`bounds`	confidence bounds on IRF estimate, specified by `level`

Identifies one and two sided impulse responses using the pseudoinverse based deconvolution algorithm. The method used to determine the pseudoinverse order can either be selected using `toep_man`, or it can be specified by specifying `mode` on the command line.

Local Functions Called: `filter_ts,scorr,toep,toep2,toep_man,toep_var`

DTW 1994

gen_kern

Syntax: `kern = gen_kern (basis, coeff, order, type);`

Inputs:

`basis`	basis functions on which the system has been expanded
`coeff`	coefficients associated with Hermite polynomials applied to the basis vectors
`order`	order of the desired kernel
`type`	`volterra` or `wiener`. Only the first letter is significant. Note, that if Volterra kernels are desired, the expansion must include terms or order up to and including the actual system order.

Outputs:

`kern`	kernel estimate

Only first and second-order Wiener kernels are presently supported.

No Local Function Calls:

DTW March 1994

half_wave

Syntax: `v = half_wave (u)`

Applies a half wave rectifier to the signal u

```
If    u > 0    v = u
      u < 0    v = 0
```

No Local Function Calls:

DTW 1991

hard_limit

Syntax: `y = hard_limit (x, h_min, h_max);`

Applies a hard limiter to the input. Bounds of the hard limiter are passed as arguments. If only one value is passed, the hard limiter is assumed to by symmetric about 0

```
If    u > h_max          y = 0
      h_min < u < h_max  y = u
      u < h_min          y = 0
```

No Local Function Calls:

DTW 1991

hard_limit_2d

Syntax: `new_points = hard_limit_2d (points, bound);`

Inputs:

`points`	An N x 2 array containing the x-y co-ordinates of the points to be limited
`bound`	An M x 2 array containing the x-y coordinates of the bounding polygon

Outputs:

`new_points`	An N x 2 array containing the points in points limited to the interior of the polygon defined by bound

Given a set of n points, stored as an $n \times 2$ matrix, and a closed polygon, represented by its m vertices in an $m + 1 \times 2$ array, bound, where the first and last vertex are the same, this function leaves all points in the interior of the polygon untouched, but moves all exterior points to the nearest point on the polygon.

No Local Function Calls:

DTW July 1993

irf_est_cross

Syntax: `[hxlin,hylin] = irf_est_cross (x,y,v,h_len,iterate);`

Inputs:

`x,y`	Inputs to a two-input nonlinear system
`v`	Output of the nonlinear system
`h_len`	(Upper bound on) the anticipated memory length of the system
`iterate`	Use iterative scheme to improve estimates (y/n)

Outputs:

`hxlin,hylin`	Impulse response estimates

Estimates the impulse responses in a two-input Wiener system based on the second-order cross-cross-correlation between the inputs `x` and `y`, and the output, `v`. The memory length of the the impulse responses, `hxlin` and `hylin`, is `h_len` samples. If iterate is `[y]`es, then an iterative technique is used to estimate and cancel interference terms in the correlation estimate.

This is intended primarily as a service routine.
It is called by: `multi_wiener`

Local Functions Called: `phixyz, proj0, tchebfit2d, toep`

DTW Apr 1992

irf_est_even

Syntax: `[hxlin,hylin] = irf_est_even (x,y,v,h_len,iterate);`

Inputs:

`x,y`	Inputs to a two-input nonlinear system
`v`	Output of the nonlinear system
`h_len`	(Upper bound on) the anticipated memory length of the system
`iterate`	Use iterative scheme to improve estimates (y/n)

Outputs:

`hxlin,hylin`	Impulse response estimates

Estimates the impulse responses in a two-input Wiener System based on the two second-order cross-correlation functions, measured between the inputs, x and y, and the output v. `h_len` is the length, in samples, of the impulse responses, `hxlin` and `hylin`. If iterate is `'[y]es'`, then an iterative technique is used to estimate and cancel interference in the correlation estimates that is caused by the presence of multiple input signals.

This is intended primarily as a service routine.
It is called by: `multi_wiener`

Local Functions Called: `phixxy, proj0, tchebfit2d, toep`

DTW Apr 1992

irf_est_odd

Syntax: `[hx, hy] = irf_est_odd (x,y,z, hlen,iterate);`

Inputs:

`x,y`	Inputs to a two-input nonlinear system
`v`	Output of the nonlinear system
`h_len`	(Upper bound on) the anticipated memory length of the system
`iterate`	Use iterative scheme to improve estimates (y/n)

Outputs:

`hxlin,hylin`	Impulse response estimates

Estimates the impulse responses in a two-input Wiener System based on the two first-order cross-correlation functions, measured between the inputs, `x` and `y`, and the output `v`. `h_len` is the length, in samples, of the impulse responses, `hxlin` and `hylin`. If iterate is `'[y]es'`, then an iterative technique is used to estimate and cancel interference in the correlation estimates that is caused by the presence of multiple input signals.

This is intended primarily as a service routine.
It is called by: `multi_wiener`

Local Functions Called: `phixy, tchebfit2d, toep`

DTW Apr 1992

iteration3

Syntax: `[k,alts] = iteration3 (phi,h, Threshold);`

Inputs:

`phi`	Third-order cross-correlation function
`h`	Initial estimate of Impulse response function
`Threshold`	Used to decide when to stop iterating (default 1×10^{-6})

Outputs:

`k`	Final impulse response estimate
`alts`	Alternate starting points for iterative searches

Performs several steps in a gradient iteration that attempts to find the first-order cross-correlation across the linear part of a Wiener system that has a third-order cross-correlation that best approximates the given function `phi`.

`alts` contain alternate starting points. They are valleys in the mean square error function that were not checked, because they were not considered 'optimal'.

This is intended primarily as a service routine.
It is called by: `wiener3`

Local Functions Called: `k3_gen, mult32`

DTW July 1992

iteration32

Syntax: `[newh, newg] = iteration32 (phi,h,g,Threshold);`

Inputs:

`phi`	Third-order cross-correlation function
`h`	Initial estimate of first impulse response function
`g`	Initial estimate of second impulse response function
`Threshold`	Used to decide when to stop iterating (default 1×10^{-6})

Outputs:

`newh`	Final impulse response estimate for first input
`newg`	Final impulse response estimate for second input

`phi` is assumed to hold the third-order cross-cross-correlation that is second-order in its first input, and first-order in its second input. `iteration32` returns `newh` and `newg`, estimates of the linear filters that are part of a two-input Wiener system, for which the third-order cross-cross-correlation best approximates that measured from the input/output data (phi). h and g should contain initial estimates of these IRFs.

Local Functions Called: `k32_gen`, `mult3_101`, `mult32`

DTW May 1993

k32_gen

Syntax: `K3 = k32_gen (h,g,gain);`

Inputs:

`h,g`	Impulse responses of linear dynamic elements in a two-input Wiener system.
`gain`	coefficient of the term in the static nonlinearity that is second-order in the first input and first order in the second input.

Outputs:

`K3`	Component of the third-order Volterra kernel resulting from the 2-1 order term in the two inputs.

Generates the third-order Volterra kernel of a two-input Wiener system, where the inputs are transformed by `h` and `g`, and the static nonlinearity is `gain` times the product of the output of the first linear system squared, and the output of the second linear system.

This is intended primarily as a service routine.
It is called by: `wiener3`

Implemented as a MEX file

DTW

k3_gen

Syntax: `K3 = k3_gen (h,gain);`

Inputs:

`h`	Impulse response of linear dynamic element of a single-input Wiener system.
`gain`	coefficient of the third-order term in the static nonlinearity

Outputs:

`K3`	Third-order Volterra kernel of the Wiener system

Generates the third-order Volterra kernel of a single-input Wiener system consisting of the linear element `h` followed by a cuber with a gain of `gain`.

This is intended primarily as a service routine.
It is called by: `wiener3`

Implemented as a MEX file

DTW

k3_sym

Syntax: `H3 = k3_sym (K3);`

Inputs:

`K3`	Third order kernel

Outputs:

`H3`	Symmetrized third-order kernel

Given a third-order kernel, stored as a long vector, this function symmetrizes it about every possible permutation of its indices.

This is intended primarily as a service routine.

It is called by: `wiener3`

Implemented as a MEX file

DTW

kernel3_slice

Syntax: `Z = kernel3_slice (k3,slice);`

Inputs:

`k3`	Third-order kernel or correlation function
`slice`	lag of slice to be displayed

Outputs:

`Z`	Single slice of `K3`

Displays a surface plot of single slice of the third-order kernel `K3`. The edges of a cube are drawn, and the surface plot is positioned within this frame to indicate the position of the slice within the original kernel. If an output is specified, the slice is returned in a matrix.

No Local Function Calls:

DTW

lnlk2

Syntax: `kernel = lnlk2 ( g,h,n);`

Inputs:

`g`	First linear element in LNL cascade
`h`	Second linear element in LNL cascade
`n`	Memory length of kernel (optional)

Outputs:

`kernel`	Second-order Volterra kernel of LNL cascade

Given the impulse responses of the two linear elements, `lnlk2` returns the second-order Volterra kernel of the cascade, assuming that the second-order coefficient in the polynomial expansion of the static-nonlinearity is 1. If `n` is specified, an `n` by `n` kernel is returned. This may either result in truncation or zero-padding of the actual kernel.

No Local Function Calls:

DTW Jan 1992

mult31

Syntax: `H2 = mult31(K3,h);`

Inputs:

`K3`	Third order kernel
`h`	vector, or first order kernel

Outputs:

`H2`	Second-order kernel, or matrix

Computes the product of a third-order kernel with a vector.

$$H_2(i,j) = \sum_{k=1}^{R} K_3(i,j,k)h(k)$$

If, as in the previous equation, the vector has R elements, then K_3 must have R^3 elements. H_2 will be an $R \times R$ matrix.

This is intended primarily as a service routine.

It is called by: `iteration32`

Implemented as a MEX file

DTW 1992

mult32

Syntax: `h = mult32(K3,H2);`

Inputs:

`K3`	Third order kernel
`H2`	matrix, or second-order kernel

Outputs:

`h`	First-order kernel, or vector

Computes the product of a third-order kernel with a matrix.

$$h(k) = \sum_{i,j=1}^{R} K_3(i,j,k)H_2(i,j)$$

If, as in the previous equation, the matrix has $R \times R$ elements, then K_3 must have R^3 elements. h will be an R element vector.

This is intended primarily as a service routine.

It is called by: `iteration3, iteration32, wiener3`

Implemented as a MEX file

DTW 1992

mult3_101

Syntax: `v = mult3_101(K3,h,g);`

Inputs:

`K3`	Third order kernel
`h,g`	vectors, or first-order kernels

Outputs:

`v`	First-order kernel, or vector

Computes the product of a third-order kernel with a two vectors.

$$v(j) = \sum_{i,k=1}^{R} K_3(i,j,k)h(i)g(k)$$

If, as in the previous equation, the vectors have length R, then K_3 must have R^3 elements. v will be an R element vector.

This is intended primarily as a service routine.
It is called by: `iteration32`

Implemented as a MEX file

DTW 1992

multi_wiener

Syntax: `[hx,hy,nl,out] = multi_wiener (x,y,z,hlen,order,control);`

Inputs:

`x,y`	System inputs
`z`	System output
`hlen`	Upper bound on the system's memory length
`order`	Degree of the highest order kernel expected in the system
`control`	Control string = `[type mode smooth iterate]` `type`: type of functions used to determine impulse responses can be either 'first', 'self' or 'cross', which correspond to the first order, second order self and second order cross input/output cross-correlations, respectively `mode`: mode used in the polynomial fitting routine. Can be either 'fixed', 'auto' or 'manual'. If fixed is chosen, a polynomial of order order is returned. If auto or manual is chosen, the best polynomial of order less or equal to order is returned. `smooth`: (yes/no)Turns smoothing on and off. `iterate`: (y/n) Turns iterative IRF estimate improvement on and off In all cases, only first letters are used. For example the string '`fmny`' corresponds to using first-order correlation functions, with manual order selection, no smoothing, and iterative estimation correction.

Outputs:

`hx hy`	Impulse responses of linear subsystems associated with inputs x and y
`nl`	Coefficients of static nonlinearity. See documentation for `tchebfit2d` and Section A.1.3 for details
`out`	Output of the the multiple input Wiener system estimated by `multi_wiener`

Estimates a multiple input Wiener system between two inputs and a single output.

The linear subsystems can be based on either the first. or second-order single-input cross-correlation functions between inputs x and y and output z. or on the second-order cross-cross-correlation between both inputs and the output.

Local Functions Called: `irf_est_cross, irf_est_even, irf_est_odd, tchebfit2d, tchebval2d`

DTW March 1992

phix2yz

Syntax: `phi = phix2yz (x,y,z,hlen);`

Inputs:

`x,y`	Input signals
`z`	Output signal
`hlen`	Maximum number of lags to be calculated +1

Outputs:

`phi` Third-order cross-cross-correlation between `x,y` and `z`. It is second-order is `x` and first-order in `y`

The means of `x,y` and `z` are subtracted prior to estimation. A biased estimate of the cross-correlation function is returned:

$$\hat{\phi}_{x^2yz}(i,j,k) = \frac{1}{N}\sum_{n=1}^{N} x_0(n-i)x_0(n-j)y_0(n-k)z_0(k)$$

where N is the length of the signals `x,y` and `z`, and the subscript 0 refers to signals that have had their average values removed. i.e.:

$$x_0(n) = x(n) - \frac{1}{N}\sum_{n=1}^{N} x(n)$$

Implemented as a MEX file

DTW June 1991 .

phix3y

Syntax: `phi = phix3y (x,y,hlen);`

Inputs:

x	Input signal
y	Output signal
hlen	Maximum number of lags to be calculated +1

Outputs:

phi	Third-order cross-correlation between x and y, stored as a vector. See Section A.1.2 for details

The means of both x and y are subtracted prior to estimation. A biased estimate of the cross-correlation function is returned:

$$\hat{\phi}_{x^3y}(i,j,k) = \frac{1}{N}\sum_{n=1}^{N} x_0(n-i)x_0(n-j)x_0(n-k)y_0(n)$$

where N is the length of the signals x and y, and the subscript 0 refers to signals that have had their average values removed. i.e.:

$$x_0(n) = x(n) - \frac{1}{N}\sum_{n=1}^{N} x(n)$$

Implemented as a MEX file

DTW June 1991 .

phi_adj_phi

Syntax: `A = phi_adj_phi (phi);`

Inputs:

`phi`	Third-order kernel or correlation function, stored as a vector

Outputs:

`A`	The adjoint operator of `phi` applied to `phi`.The eigenvectors of `A` are the singular vectors of `phi`. The eigenvalues of `A` are the squares of the singular values of `phi`.

This is intended primarily as a service routine.
It is called by: `wiener3`

No Local Function Calls:

DTW June 1992

phixxy

Syntax: `phi = phixxy (x,y,hlen);`

Inputs:

`x`	Input signal
`y`	Output signal
`hlen`	Maximum number of lags to be calculated +1

Outputs:

`phi`	Second order cross-correlation between `x` and `y`

The means of both `x` and `y` are subtracted prior to estimation. A biased estimate of the cross-correlation function is returned:

$$\hat{\phi}_{xxy}(i,j) = \frac{1}{N}\sum_{n=1}^{N} x_0(n-i)x_0(n-j)y_0(n)$$

where N is the length of the signals `x` and `y`, and the subscript 0 refers to signals that have had their average values removed. i.e.:

$$x_0(n) = x(n) - \frac{1}{N}\sum_{n=1}^{N} x(n)$$

Implemented as a MEX file

DTW June 1991 .

phixy

Syntax: `phi = phixy (x,y,hlen);`

Inputs:

`x`	Input signal
`y`	Output signal
`hlen`	Maximum number of lags to be calculated +1

Outputs:

`phi`	First order cross-correlation between `x` and `y`

Computations are performed in the frequency domain, as in the *System Identification Toolbox* routine `covf` However unlike that function, only the input-output cross-correlation is computed.

No Local Function Calls:

DTW Aug 1993

phixyz

Syntax: `phi = phixyz (x,y,z,hlen);`

Inputs:

`x,y`	Input signals
`z`	Output signal
`hlen`	Maximum number of lags to be calculated +1

Outputs:

`phi`	Second-order cross-cross-correlation between inputs x and y, and output z

The means of `x,y` and `z` are subtracted prior to estimation. A biased estimate of the cross-correlation function is returned:

$$\hat{\phi}_{xyz}(i,j) = \frac{1}{N}\sum_{n=1}^{N} x_0(n-i)y_0(n-j)z_0(n)$$

where N is the length of the signals `x,y` and `z`, and the subscript 0 refers to signals that have had their average values removed. i.e.:

$$x_0(n) = x(n) - \frac{1}{N}\sum_{n=1}^{N} x(n)$$

Implemented as a MEX file

DTW June 1991 .

proj0

Syntax: `out = proj0 (x,y)`

Inputs:

x	Vector to be Projected
y	Basis for Projection

Outputs:

out	projection of `(x - mean(x))` onto `(y - mean(y))`

This is intended primarily as a service routine.

It is called by: `irf_est_cross ,irf_est_even`

No Local Function Calls:

DTW August 1993

tchebconvert

Syntax: `pars = tchebconvert ( coeff )`

Inputs:

`coeff` vector of coefficients describing a tchebyshev polynomial. See section A.1.3 and the routine `tchebfit` for more details

Outputs:

pars The polynomial is returned with coefficients describing it in terms of increasing powers of x. i.e. the i'th element of pars contains the coefficient of x to the i-1.

No Local Function Calls:

DTW

tchebconvert2d

Syntax: `pars = tchebconvert 2d( coeff );`

Inputs:

`coeff` vector of coefficients describing a two-variable tchebyshev polynomial. See section A.1.3 and the routine `tchebfit2d` for more details

Outputs:

`pars` The polynomial is returned with coefficients describing it in terms of the functions $x^n y^m$.

The vector pars contains terms in the following order:

$[1\, x\, y\, x^2\, xy\, y^2\, x^3\, x^2y\, xy^2\, y^3 \ldots]$

No Local Function Calls:

DTW

tchebfit

Syntax: `[coeff,vaf,out]=tchebfit(x,y,order, mode);`

Inputs:

`x`	Input to static Nonlinearity
`y`	Output from static Nonlinearity
`order`	Maximum order of the polynomial to be estimated
`mode`	Method used to choose optimal polynomial order:
	'`fixed`': a polynomial of order `order` is returned
	'`manual`': polynomials of orders 0 through `order` are calculated. The variance accounted for by estimated polynomials is displayed, and the used is asked to select the optimal order
	'`automatic`': polynomials of orders 0 through `order` are calculated. The order is selected automatically
	in all cases only the first letter is significant. The default mode is `fixed`.

Outputs:

`coeff`	A vector consisting of the minimum and maximum values of `x`, followed by the coefficients of the tchebyshev polynomial functions. For further information, see Section A.1.3
`vaf`	The percentage of the variance of `y` accounted for by the polynomial
`out`	The polynomial applied to the input signal `x`

No Local Function Calls:

DTW Sep 1991

tchebfit2d

Syntax: `[coeff,vaf,out]= tchebfit2d (x,y,z,order, mode);`

Inputs:

`x,y`	Inputs to static nonlinearity
`z`	Output from static nonlinearity
`order`	Maximum order of the polynomial to be estimated
`mode`	Method used to choose optimal polynomial order:
	'`fixed`': a polynomial of order order is returned
	'`manual`': polynomials of orders 0 through `order` are calculated. The variance accounted for by estimated polynomials is displayed, and the used is asked to select the optimal order
	'`automatic`': polynomials of orders 0 through `order` are calculated. The order is selected automatically
	in all cases only the first letter is significant. The default mode is `fixed`.

Outputs:

`coeff`	A vector describing the two-input tchebyshev polynomial. For a description of its format, see Section A.1.3
`vaf`	The percentage of the variance of y accounted for by the polynomial
`out`	The polynomial applied to the input signals `x` and `y`

No Local Function Calls:

DTW Oct 1991

tchebval

Syntax: `y=tchebval(coeff,x,mode);`

Inputs:

`coeff`	A vector describing a single-input tchebyshev polynomial, such as those returned by `tchebfit`. For details, see Section A.1.3
`x`	Input signal to be transformed.
`mode`	Method used to treat points outside of the domain of definition of the polynomial '`clip`':points are hard-limited to the minimum and maximum of the input domain '`extend`': points are not clipped. This can be lead to unpredictable results, particularly with high order polynomials in all cases only the first letter is significant. The default mode is `clip`.

Outputs:

`y`	The input `x` transformed by the tchebyshev polynomial

Local Functions Called: `hard_limit`

DTW Sep 1991

tchebval2d

Syntax: `y=tchebval2d(coeffs,x,y,mode,bound_id);`

Inputs:

`coeff`	A vector describing a two-input tchebyshev polynomial, such as those returned by `tchebfit2d`. For details, see Section A.1.3
`x,y`	Input signals to be transformed.
`mode`	Method used to treat points outside of the domain of definition of the polynomial '`clip`': points are hard-limited to the minimum and maximum of the input domain '`extend`': points are not clipped. This can be lead to unpredictable results, particularly with high order polynomials in all cases only the first letter is significant. The default mode is `clip`.
`bound_id`	identity of the bounding polygon (0 through the number of bounds included in `coeffs`. 1 is the default value. If `bound_id` = 0, a rectangular domain defined by the first four elements of `coeff` is used.

Outputs:

y	Inputs x and y transformed by the tchebyshev polynomial

Local Functions Called: `hard_limit, hard_limit_2d`

DTW Oct 1991

tchebplot

Syntax: `[dom,ran]=tchebplot(poly,color);`

Inputs:

`poly`	coefficients of a tchebyshev polynomial, as returned by `tchebfit`
`color`	colour of the plotted line

Outputs:

`dom`	domain values of the points used to generate the plot
`ran`	range values of the points used to generate the plot

Local Functions Called: `tchebval`

DTW Sep 1991

tchebplot2d

Syntax: `[X,Y,Z] = tchebplot2d ( poly, control, bound);`

Inputs:

`poly`	coefficients of a two-dimensional tchebyshev polynomial, as returned by `tchebfit2d`
`control`	a string that controls the appearance of the plot (see below)
`bound`	`[bound_ids NaN x_size y_size NaN domain_bounds]`

Outputs:

`X,Y,Z` matrices of the `X,Y` and `Z` values of the points used to generate the polynomial surface. If only one output is specified, `Z` is returned.

`control:` `[plt_type dom_mode bound_mode bound_color]`

`plt_type:`	can be '`mesh`', '`surf`', '`lsurf`' or '`contour`', corresponding to the MATLAB surface plotting functions. Only the first character is significant in each case.
`dom_mode:`	can either be '`clip`' or '`extend`'. It controls how the polynomial is applied to points outside of its domain of definition. They are either clipped back to the domain, or the polynomial domain is extended, so that the points are simply evaluated. See `tchebval2d` for more details.
`bound_mode`	can either be '`contour`', '`rectangle`', or '`full`'. `contour` limits the domain to a specified contour, which is passed in `bounds`. `rectangle` limits the domain to a rectangle defined by the first four elements of `coeff`. `full` sets the domain to that specified in the last four elements passed in the vector `bounds`
`bound_color`	specifies which colour is used to plot the clipping boundary

Local Functions Called: `hard_limit, hard_limit_2d, tchebval2d`

DTW Oct 1991

toep

Syntax: `x = toep (r,b);`

Inputs:

- `r` vector used to describe a Toeplitz matrix. Usually the autocorrelation function of an input signal
- `b` right hand side of Toeplitz matrix equation. Usually the input-output cross-correlation measured across a dynamic linear system.

Outputs:

- `x` solution of Toeplitz equation. If `r` and `b` are the input autocorrelation and input-output cross-correlation functions, `x` will be the impulse response.

This function solves the matrix equation $Tx = b$, where T is a Toeplitz matrix, described by the vector r. i.e. $T(i,j) = r(|i-j|+1)$. Levinson's [20] algorithm is used.

No Local Function Calls:

EJP Jan 1991

toep2

Syntax: `x = toep (r,b,resid);`

Inputs:

`r`	vector used to describe a Toeplitz matrix. Usually the autocorrelation function of an input signal
`b`	right hand side of Toeplitz matrix equation. Usually the input-output cross-correlation measured across a dynamic linear system.
`resid`	(Optional) channel of residuals. Used to estimate noise statistics.

Outputs:

`x`	solution of Toeplitz equation. If `r` and `b` are the input autocorrelation and input-output cross-correlation functions, `x` will be the impulse response.

This function solves the matrix equation $Tx = b$, where T is a Toeplitz matrix, described by the vector r. i.e. $T(i,j) = r(|i-j|+1)$. Instead of using the exact inverse of T, a pseudo-inverse may be employed. The use of a full-inverse as opposed to a pseudo-inverse, as well as the method used to choose the order of the pseudo-inverse, is controlled by a series of global variables. For details, see `toep_man.m`

Local Functions Called: `toep`

DTW Sept 1994

toep_man

Syntax: `toep_man(option,parameter)`

Inputs:

`option`	control string
`parameter`	optional parameter, dependent on which option is selected.

Valid options are:

- `auto` Enables automatic order selection.
- `fixed` Use a fixed pseudoinverse order
- `fixed_order` specify which order to use
- `full` Uses exact inverse in all deconvolutions
- `gui` Initializes all global variables, if necessary, and creates a graphical user interface
- `help` Access to help systems
- `init` Initializes all global variables
- `manual` manual order selection, based on displayed thresholds
- `status` Displays the value of all global variables, as well as a brief description of the current order selection mode.
- `verbose` Toggles verbose mode (starts on)
- `visual_mode` User can manipulate pseudoinverse order using sliders. The current IRF is plotted together with the full inverse solution.

For detailed information, type `toep_man('help')`, or `toep_man('gui')` and then examine the help menu on the resulting figure window.

Local Functions Called: `input_d,input_d1`

DTW Sept 1994

wiener1

Syntax: `[irf, poly, vaf, out] = wiener1 (x,v,irflen,order, mode);`

Inputs:

`x`	system input
`v`	system output
`irflen`	memory length of the linear dynamic element
`order`	maximum order for the polynomial nonlinearity
`mode`	method used to select polynomial order. See `tchebfit`

Outputs:

`irf`	impulse response of the dynamic linear subsystem of the Wiener cascade
`poly`	static nonlinear subsystem, represented as a tchebyshev polynomial. For details regarding the format of this vector, see Section A.1.3
`vaf`	percentage of the variance of `v` accounted for by the Wiener cascade output
`out`	output of the Wiener cascade applied to `x`

Fits a Wiener cascade between input `x` and output `v`, using the first-order cross-correlation between `x` and `v`, as well as the input-autocorrelation, to estimate the dynamic linear part of the system.

Local Functions Called: `phixy, tchebfit, tchebval, toep`

DTW Jan. 1992

wiener2

Syntax: `[irf, poly, vaf,out] = wiener2 (x,v,irflen,order,mode);`

Inputs:

`x`	system input
`v`	system output
`irflen`	memory length of the linear dynamic element
`order`	maximum order for the polynomial nonlinearity
`mode`	method used to select polynomial order. See `tchebfit`

Outputs:

`irf`	impulse response of the dynamic linear subsystem of the Wiener cascade
`poly`	static nonlinear subsystem, represented as a tchebyshev polynomial. For details regarding the format of this vector, see Section A.1.3
`vaf`	percentage of the variance of `v` accounted for by the Wiener cascade output
`out`	output of the Wiener cascade applied to `x`

Fits a Wiener cascade between input `x` and output `v`, using the second-order cross-correlation between `x` and `v`, as well as the input-autocorrelation, to estimate the dynamic linear part of the system.

Local Functions Called: `phixy, phixxy, tchebfit, tchebval, toep`

DTW Jan. 1992

wiener3

Syntax: `[irf,poly,vf,out] = wiener3 (x,v,numlags,order,mode,tol);`

Inputs:

`x`	system input
`v`	system output
`irflen`	memory length of the linear dynamic element
`order`	maximum order for the polynomial nonlinearity
`mode`	method used to select polynomial order. See `tchebfit`
`tol`	tolerance used to halt gradient search algorithm that is used to find the optimal IRF. Default value is (1×10^{-6})

Outputs:

irf	impulse response of the dynamic linear subsystem of the Wiener cascade
poly	static nonlinear subsystem, represented as a tchebyshev polynomial. For details regarding the format of this vector, see Section A.1.3
vaf	percentage of the variance of **v** accounted for by the Wiener cascade output
out	output of the Wiener cascade applied to **x**

Fits a Wiener cascade between input **x** and output **v**, using the third-order cross-correlation between **x** and **v**, as well as the input-autocorrelation, to estimate the dynamic linear part of the system.

Local Functions Called: **`iteration3, phixy, phix3y, phi_adj_phi, tchebfit, tchebval, toep`**

DTW June 1992

Appendix B

Details of the Analogue Nonlinear System

Figure B.1, a repeat of Figure 6.1, shows a block diagram of the analogue system used in the experiments described in Section 6.1. Details regarding the linear filters are presented in Table B.1.

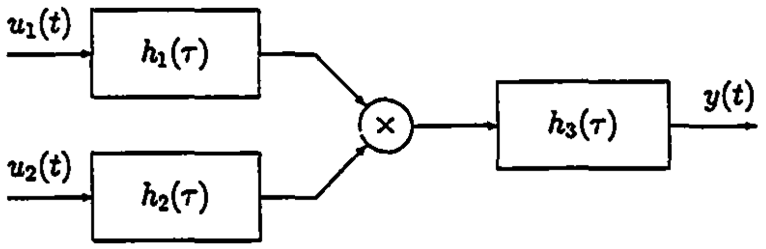


Figure B.1: Block diagram of the electronic system used in the experimental verification of the eigenvector method (repeat of Figure 6.1)

All of the filters, except the eighth order Bessel filter, were realized using a second-order Sallen-Key structure [35]. A circuit diagram for a low-pass Sallen-Key filter is shown in Figure B.2. To obtain a high-pass filter, the resistors marked R and capacitors marked C are interchanged. The remaining filter was a commercially produced anti-aliasing filter [1].

[1]Frequency Devices 902LPF

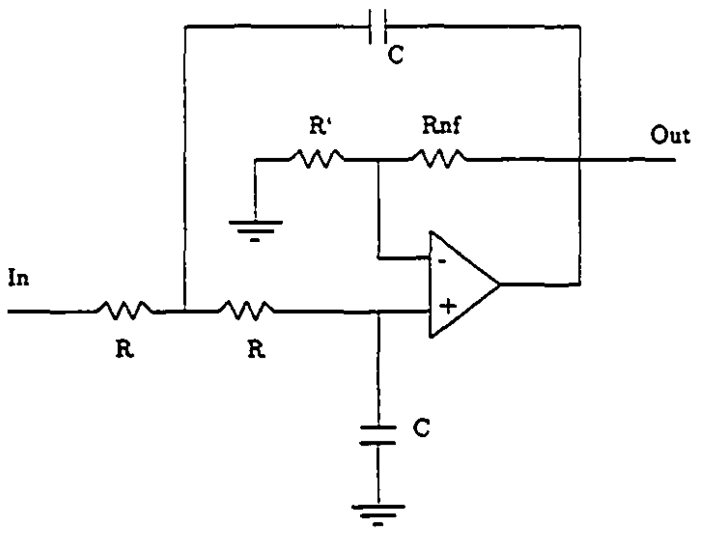


Figure B.2: Circuit diagram of the second-order Sallen-Key filters used to construct the analogue nonlinear system

We estimated impulse responses for the linear filters, using the techniques described in Chapter 3. These are presented in Figure 6.3. A nonlinear optimization method was then used to estimate filter parameters for the identified IRFs. These are presented, along with the design values, in Table B.1.

			Cut-Off (Hz)		Ripple (Db)	
Filter	Type	Order	Design	Est.	Design	Est.
LPF-1	Low-Pass Butterworth	2	21.16	19.31	N/A	N/A
LPF-2	Low-Pass Butterworth	2	10.26	10.58	N/A	N/A
LPF-3	Low-Pass Bessel	8	30.00	30.00	N/A	N/A
HPF-1	High-Pass Chebyshev Type I	2	4.82	4.58	2	1.65
HPF-2	High-Pass Chebyshev Type I	2	7.23	7.27	2	1.35

Table B.1: Filters used as linear elements in the analog simulation experiment

Bibliography

[1] G.C. Agarwal and G.L. Gottlieb. Mathematical modeling and simulation of the postural control loop: part I. *CRC Critical Reviews in Biomedical Engineering*, 8(1):93–134, 1982.

[2] G.C. Agarwal and G.L. Gottlieb. Mathematical modeling and simulation of the postural control loop: part II. *CRC Critical Reviews in Biomedical Engineering*, 11(2):113–154, 1984.

[3] G.C. Agarwal and G.L. Gottlieb. Mathematical modeling and simulation of the postural control loop: part III. *CRC Critical Reviews in Biomedical Engineering*, 12(1):49–93, 1985.

[4] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19:716–723, 1974.

[5] J.S. Bendat and A.G. Piersol. *Random Data, analysis, and measurement proceedures*. John Wiley and Sons, New York, 2'nd edition, 1986.

[6] S.A. Billings and S.Y. Fakhouri. Identification of systems containing linear dynamic and static nonlinear elements. *Automatica*, 18:15–26, 1982.

[7] S.A. Billings and G.N. Jones. Orthogonal least-squares parameter estimation algorithms for non-linear stochastic systems. *International Journal of Systems Science*, 23(7):1019–1032, 1992.

[8] S.A. Billings and W.S.F. Voon. A prediction-error and stepwise-regression estimation algorithm for non-linear systems. *International Journal of Control*, 44:803–822, 1986.

[9] S. Boyd and L.O. Chua. Fading memory and the problem of approximating nonlinear operators with Volterra series. *IEEE transactions on Circuits and Systems*, CAS-32(11):1150–1161, 1985.

[10] J.J. Bussgang. Crosscorrelation functions of amplitude-distorted Gaussian signals. Technical Report 216, MIT Electrical Research Lab, 1952.

[11] P.E. Caines. *Linear Stochastic Systems*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, New York, 1988.

[12] H.W. Chen. Modeling and identification of parallel nonlinear systems: Structural classification and parameter estimation methods. *Proceedings of the IEEE*, 83(1):39–66, 1995.

[13] H.W. Chen, L.D. Jacobsen, and J.P. Gaska. Structural classification of multi-input nonlinear systems. *Biological Cybernetics*, 63:341–357, 1990.

[14] S. Chen, S.A. Billings, and P.M. Grant. Recursive hybrid algorithm for non-linear system identification using radial basis function networks. *International Journal of Control*, 55(5):1051–1070, 1992.

[15] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. The MIT Electrical Engineering and Computer Science Series. McGraw Hill, New York, 1'st edition, 1990.

[16] B.L.R. de Moor and G.H. Golub. The restricted singular value decomposition: Properties and applications. *SIAM Journal of Matrix Analysis and Applications*, 12(3):401–425, 1991.

[17] T.D. Doukoglou and I.W. Hunter. Estimation of Volterra kernels using matrix equation approach. Biorobotics Lboratory, McGill University, Internal Report, Mar. 1990.

[18] *Proceedings of the IEEE EMBS*, volume 13. October 1991.

[19] A.S. French. Practical nonlinear system analysis by Wiener kernel estimation in the frequency domain. *Biological Cybernetics*, 24:111–119, 1976.

[20] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins Series in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, 2'nd edition, 1989.

[21] Y. Goussard, W.C. Krenz, L. Stark, and G. Demoment. Practical identification of functional expansions of nonlinear systems submitted to non-Gaussian inputs. *Annals of Biomedical Engineering*, 19:401–427, 1991.

[22] W. Greblicki. Non-parametric orthogonal series identification of Hammerstein systems. *International Journal of Systems Science*, 20:2355–2367, 1989.

[23] W. Greblicki. Nonparametric identification of Wiener systems. *IEEE Transactions on Information Theory*, 38(5):1487–1493, 1992.

[24] W. Greblicki and M. Pawlak. Hammerstein system identification by nonparametric regression estimation. *International Journal of Control*, 45(1):343–354, 1987.

[25] W. Greblicki and M. Pawlak. Nonparametric identification of Hammerstein systems. *IEEE Transactions on Information Theory*, 35(2):409–418, 1989.

[26] W. Greblicki and M. Pawlak. Recursive nonparametric identification of Hammerstein systems. *Journal of the Franklin Institute*, 326(4):461–481, 1989.

[27] W. Greblicki and M. Pawlak. Nonparametric identification of a cascade nonlinear time series system. *Signal Processing*, 22:61–75, 1991.

[28] B.L. Ho and R.E. Kalman. Effective construction of linear, state-variable models from input/output functions. *Regelungstechnik*, 14:545–548, 1966.

[29] I.W. Hunter and R.E. Kearney. Invariance of ankle dynamic stiffness during fatiguing muscle contractions. *Journal of Biomechanics*, 16:985–991, 1983.

[30] I.W. Hunter and R.E. Kearney. Two-sided linear filter identification. *Medical and Biological Engineering and Computing*, 21:203–209, 1983.

[31] I.W. Hunter and M.J. Korenberg. The identification of nonlinear biological systems: Wiener and Hammerstein cascade models. *Biological Cybernetics*, 55:135–144, 1986.

[32] I.W. Hunter, S. Lafontaine, T.D. Doukoglou, L.A. Jones, M.J. Korneberg, P.M.F. Nielsen, and R.F. Kirsch. Micro-robotics and the study of muscle: Special problems in control, system identification, and modelling. In Cs. Bánysász and L. Keviczky, editors, *Identification and System Parameter Estimation*, volume 9, pages 197–202. International Federation of Automatic Control, 1991.

[33] E. Jankowska. Interneuronal relay in spinal pathways from proprioceptors. *Progress in Neurobiology*, 38:335–378, 1992.

[34] R.A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2:18–21, 1973.

[35] D.E. Johnson and J.L. Hilburn. *Rapid Practical Designs of Active Filters*. John Wiley and Sons, New York, 1975.

[36] R.E. Kearney and I.W. Hunter. System identification of human stretch reflex dynamics: tibialis anterior. *Experimental Brain Research*, 56:40–49, 1984.

[37] R.E. Kearney and I.W. Hunter. Nonlinear identification of stretch reflex dynamics. *Annals of Biomedical Engineering*, 16:79–94, 1988.

[38] R.E. Kearney and I.W. Hunter. System identification of human joint dynamics. *CRC Critical Reviews in Biomedical Engineering*, 18:55–87, 1990.

[39] R.E. Kearney, I.W. Hunter, P.L. Weiss, and K. Spring. Tilt-table/ankle-actuator system for the study of vestibulospinal reflexes. *Medical and Biological Engineering and Computing*, 21:301–305, 1983.

[40] R.E. Kearney and R.B. Stein. Influence of perturbation properties on the identification of stretch reflexes at the human ankle joint. In *Proceedings of the IEEE EMBS*, volume 15, pages 1169–1170, 1993.

[41] R.E. Kearney and R.B. Stein. Modulation of human stretch reflex by random perturbations. In *Society for Neuroscience Abstracts*, volume 19, page 141, 1993.

[42] R.E. Kearney, R.B. Stein, and L. Parmeswaran. Differential identification of passive and reflex mechanisms in human ankle stiffness dynamics. In *Proceedings of the IEEE EMBS*, volume 16, pages 430–431, 1994.

[43] R.E. Kearney, R.B. Stein, and L. Parmeswaran. Identification of passive and reflex components of human dynamic ankle stiffness. In *Society for Neuroscience Abstracts*, volume 20, page 1752, 1994.

[44] R.F. Kirsch and R.E. Kearney. Identification of time-varying dynamics of the human triceps surae stretch reflex II. rapid imposed movement. *Experimental Brain Research*, 97:128–138, 1993.

[45] R.F. Kirsch, R.E. Kearney, and J.B. MacNeil. Identification of time-varying dynamics of the human triceps surae stretch reflex I. rapid isometric contraction. *Experimental Brain Research*, 97:115–127, 1993.

[46] R.F. Kirsch and W.Z. Rymer. Neural compensation for muscular fatigue: Evidence for significant force regulation in man. *Journal of Neurophysiology*, 57(6):1893–1910, 1987.

[47] K. Konstantinides and K. Yao. Statistical analysis of effective singular values in matrix rank determination. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(5):757–763, 1988.

[48] M.J. Korenberg. Cross-correlated analysis of neural cascades. *Proceedings of the Annual Rocky Mountain Bioengineering Symposium*, 1:47–52, 1973.

[49] M.J. Korenberg. Identification of biological cascades of linear and static nonlinear elements. *Proceedings of the Midwest Symposium on Circuit Theory*, 18.2:1–9, 1973.

[50] M.J. Korenberg. Obtaining differential equation, functional expansion or cascade representations for nonlinear biological systems. *Proceedings of the New England Bioengineering Conference*, 1:237–245, 1973.

[51] M.J. Korenberg. Statistical identification of parallel cascades of linear and nonlinear systems. In *Identification and System Parameter Estimation*, volume 1, pages 669–674. International Federation of Automatic Control, 1982.

[52] M.J. Korenberg. Statistical identification of Volterra kernels of high order systems. *ICAS'84*, pages 570–575, 1984.

[53] M.J. Korenberg. Identifying noisy cascades of linear and static nonlinear systems. In *Identification and System Parameter Estimation*, volume 4, pages 421–426. International Federation of Automatic Control, 1985.

[54] M.J. Korenberg. Identifying nonlinear difference equation and functional expansion representations: The fast orthogonal algorithm. *Annals of Biomedical Engineering*, 16:123–142, 1988.

[55] M.J. Korenberg. Fast orthogonal algorithms for nonlinear system identification and time-series analysis. In Marmarelis [70], pages 165–178.

[56] M.J. Korenberg. Parallel cascade identification and kernel estimation for nonlinear systems. *Annals of Biomedical Engineering*, 19:429–455, 1991.

[57] M.J. Korenberg. Recent advances in the identification of nonlinear sytems: minimum-variance approximation by Hammerstein models. In EMBS13 [18], pages 2258–2259.

[58] M.J. Korenberg. Identification of nonlinear systems. *Proceedings of the IFAC Conference on Modeling and Control in Biomedical Systems*, 1:541–543, 1994.

[59] M.J. Korenberg, S.B. Bruder, and P.J. McIlroy. Exact orthogonal kernel estimation from finite data records: Extending Wiener's identification of nonlinear systems. *Annals of Biomedical Engineering*, 16:201–214, 1988.

[60] M.J. Korenberg and I.W. Hunter. The identification of nonlinear biological systems: LNL cascade models. *Biological Cybernetics*, 55:125–134, 1986.

[61] M.J. Korenberg and I.W. Hunter. The identification of nonlinear biological systems: Wiener kernel approaches. *Annals of Biomedical Engineering*, 18:629–654, 1990.

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

[63] I.J. Leontaritis and S.A. Billings. Input-output parametric models for non-linear systems part I: deterministic non-linear systems. *International Journal of Control*, 41(2):303–328, 1985.

[64] I.J. Leontaritis and S.A. Billings. Input-output parametric models for non-linear systems part II: stochasitc non-linear systems. *International Journal of Control*, 41(2):329–344, 1985.

[65] L. Ljung. *System Identification: Theory for the user*. Prentice Hall Information and System Science Series. Prentice Hall Inc., Englewood Cliffs, New Jersey, 1987.

[66] J.B. MacNeil. Identification of time-varying human joint dynamics. Master's thesis, McGill University, Departments of Mechanical Engineering and Biomedical Engineering, 1990.

[67] J.B. MacNeil, R.E. Kearney, and I.W. Hunter. Identification of time-varying biological systems from ensemble data. *IEEE Transactions on Biomedical Engineering*, 39(12):1213–1225, 1992.

[68] P.Z. Marmarelis and V.Z. Marmarelis. *Analysis of Physiological Systems.* Computers in Biology and Medecine. Plenum Press, New York, 1978.

[69] P.Z. Marmarelis and K.I. Naka. Identification of multi-input biological sytems. *IEEE Transactions of Biomedical Engineering*, BME-21:88–101, 1974.

[70] V.Z. Marmarelis, editor. *Advanced Methods of Physiological System Modeling*, volume 2. Plenum Press, New York, 1989.

[71] V.Z. Marmarelis. Identification of nonlinear biological systems using Laguerre expansions of kernels. *Annals of Biomedical Engineering*, 21(6):573–589, 1993.

[72] V.Z. Marmarelis. Nonlinear modeling of physiological systems using principal dynamic modes. In V.Z. Marmarelis, editor, *Advanced Methods of Physiological System Modeling*, volume 3, pages 1–27. Plenum Press, New York, 1994.

[73] V.Z. Marmarelis and M.E. Orme. Modeling of neuronal systems by use of neuronal modes. *IEEE Transactions on Biomedical Engineering*, 40(11):1149–1158, 1993.

[74] M. Moonen and J. Ramos. A subspace algorithm for balanced state space system identification. *IEEE Transactions on Automatic Control*, 38(11):1727–1729, 1993.

[75] R.L. Morier, P.L. Weiss, and R.E. Kearney. Low inertia, rigid limb fixation using glass fibre casting bandage. *Medical and Biological Engineering and Computing*, 28:96–99, 1990.

[76] B. Ottersten and M. Viberg. A subspace based instrumental variable method for state-space system identification. *Proceedings of SYSID'94, 10th IFAC Symposium on System Identification*, 2:139–144, 1994.

[77] G. Palm. On representation and approximation of nonlinear sytems. *Biological Cybernetics*, 34:49–52, 1979.

[78] G. Palm and T. Poggio. The Volterra representation and the Wiener expansion: validity and pitfalls. *SIAM Journal of Applied Mathematics*. 33:195–216. 1977.

[79] L Parameswaran and R.E. Kearney. The effects of passive joint movement on stretch reflex gain at the human ankle. Submitted to IEEE EMBS 21. 1995.

[80] M. Pawlak. On the series expansion approach to the identification of Hammerstein systems. *IEEE Transactions on Automatic Control*. AC-36(6):763–767, 1991.

[81] M. Pawlak and W. Greblicki. Nonparametric estimation of a class of nonlinear time series models. In G. Roussas, editor, *Nonparametric Functional Estimation and Related Topics*, pages 541–552. Kluwer Academic Publishers, Amsterdam, 1991.

[82] E.J. Perreault, I.W. Hunter, and R.E. Kearney. Quantitative analysis of four EMG amplifiers. *Journal of Biomedical Engineering*, 15:413–419, 1993.

[83] D.R. Powell and J.R. Macdonald. A rapidly convergent iterative method of the solution of the generalized nonlinear least squares problem. *Comp J*, 15:148–155, 1972.

[84] W.J. Rugh. *Nonlinear system theory, the Volterra/Wiener approach*. Johns Hopkins Series in Information Sciences and Systems. The Johns Hopkins University Press, Baltimore, 1981.

[85] J. Shi and H.H. Sun. Nonlinear system identification for cascaded block model: An application to electrode polarization impedance. *IEEE Transactions on Biomedical Engineering*, 37(6):574–587, 1990.

[86] R.B. Stein, S.J. Deserres, and R.E. Kearney. Modulation of stretch reflexes during behaviour. In W.R. Ferrell and U. Proske, editors, *Neural Control of Movement*, pages 151–158. Plenum Press, New York, 1995.

[87] R.B. Stein and R.E. Kearney. Nonlinear behavior of muscle reflexes at the human ankle. *Journal of Neurophysiology*, 73(1):65–72, 1995.

[88] G.W. Stewart. Stochastic perturbation theory. *SIAM Review*, 32(4):579–610, 1990.

[89] G.W. Stewart and J.G. Sun. *Matrix Perturbation Theory*. Computer Science and Scientific Computing. Academic Press, San Diego, 1990.

[90] H.H. Sun and J.H. Shi. New algorithm for Korenberg-Billings model of nonlinear system identification. In Marmarelis [70], pages 179–200.

[91] A.L. Swindlehurst, B. Ottersten, R. Roy, and T. Kailath. Multiple invariance ESPRIT. *IEEE Transactions on Signal Processing*, 40(4):867–881, 1992.

[92] J.P. Trainor. Identification of time-varying human joint dynamics during an electrically stimulated twitch. Master's thesis, McGill University, Departments of Mechanical Engineering and Biomedical Engineering, 1994.

[93] A.J. van der Veen, F. Deprettere, and A.L. Swindlehurst. Subspace-based signal analysis using singular value decomposition. *Proceedings of the IEEE*, 81(9):1277–1308, 1993.

[94] P. van Overschee and B. de Moor. Subspace algorithms for the stochastic identification problem. *Automatica*, 29(3):649–660, 1993.

[95] P. van Overschee and B. de Moor. N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75–93, 1994.

[96] M. Verhaegen. Subspace model identification part 3. analysis of the ordinary output-error state-space model identification algorithm. *International Journal of Control*, 58(3):555–586, 1993.

[97] M. Verhaegen. Identification of the deterministic part of MIMO state space models given in innovations form from input-output data. *Automatica*, 30(1):61–74, 1994.

[98] M. Verhaegen and P. DeWilde. Subspace identification part 2. analysis of the elementary output-error state-space model identification algorithm. *International Journal of Control*, 56(5):1211–1241, 1992.

[99] M. Verhaegen and P. DeWilde. Subspace model identification part 1. the output-error state-space model identification clasdd of algorithms. *International Journal of Control*, 56(5):1187–1210, 1992.

[100] M. Verhaegen and D. Westwick. Identifying MIMO Hammerstein systems in the context of subspace model identification methods. International Journal of Control, Accepted for Publication.

[101] M. Verhaegen and D. Westwick. Identifying MIMO Wiener systems using subspace model identification methods. Signal Processing, Submitted.

[102] M. Viberg and B. Ottersten. Sensor array processing based on subspace fitting. *IEEE Transactions on Signal Processing*, 39(5):1110–1121, 1991.

[103] V. Volterra. *Theory of functionals and of integral and integro-differential equations*. Dover, New York, 1959.

[104] P.L. Weiss, I.W. Hunter, and R.E. Kearney. Rigid polyurethane foam casts for the fixation of human limbs. *Medical and Biological Engineering and Computing*, 22:603–604, 1984.

[105] P.L. Weiss, I.W. Hunter, and R.E. Kearney. Human ankle joint stiffness over the full range of muscle activation levels. *Jounral of Biomechanics*, 21:539–544, 1988.

[106] P.L. Weiss, R.E. Kearney, and I.W. Hunter. Position dependence of ankle joint dynamics - I. passive mechanics. *Journal of Biomechanics*, 19:727–735, 1986.

[107] P.L. Weiss, R.E. Kearney, and I.W. Hunter. Position dependence of ankle joint dynamics - II active mechanics. *Journal of Biomechanics*, 19:737–751, 1986.

[108] P.L. Weiss, R.E. Kearney, and I.W. Hunter. Position dependence of stretch reflex dynamics at the human ankle. *Experimental Brain Research*, 63:49–59, 1986.

[109] D.T. Westwick, E.J. Perreault, and R.E. Kearney. Estimation of Wiener kernels from non-white inputs: sampling rate considerations. In *Proceedings of Canadian Medical and Biological Engineering Conference*, volume 17, pages 125–126, May 1991.

[110] D.T. Westwick, E.J. Perreault, and R.E. Kearney. Sampling rate considerations in nonlinear system identification. In EMBS13 [18], pages 2303–2304.

[111] D.T. Westwick, E.J. Perreault, and R.E. Kearney. New approaches to the identification of stretch reflex dynamics. In *Proceedings of the Canadian Medical and Biological Engineering Conference*, volume 18, pages 56–57, June 1992.

[112] Southwell W.H. Fitting data to nonlinear functions with uncertainties in all measurement variables. *Comp J*, 19:69–73, 1975.

[113] N. Wiener. *Nonlinear problems in random theory*. Technology Press Research Monographs. Wiley, New York, 1958.

[114] T. Wigren. Convergence analysis iof recursive identification algorithms based on the nonlinear Wiener model. *IEEE Transactions on Automatic Control*, 39:2191–2206, 1994.

[115] S. Yasui. The use of clipped input information in multidimensional crosscorrelation for estimating Wiener-like kernels of non-linear systems. *International Journal of Systems Science*, 17:1421–1434, 1986.

[116] S. Yasui, W. David, and K.I. Naka. Spatio-temporal receptive field measurement of retinal neurons by random patern stimulation and cross correlation. *IEEE Transactions on Biomedical Engineering*, BME-26:263–272, 1979.

[117] B.Y. Yunan. Design and construction of a high performance electro-hydraulic actuating system used to identify the human ankle joint mechanics. Master's thesis, McGill University, Department of Mechanical Engineering, 1989.

[118] HP. Zeiger and A.J. McEwen. Approximate linear realization of given dimension via Ho's algorithm. *IEEE Transactions on Automatic Control*, 19(2):153, 1974.