An Efficient ARX Model Selection Procedure Applied to Autonomic Heart Rate Variability

by

Michael H. Perrott

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

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1992

Author .....
Department of Electrical Engineering and Computer Science
May 15, 1992

Certified by .....
Richard J. Cohen
Professor
Thesis Supervisor

Accepted by .....
Campbell L. Searle
Chairman, Departmental Committee on Graduate Students

An Efficient ARX Model Selection Procedure Applied to Autonomic Heart Rate Variability

by

Michael H. Perrott

Submitted to the Department of Electrical Engineering and Computer Science
on May 15, 1992, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

This thesis is geared toward the introduction and explanation of a novel algorithm that numerically estimates the transfer function relationship (in the form of 'impulse response' estimates) between input and output data sequences. In particular, the algorithm assumes that the relationship between any number of specified inputs and some chosen output is linear and time-invariant, and, further, that it can be placed into the ARX (AutoRegressive, eXtra-input) structure (in biomedical circles, this structure is often referred to as an ARMA (AutoRegressive, Moving Average) model). Any source of corrupting noise in the output data is assumed to be gaussian and uncorrelated with the inputs.

Using the outlined assumptions, the algorithm goes through several steps in determining the best ARX model to fit the given data. First, the algorithm uses the least squares procedure to estimate the parameters associated with a selected 'maximal model', which leads to an estimate of the corrupting noise variance. This 'maximal model' is chosen by the user, and is assumed to be of greater order (i.e. contain more parameters) than necessary to describe the relationship between the given inputs and output. In order to determine whether a lower order model can be selected, a series of consecutive hypothesis tests are performed as parameters are removed one at a time from the maximal model. Using the results of these tests, the algorithm outputs the lowest order model such that the change in prediction error between this model and the maximal model can be attributed to noise.

The novelty of the introduced algorithm rests in its efficiency of searching over lower order model sets. The classical approach to this problem has been simply to choose lower order models that contain arbitrary combinations of parameters, thus requiring the user to guess at which parameterizations might be most likely. Rather than taking such a brute force approach, the discussed algorithm determines which parameters are most likely to be unnecessary based on maximal model estimates, and then removes the parameters in order of least importance. This procedure is not only more efficient computationally, but it also allows for the selection of models that would otherwise not be considered.

As a means of verifying the effectiveness of the discussed algorithm, the influence of changes in respiration and arterial blood pressure on changes in heart rate is evaluated for a few experimental data sets. Through this application, the introduced algorithm is shown to be an effective tool for numerical transfer function analysis.

Thesis Supervisor: Richard J. Cohen
Title: Professor

0.1 Acknowledgements

There are many people I would like to thank for their help, friendship, and support as I undertook this research project. My thesis supervisor, Richard Cohen, had tremendous patience with me while I took more than a few tangents in my quest to understand the fundamentals of estimation theory. Kazuo Yana introduced me to this field while visiting MIT from Japan and has been a source of consistent encouragement and support ever since. Marvin Appel was always willing to listen to my ideas and, through our many discussions, provided me with many valuable insights on transfer function analysis. Tom Mullen was extremely generous with his time as he reviewed this thesis and offered his comments.

On a more personal note, I would like to thank my parents, Charles and Emily, for their encouragement and support during the times that I struggled here. I would also like to thank God, both for bringing me here and for getting me through the difficult times that came with it — I honestly believe that my life began when I placed my faith in His Son, Jesus Christ.

Contents

List of Figures

Chapter 1

Introduction

1.1 Objective

The focus of this thesis is on numerical transfer function analysis between experimental input and output data sets, where the output is assumed to have been corrupted by a gaussian noise source that is uncorrelated (on average) with the inputs. In regards to this analysis, 'impulse response' estimates are sought after to quantify an assumed linear, time-invariant system model between the chosen output sequence and considered input sequences. In order to incorporate a non-iterative, least squares procedure as the means of estimation, it will be assumed that the system model can be placed into the ARX (AutoRegressive, eXtra-input) model structure. With this restriction in mind, our goal is expressed as the determination of the best ARX model to describe the relationship between a given set of input and output data.

1.2 Overview of Standard Techniques of System Identification

The current approach to ARX model selection incorporates what is known as 'information criterion' evaluation. Defining residual error norm as a measure of the discrepancy between the predicted output of a chosen ARX model and the actual output, it is a characteristic of least squares estimation that residual error norm monotonically decreases as higher order models are considered. Unfortunately, if the model order is chosen too high, overfitting occurs to the corrupting noise that exists in the output sequence. The problem associated

with the selection of such an 'overparameterized' model is that it will perform poorly when applied to a new data set that has been corrupted differently by noise. As a means of trying to avoid the selection of such a model, information criterions such as AIC (Akaike's theoretic information criterion) have been developed which determine the optimal model through a minimization procedure. Essentially, these techniques penalize the amount of residual error norm occurring with a given model by the number of parameters used to achieve it. Thus, use of these criterions lead to a model that is the best compromise between having a low value of residual error norm and a small number of parameters. This method of model selection is often called 'objective' since the criterion function is not tunable by the user.

Unfortunately, there is more at issue than just the model order when trying the pick the best ARX model — the proper parameter *combination* must also be determined. Information criterions in and of themselves offer no efficient solution to this problem, so that a typical approach has been for the user to arbitrarily select some set of parameter combinations, and then choose the one that minimizes the given criterion function. When dealing with high order models, this approach can lead to very compromised results.

1.3 Proposed Method of System Identification

The proposed ARX model selection procedure presented in this thesis, which will be referred to as the APR (ARX Parameter Reduction) Algorithm, incorporates a novel 'parameter ordering' procedure aimed at determining the lowest order ARX parameter *combination* to represent the input to output relationship. Although this ordering procedure can be directly incorporated into the information criterion approach of model selection, we have applied it to an alternative method known as hypothesis testing.

Hypothesis testing approaches the problem of model selection with a different philosophy than that of information criterion evaluation. Rather than striving for the model with the best compromise between low residual error norm and the fewest number of parameters, this technique attempts to evaluate whether changes that occur in residual error norm can be attributed to the corrupting noise in the output. Specifically, a 'maximal model' is selected by the user that is assumed to be of higher order than required to represent the given system model. The residual error norm calculated for this model is then used to estimate the corrupting noise variance. As parameters are removed from the maximal model

according to the previously mentioned ordering procedure, a series of hypothesis tests are performed to determine the lowest parameter set such that the overall change in residual error norm can be ascribed to noise. This testing is done at a user specified 'confidence level', such that higher confidence levels result in the selection of lower order models. As such, this procedure is referred to as being 'subjective'. Since the number of parameters in a given estimation model directly corresponds to how much 'detail' can be represented in the estimated model, adjustment of the confidence level allows the user to have control over the level of detail considered by the estimation procedure.

1.4 Outline of Thesis

The fundamental concepts developed in this thesis are organized in the following format:

- Development of APR algorithm,
- Measure of Performance,
- Verification.

Development of the proposed model selection procedure is divided between chapters 2, 3, and 4. Chapter 2 is meant to give the reader an overview of system identification, least squares, and the ARX model. Building on this information, Chapter 3 outlines the concepts behind hypothesis testing and information criterion evaluation, and also presents the ordering procedure used by the APR algorithm. Chapter 4 strives to provide a more concrete understanding of the principles used in Chapters 2 and 3.

The considered measure of performance in our model selection procedure will be the amount of error associated with its 'impulse response' estimates. Presenting a means of evaluating this error, Chapter 5 will be devoted to the theory behind calculating error boundaries associated with the produced estimates.

In order to provide verification of the APR algorithm's ability to successfully perform ARX model selection, a chapter on simulated results and a chapter on experimental results are included. Chapter 6 revolves around the application of the APR algorithm to a fairly general simulation model under different configurations. Chapter 7 incorporates the algorithm in estimating impulse responses associated with a few experimental data sets involving variations in human respiration, arterial blood pressure, and heart rate.

Chapter 2

Parametric Estimation in the Presence of a Known Model

2.1 System Identification Basics

This thesis is geared toward the explanation and implementation of a novel technique to perform system identification when dealing with linear, time-invariant systems. As a step toward explaining this statement, we define a system to be any device or algorithm that creates an output sequence based on any number of known input sequences and some unobservable 'noise' sequence. To illustrate, consider Figure 2-1, which specifies that an output sequence, $\mathbf{y}$, is created upon the input sequences, $\mathbf{x_i}$, and noise sequence, $\mathbf{e}$, passing through the system, H_P. To formulate this process, we would write:

$$\mathbf{y} = H_P(\mathbf{x_1}, \cdots, \mathbf{x_n}, \mathbf{e}). \tag{2.1}$$

In the physical realm, $H_P(\cdot)$ is implemented via physical elements such as motors, transistors, capacitors, etc. In the case that our system is an algorithm, a computer program would implement $H_P(\cdot)$.

At a more abstract level, the behavior of a broad class of systems can actually be expressed in the form of a mathematical function, which shall, in such case, be referred to as the *model* of the system, denoted by $H(\cdot)$. In the context of this thesis, we will be concerned with systems that can be modeled and also exhibit the behavior of being linear and time-invariant.

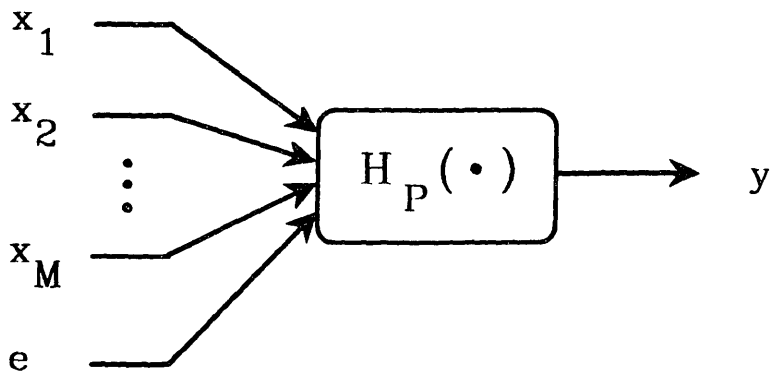


Figure 2-1: A general system model

The value of obtaining an accurate model to represent a system is significant when attempting to design or analyze its behavior. The best way to obtain an appropriate mathematical description is often through a brute force derivation procedure that makes use of physical laws such as Newtonian mechanics, Ohm's Law, etc. Unfortunately, this process is only feasible when the system under study is very well understood and fairly simple in structure. Many systems, such as the one considered in this thesis, do not fall into this category — they are just too complicated! When this first option falls through, a more 'empirical' procedure must be used. Rather than directly deriving a $H(\cdot)$, we are forced to *search* for an expression that closely approximates its mapping procedure. It is the end result of this procedure, i.e. the establishment of an $H(\cdot)$ that suitably describes the system's mapping characteristics, that defines the term 'identification'. Thus, system identification is the field dedicated to obtaining a mathematical expression, or model, that adequately describes a chosen system's procedure of forming an output based on given input sequences.

There are two main roads to take in the system identification world — parametric and nonparametric. The reader is invited to explore current textbooks to gain specific insight into the advantages each of these areas offer [5]. In general, nonparametric techniques are concerned with the retrieval of qualitative information about the behavior of a system. A 'picture' of $H(\cdot)$ is sought after rather than an explicit formula. Parametric techniques, on

the other hand, tend to be more quantitative in nature — they strive after mathematical expressions and thus offer an advantage in precision. As an added bonus, pictures can often be obtained from the produced expressions that are every bit as qualitative as could be offered by their nonparametric counterparts. Due to their attention to detail, however, parametric techniques often suffer the disadvantage of being more time consuming and complex.

This thesis introduces a new procedure to perform parametric system identification for the class of systems described as linear and time-invariant. The presented algorithm can be subdivided into two main components. The first component, perhaps better stated as a procedure, concerns the determination of optimal parameter values given a model structure. The technique used to achieve such a result, namely 'least squares', is not, by any means, new. However, the second procedure, which is concerned with obtaining the actual model structure itself, is new. These operations interplay on each other, necessitating a concrete understanding of both in order to understand the overall identification algorithm. Thus, for clarity, we have divided explanation of the presented technique into three chapters. This chapter will cover the basics of 'least squares' estimation applied to a specified model. The following chapter presents a search algorithm that uses 'least squares' in determining the specific model parameterization that will most efficiently describe the behavior of an investigated linear, time-invariant system. The third of the chapters is intended to fill in details left out of the first two.

2.2 Model Assumptions when using Least Squares

As described in the introductory section, our definition of a system essentially refers to the mapping from input and noise sequences to an output. It is assumed that we have knowledge of the input and output sequences, but that the noise sequence is unknown. Thus, in obtaining an estimate of the model of the system, the noise acts as a corrupting influence.

In order for least squares to be applicable to an estimation problem, input and output data must be placed into a series of finite length vectors. In particular, it must be the case that the output vector can be constructed from the addition of scaled vectors that are formed from input and/or output data, with a noise vector accounting for any discrepancy.

To illustrate this, consider an output, $\mathbf{y}$, that is formed by the addition of vectors $\mathbf{v}_i$ $(1 \leq i \leq M)$ and $\mathbf{e}$. Each of these vectors will be defined to contain N elements. Aside from the noise, a scaling parameter, p_i $(1 \leq i \leq M)$, is associated with each vector that is used to form $\mathbf{y}$. Altogether, we can express this relationship as:

$$\mathbf{y} = \sum_{i=1}^{M} p_i \mathbf{v}_i + \mathbf{e}. \tag{2.2}$$

The above equation can be placed into matrix notation by defining the following:

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_M \end{bmatrix}, \qquad \boldsymbol{\theta} = \begin{bmatrix} p_1 \\ \vdots \\ p_M \end{bmatrix}. \tag{2.3}$$

Using these definitions, the least squares procedure is geared toward estimating the parameters, $\boldsymbol{\theta}$, associated with a model that has the following structure:

$$\mathbf{y} = \mathbf{\Phi}\boldsymbol{\theta} + \mathbf{e}, \tag{2.4}$$

in which we have combined the previous two equations. Hereafter, we refer to $\mathbf{\Phi}$ as the 'data matrix' and $\boldsymbol{\theta}$ as the 'parameter vector'.

2.3 Implementation of Least Squares

The least squares procedure yields optimal results if certain characteristics or properties are assumed of the corrupting noise source. The reader is advised to consult reference [7] to gain a proper understanding of the following statement: The least squares procedure yields a 'maximum-likelihood' estimator which achieves the Cramer-Rao bound in estimator variance (the theoretical lower limit) if the corrupting noise sequence is white and gaussian.

We will briefly describe the equations used to arrive at least squares estimates, deferring the reader to chapter 4 for the derivation of this method. This procedure is remarkably simple to describe using matrix notation. We denote $\hat{\boldsymbol{\theta}}$ as the resulting estimates of the following least squares operation:

$$\hat{\boldsymbol{\theta}} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{y}. \tag{2.5}$$

Note that the above solution exists if and only if the matrix $\mathbf{\Phi}^T\mathbf{\Phi}$ is invertible. This condition is equivalent to stating that all the columns in $\mathbf{\Phi}$ must be independent.

Given that we have estimated the parameters of a given system, it is often useful to construct an estimate of the 'uncorrupted' output of the system:

$$\hat{\mathbf{y}}_u = \mathbf{\Phi}\hat{\boldsymbol{\theta}}. \tag{2.6}$$

We can express the actual output of the system in terms of this 'uncorrupted' output estimate:

$$\mathbf{y} = \hat{\mathbf{y}}_u + \mathbf{error}. \tag{2.7}$$

Comparing the above expression with equation 2.4, we note that perfect results for our estimation procedure, i.e. $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$, leads to an **error** vector that is equal to the noise vector **e**.

2.4 Example 1

To gain a better understanding of the principles discussed above, we now introduce an example. In a computer simulation, we formed a system that had the following model description:

$$\mathbf{y} = p_1\mathbf{x} + p_2\mathbf{x}^2 + \mathbf{e}. \tag{2.8}$$

Note that this system has associated with it an $H_P(\cdot) = H(\cdot) = p_1\mathbf{x} + p_2\mathbf{x}^2 + \mathbf{e}$, so that the system and its corresponding model description are equivalent. The parameters associated with the model are p_1 and p_2. These were arbitrarily chosen to be -0.9 and 0.5, respectively, in our simulation.

The input into the system, $\mathbf{x}$, was selected as the finite sequence:

$$\begin{bmatrix} -1.0 \\ -0.95 \\ -0.90 \\ \vdots \\ 0.95 \\ 1.0 \end{bmatrix}.$$

The noise source, **e**, was generated via a random number generating program that caused **e** to take on the characteristics of a white, gaussian noise process with variance = 0.0626. We chose the length of this noise sequence to be the same as that of the input sequence. The system output, **y**, was computed via the given model structure and the generated input and noise sequences.

To arrive at an estimate of parameters p_1 and p_2 from the input and output data, we defined the following:

$$\Phi = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 \end{bmatrix}, \qquad \boldsymbol{\theta} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}. \tag{2.9}$$

The least squares procedure was then implemented to produce estimates of the parameters contained within $\boldsymbol{\theta}$. The result of this operation was:

$$\hat{\boldsymbol{\theta}} = \left(\begin{bmatrix} \mathbf{x}^T \\ \mathbf{x}^{2T} \end{bmatrix} \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{x}^T \\ \mathbf{x}^{2T} \end{bmatrix} \mathbf{y} = \begin{bmatrix} -0.9096 \\ 0.5250 \end{bmatrix}. \tag{2.10}$$

For the purposes of evaluating the effectiveness of our estimation procedure, we made plots of the following:

1. 'uncorrupted output': $\mathbf{y}_u = \Phi\boldsymbol{\theta}$,
2. estimated 'uncorrupted output': $\hat{\mathbf{y}}_u = \Phi\hat{\boldsymbol{\theta}}$,
3. system output: **y**.

Accordingly, these are shown in Figure 2-2.

2.5 Discrete, Linear, Time-Invariant Systems

Although least squares can be used to estimate parameters associated with a fairly broad class of systems, our discussion will center around its application to models that exhibit the properties of being discrete, linear, and time-invariant. A detailed explanation of these properties requires a book in and of itself, and, in fact, the reader is guided to [11] as only one of many possible references. To briefly explain, however, we state that a discrete model satisfies the condition that it operates on and produces only sequences (as opposed to continuous waveforms). Thus, if the model is intended to represent a continuous time

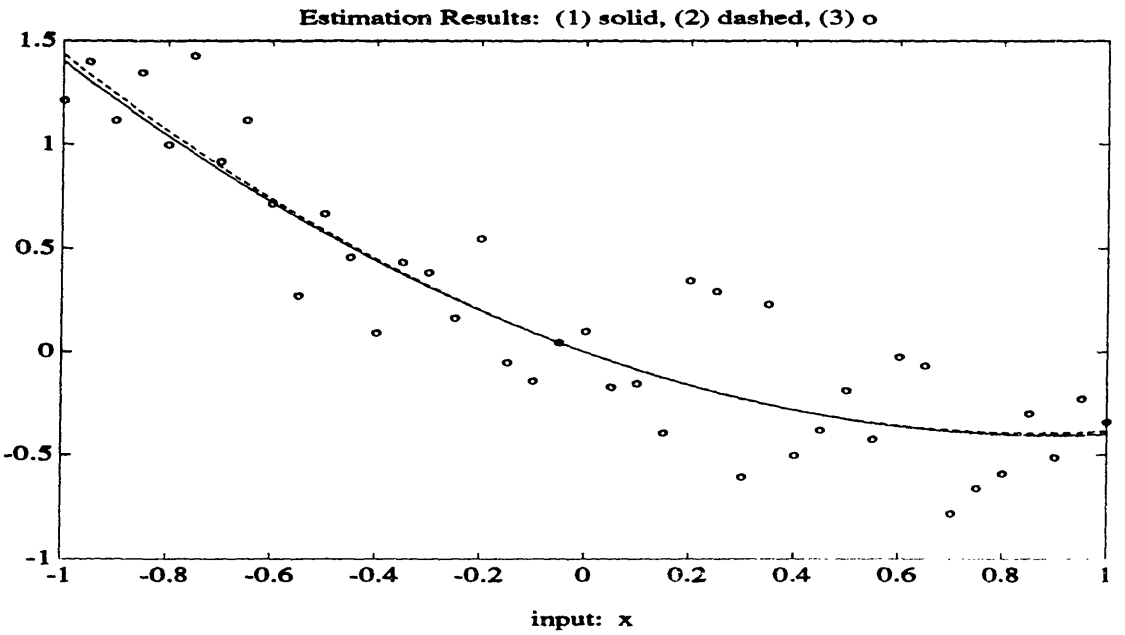


Figure 2-2: Estimation results for a correctly parameterized model

system, samples of the inputs and output of the system must be taken before the model can be used. Linearity and time-invariance are used to describe a general framework that the model is constrained to work within when mapping inputs to the output. Essentially, all of these properties can be summarized by simply stating that discrete, linear, time-invariant systems act according to the following equation:

$$\mathbf{y}[k] = \sum_{i=-\infty}^{+\infty} \mathbf{h}_1[i]\mathbf{x}_1[k-i] + \cdots + \sum_{i=-\infty}^{+\infty} \mathbf{h}_M[i]\mathbf{x}_M[k-i] + \sum_{i=-\infty}^{+\infty} \mathbf{h}_e[i]\mathbf{e}[k-i]. \qquad (2.11)$$

Where we have indicated that any chosen output of this type of system can be expressed as a sum of 'convolutions' of M inputs and one noise source with their respective 'impulse responses', $\mathbf{h}$. It should be noted that, in general, these impulse responses may be infinite in duration.

In order to simplify the analysis of discrete, linear, time-invariant systems, a powerful tool is often incorporated — the Z transform. As a general form of notation, we define $u(z)$ as the Z transform of $\mathbf{u}$, and compute this quantity as:

$$u(z) = \sum_{i=-\infty}^{+\infty} \mathbf{u}[k]z^{-i}. \qquad (2.12)$$

Use of this tool enables us to write equation 2.11 as a series of multiplications:

$$y(z) = h_1(z)x_1(z) + \cdots + h_M(z)x_M(z) + h_e(z)e(z). \tag{2.13}$$

This type of formulation leads to a convenient block diagram representation, as shown in Figure 2-3. As a matter of notation, we will refer to the Z transform of impulse responses

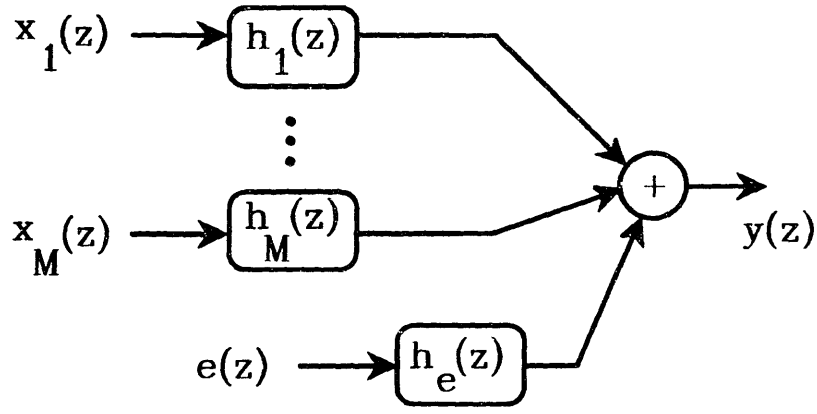


Figure 2-3: The linear, time-invariant (LTI) model

as transfer functions. Intuitively, observation of the above diagram reveals that the transfer functions associated with any given system act to map, or *transfer*, the input and noise sequences into the output sequence.

Given that we are dealing with systems that act on real inputs to produce real outputs (implying that their impulse responses are also real), it is important to realize that all of the transforms in equation 2.13 can be placed in the following form:

$$u(z) = \frac{b(z)}{a(z)} = \frac{b(z)}{1 + a^*(z)},$$

where $a(z)$ and $b(z)$ are polynomials in z with real coefficients. As a simple illustration of this property, consider the Z transform of the following impulse response:

$$\mathbf{h}[k] = 5 \cdot 0.5^k,\ k \geq 0\ (= 0, \text{otherwise}) \Longrightarrow h(z) = 5\sum_{k=0}^{\infty} 0.5^k z^{-k} = \frac{5}{1 - 0.5z^{-1}}$$

We will make use of this form in the case of transfer functions, leading us to express equation 2.13 as:

$$y(z) = \frac{b_{h_1}(z)}{a_{h_1}(z)}x_1(z) + \cdots + \frac{b_{h_M}(z)}{a_{h_M}(z)}x_M(z) + \frac{b_{h_e}(z)}{a_{h_e}(z)}e(z). \tag{2.14}$$

The relation expressed by equation 2.14 must be manipulated before it can be placed directly in the least squares format stated in equation 2.4. In striving toward this means, we write:

$$\frac{a_{h_e}(z)}{b_{h_e}(z)}y(z) = \frac{a_{h_e}(z)}{b_{h_e}(z)}\frac{b_{h_1}(z)}{a_{h_1}(z)}x_1(z) + \cdots + \frac{a_{h_e}(z)}{b_{h_e}(z)}\frac{b_{h_M}(z)}{a_{h_M}(z)}x_M(z) + e(z). \tag{2.15}$$

Given the fact that all the transfer functions expressed in equation 2.14 are strictly stable, along with the constraint that the transfer function corresponding to the noise dynamics, $\frac{b_{h_e}(z)}{a_{h_e}(z)}$, is minimum phase (implying that $\frac{a_{h_e}(z)}{b_{h_e}(z)}$ is strictly stable), equation 2.15 can be *approximated* by the following:

$$(1 + a^*(z))y(z) = b_1(z)x_1(z) + \cdots + b_M(z)x_M(z) + e(z), \tag{2.16}$$

where we are restricting $1 + a^*(z)$, $b_1(z)$, etc. to be *finite* polynomials in z with real coefficients (the fact that we are restricting the polynomials to be finite in order is what causes the expression to be an approximation). In explaining the above statement, note that specifying the noise dynamics to be minimum phase is required so that the transfer functions produced in equation 2.15 from equation 2.14 *remain* strictly stable. Given this fact, we can approximate each transfer function with a finite polynomial in z. To show this, consider the general Z transform relation for causal impulse responses:

$$h(z) = \sum_{k=0}^{\infty} \mathbf{h}[k]z^{-k} = \frac{b(z)}{a(z)}$$

One sees from the above expression that all transfer functions are identically equal to a corresponding infinite polynomial in z, whose coefficients correspond to different time increments in their respective impulse responses. By specifying that a given transfer function is strictly stable, we are enforcing the condition that its impulse response *decays*. Thus, for such a transfer function, its corresponding polynomial has coefficients that become

progressively smaller with increasing powers in z — the infinite polynomial can therefore be *approximated* with a finite polynomial. As an illustration of this behavior, consider the previously used example:

$$5\sum_{i=0}^{\infty} 0.5^k z^{-k} = \frac{5}{1-0.5z^{-1}}$$

Expanding the infinite polynomial on the left:

$$5(1 + 0.5z^{-1} + 0.25z^{-2} + .125z^{-3} + .0625z^{-4} + \cdots),$$

we note the approximation:

$$5(1 + 0.5z^{-1} + 0.25z^{-2} + .125z^{-3}) \approx \frac{5}{1-0.5z^{-1}}$$

In the control and signal processing literature, the structure expressed in equation 2.16 is known as the ARX model. In biomedical circles, it is referred to as the ARMA model. Although presented as an approximation to the general expression found in equation 2.14, this structure is extremely versatile in representing a broad class of LTI systems. The underlying reason for introducing this model is that it is of a form that can be directly placed into the least squares format, as will be discussed in the next section.

2.6 Least Squares applied to the ARX model

Equation 2.16 is easily manipulated to arrive at the relation:

$$y(z) = -a^*(z)y(z) + b_1(z)x_1(z) + \cdots + b_M(z)x_M(z) + e(z). \tag{2.17}$$

It is also straightforward to then inverse transform the above relation, obtaining the 'difference equation' format shown below:

$$\mathbf{y}[k] = \sum_{i=1}^{p} -\mathbf{a}^*[i]\mathbf{y}[k-i] + \sum_{i=s_1}^{f_1} \mathbf{b}_1[i]\mathbf{x}_1[k-i] + \cdots + \sum_{i=s_M}^{f_M} \mathbf{b}_M[i]\mathbf{x}_M[k-i] + \mathbf{e}[k]. \tag{2.18}$$

If we increment k over N values (i.e. $0 \leq k \leq N-1$), and place the corresponding samples of output and input values into vectors as such:

$$\mathbf{y}[k-i] \equiv \begin{bmatrix} \mathbf{y}[-i] \\ \mathbf{y}[-i+1] \\ \vdots \\ \mathbf{y}[-i+N-1] \end{bmatrix}, \; \mathbf{x}_i[k-i] \equiv \begin{bmatrix} \mathbf{x}_i[-i] \\ \mathbf{x}_i[-i+1] \\ \vdots \\ \mathbf{x}_i[-i+N-1] \end{bmatrix},$$

we can consider the above equation as the formation of an output vector, $\mathbf{y}[k]$, from the sum of *vectors* $\mathbf{y}[k-1]$, $\mathbf{y}[k-2]$, $\mathbf{x}_1[k-s_1]$, etc. Combining these vectors to form matrices, we define:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}[k-1] & \mathbf{y}[k-2] & \cdots & \mathbf{y}[k-p] \end{bmatrix},$$

$$\mathbf{X}_1 = \begin{bmatrix} \mathbf{x}_1[k-s_1] & \cdots & \mathbf{x}_1[k-f_1] \end{bmatrix}, \; \cdots \; \mathbf{X}_M = \begin{bmatrix} \mathbf{x}_M[k-s_M] & \cdots & \mathbf{x}_M[k-f_M] \end{bmatrix}. \tag{2.19}$$

Corresponding to the above data matrices, $\mathbf{a}^*$, $\mathbf{b}_1$, $\mathbf{b}_2$, etc., are vectors that contain the parameters associated with the model expressed in equation 2.18. To arrive at the least squares format, we simply denote:

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_M & \mathbf{Y} \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_M \\ -\mathbf{a}^* \end{bmatrix}, \tag{2.20}$$

and then rewrite equation 2.18 as:

$$\mathbf{y} = \mathbf{\Phi}\boldsymbol{\theta} + \mathbf{e}, \tag{2.21}$$

which is in the least squares format.

2.7 Example 2

To get a better idea of some of the issues involved with ARX modeling of LTI systems, we present the following example. Suppose we are given the single input system illustrated in Figure 2-4, which relates that a given input, $\mathbf{x}_1$, passes through two paralleled system blocks with transfer functions $h_{1P}(z)$ and $h_{1S}(z)$. We will specify that the impulse responses that

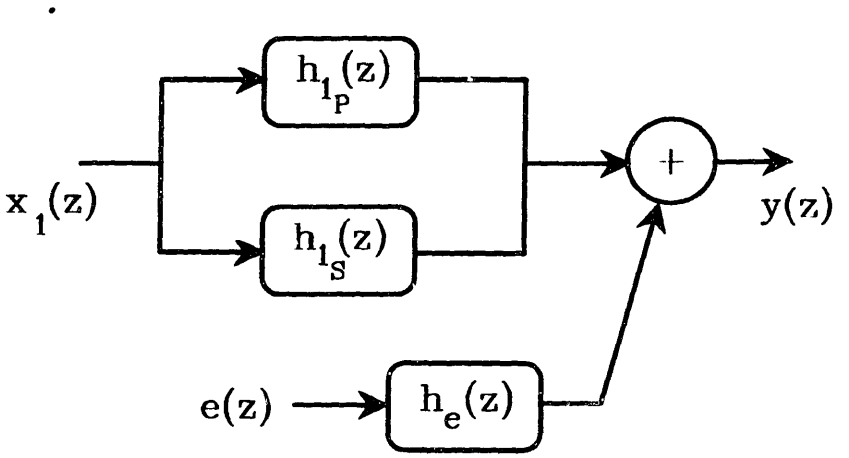


Figure 2-4: Example of a single input, LTI model

correspond to these transfer functions are:

$$\mathbf{h}_{1P}[k] = A_P \cdot \tau_P^{k-2},\ k \geq 2\ (= 0, \text{otherwise}) \Longrightarrow h_{1P}(z) = \frac{z^{-2}A_P}{1-\tau_P z^{-1}},$$

$$\mathbf{h}_{1S}[k] = A_S \cdot \tau_S^{k-7},\ k \geq 7\ (= 0, \text{otherwise}) \Longrightarrow h_{1S}(z) = \frac{z^{-7}A_S}{1-\tau_S z^{-1}}.$$

The noise sequence, **e**, passes through the transfer function $h_e(z)$ before contributing to the output **y**, with noise dynamics being specified as:

$$h_e(z) = \frac{1}{1-(\tau_P+\tau_S)z^{-1}+\tau_P\tau_S z^{-2}}.$$

The impulse responses associated with the paralleled system blocks are illustrated in Figure 2-5. Note the fact that the system has delays associated with it — i.e. the impulse responses within the paralleled blocks do not start at $k = 0$. The values of these delays are directly reflected in the numerators of $h_{1P}(z)$ and $h_{1S}(z)$.

Based on the given information, we can construct an expression that leads to the ARX parameterization of this system:

$$y(z) = \left(\frac{A_P z^{-2}}{1-\tau_P z^{-1}} + \frac{A_S z^{-7}}{1-\tau_S z^{-1}}\right) x_1(z) + \left(\frac{1}{1-(\tau_P+\tau_S)z^{-1}+\tau_P\tau_S z^{-2}}\right) e(z), \quad (2.22)$$

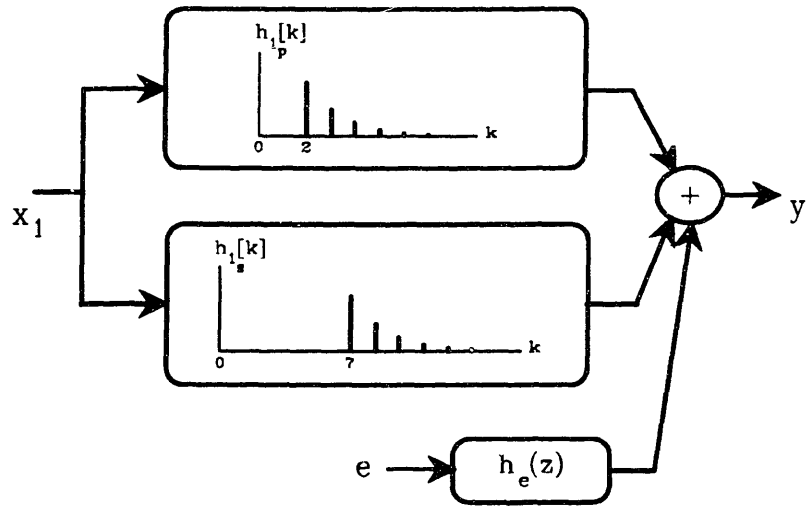


Figure 2-5: Sketch of impulse responses associated with example model

which is immediately rearranged as:

$$\left(1-(\tau_P+\tau_S)z^{-1}+\tau_P\tau_S z^{-2}\right)y(z)=\left(A_P z^{-2}-A_P\tau_S z^{-3}+A_S z^{-7}-A_S\tau_P z^{-8}\right)x_1(z)+e(z). \tag{2.23}$$

Inspection of equation 2.23 reveals that the given system has a model that can be directly placed into the ARX structure without approximation. As a sidenote, one should observe that this was the case because of the specified noise dynamics. For convenience, let us now define:

$$a^*(z)=-(\tau_P+\tau_S)z^{-1}+\tau_P\tau_S z^{-2},$$
$$b_1(z)=A_P z^{-2}-A_P\tau_S z^{-3}+A_S z^{-7}-A_S\tau_P z^{-8}.$$

With the aid of the above definitions, we can inverse Z transform equation 2.23 to obtain the difference equation:

$$\mathbf{y}[k]=-\sum_{i=1}^{2}\mathbf{a}^*[i-1]\cdot\mathbf{y}[k-i]+\sum_{i=0}^{8}\mathbf{b}_1[i]\cdot\mathbf{x}_1[k-i]+\mathbf{e}[k], \tag{2.24}$$

where we have implicitly defined the parameter vectors:

$$\mathbf{a}^* = \begin{bmatrix} -(\tau_P + \tau_S) \\ \tau_P \tau_S \end{bmatrix}, \qquad \mathbf{b}_1 = \begin{bmatrix} 0 \\ 0 \\ A_P \\ -A_P\tau_S \\ 0 \\ 0 \\ 0 \\ A_S \\ -A_S\tau_P \end{bmatrix}. \tag{2.25}$$

(Note that the starting index of all vectors is 0). One should note the fact that there are zeros occurring in the above $\mathbf{b}_1$ vector — we will refer to such entries as 'gaps'. It is useful to observe that these 'gaps' were caused by the delays that occurred in the paralleled system blocks (via the numerators in $h_{1P}(z)$ and $h_{1S}(z)$). Also, notice that if the delay in each system block were adjusted, the location of the 'gaps' within the $\mathbf{b}_1$ vector would change, but the $\mathbf{a}^*$ vector would remain unaffected. We will expand this observation into an assumption as we deal with the identification of LTI systems — namely, that delays occurring in the system will cause 'gaps' in the $\mathbf{b}_i$ vectors, but will have no effect on the $\mathbf{a}^*$ parameters.

To convert equation 2.24 into the least squares format, we specify:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}[k-1] & \mathbf{y}[k-2] \end{bmatrix}, \qquad \mathbf{X}_1 = \begin{bmatrix} \mathbf{x}_1[k] & \mathbf{x}_1[k-1] & \cdots & \mathbf{x}_1[k-8] \end{bmatrix}, \tag{2.26}$$

and, in turn, form the data matrix and parameter vector:

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{Y} \end{bmatrix}, \qquad \boldsymbol{\theta} = \begin{bmatrix} \mathbf{b}_1 \\ -\mathbf{a}^* \end{bmatrix}. \tag{2.27}$$

Making use of these definitions, equation 2.24 falls directly into the least squares format:

$$\mathbf{y} = \mathbf{\Phi}\boldsymbol{\theta} + \mathbf{e}. \tag{2.28}$$

2.8 The Issue of Overparameterization

Although the results obtained in the previous example were valid, they were somewhat inefficient in the stated form. In effect, we had 'overparameterized' the system — many of the entries in the $\mathbf{b}_1$ parameter vector had value zero and would therefore cause their

corresponding data matrix vectors (i.e. columns in $\mathbf{X}_1$ that were associated with these parameters) to be scaled by zero. Obviously, these 'zeroed-out' vectors have no influence on the formation of $\mathbf{y}$, the output vector, and can therefore be removed from the model description. In following through with this thought, let us define a new data matrix and parameter vector for the previous example. Consider:

$$\mathbf{b}_{1t} = \begin{bmatrix} A_P \\ -A_P\tau_S \\ A_S \\ -A_S\tau_P \end{bmatrix}, \quad \mathbf{X}_{1t} = \begin{bmatrix} \mathbf{x}_1[k-2] & \mathbf{x}_1[k-3] & \mathbf{x}_1[k-7] & \mathbf{x}_1[k-8] \end{bmatrix},$$
$$\mathbf{b}_{1ex} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{X}_{1ex} = \begin{bmatrix} \mathbf{x}_1[k] & \mathbf{x}_1[k-1] & \mathbf{x}_1[k-4] & \mathbf{x}_1[k-5] & \mathbf{x}_1[k-6] \end{bmatrix}. \tag{2.29}$$

Using the above definitions, the data matrix and parameter vector could be reconfigured as:

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_t & \vdots & \mathbf{\Phi}_{ex} \end{bmatrix} = \begin{bmatrix} \mathbf{X}_{1t} & \mathbf{Y} & \vdots & \mathbf{X}_{1ex} \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_t \\ \cdots \\ \boldsymbol{\theta}_{ex} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{1t} \\ \mathbf{a}^* \\ \cdots \\ \mathbf{b}_{1ex} \end{bmatrix}, \tag{2.30}$$

so that equation 2.28 is restated as:

$$\mathbf{y} = \begin{bmatrix} \mathbf{\Phi}_t & \mathbf{\Phi}_{ex} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}_t \\ \boldsymbol{\theta}_{ex} \end{bmatrix} + \mathbf{e} = \mathbf{\Phi}_t\boldsymbol{\theta}_t + \mathbf{\Phi}_{ex}\boldsymbol{\theta}_{ex} + \mathbf{e}. \tag{2.31}$$

A key point to notice about the above equation is that $\boldsymbol{\theta}_{ex}$ is defined to have entries that are all *zero*. Thus, it is equivalent to write the system description as:

$$\mathbf{y} = \mathbf{\Phi}_t\boldsymbol{\theta}_t + \mathbf{e}. \tag{2.32}$$

We will refer to such parameterizations, namely, those consisting only of 'true' parameters, as minimal models.

As we move on to the next chapter, it will be important to keep in mind the characteristics we have just observed in ARX parameterizations. In effect, we can separate model parameters into two groups. The term 'extra' parameters will be used to designate *zero*

valued (or very close to zero in terms of their absolute value) parameter vector entries. Conversely, 'true' parameters will be defined to take on *nonzero* values. If one glances back at the previous example, some facts can be inferred about these two sets. Equation 2.25 reveals that $\mathbf{b}_i$ parameters (i.e. parameters contained within any of the $\mathbf{b}_i$ vectors) can have 'gaps' ('extra' parameter entries) that are caused by delays occurring in the investigated system. However, the $\mathbf{a}^*$ parameters were unaffected by these delays and will thus be assumed to never contain such 'gaps'. These observations will take on an important role as we strive to develop an efficient model selection procedure.

Chapter 3

Techniques of ARX Model Selection

3.1 Introduction

This chapter is dedicated to the explanation of an efficient model search algorithm, which we will call the APR (ARX Parameter Reduction) algorithm, that makes use of least squares estimation. Our search will entail the determination of a 'minimal' ARX parameterization to represent a given system. In defining 'minimal', we are describing a model that adequately describes the given system's behavior without containing 'extra' parameters — i.e. parameters whose absolute value is very close or equivalent to *zero*. To determine such a model, a search procedure must be implemented that effectively compares different parameterizations. Using such a procedure, the parameterization that adequately describes the specified system and contains the fewest number of parameters is considered to be 'minimal'.

In order to understand the issues involved with searching for a suitable model structure, we must consider the characteristics of 'nonminimal' realizations. On opposite sides of the ideal, we have two cases: 'underparameterization' and 'overparameterization'. Underparameterized models lack the ability to represent their associated system's dynamics, while overparameterized models lack the ability to *predict* their system's behavior. To explore both possibilities, we will develop a means of evaluating the performance of chosen model structures, thus establishing a quantified procedure that will indicate how well the structure represents the given system. It is precisely for this purpose that we will incorporate least

squares estimation, along with evaluation tools known as information criterion selection and hypothesis testing.

3.2 Estimation Basics

To develop a mathematical procedure that determines the best model to represent a given system, we must quantify this condition. In other words, "what does it mean for a given model to be able to describe its associated system's behavior?" This question is somewhat subjective, but our response to it is the following: "a model that closely represents its associated system can produce, in the context of estimation, an uncorrupted output that closely matches the system's uncorrupted output." To explain this statement, we define the following:

$$\mathbf{y} = \mathbf{y}_u + \mathbf{e}, \tag{3.1}$$

and

$$\mathbf{y} = \hat{\mathbf{y}}_u + \mathbf{error}. \tag{3.2}$$

Equation 3.1 relates to the systems we will be concerned with — their output, $\mathbf{y}$, can be divided into two components. The first of these components, the 'uncorrupted output' $\mathbf{y}_u$, describes the portion of output that is exclusively formed by known *inputs* passing through the system. The other component, $\mathbf{e}$, is referred to as noise and is defined to be independent of any input activity. It is important to note the significance of this decomposition — we have separated the system output into one component that can be precisely predicted from the given system inputs (assuming we were given the system model), and one that is completely unknown (although we will assume that $\mathbf{e}$ has some characteristic behavior properties). Unfortunately, in the absence of a known system model, the fact that $\mathbf{e}$ is unknown causes $\mathbf{y}_u$ to also be unknown. In fact, given our definitions, we gain the same amount of information in determining $\mathbf{y}_u$ or $\mathbf{e}$ or the system model (since the inputs are known) — determining any of these quantities allows us to solve for any of the others.

Equation 3.2 brings us into the estimation world. In explaining its relevance, we must first clarify our assumptions. We will always assume that our only information relating to model identification is the following:

- the considered system is LTI and also capable of being represented with the ARX

model structure,

- all inputs of interest are known, and are given in the form of discrete sequences (all unknown inputs will contribute to the noise vector),
- the output, $\mathbf{y}$, is also known and given in the form of a discrete sequence,
- the unknown noise sequence, $\mathbf{e}$, will be assumed to have the properties of being white and gaussian.

With equation 3.2 in mind, our strategy will be to make use of the above information to estimate $\mathbf{y}_u$, implicitly identifying the model associated with the given system in the process. The error sequence, **error**, will be used to account for any discrepancy between $\hat{\mathbf{y}}_u$ and the system output, $\mathbf{y}$. As we noted in chapter 2, a perfect estimation procedure (i.e. one that found a $\hat{\mathbf{y}}_u$ that was identical to $\mathbf{y}_u$) would lead to **error** = $\mathbf{e}$.

Given that we have decided to estimate $\mathbf{y}_u$ in order to determine the model associated with a given system, our next step is to describe the assumptions and methodology associated with this procedure. As explained in chapter 2, we will always assume that the desired model can be placed into the least squares format:

$$\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{e} \implies \mathbf{y}_u = \boldsymbol{\Phi}\boldsymbol{\theta}, \tag{3.3}$$

in which $\boldsymbol{\theta}$ represents the system model parameters (knowledge of which completely determines the model structure), and $\boldsymbol{\Phi}$ represents input and output sequence information that the model parameters act on to form $\mathbf{y}_u$. There is a subtle point to be made regarding this last statement. We had previously specified that $\mathbf{y}_u$ could be completely constructed from the input sequences and system model alone. Thus, one might be led to ask why we have allowed the inclusion of output data in forming $\boldsymbol{\Phi}$. The answer to this question is found in the chapter 2 derivation of the ARX model structure. One should notice, upon observing the relevant section, that as a *restructuring* of the general discrete, LTI system model which considers only inputs and noise when determining the corresponding output, the ARX model uses input *and* output information (along with a noise sequence) to form each output value. An optimal method of estimating the model parameter values in equation 3.3, as outlined in chapter 2, involves the incorporation of the least squares procedure.

We are now in a position to talk about model selection. Given that we have chosen

a specific parameterization, we could calculate its parameter values by forming an appropriate Φ and then using least squares to obtain the required estimates. We could then evaluate how well the model performed (i.e. how closely the uncorrupted output estimate compared with the actual uncorrupted output). Upon repeating this procedure with different parameterizations, we could then use some criterion to evaluate, on the basis of each model's performance, which one should be selected. The catch of this previous description lies in the fact that the uncorrupted output is unknown and cannot be used for comparison purposes. The next two sections will be devoted toward overcoming this obstacle.

3.3 Example 1 Revisited

Let us return to the first example of chapter 2. Recall that we were given the system:

$$\mathbf{y} = p_1\mathbf{x} + p_2\mathbf{x}^2 + \mathbf{e}, \tag{3.4}$$

where we chose $p_1 = -0.9$ and $p_2 = 0.5$. Obviously, since both p_1 and p_2 are nonzero, the minimal model that would correspond to this system would simply be the system description itself (i.e. the above equation).

3.3.1 Underparameterization

Underparameterization refers to a model's lack of ability to represent a given system's behavior. To clarify this statement, let us consider the following estimation model:

$$\mathbf{y} = \hat{p}_1\mathbf{x} + \mathbf{error}, \tag{3.5}$$

in which $\hat{p}_1$ will be chosen such that the above equation does its best to describe the defined system's behavior — i.e. so that

$$\hat{\mathbf{y}}_u = [\mathbf{x}][\hat{p}_1] \approx \mathbf{y}_u = \Phi\boldsymbol{\theta} = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}.$$

As noted before, we normally don't have access to $\mathbf{y}_u$, so as a compromise we will simply choose $\hat{p}_1$ such that the *energy* of the **error** sequence is minimized. For convenience, we

represent this quantity as:

$$\|\mathbf{error}\|_2^2 = \sum_{i=0}^{N-1} (\mathbf{error}[i])^2, \tag{3.6}$$

where we are assuming the sequence consists of N elements. (As a matter of notation, we will refer to $\|\cdot\|_2$ as the 'l_2 norm' of a sequence, and the energy of a sequence as the square of its l_2 norm.) The motivation for minimizing $\|\mathbf{error}\|_2^2$ rests in the fact that it leads directly to the implementation of least squares. Thus, we write:

$$\min_{\hat{p_1}}(\|\mathbf{error}\|_2^2) \implies \hat{p_1} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}. \tag{3.7}$$

Using the simulation data generated in Example 1, we performed the above operation and obtained $\hat{p_1} = -0.91$. To examine how well the specified model performed, we made plots of the following:

1. actual 'uncorrupted output': $\mathbf{y}_u = \mathbf{\Phi\theta}$,
2. estimated 'uncorrupted output': $\hat{\mathbf{y}}_u = [\mathbf{x}]\,[\hat{p_1}]$,
3. system output: $\mathbf{y}$,

which are shown in Figure 3-1. Inspection of this graph illustrates that the selected model did a poor job of representing the system's behavior — $\hat{\mathbf{y}}_u$ is not very well matched to $\mathbf{y}_u$. Essentially, the problem with this parameterization is that it lacks the *degrees of freedom* necessary to represent $\mathbf{y}_u$.

3.3.2 Overparameterization

Overparameterization refers to the selection of a model that has more parameters than necessary to describe the given system's behavior. To explore this issue, let us now consider the following estimation model:

$$\mathbf{y} = \hat{p_1}\mathbf{x} + \hat{p_2}\mathbf{x}^2 + \hat{p_3}\mathbf{x}^3 + \hat{p_4}\mathbf{x}^4 + \hat{p_5}\mathbf{x}^5 + \mathbf{error}_5. \tag{3.8}$$

Again, we wish to choose the model parameters (in this case, $\hat{p_1}, \hat{p_2}, \hat{p_3}$, $\hat{p_4}$, and $\hat{p_5}$) such that $\mathbf{y}_u$ is best represented. Reverting again to matrix notation, this desire is expressed as:

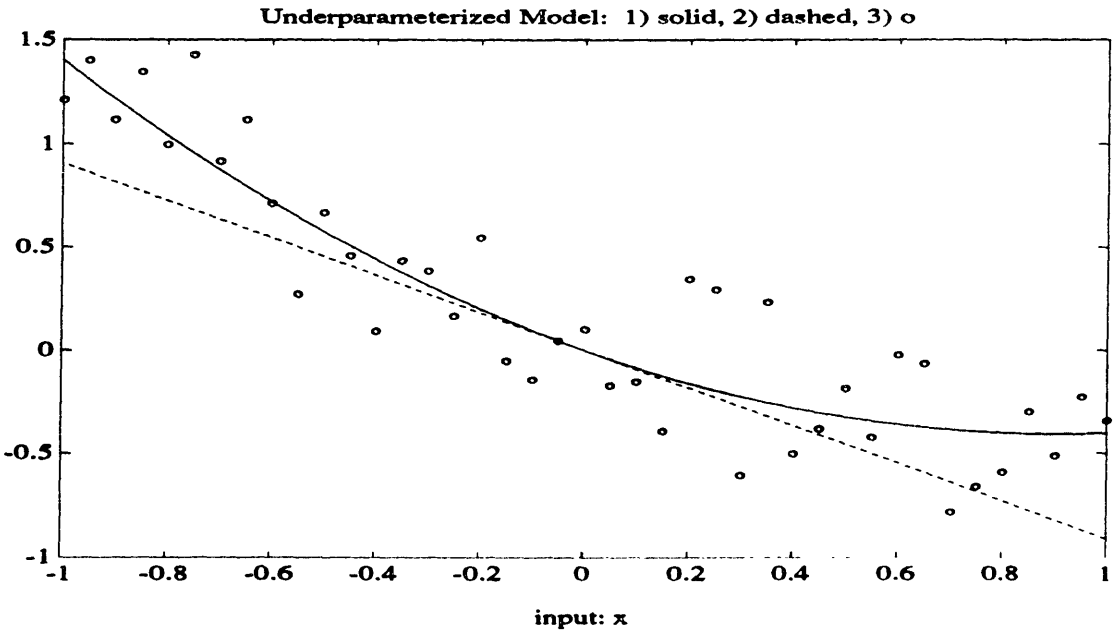


Figure 3-1: Estimation results for an underparameterized model

$$\hat{\mathbf{y}}_u = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 & \mathbf{x}^3 & \mathbf{x}^4 & \mathbf{x}^5 \end{bmatrix} \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \hat{p}_3 \\ \hat{p}_4 \\ \hat{p}_5 \end{bmatrix} \approx \mathbf{y}_u = \boldsymbol{\Phi\theta} = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}.$$

Since we normally don't have access to $\mathbf{y}_u$, we again choose parameter values such that the l_2 norm of the $\mathbf{error}_5$ sequence is minimized (which is equivalent to minimizing its *energy*), as solved by least squares:

$$\begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \hat{p}_3 \\ \hat{p}_4 \\ \hat{p}_5 \end{bmatrix} = \left(\begin{bmatrix} \mathbf{x}^T \\ \mathbf{x}^{2T} \\ \mathbf{x}^{3T} \\ \mathbf{x}^{4T} \\ \mathbf{x}^{5T} \end{bmatrix} \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 & \mathbf{x}^3 & \mathbf{x}^4 & \mathbf{x}^5 \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{x}^T \\ \mathbf{x}^{2T} \\ \mathbf{x}^{3T} \\ \mathbf{x}^{4T} \\ \mathbf{x}^{5T} \end{bmatrix} \mathbf{y}. \tag{3.9}$$

Using the simulation data generated in Example 1, we performed the above operation and obtained:

$$\begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \hat{p}_3 \\ \hat{p}_4 \\ \hat{p}_5 \end{bmatrix} = \begin{bmatrix} -0.40 \\ 0.62 \\ -2.18 \\ -0.13 \\ 1.83 \end{bmatrix}.$$

Similar to the previous section, we made plots of the following:

1. actual 'uncorrupted output': $\mathbf{y}_u = \Phi\boldsymbol{\theta}$,

2. estimated 'uncorrupted output': $\hat{\mathbf{y}}_u = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 & \mathbf{x}^3 & \mathbf{x}^4 & \mathbf{x}^5 \end{bmatrix} \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \hat{p}_3 \\ \hat{p}_4 \\ \hat{p}_5 \end{bmatrix}$,

3. system output: $\mathbf{y}$,

which are shown in Figure 3-2. Inspection of this graph illustrates that the above model

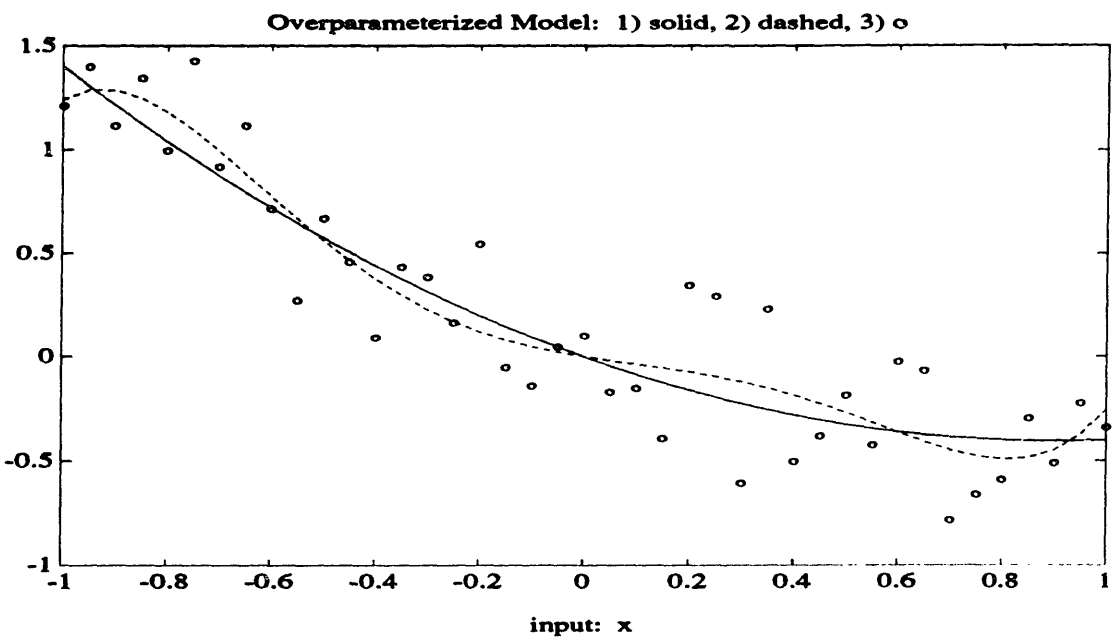


Figure 3-2: Estimation results for overparameterized model

has *more* degrees of freedom than necessary to represent the specified $\mathbf{y}_u$. As a result, the model is unnecessarily sensitive toward accommodating the noise induced features in the output, and would thus do a poor job of *predicting* the system's behavior under a different set of data.

3.4 Using Residual Error to Determine Model Order

As observed in the previous example, it is undesirable to under or overparameterize a system. It is therefore our goal to select a model that contains the minimum number of parameters necessary to represent the given system. As mentioned before, the only practical strategy toward achieving this goal is to choose different parameterizations and compare how well they perform. Although we have defined performance in terms of how well the given model represents the system's uncorrupted output, this signal will be unknown to us in the context of estimation. Thus, we are forced to pick a different criterion to examine as we compare the different parameterizations. This is not to say that our previous definition of performance is void — rather, we simply need to find a criterion that will, however indirectly, give us an idea of how well $\mathbf{y}_u$ is being represented for any given model.

We have been consistently using least squares to determine the model parameter estimates — it is only fitting to pick some criterion associated with this procedure to evaluate model performance. Since least squares is geared toward the minimization of the l_2 norm of the **error** sequence in any given estimation model, a natural quantity to observe is precisely the resulting $||\mathbf{error}||_2^2$ under different parameterizations, which will be referred to as the 'residual error norm'.

3.4.1 Residual Error Norms as a Function of Model Order

Let us return again to the system portrayed in Example 1. Using the same simulation data pertaining to the system:

$$\mathbf{y} = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \mathbf{e},$$

we will attempt to describe its behavior with models of increasing *order* (we will define 'order' to be the number of parameters in any given model). Proceeding, let us consider the model structures:

$$
\begin{aligned}
&(0)\ \mathbf{y} = \mathbf{error}_0, && (3)\ \mathbf{y} = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 & \mathbf{x}^3 \end{bmatrix} \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \hat{p}_3 \end{bmatrix} + \mathbf{error}_3, \\
&(1)\ \mathbf{y} = [\mathbf{x}][\hat{p}_1] + \mathbf{error}_1, && (4)\ \mathbf{y} = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 & \mathbf{x}^3 & \mathbf{x}^4 \end{bmatrix} \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \hat{p}_3 \\ \hat{p}_4 \end{bmatrix} + \mathbf{error}_4, \\
&(2)\ \mathbf{y} = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 \end{bmatrix} \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \end{bmatrix} + \mathbf{error}_2, && (5)\ \mathbf{y} = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 & \mathbf{x}^3 & \mathbf{x}^4 & \mathbf{x}^5 \end{bmatrix} \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \hat{p}_3 \\ \hat{p}_4 \\ \hat{p}_5 \end{bmatrix} + \mathbf{error}_5.
\end{aligned}
\tag{3.10}
$$

(where the quantity in parenthesis denotes the number of parameters included in the specified model) Performing least squares on these estimation models led to different error sequences, $\mathbf{error}_i$. For the given simulation data, we calculated $\|\mathbf{error}_i\|_2^2$ for each parameterization, which led to Figure 3-3. The precise residual error norm values for each

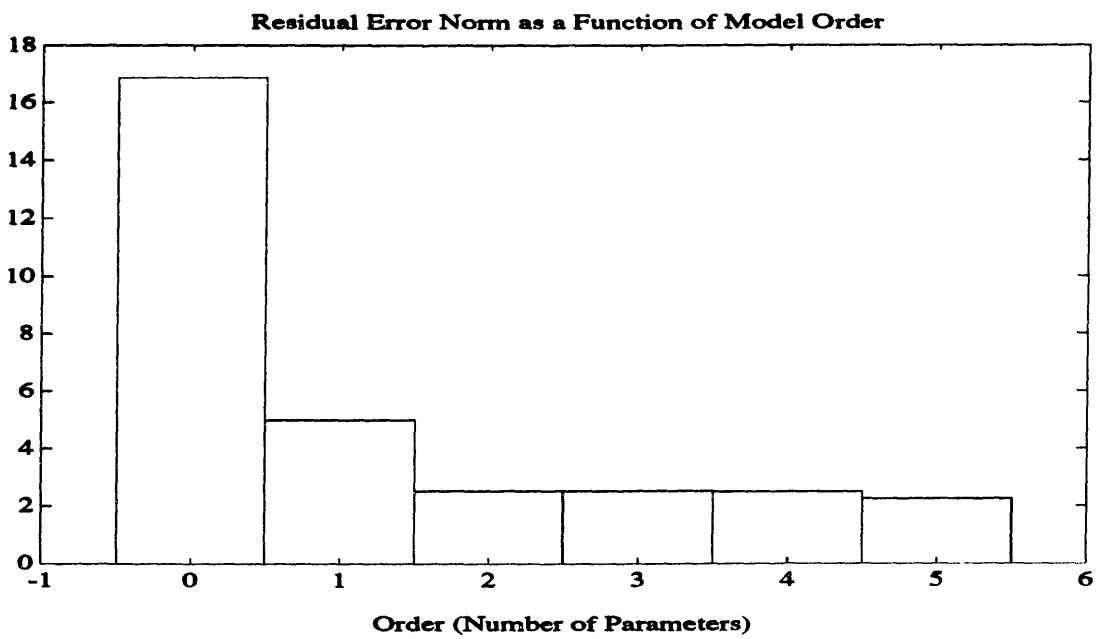


Figure 3-3: Plot of residual error norm versus model order

parameterization were:

$$
(0)\ 16.867,\quad (1)\ 4.995,\quad (2)\ 2.505,\quad (3)\ 2.502,\quad (4)\ 2.495,\quad (5)\ 2.245 \tag{3.11}
$$

One should observe that as the 'true' parameters were added to the estimation model (corresponding to (1) and (2)), the residual error dropped dramatically. Once the 'true' parameters had been included, the 'extra' parameters served only to lower the residual error norm slightly.

This illustration touches upon a very important idea — 'extra' parameters are generally less effective at reducing residual error than 'true' parameters as they are included in the estimation model (we are assuming that the 'true' parameters have been 'clumped' together and added to the model structure before their 'extra' counterparts). There are two common methods often employed that exploit this fact in order to discriminate between these parameter sets — hypothesis testing and information criterion evaluation.

3.5 Hypothesis Testing

'Hypothesis testing' is often classified as being a *subjective* criterion for parameter discrimination. This technique is built around the use of assumed characteristics in the corrupting noise process in evaluating changes in residual error norms. In the context of this thesis, we will assume that $\mathbf{e}$ is white and gaussian, and has variance σ_e^2 (this is actually the standard assumption in most estimation problems). Given this fact, we can specify a probability distribution associated with the change occurring in $\|\mathbf{error}\|_2^2$ upon the inclusion of 'extra' parameters. The results that follow are derived in chapter 4.

Let us consider the system:

$$\mathbf{y} = [\boldsymbol{\Phi}_t][\boldsymbol{\theta}_t] + \mathbf{e}, \tag{3.12}$$

for which we will consider the estimation models:

$$\mathbf{y} = [\boldsymbol{\Phi}_t]\left[\hat{\boldsymbol{\theta}}_t\right] + \mathbf{error}_t, \tag{3.13}$$

and

$$\mathbf{y} = \begin{bmatrix} \boldsymbol{\Phi}_t & \boldsymbol{\Phi}_{ex} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\theta}}_t \\ \hat{\boldsymbol{\theta}}_{ex} \end{bmatrix} + \mathbf{error}_{t+ex}, \tag{3.14}$$

in which we have used the notation introduced in chapter 2 — $\boldsymbol{\theta}_t$ represents the 'true' parameters of the system, $\hat{\boldsymbol{\theta}}_t$ their estimates, and $\hat{\boldsymbol{\theta}}_{ex}$ the estimates that correspond to 'extra', or unneeded, parameters.

Assuming that we had data corresponding to the above system, the least squares procedure would lead to estimates associated with the given models such that their **error** sequences would have the following property:

$$\frac{\|\mathbf{error}_t\|_2^2 - \|\mathbf{error}_{t+ex}\|_2^2}{\sigma_e^2} \rightsquigarrow \chi^2_{dim(\hat{\boldsymbol{\theta}}_{ex})}, \tag{3.15}$$

where $\rightsquigarrow$ denotes: "is distributed according to." Thus, the above equation relates that the left side takes on, in a statistical sense, values that are distributed according to the probability density function given on the right. We have designated $\chi^2_{dim(\hat{\boldsymbol{\theta}}_{ex})}$ to represent the Chi-Square probability density with degrees of freedom equal to the dimension of (i.e. the number of elements contained within) the 'extra' parameter vector $\hat{\boldsymbol{\theta}}_{ex}$. The relation expressed in equation 3.15 holds even if the $\mathbf{error}_t$ sequence is associated with a model that contains 'extra' parameters, so long as it also contains *all* of the 'true' parameters (we will defer any proof of this statement until chapter 4). Further explaining this equation, the notation $\mathbf{error}_{t+ex}$ refers to a model that contains all of the parameters corresponding to $\mathbf{error}_t$, to which an additional set of 'extra' parameters has been included. Thus, the normalized difference between residual error norms occurring before and after the inclusion of a set of 'extra' parameters is distributed according to the Chi-Square distribution. This property can be made use of in the context of a hypothesis test that uses changes in residual variance to evaluate whether a set of parameters is 'extra'.

Proceeding, let us consider the following null hypothesis and its alternate:

- h_0: All of the considered parameters are 'extra',
- h_{alt}: Not all of the considered parameters are 'extra'.

To decide between h_0 and h_{alt}, we make use of equation 3.15 and evaluate the validity of the following expression:

$$\frac{\|\mathbf{error}_t\|_2^2 - \|\mathbf{error}_{t+ex}\|_2^2}{\sigma^2} \stackrel{?}{>} \mu_{ex}, \tag{3.16}$$

where we define μ_{ex} in terms of a specified confidence level, η (which must takes on values between 0.0 and 1.0):

$$\eta = \int_0^{\mu_{ex}} \chi^2_{dim(\hat{\boldsymbol{\theta}}_{ex})}(x_0)dx_0. \tag{3.17}$$

Equation 3.17 is illustrated in Figure 3-4 for the case where $dim(\hat{\theta}_{ex}) = 2$ (The area represented by the shaded region is equivalent to η).

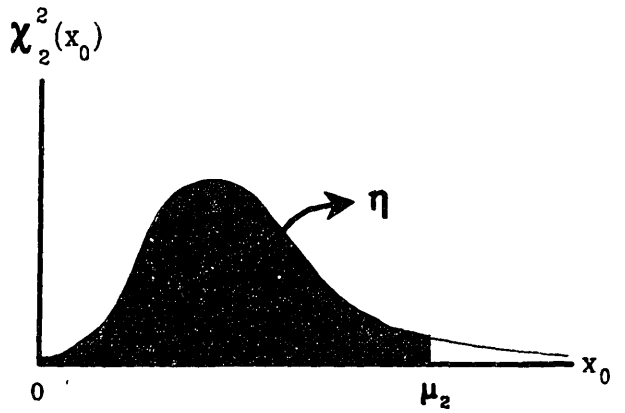


Figure 3-4: Plot of Chi Square probability density function with two degrees of freedom

To perform the hypothesis test, we simply evaluate whether equation 3.16 is valid for the given data. If so, h_{alt} is accepted to be the valid statement. Otherwise, h_0 is chosen.

3.5.1 Example — Basic Hypothesis Test

Let us return again to our continuing example. Recalling the maximal estimation model considered in section 3.3.2, let us now denote:

$$\theta_t = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \implies \hat{\theta}_t = \begin{bmatrix} \hat{p_1} \\ \hat{p_2} \end{bmatrix}, \quad \hat{\theta_{ex}} = \begin{bmatrix} \hat{p_3} \\ \hat{p_4} \\ \hat{p_5} \end{bmatrix}.$$

Corresponding to the above parameter vectors, we have the data matrices:

$$\Phi_t = \begin{bmatrix} \mathbf{x} & \mathbf{x}^2 \end{bmatrix}, \quad \Phi_{ex} = \begin{bmatrix} \mathbf{x}^3 & \mathbf{x}^4 & \mathbf{x}^5 \end{bmatrix}.$$

To make use of equation 3.16, we must first pick a confidence level between 0.0 and 1.0 — we'll choose 0.95 for this example. The establishment of this confidence level leads directly to a value of $\mu_{ex} = 7.815$, which corresponds to the integration of the Chi Square distribution with 3 degrees of freedom (since $dim(\hat{\theta}_e) = 3$, in this case). Recalling the data

found in equation 3.11:

$$(0)\ 16.867,\quad (1)\ 4.995,\quad (2)\ 2.505,\quad (3)\ 2.502,\quad (4)\ 2.495,\quad (5)\ 2.245,$$

and the fact (from chapter 2) that $\sigma_e^2 = 0.0626$ for the simulated noise (one should note that the value of noise variance will not be given to us in the context of estimation, but a simple and accurate procedure to estimate this value is introduced in the next section and derived in chapter 4), we now write:

$$\frac{\|\mathbf{error}_t\|_2^2 - \|\mathbf{error}_{t+ex}\|_2^2}{\sigma_e^2} = \frac{2.505 - 2.245}{0.0626} = 4.153 \overset{?}{>} 7.815, \tag{3.18}$$

which is obviously not true! Therefore, h_0 is correctly accepted — all of the parameters associated with $\hat{\boldsymbol{\theta}}_{ex}$ are 'extra'.

3.5.2 Example — Model Selection using Hypothesis Testing

Let us now adapt the previous illustration to the problem of determining an estimation model that contains only 'true' parameters when the number of these parameters is unknown. A natural way to proceed toward the solution of such a problem would be to select a 'maximal' model, i.e. a model that clearly overparameterizes the given system, and then eliminate those parameters that are deemed unnecessary. The value of this approach lies in the fact that it can directly incorporate the method summarized in equation 3.16.

For convenience, we introduce a modification tc the previously used notation. In the normal context of estimation, we will be unaware of how the parameter vector, $\hat{\boldsymbol{\theta}}$, separates into $\hat{\boldsymbol{\theta}}_t$ and $\hat{\boldsymbol{\theta}}_{ex}$. Therefore, let us simply denote estimation vectors according to how many parameters they contain. In order to keep track of the actual parameters contained within a specified parameter vector, we will select some sort of ordering procedure prior to building each vector.

Defining a maximal model set that contains the five parameters $\hat{p}_1$, $\hat{p}_2$, $\cdots$ $\hat{p}_5$, we will specify, for this example, the following ordering procedure:

$$\hat{p}_1 \leftarrow \hat{p}_2 \leftarrow \hat{p}_3 \leftarrow \hat{p}_4 \leftarrow \hat{p}_5 \leftarrow,$$

in which the arrows indicate the order in which we will *remove* parameters from the maximal

model when forming parameter vectors. This procedure was illustrated in equation 3.10, for which we now write:

$$\begin{aligned} &(0)\ \mathbf{y} = \mathbf{\Phi_0}\hat{\boldsymbol{\theta}}_0 + \mathbf{error_0} \qquad (3)\ \mathbf{y} = \mathbf{\Phi_3}\hat{\boldsymbol{\theta}}_3 + \mathbf{error_3} \\ &(1)\ \mathbf{y} = \mathbf{\Phi_1}\hat{\boldsymbol{\theta}}_1 + \mathbf{error_1} \qquad (4)\ \mathbf{y} = \mathbf{\Phi_4}\hat{\boldsymbol{\theta}}_4 + \mathbf{error_4} \\ &(2)\ \mathbf{y} = \mathbf{\Phi_2}\hat{\boldsymbol{\theta}}_2 + \mathbf{error_2} \qquad (5)\ \mathbf{y} = \mathbf{\Phi_5}\hat{\boldsymbol{\theta}}_5 + \mathbf{error_5} \end{aligned} \tag{3.19}$$

To determine which parameters can be removed from the maximal model set, $\hat{\boldsymbol{\theta}}_5$, we apply equation 3.16 in a modified notation:

$$\frac{\|\mathbf{error}_\kappa\|_2^2 - \|\mathbf{error}_{max}\|_2^2}{\sigma_e^2} \overset{?}{>} \mu_{(max-\kappa)}. \tag{3.20}$$

We now demonstrate this equation using the data given in equation 3.11:

$$(0)\ 16.867,\quad (1)\ 4.995,\quad (2)\ 2.505,\quad (3)\ 2.502,\quad (4)\ 2.495,\quad (5)\ 2.245,$$

and associated Chi Square values selected at the 0.95 confidence level:

$$\begin{aligned} &(0)\ \tfrac{16.867-2.245}{0.0626} = 233.58 \overset{?}{>} \mu_5 = 11.070, \quad (3)\ \tfrac{2.502-2.245}{0.0626} = 4.11 \overset{?}{>} \mu_2 = 5.991, \\ &(1)\ \tfrac{4.995-2.245}{0.0626} = 43.93 \overset{?}{>} \mu_4 = 9.488, \qquad (4)\ \tfrac{2.495-2.245}{0.0626} = 3.99 \overset{?}{>} \mu_1 = 3.841 \\ &(2)\ \tfrac{2.505-2.245}{0.0626} = 4.15 \overset{?}{>} \mu_3 = 7.815, \end{aligned} \tag{3.21}$$

To determine which model should be selected, we simply choose the one that contains the fewest parameters while remaining within its associated Chi Square limits (i.e. for which $\mu_{(max-\kappa)}$ remains less than the quantity compared to it). For the above example, the chosen model would thus be correctly selected as $\kappa = 2$!

The previous operation can be expressed from an alternative viewpoint — namely, we are constructing a boundary within which changes in residual error from the maximal model can be 'explained' by noise at a specified confidence level using the Chi Square density function. Upon calculating the resulting residual error norms, we simply choose the smallest parameter set that remains within these bounds. To illustrate this viewpoint for the given example, we have included Figure 3-5, from which we again observe that $\kappa = 2$ is selected.

As a general observation, note that upon removing $\hat{p}_5$ (corresponding to (4)), the residual error changed by an amount that was not explainable by noise at the specified confidence

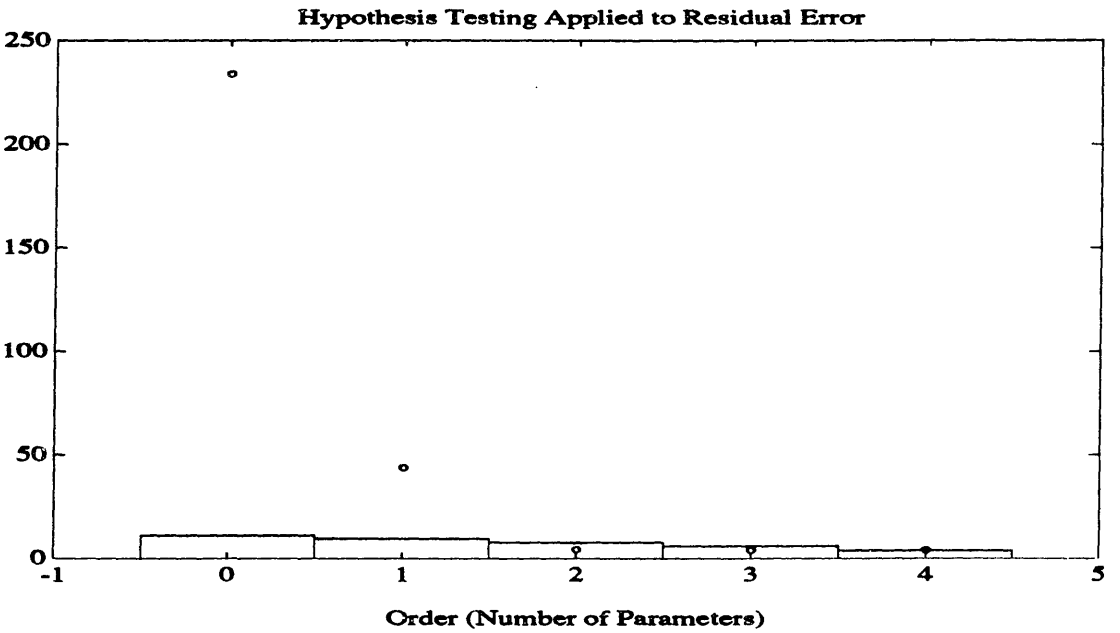


Figure 3-5: Illustration of parameter selection using hypothesis testing, where the bar graph corresponds to the Chi Square bounds occurring at a 0.95 level of confidence, and the 'o' points correspond to the norm of the resulting 'normalized' residual error difference between the maximal model and lower order models

level (i.e. $3.99 > \mu_1 = 3.841$). One could easily infer from this fact that $\hat{p}_5$ was not 'extra', but rather a 'true' parameter that should be retained within the model. In this case, since we know that $\boldsymbol{\theta}_t$ does not, in fact, contain p_5, such an inference would be false. Generally speaking, when the residual error norm changes enough to give an 'extra' parameter the impression of being a 'true' parameter, we will refer to this change as an 'outlier'. To protect ourselves from accepting these unnecessary parameters as being 'true', we will always assume that the 'true' parameters have been clumped together in the ordering procedure (we will discuss this in more detail later), and thus choose the minimal parameter *set* that stays within Chi Square limits (as portrayed in this example).

A second point to make is that we used the noise variance, σ_e^2, to normalize our changes in residual error norms. Unfortunately, as was indicated at the beginning of this chapter, we do not have access to this value in the normal context of estimation. However, as will be derived in chapter 4, it is straightforward to *estimate* this variance with the residual error norm obtained for any model that contains all of the 'true' parameters associated with the system. Since we are assuming this fact of the maximal model, we can estimate the noise

variance as:

$$\hat{\sigma_e^2} = \frac{1}{N - k_{max}} \|\mathbf{error}_{k_{max}}\|_2^2, \tag{3.22}$$

where N represents the number of samples used in the estimation process (i.e. the number of elements in $\mathbf{error}_{k_{max}}$). For our example, $N = 41$ and $k_{max} = 5$, so that the simulation results yield:

$$\hat{\sigma_e^2} = \frac{2.245}{41 - 5} = 0.0624,$$

which is extremely close to the true value of $\sigma_e^2 = 0.0626$!

3.6 Information Criterions

This section outlines an alternative method for determining a 'minimal' model associated with a given system — information criterion evaluation. Model selection based on this technique is often referred to as being 'objective', in the sense that the user need not specify a confidence level as was required by hypothesis testing. Overall, the strategy of this approach is as follows: compute the residual errors associated with a set of selected models (in precisely the same manner as presented in the hypothesis testing section), weight the obtained residual errors by a function that considers the number of parameters associated with each error, and then choose the model that minimizes the 'weighted' errors. There are two weighting functions that are currently popular: Akaike's Information Theoretic Criterion (AIC) and Rissanen's minimum description length (MDL). Both of these functions are very similar and straightforward to implement.

Generally speaking, use of AIC is known to lead to overparameterized models. However, we will include it for completeness. The following equation can be found in [6]:

$$AIC(\kappa) \approx (1 + 2 * \kappa/N) * \frac{\|\mathbf{error}_\kappa\|_2^2}{N}, \tag{3.23}$$

with N representing the number of samples (i.e. the number of elements in $\mathbf{error}_k$) used in the estimation procedure. To illustrate the use of this equation, we need simply make use of the data found in equation 3.11:

(0) 16.867, (1) 4.995, (2) 2.505, (3) 2.502, (4) 2.495, (5) 2.245,

and note the fact that $N = 41$ for that example. Thus, we write the result of weighting the data set via AIC:

$$(0)\,0.411,\quad (1)\,0.128,\quad (2)\,0.067,\quad (3)\,0.070,\quad (4)\,0.073,\quad (5)\,0.068, \tag{3.24}$$

These values are illustrated in Figure 3-6, inspection of which leads correctly to choosing $\kappa = 2$ as the model that minimizes the above criterion.

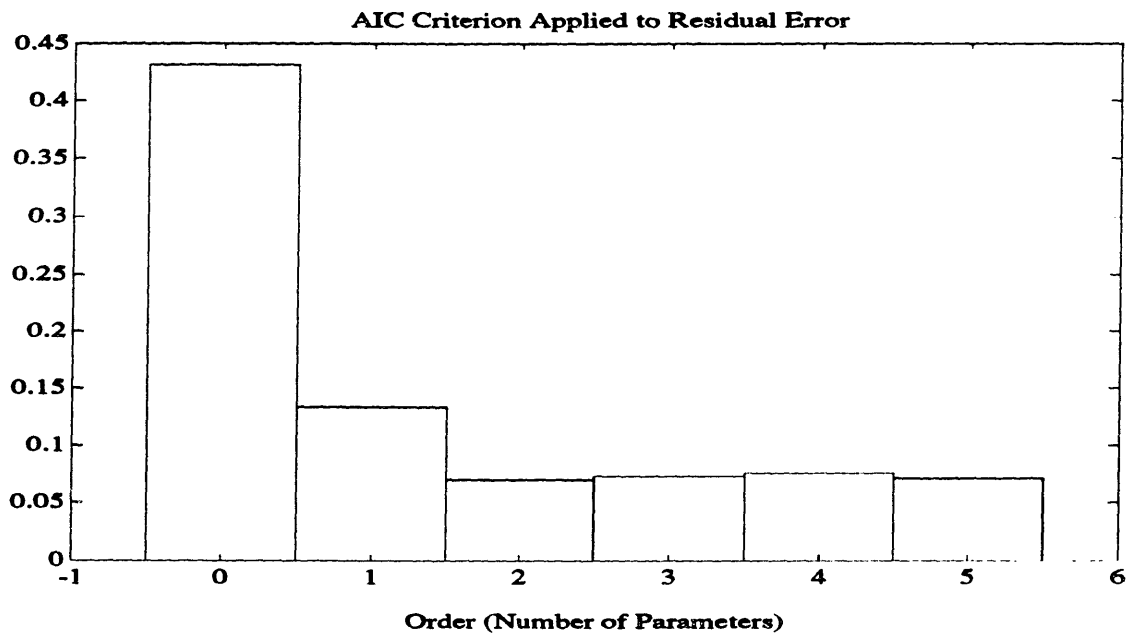


Figure 3-6: Illustration of parameter selection using AIC evaluation

Rissanen's MDL criterion has a very similar structure to AIC. Specifically, the following equation is also contained in [6]:

$$MDL(\kappa) \approx (1 + \log(N) * \kappa/N) * \frac{\|\mathbf{error}_\kappa\|_2^2}{N}. \tag{3.25}$$

The result of weighting the same data set according to MDL was:

$$(0)\,0.411,\quad (1)\,0.133,\quad (2)\,0.072,\quad (3)\,0.078,\quad (4)\,0.083,\quad (5)\,0.080, \tag{3.26}$$

the corresponding plot being shown in Figure 3-7. The MDL criterion also leads to correctly selecting the model $k = 2$ for this example.

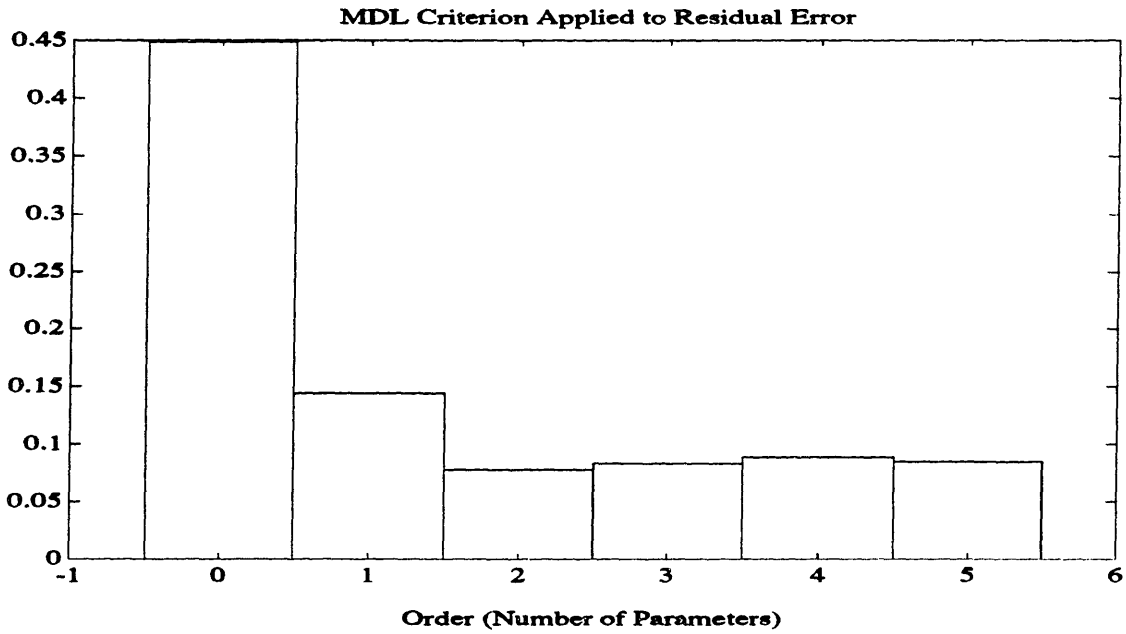


Figure 3-7: Illustration of parameter selection using MDL evaluation

3.7 Comparing the Evaluation Techniques

An important distinction to be made between hypothesis testing and information criterion evaluation lies in the basic philosophy difference between these techniques. Hypothesis testing is directly built upon the principle of making use of a specified noise model to establish quantitative limits on how much the noise will affect the residual error norm. Upon removing a set of parameters, the change in residual error norm is compared to these limits in order to decide whether those parameters are only serving to fit noise. In such a case, the parameters are considered 'extra'. Thus, hypothesis testing focuses on the principle that each parameter belongs in one of two groups, 'true' or 'extra', with assumed noise properties being used to make the distinction. In contrast, information criterions choose to focus on the task of gaining the maximum residual error reduction in the fewest number of parameters. To achieve this goal, these techniques incorporate some type of residual error weighting function that penalizes error reduction by the number of parameters required to achieve it. Parameters are not classified as being 'extra' or 'true', but rather the final model is chosen by simply comparing all of the considered models' performances in the context of the chosen criterion. However, the subtlety of the previous statement is that information criterions *do*, in fact, lead to the separation of parameters into 'true' and 'extra' sets when used to choose

the lowest order model out of a consistent class of model structures (i.e. when used to choose the lowest order ARX model, for instance). Information criterions are very useful when trying to compare models with different structures (i.e. comparing the performance of an ARX model vs. some nonlinear model for representing a given system), but in the context of this thesis, ARX models are the only considered parameterization structure. Therefore, for our purposes, comparison between AIC, MDL, and hypothesis testing really comes down to the question: "which method yields the most accurate threshold level at which parameters will be divided into 'true' or 'extra' groups?"

We previously mentioned that hypothesis testing was often thought of as being subjective since a confidence level is required to be chosen. In contrast, information criterions do not require such prior specification, and are thus often thought of as being objective. However, this is slightly misleading in the sense that *all* of the techniques presented are subjective — information criterions simply have their confidence level implicitly chosen upon specifying a criterion function (i.e. AIC or MDL). Hypothesis testing is based on theory that offers some very intuitive insights about least squares estimation (which are presented in chapter 4). This fact, along with its allowance for the user to directly fine tune, in a consistent and intuitive fashion, the threshold at which parameters are considered to be 'true' or 'extra', has led us to choose hypothesis testing as the primary method of evaluation in this thesis.

3.8 The Problem of Ordering

In presenting the hypothesis testing and information criterion approaches to parameter selection, we have thus far ignored a fundamental problem associated with both methods. Specifically, both techniques require a prespecified ordering procedure to be used when building models for parameter evaluation. To the author's knowledge, there is currently no objective manner to determine such a procedure. Consequently, an investigator wishing to use AIC, MDL, or hypothesis testing must arbitrarily select model parameterizations to be compared. In order to complete the identification in a reasonable amount of time, the number of considered model structures must be kept to a low number. Thus, in consideration of the fact that, under such circumstances, it would be very unlikely for the investigator to actually pick the minimal model as one of those considered, it will generally be the case that all of the techniques (AIC, MDL, etc.) fail to find the best model to represent the given

system. In other words, without an *intelligent* search procedure, all of the given parameter evaluation tools will, more than likely, lead to the selection of an over or underparameterized model.

Optimal system identification entails the selection of a model that contains no parameters whose actual value is equal or very close to zero. To isolate such a model, different parameterizations must be chosen and their performance compared with some criterion. Alternatively, we can view this search procedure as the removal of different parameter sets from some 'maximal' model that contains a union of all parameters considered, from which comparisons on performance can be made. Practically speaking, each of the previous two statements is equivalent when using some sort of parameter selection process (i.e. MDL, AIC, etc. are only applicable to the comparison of selected, distinct models), but an approach is suggested by the second statement that we will make use of. In particular, the focus of our ordering procedure will be the task of determination of which parameters are *most likely* to be 'extra' parameters *within* a given maximal model.

A natural way to evaluate the relative likelihood that a specific parameter within a maximal model is 'extra' is to simply look at the estimate of that parameter that resulted from running the maximal model through the estimation process. Since its actual value is defined to be zero, it stands to reason that the estimate of any 'extra' parameter should generally have a smaller absolute value than the estimates of 'true' parameters. Thus, to determine an ordering procedure, one could simply estimate the parameters within the given maximal model and then order the parameters according to the estimate values (our object is to cluster the 'true' parameters together in such a procedure). This method would be very effective in the case where noise has a small effect on the estimation procedure, but for the purposes of increasing robustness, we will improve upon it by considering the assumed noise characteristics in our calculations.

Let us now examine, in a quantitative fashion, the characteristics of how noise corrupts estimates in the least squares procedure. Recall the following two equations:

$$\mathbf{y} = \mathbf{\Phi\theta} + \mathbf{e}, \tag{3.27}$$

and

$$\mathbf{y} = \mathbf{\Phi}\hat{\boldsymbol{\theta}} + \mathbf{error}. \tag{3.28}$$

(Assuming the data matrix, Φ, in each of these equations corresponded to the chosen maximal model, parameters that were 'extra', by definition, would be defined as all $\theta[i]$ that were equivalent to zero.) The second equation represents the estimation results, with least squares being used to determine $\hat{\theta}$ as:

$$\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}. \tag{3.29}$$

If we substitute equation 3.27 into equation 3.29, we obtain:

$$\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T (\Phi\theta + \mathbf{e}) = \theta + (\Phi^T \Phi)^{-1} \Phi^T \mathbf{e}, \tag{3.30}$$

which directly specifies the manner in which noise influences the least squares estimates.

The following facts will be useful to keep in mind as we examine the relationship depicted in equation 3.30:

1. in general, if we define $z = ax + by$, where x and y are random variables, then

$$E(z) = aE(x) + bE(y),$$

where $E(\cdot)$ designates the *mean* of a random variable,

2. again, if we define $z = ax + by$, and additionally specify that x and y are *uncorrelated* random variables, then

$$\sigma_z^2 = a^2\sigma_x^2 + b^2\sigma_y^2,$$

where σ^2 denotes the *variance* of a random variable,

3. the elements within $\mathbf{e}$ are assumed to be uncorrelated random variables, i.e.:

$$E(\mathbf{e}[i]\,\mathbf{e}[j]) = E(\mathbf{e}[i])\,E(\mathbf{e}[j]),\ i \neq j,$$

(This assumption is based on designating $\mathbf{e}$ to be 'white'. As a sidenote, it is a property of gaussian processes (which is an assumed property of $\mathbf{e}$), that uncorrelation implies independence, so that we also have:

$$p_e(\mathbf{e}[i]\,\mathbf{e}[j]) = p_e(\mathbf{e}[i])\,p_e(\mathbf{e}[j]),\ i \neq j)$$

4. based on 3., the mean and variance of each element within $\mathbf{e}$ are identical and will be designated as m_e ($= 0$) and σ_e^2, respectively.

5. since each element within $\mathbf{e}$ is a gaussian random variable, the addition of any of these variables leads to a random variable that is also gaussian (this is simply a property of guassian random variables),

Now, for ease in notation, let us define:

$$\mathbf{C} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T, \tag{3.31}$$

so that equation 3.30 becomes:

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta} + \mathbf{Ce}. \tag{3.32}$$

If we additionally define the quantity $\mathbf{v} = \mathbf{Ce}$, we should note the fact that $\mathbf{v}$ is composed of the sum of random variables contained within $\mathbf{e}$, and as such is gaussian. Using this definition, equation 3.30 is once more rewritten as:

$$\hat{\boldsymbol{\theta}}[i] = \boldsymbol{\theta}[i] + \mathbf{v}[i], \tag{3.33}$$

where $[i]$ designates the i^{th} element of its associated vector.

In order to understand how $\mathbf{v}$ effects the estimate results, we need to determine its first and second order moments, i.e. its mean and variance. We'll do this in the form of an example, arbitrarily considering the following:

$$\begin{bmatrix} \mathbf{v}[1] \\ \mathbf{v}[2] \end{bmatrix} = \begin{bmatrix} \mathbf{C}[1,1] & \mathbf{C}[1,2] & \mathbf{C}[1,3] \\ \mathbf{C}[2,1] & \mathbf{C}[2,2] & \mathbf{C}[2,3] \end{bmatrix} \begin{bmatrix} \mathbf{e}[1] \\ \mathbf{e}[2] \\ \mathbf{e}[3] \end{bmatrix}.$$

By properties 1., 2., and 4. outlined above, we can immediately write:

$$E(\mathbf{v}[1]) = (\mathbf{C}[1,1] + \mathbf{C}[1,2] + \mathbf{C}[1,3])\, m_e = 0,$$
$$\sigma^2_{\mathbf{v}[1]} = (\mathbf{C}[1,1]^2 + \mathbf{C}[1,2]^2 + \mathbf{C}[1,3]^2)\, \sigma_e^2 = \textstyle\sum_j \mathbf{C}[1,j]^2 \sigma_e^2 = \|\mathbf{C}[1]\|_2^2 \sigma_e^2,$$

and

$$E(\mathbf{v}[2]) = (\mathbf{C}[2,1] + \mathbf{C}[2,2] + \mathbf{C}[2,3])\, m_e = 0,$$
$$\sigma^2_{\mathbf{v}[2]} = (\mathbf{C}[2,1]^2 + \mathbf{C}[2,2]^2 + \mathbf{C}[2,3]^2)\, \sigma_e^2 = \textstyle\sum_j \mathbf{C}[2,j]^2 \sigma_e^2 = \|\mathbf{C}[2]\|_2^2 \sigma_e^2,$$

where $\|\mathbf{C}[i]\|_2^2$ represents the square of the l_2 norm of the i^{th} row of $\mathbf{C}$. It is straightforward to generalize the above results as follows:

$$E(\mathbf{v}[i]) = 0, \qquad \sigma^2_{\mathbf{v}[i]} = \|\mathbf{C}[i]\|_2^2 \sigma_e^2, \tag{3.34}$$

so that we observe each parameter estimate, $\hat{\boldsymbol{\theta}}[i]$, is corrupted by a gaussian random variable, $\mathbf{v}[i]$, with zero mean and a variance that varies according to the l_2 norm of the corresponding row in $\mathbf{C}$. However, if it is our desire to compare different parameter estimates in order to decide which are most likely to be 'extra', it would be preferable to have the same degree of noise corruption in each estimate. The justification for this argument will hopefully become clear as we proceed.

In order to put all of the parameter estimates on equal footing with respect to noise corruption, let us modify equation 3.33. Recalling property two of the previously outlined facts, we note that scaling a random variable leads to the scaling of that variable's variance by a square factor. In other words, given $\sigma_z^2 = 1$, then $\sigma_{az}^2 = a^2$. Using this principle, we modify equation 3.33 as:

$$\frac{\hat{\boldsymbol{\theta}}[i]}{\|\mathbf{C}[i]\|_2} = \frac{\boldsymbol{\theta}[i]}{\|\mathbf{C}[i]\|_2} + \frac{\mathbf{v}[i]}{\|\mathbf{C}[i]\|_2}, \tag{3.35}$$

noting that

$$var(\frac{\mathbf{v}[i]}{\|\mathbf{C}[i]\|_2}) = \sigma_e^2.$$

Thus, equation 3.35 depicts a 'normalization' procedure whereby each of the parameter estimates are scaled such that the corrupting noise variance is the same for each.

We are now in a position to compare the relative likelihood of estimates being 'extra'. One should note that a rigorous approach to this problem would prove unpractical (i.e. one that considered the dependencies between parameter estimates). However, the simplified approach about to be outlined has achieved excellent results. With this qualifying statement in mind, let us begin by renotating equation 3.35 as:

$$\tilde{\hat{\boldsymbol{\theta}}}[i] = \tilde{\boldsymbol{\theta}}[i] + \tilde{\mathbf{v}}[i], \tag{3.36}$$

where the ˜ symbol has been used to indicate that the corresponding vector entry has been 'normalized', and consider the example:

$$\begin{bmatrix} -.9 \\ .5 \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{\theta}}[1] \\ \tilde{\boldsymbol{\theta}}[2] \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{v}}[1] \\ \tilde{\mathbf{v}}[2] \end{bmatrix}.$$

If we momentarily assumed that both of the above parameters had an actual value of zero, the above equation would turn into:

$$\begin{bmatrix} -.9 \\ .5 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{v}}[1] \\ \tilde{\mathbf{v}}[2] \end{bmatrix},$$

which would state that both of the normalized estimates are realizations of the corrupting noise variables, $\tilde{\mathbf{v}}[i]$. This point is illustrated graphically in Figure 3-8, which depicts the gaussian probability density function, $f_{\mathbf{v}[i]}(\mathbf{v}_0[i])$, associated with each noise variable, along with their supposed realizations, $\hat{\tilde{\boldsymbol{\theta}}}[i]$. Glancing at the given picture, we now ask

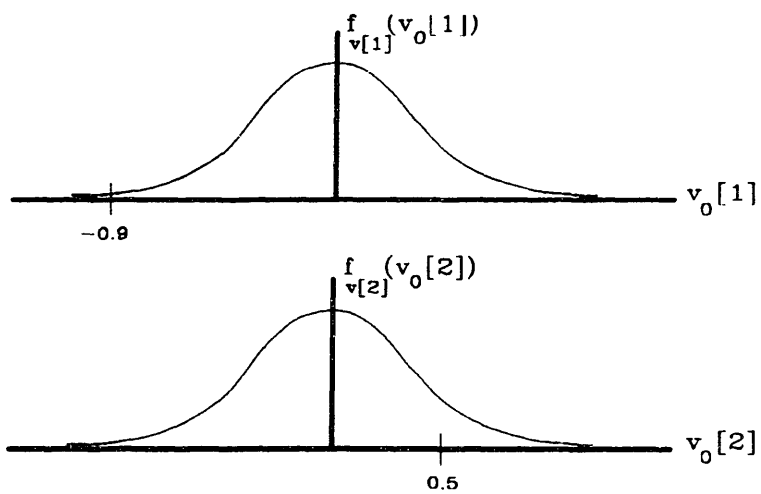


Figure 3-8: Probability density functions associated with two parameters estimates under the hypothesis that their true values are zero

the question: "Which parameter was less likely to have had an actual value of zero?"

Answering this question is quite easy if we consider nothing more than the above plots (i.e. we ignore any interdependence between the parameter estimates) — since each of the $\tilde{\mathbf{v}}[i]$ have identical, gaussian probability densities (via the fact that both have zero mean and variance σ_ϵ^2), the parameter with normalized estimate of highest magnitude is the least likely to be 'extra'. Thus, in the above diagram, since $|-0.9| > |0.5|$, $\boldsymbol{\theta}[1]$ is less likely than $\boldsymbol{\theta}[2]$ to be 'extra'.

Generalization of the above results leads directly to a straightforward ordering procedure. Our desire is to clump the 'true' parameters together such that they will be removed from the maximal model after the 'extra' parameters. Thus, assuming that we have performed the normalization procedure depicted in equation 3.35, we strive toward this goal by ordering the parameters according to the magnitude of their normalized estimates, such that the parameters most likely to be 'extra' are removed first. For instance, if it were true for the case of three parameters that:

$$|\hat{\tilde{\boldsymbol{\theta}}}[2]| > |\hat{\tilde{\boldsymbol{\theta}}}[1]| > |\hat{\tilde{\boldsymbol{\theta}}}[3]|,$$

then the ordering procedure would be:

$$\hat{\boldsymbol{\theta}}[2] \leftarrow \hat{\boldsymbol{\theta}}[1] \leftarrow \hat{\boldsymbol{\theta}}[3] \leftarrow .$$

(where, again, the arrows indicate the direction of *removing* parameters from the maximal model) For reasons soon to be explained, we will refer to this process of scaling the estimate values for the purposes of ordering as 'S/N normalization'.

3.8.1 Example

Returning again to the ongoing example, the following results were obtained for the maximal model, $\mathbf{y} = \boldsymbol{\Phi}_5\hat{\boldsymbol{\theta}}_5 + \mathbf{error}_5$, using different simulation data:

$$\begin{bmatrix} \hat{\boldsymbol{\theta}}_5[1] \\ \hat{\boldsymbol{\theta}}_5[2] \\ \hat{\boldsymbol{\theta}}_5[3] \\ \hat{\boldsymbol{\theta}}_5[4] \\ \hat{\boldsymbol{\theta}}_5[5] \end{bmatrix} = \begin{bmatrix} -1.203 \\ 0.477 \\ 1.623 \\ 0.059 \\ -1.425 \end{bmatrix}, \quad \begin{bmatrix} \|\mathbf{C}_5[1]\|_2 \\ \|\mathbf{C}_5[2]\|_2 \\ \|\mathbf{C}_5[3]\|_2 \\ \|\mathbf{C}_5[4]\|_2 \\ \|\mathbf{C}_5[5]\|_2 \end{bmatrix} = \begin{bmatrix} 1.161 \\ 1.170 \\ 4.368 \\ 1.498 \\ 3.665 \end{bmatrix} \implies \begin{bmatrix} \hat{\tilde{\boldsymbol{\theta}}}_5[1] \\ \hat{\tilde{\boldsymbol{\theta}}}_5[2] \\ \hat{\tilde{\boldsymbol{\theta}}}_5[3] \\ \hat{\tilde{\boldsymbol{\theta}}}_5[4] \\ \hat{\tilde{\boldsymbol{\theta}}}_5[5] \end{bmatrix} = \begin{bmatrix} -1.036 \\ 0.408 \\ 0.372 \\ 0.039 \\ -0.389 \end{bmatrix}.$$

By inspection of the above $\hat{\tilde{\boldsymbol{\theta}}}_5$ vector, the ordering procedure would be:

$$\hat{\boldsymbol{\theta}}[1] \leftarrow \hat{\boldsymbol{\theta}}[2] \leftarrow \hat{\boldsymbol{\theta}}[5] \leftarrow \hat{\boldsymbol{\theta}}[3] \leftarrow \hat{\boldsymbol{\theta}}[4] \leftarrow,$$

which leads to estimators of the 'extra' parameters, $\hat{\boldsymbol{\theta}}[3]$, $\hat{\boldsymbol{\theta}}[4]$, and $\hat{\boldsymbol{\theta}}[5]$ being removed from the maximal model before the 'true' estimators, $\hat{\boldsymbol{\theta}}[1]$ and $\hat{\boldsymbol{\theta}}[2]$.

3.8.2 Further Observations

Although the previous ordering procedure was presented in the context of comparing the relative likelihood of maximal model parameters being 'extra', there is an alternate way of interpreting this method. In general, 'signal to noise ratio' is defined as signal energy divided by noise energy. Applying this ratio to the estimation representation found in equation 3.33, we write:

$$S/N(i) = \frac{\|\boldsymbol{\theta}[i]\|_2^2}{\|\mathbf{v}[i]\|_2^2} = \left(\frac{\|\boldsymbol{\theta}[i]\|_2}{\|\mathbf{v}[i]\|_2}\right)^2 = \left(\frac{\|\boldsymbol{\theta}[i]\|_2}{\|\mathbf{C}[i]\|_2 \sigma_e}\right)^2, \tag{3.37}$$

where the rightmost part of the above equation is based on a statistical point of view. Since we don't have access to the actual parameter values in the context of estimation, we rely on the following approximation in our analysis:

$$S/N(i) = \left(\frac{\|\hat{\boldsymbol{\theta}}[i] - \mathbf{v}[i]\|_2}{\|\mathbf{C}[i]\|_2 \sigma_e}\right)^2 \approx \left(\frac{\|\hat{\boldsymbol{\theta}}[i]\|_2}{\|\mathbf{C}[i]\|_2 \sigma_e}\right)^2 = \frac{1}{\sigma_e^2}\left(\frac{\hat{\boldsymbol{\theta}}[i]}{\|\mathbf{C}[i]\|_2}\right)^2. \tag{3.38}$$

Note that the right side of the above equation is directly proportional to the squared value of each normalized estimate, $\hat{\tilde{\boldsymbol{\theta}}}[i]$. Thus, we also make the claim that the above ordering procedure effectively adds parameters with high signal to noise ratios (i.e. 'true' parameters) before those with low signal to noise ratios ('extra' parameters). One should note that the approximation introduced in equation 3.38 is only valid in the case of high signal to noise ratios (i.e. the noise energy must be much smaller than the squared parameter value, $\|\mathbf{v}[i]\|_2^2 << \boldsymbol{\theta}[i]^2$). However, our interest lies mainly in determining which estimators have the *highest relative* signal to noise ratio, a task for which equation 3.38 is well suited.

With the establishment of an ordering procedure, it becomes the choice of the investigator as to whether hypothesis testing or information criterion selection is used to determine the division between 'true' and 'extra' parameters. The reader might also be curious as to

whether hypothesis testing could be directly applied to the normalized estimators, $\hat{\tilde{\boldsymbol{\theta}}}$ (i.e. evaluate directly whether or not a parameter is 'true' by determining if its associated $|\hat{\tilde{\boldsymbol{\theta}}}[i]|$ is large enough in magnitude compared to the estimated noise variance to justify such a claim). Although such a method could easily be implemented, it is not desirable in the context of ARX model identification, which is the intended application for this technique. This fact will become clear in the following section.

3.9 The ARX Model Revisited

To extend the above results to an ARX model selection procedure, we must take a closer look at characteristic properties of the parameters associated with these models. Recall the difference equation format of the ARX model given in chapter 2:

$$\mathbf{y}[k] = \sum_{i=1}^{p} -\mathbf{a}^*[i]\mathbf{y}[k-i] + \sum_{i=s_1}^{f_1} \mathbf{b}_1[i]\mathbf{x}_1[k-i] + \cdots + \sum_{i=s_M}^{f_M} \mathbf{b}_M[i]\mathbf{x}_M[k-i] + \mathbf{e}[k], \quad (3.39)$$

whose corresponding Z transform is:

$$(1 + a^*(z))y(z) = b_1(z)x_1(z) + \cdots + b_M(z)x_M(z) + e(z). \quad (3.40)$$

Equation 3.39 leads directly to the least squares format:

$$\boldsymbol{\Phi} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_M & \mathbf{Y} \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_M \\ -\mathbf{a}^* \end{bmatrix} \implies \mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{e}. \quad (3.41)$$

Note that the ARX model contains two *types* of parameters: $\mathbf{a}^*$ and $\mathbf{b}_i$. The distinction between the two is best seen through a simple example.

Consider the system represented by the Z transform relation:

$$y(z) = \frac{\mathbf{b}_1[0] + \mathbf{b}_1[1]z^{-1}}{1 + \mathbf{a}^*[1]z^{-1}} x_1(z),$$

where we have, for the purposes of clarity, left out the noise term $\mathbf{e}$. Also, let us consider the input to be a 'unit impulse', such that $x_1(z) = 1$. The resulting output response, referred

to as the 'impulse response' of the system, is then expressed as:

$$y(z) = \mathbf{b}_1[0]\left(\frac{1}{1+\mathbf{a}^*[1]z^{-1}}\right) + z^{-1}\mathbf{b}_1[1]\left(\frac{1}{1+\mathbf{a}^*[1]z^{-1}}\right) \tag{3.42}$$

The inverse transform of the above equation is:

$$\mathbf{y}[k] = \mathbf{b}_1[0](\mathbf{a}^*[1])^k\mathbf{u}[k] + \mathbf{b}_1[1](\mathbf{a}^*[1])^{k-1}\mathbf{u}[k-1], \tag{3.43}$$

where, by definition, $\mathbf{u}[k-i]$, the unit step function, takes on values such that:

$$\mathbf{u}[k-i] = 1,\ k \geq i; \qquad \mathbf{u}[k-i] = 0,\ k < i. \tag{3.44}$$

Figure 3-9 is intended to illustrate equation 3.43. Explaining this diagram, the parameters

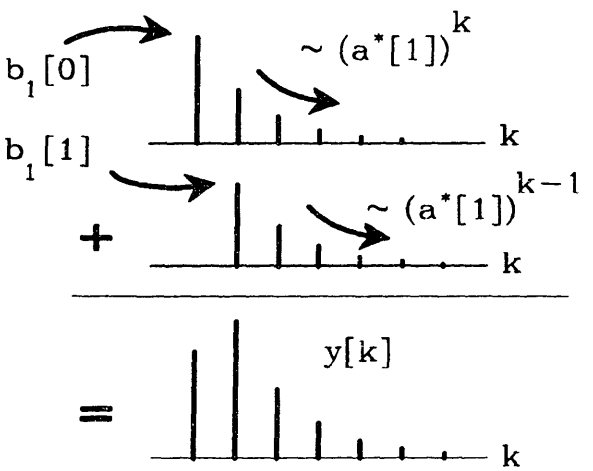


Figure 3-9: Illustration of the influence of 'a' and 'b' parameters in forming an impulse response

$\mathbf{b}_1[0]$ and $\mathbf{b}_1[1]$ have the effect of shifting and scaling a response 'shape' determined by the $\mathbf{a}^*[1]$ parameter. Note that the $\mathbf{b}_i$ parameters can effectively take on any value, while the $\mathbf{a}^*[1]$ parameter must have an absolute value less than one in order to have a decaying $\mathbf{y}[k]$ response.

Because of the differences in their effect on system dynamics, comparing $\mathbf{b}_i$ and $\mathbf{a}^*$ parameters in order to determine a minimal model is not a very well defined task — effectively, it's like trying to compare apples and oranges! For instance, in the context of estimation, suppose we were given:

$$\hat{\mathbf{b}}_1[0] = 10.0, \quad \hat{\mathbf{b}}_1[1] = 5.0, \quad \hat{\mathbf{a}}^*[1] = 0.9,$$

from which we would like to determine the *relative* likelihood of each parameter being 'extra'. In considering the magnitude of each estimate (temporarily ignoring the developed S/N normalization procedure), one might try to argue that $\hat{\mathbf{b}}_1[0]$ and $\hat{\mathbf{b}}_1[1]$ were far more likely than $\hat{\mathbf{a}}^*[1]$ to be 'true'. However, $\hat{\mathbf{a}}^*[1]$ is probably far from being 'extra' in this case — a drastic change in the shape of the 'impulse response' would occur if it were removed from the model! Therefore, an ordering procedure that considers both $\mathbf{a}^*$ and $\mathbf{b}$ parameters cannot rely directly on estimate values for comparisons of importance between these sets.

3.10 ARX Model Selection Using The APR Algorithm

The determination of a minimal estimation model entails the incorporation of a search procedure and an evaluation tool (such as hypothesis testing). Previously, we developed an intelligent search method that made use of the estimation results obtained for a specified maximal model. Unfortunately, this procedure could not be directly applied to ARX model selection since comparison between $\mathbf{a}^*$ and $\mathbf{b}_i$ estimation values is not insightful. However, it is fairly straightforward to modify the developed search procedure in order to accomplish our task — we simply consider the $\mathbf{a}^*$ and $\mathbf{b}_i$ groups separately.

Since the $\mathbf{a}^*$ parameters have the characteristic of being more likely to be 'extra' with increasing index (this is based on the assumption that no 'gaps' occur in the $\mathbf{a}$ polynomial), a natural way to begin the search for a minimal model is to remove these parameters from the specified maximal model in just that order. In other words, beginning with the last index, remove one $\mathbf{a}^*$ parameter at a time and observe the resulting residual error values. Upon the removal of all of the $\mathbf{a}^*$ parameters, hypothesis testing or information criterion could then be used on the resulting residual error sequence to determine which parameters were 'true'.

Upon removing all of the 'extra' $\mathbf{a}^*$ parameters, we could then proceed to determine

which $\mathbf{b}_i$ parameters are 'extra'. In order to do this intelligently, we could incorporate the search procedure outlined previously. To clarify, $\mathbf{b}_i$ estimates associated with the maximal model after having its 'extra' $\mathbf{a}^*$ parameters removed (which will be referred to as the *reduced* estimation model) would be S/N normalized so that the relative 'extra' likelihood of each parameter could be evaluated. The $\mathbf{b}_i$ parameters would then be removed from the reduced estimation model one at a time in the order obtained via their S/N normalized values, with hypothesis testing or information criterion selection being used in conclusion to evaluate which parameters were, in fact, 'true'.

The overall ARX model selection process, referred to as the APR algorithm, is illustrated as follows:

Remove $\mathbf{a}^*$ parameters one at a time,
beginning with those at later indices
$\Downarrow$
Given the resulting residual error norm sequence,
use hypothesis testing or an information criterion
to determine *reduced* model
$\Downarrow$
Based on reduced model estimates, order the $\mathbf{b}_i$
parameters according to their S/N normalization values
$\Downarrow$
Use hypothesis testing or an information criterion
to determine *minimal* model

3.10.1 Example

To provide an example of the above approach with hypothesis testing as our evaluation tool, consider now the ARX model specified below:

$$\begin{bmatrix} \mathbf{a}^*[1] \\ \mathbf{a}^*[2] \\ \mathbf{a}^*[3] \end{bmatrix} = \begin{bmatrix} -1.77 \\ 1.45 \\ -0.55 \end{bmatrix} \quad \begin{bmatrix} \mathbf{b}_1[0] \\ \mathbf{b}_1[1] \\ \mathbf{b}_1[2] \\ \mathbf{b}_1[5] \end{bmatrix} = \begin{bmatrix} 10.0 \\ -16.4 \\ 8.29 \\ -3.20 \end{bmatrix} \quad \begin{bmatrix} \mathbf{b}_2[2] \\ \mathbf{b}_2[4] \end{bmatrix} = \begin{bmatrix} 8.0 \\ -6.54 \end{bmatrix}.$$

Note that this system could also be represented with the block diagram structure shown in Figure 3-10. In order to perform an estimation procedure with respect to this system, we generated three uncorrelated, white gaussian noise sequences corresponding to $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{e}$ on a computer. We then ran these sequences through a computer simulation of the above system, producing the $\mathbf{y}$ sequence in the process. It should be noted that the noise

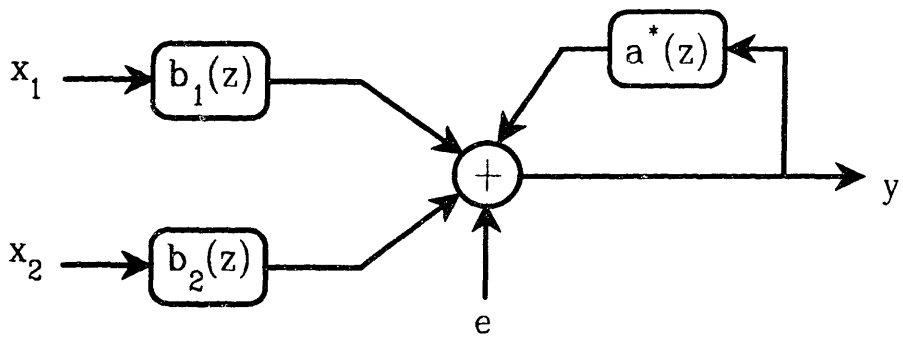


Figure 3-10: A two input ARX model structure

energy, $\|\mathbf{e}\|_2^2$, was chosen to be fairly large in magnitude with respect to the energy of the inputs in order that estimation corruption be plainly seen.

In order to proceed through the described method, a maximal model set was required to be specified. We arbitrarily chose the following as this set:

$$\hat{\mathbf{a}}_{10}^{*} = \begin{bmatrix} \hat{\mathbf{a}}[1] \\ \hat{\mathbf{a}}^{*}[2] \\ \hat{\mathbf{a}}^{*}[3] \\ \hat{\mathbf{a}}^{*}[4] \\ \hat{\mathbf{a}}^{*}[5] \\ \hat{\mathbf{a}}^{*}[6] \\ \hat{\mathbf{a}}^{*}[7] \\ \hat{\mathbf{a}}^{*}[8] \\ \hat{\mathbf{a}}^{*}[9] \\ \hat{\mathbf{a}}^{*}[10] \end{bmatrix} \quad \hat{\mathbf{b}}_{1_{10}} = \begin{bmatrix} \hat{\mathbf{b}}_1[0] \\ \hat{\mathbf{b}}_1[1] \\ \hat{\mathbf{b}}_1[2] \\ \hat{\mathbf{b}}_1[3] \\ \hat{\mathbf{b}}_1[4] \\ \hat{\mathbf{b}}_1[5] \\ \hat{\mathbf{b}}_1[6] \\ \hat{\mathbf{b}}_1[7] \\ \hat{\mathbf{b}}_1[8] \\ \hat{\mathbf{b}}_1[9] \end{bmatrix} \quad \hat{\mathbf{b}}_{2_{10}} = \begin{bmatrix} \hat{\mathbf{b}}_2[0] \\ \hat{\mathbf{b}}_2[1] \\ \hat{\mathbf{b}}_2[2] \\ \hat{\mathbf{b}}_2[3] \\ \hat{\mathbf{b}}_2[4] \\ \hat{\mathbf{b}}_2[5] \\ \hat{\mathbf{b}}_2[6] \\ \hat{\mathbf{b}}_2[7] \\ \hat{\mathbf{b}}_2[8] \\ \hat{\mathbf{b}}_2[9] \end{bmatrix},$$

which were combined to form:

$$\hat{\boldsymbol{\theta}}_{30} = \begin{bmatrix} \hat{\mathbf{b}}_{1_{10}} \\ \hat{\mathbf{b}}_{2_{10}} \\ -\hat{\mathbf{a}}^*_{10} \end{bmatrix}$$

In order to use hypothesis testing as the evaluation tool, a confidence level was also needed, which was chosen to be 0.95.

Using the input and output data sequences from our simulation, we created the data matrix, $\boldsymbol{\Phi}_{30}$ (in completely analogous fashion as outlined at the end of chapter 2), thereby forming the maximal estimation model:

$$\mathbf{y} = \boldsymbol{\Phi}_{30}\hat{\boldsymbol{\theta}}_{30} + \mathbf{error}_{30}. \tag{3.45}$$

Note that the actual system model is a subset of this overly specified model, namely:

$$\mathbf{y} = \boldsymbol{\Phi}_9\boldsymbol{\theta}_9 + \mathbf{e}, \qquad \boldsymbol{\theta}_9 = \begin{bmatrix} \mathbf{b}_1[0] \\ \mathbf{b}_1[1] \\ \mathbf{b}_1[2] \\ \mathbf{b}_1[5] \\ \mathbf{b}_2[2] \\ \mathbf{b}_2[4] \\ -\mathbf{a}^*[1] \\ -\mathbf{a}^*[2] \\ -\mathbf{a}^*[3] \end{bmatrix}, \tag{3.46}$$

To gain a feel for the amount of corruption occurring in the above estimation procedure under the given noise conditions, we performed least squares in order to estimate each parameter value within the maximal set:

$$\hat{\mathbf{a}}^*_{10} = \begin{bmatrix} -1.70 \\ 1.39 \\ -0.57 \\ 0.14 \\ -0.09 \\ -0.08 \\ 0.25 \\ -0.35 \\ 0.31 \\ -0.15 \end{bmatrix} \quad \hat{\mathbf{b}}_{1_{10}} = \begin{bmatrix} 8.84 \\ -14.16 \\ 8.19 \\ -0.76 \\ -0.07 \\ -4.09 \\ 0.19 \\ 1.12 \\ -2.49 \\ 1.53 \end{bmatrix} \quad \hat{\mathbf{b}}_{2_{10}} = \begin{bmatrix} 0.04 \\ 1.10 \\ 8.60 \\ 1.19 \\ -6.22 \\ -0.86 \\ -0.11 \\ 0.21 \\ -0.65 \\ 0.51 \end{bmatrix}.$$

One should note that corruption is significant in these estimates — twenty one of the above parameters have actual values of zero, yet many of these 'extra' parameters have fairly large estimate values ! It is our desire to remove these unnecessary parameters in order to obtain the minimal model consisting of nine 'true' parameters.

The first step toward discarding extra parameters from the maximal model entailed removal of the $\mathbf{a}^*$ parameters. Proceeding with this operation, we pulled off one $\mathbf{a}^*$ parameter at a time (starting with those at the highest index), and observed the resulting error norms. Expressing this operation as an ordering procedure, we write:

$$\{\mathbf{b}_i\} \leftarrow \mathbf{a}^*[1] \leftarrow \mathbf{a}^*[2] \leftarrow \mathbf{a}^*[3] \leftarrow \cdots \mathbf{a}^*[10] \leftarrow,$$

where the arrows indicate the direction of *removing* parameters, and the notation '$\{\mathbf{b}_i\}$' denotes the entire set of $\mathbf{b}_i$ parameters. The residual error norms resulting from this procedure were as follows:

$$\begin{aligned}
\|\mathbf{error}_{30}\|_2^2 &= 12803.84 \\
\|\mathbf{error}_{29}\|_2^2 &= 13524.94 \\
\|\mathbf{error}_{28}\|_2^2 &= 13538.42 \\
\|\mathbf{error}_{27}\|_2^2 &= 13547.61 \\
\|\mathbf{error}_{26}\|_2^2 &= 13551.04 \\
\|\mathbf{error}_{25}\|_2^2 &= 13558.24 \\
\|\mathbf{error}_{24}\|_2^2 &= 13616.85 \\
\|\mathbf{error}_{23}\|_2^2 &= 13620.90 \\
\|\mathbf{error}_{22}\|_2^2 &= 18030.18 \\
\|\mathbf{error}_{21}\|_2^2 &= 38581.12 \\
\|\mathbf{error}_{20}\|_2^2 &= 98346.91
\end{aligned}$$

The above data set was then used for two purposes — obtaining an estimate of the noise variance, $\hat{\sigma}_e^2$, and determining the 'extra' $\mathbf{a}^*$ parameters with the use of hypothesis testing. Recall that the residual error norm corresponding to the maximal model is used to estimate the noise variance, which, in this case, led to:

$$\hat{\sigma}_e^2 = \frac{\|\mathbf{error}_{max}\|_2^2}{N - max} = \frac{\|\mathbf{error}_{30}\|_2^2}{200 - 30} = 75.32.$$

(The actual noise variance in our simulation was $\sigma_e^2 = 78.43$, implying the above estimation procedure had acceptable results.) Given this estimated noise variance, we then proceeded to normalize the residual error norms determined above in order to compare them to their

associated Chi Square limits at 0.95 confidence:

$$\frac{\|error_\kappa\|_2^2 - \|error_{max}\|_2^2}{\hat{\sigma}_e^2} = \begin{cases} (29)\ 9.57 \\ (28)\ 9.75 \\ (27)\ 9.88 \\ (26)\ 9.92 \\ (25)\ 10.02 \\ (24)\ 10.79 \\ (23)\ 10.85 \\ (22)\ 69.50 \\ (21)\ 342.25 \\ (20)\ 1135.78 \end{cases} \overset{?}{>} \mu_{(max-\kappa)} = \begin{cases} \mu_1 = 3.84 \\ \mu_2 = 5.99 \\ \mu_3 = 7.82 \\ \mu_4 = 9.49 \\ \mu_5 = 11.07 \\ \mu_6 = 12.59 \\ \mu_7 = 14.07 \\ \mu_8 = 15.51 \\ \mu_9 = 16.92 \\ \mu_{10} = 18.31 \end{cases}$$

(where the value of κ in each row is specified within parenthesis.) The above sequences are plotted in Figure 3-11 to allow a visualization of the indicated comparison process. Inspection of this data leads to the correct conclusion that seven **a*** parameters could be

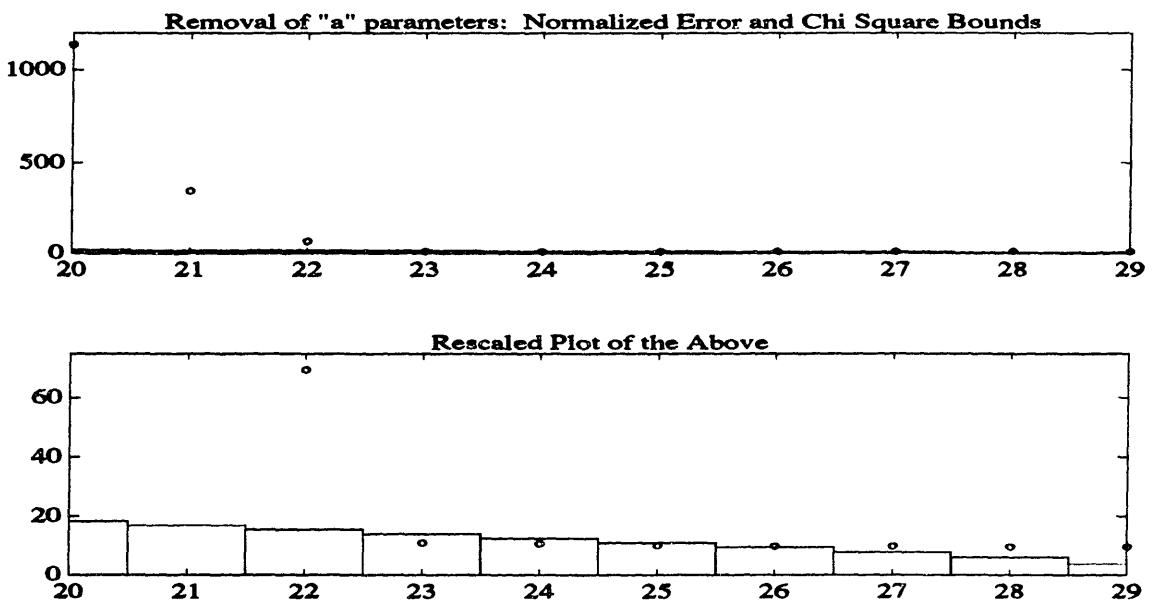


Figure 3-11: Comparison of changes in residual error due to removal of 'a' parameters with their Chi Square bounds

removed from the maximal model (i.e. for (23): $10.85 < 14.07$)! Note that removal of the first **a*** parameter resulted in a change of residual error norm that was greater than its Chi Square limit ($9.57 > \mu_1 = 3.84$) — this was an outlier. Since we chose the smallest *set* of parameters that lead to an error norm within the given limits, however, this outlying value

had negligible effect. Thus, we obtained the *reduced* estimation model:

$$\mathbf{y} = \mathbf{\Phi}_{23}\hat{\boldsymbol{\theta}}_{23} + \mathbf{error}_{23}$$

We then used the parameter estimates from the reduced model to order the $\mathbf{b}_i$ parameters. This operation was carried out by S/N normalizing their parameter estimates accordingly, the results of which were as follows:

$$\frac{|\hat{\boldsymbol{\theta}}_{23}[i]|}{\|\mathbf{C}[i]\|_2} = \begin{cases} |\hat{\mathbf{b}}_1[0]| = 113.41 \\ |\hat{\mathbf{b}}_1[1]| = 139.67 \\ |\hat{\mathbf{b}}_1[2]| = 65.57 \\ |\hat{\mathbf{b}}_1[3]| = 0.32 \\ |\hat{\mathbf{b}}_1[4]| = 13.07 \\ |\hat{\mathbf{b}}_1[5]| = 45.37 \\ |\hat{\mathbf{b}}_1[6]| = 11.93 \\ |\hat{\mathbf{b}}_1[7]| = 10.57 \\ |\hat{\mathbf{b}}_1[8]| = 1.80 \\ |\hat{\mathbf{b}}_1[9]| = 6.16 \\ |\hat{\mathbf{b}}_2[0]| = 1.32 \\ |\hat{\mathbf{b}}_2[1]| = 12.84 \\ |\hat{\mathbf{b}}_2[2]| = 105.37 \\ |\hat{\mathbf{b}}_2[3]| = 10.80 \\ |\hat{\mathbf{b}}_2[4]| = 67.61 \\ |\hat{\mathbf{b}}_2[5]| = 3.57 \\ |\hat{\mathbf{b}}_2[6]| = 0.23 \\ |\hat{\mathbf{b}}_2[7]| = 0.58 \\ |\hat{\mathbf{b}}_2[8]| = 0.10 \\ |\hat{\mathbf{b}}_2[9]| = 0.82 \end{cases}$$

(One should note in general that, although $\mathbf{a}^*$ parameters will generally be included in the reduced estimation model (in this case, there were three), we need only consider the $\mathbf{b}_i$ parameter estimates in the S/N normalization procedure.) Recalling that normalized estimates with highest absolute value are considered least likely to be 'extra', we used the above data to obtain the following ordering procedure:

$$\{\mathbf{a}^*\} \leftarrow \mathbf{b}_1[1] \leftarrow \mathbf{b}_1[0] \leftarrow \mathbf{b}_2[2] \leftarrow \mathbf{b}_2[4] \leftarrow \mathbf{b}_1[2] \leftarrow \mathbf{b}_1[5] \leftarrow \mathbf{b}_1[4] \leftarrow \cdots \mathbf{b}_2[8] \leftarrow .$$

Using this ordering specification, we proceeded to remove parameters one at a time from the reduced estimation model. Upon obtaining all of the residual error norms, we normalized their values for the purposes of performing hypothesis testing. The results of this operation were as follows:

$$\frac{\|error_\kappa\|_2^2 - \|error_{max}\|_2^2}{\hat{\sigma}_e^2} = \begin{cases} (23)\ 10.848 \\ (22)\ 10.848 \\ (21)\ 10.849 \\ (20)\ 10.851 \\ (19)\ 10.856 \\ (18)\ 10.865 \\ (17)\ 10.889 \\ (16)\ 10.9 \\ (15)\ 11.2 \\ (14)\ 11.8 \\ (13)\ 13.2 \\ (12)\ 14.6 \\ (11)\ 17.6 \\ (10)\ 19.5 \\ (9)\ 22.4 \\ (8)\ 51.0 \\ (7)\ 158 \\ (6)\ 226 \\ (5)\ 423 \\ (4)\ 613 \\ (3)\ 845 \end{cases} \overset{?}{>} \mu_{(max-\kappa)} = \begin{cases} \mu_7 = 14.1 \\ \mu_8 = 15.5 \\ \mu_9 = 16.9 \\ \mu_{10} = 18.3 \\ \mu_{11} = 19.7 \\ \mu_{12} = 21.0 \\ \mu_{13} = 22.4 \\ \mu_{14} = 23.7 \\ \mu_{15} = 25.0 \\ \mu_{16} = 26.3 \\ \mu_{17} = 27.6 \\ \mu_{18} = 28.9 \\ \mu_{19} = 30.1 \\ \mu_{20} = 31.4 \\ \mu_{21} = 32.7 \\ \mu_{22} = 33.9 \\ \mu_{23} = 35.2 \\ \mu_{24} = 36.4 \\ \mu_{25} = 37.7 \\ \mu_{26} = 38.9 \\ \mu_{27} = 40.1 \end{cases}$$

A plot of these sequences is shown in Figure 3-12. Inspection of the data leads immediately to the conclusion that the smallest parameter set to stay within Chi Square limits contains nine parameters — these precisely correspond to the 'true' parameters of our system model! Obtaining the resulting parameter estimates for the chosen minimal estimation model (with zeros being substituted for all excluded parameters) led to the following:

$$\hat{\mathbf{a}}_{10}^* = \begin{bmatrix} -1.73 \\ 1.39 \\ -0.48 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \hat{\mathbf{b}}_{1_{10}} = \begin{bmatrix} 8.90 \\ -14.7 \\ 8.37 \\ 0 \\ 0 \\ -3.49 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \hat{\mathbf{b}}_{2_{10}} = \begin{bmatrix} 0 \\ 0 \\ 8.50 \\ 0 \\ -6.70 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

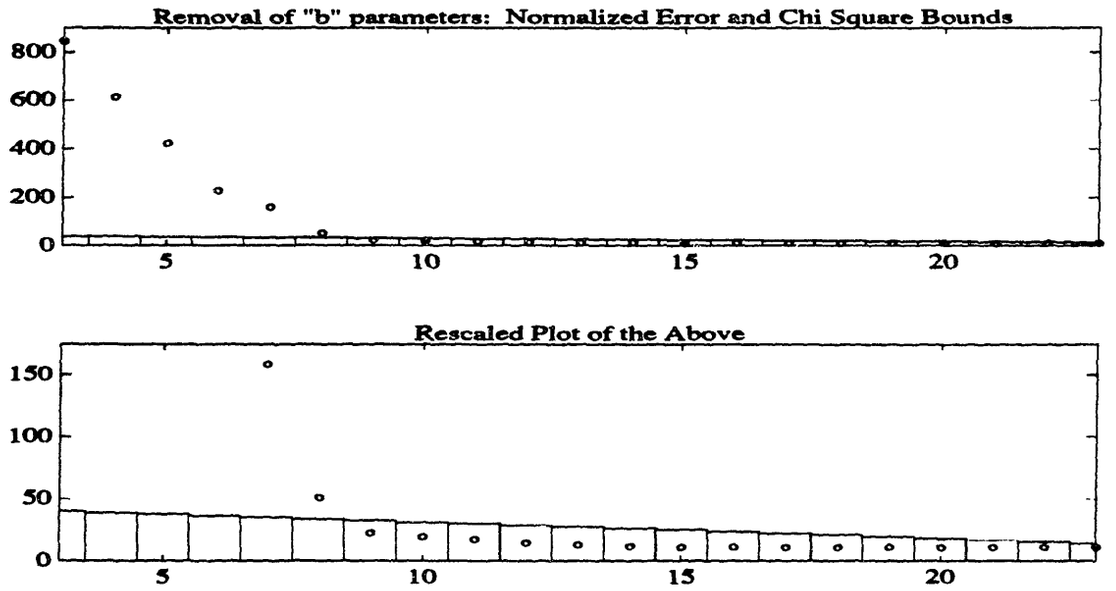


Figure 3-12: Comparison of changes in residual error due to removal of 'b' parameters with their Chi Square bounds

Thus, the algorithm precisely identified the 'true' parameters within the specified maximal model!

Chapter 4

White Noise, Least Squares, and Hypothesis Testing Revisited

4.1 Introduction

This chapter is geared toward providing a more solid understanding of some of the principles used in the previous two chapters. In particular, our desire is to provide the reader with a better understanding of least squares and the role it plays in hypothesis testing. It is the author's opinion that the most straightforward and intuitive way to achieve this goal is to develop these concepts through the use of linear algebra.

4.2 Introductory Examples

As we explore the least squares procedure, it will be useful to have at our disposal a few examples pertaining to its associated model structure:

$$\mathbf{y} = \mathbf{\Phi}\boldsymbol{\theta} + \mathbf{e} \;=\; \mathbf{y}_u + \mathbf{e}. \tag{4.1}$$

Therefore, let us consider the following ARX models placed in difference equation format:

$$a)\;\; \mathbf{y}[k] = \mathbf{b}_1[0]\mathbf{x}_1[k] + \mathbf{e}[k], \tag{4.2}$$

and

$$b)\;\; \mathbf{y}[k] = \mathbf{b}_1[0]\mathbf{x}_1[k] + \mathbf{b}_1[1]\mathbf{x}_1[k-1] + \mathbf{e}[k], \tag{4.3}$$

Also, for the purposes of simplicity, let us specify, for both systems, the input sequence as:

$$\mathbf{x}_1[k] = \delta[k], \text{ where we define: } \delta[k] = 1,\ k = 0;\ \delta[k] = 0, \text{ otherwise.} \tag{4.4}$$

Together with the unknown noise term, $\mathbf{e}$ (which will be assumed different for each of the above systems), the specified input sequence acts to generate each output sequence, $\mathbf{y}$.

In order to convert the difference equation formats of our example into the least squares structure, we define the following:

$$a)\ \ \Phi = [\,\mathbf{x}_1[k]\,], \boldsymbol{\theta} = [\,\mathbf{b}_1[0]\,], \tag{4.5}$$

and

$$b)\ \ \Phi = \begin{bmatrix} \mathbf{x}_1[k] & \mathbf{x}_1[k-1] \end{bmatrix}, \boldsymbol{\theta} = \begin{bmatrix} \mathbf{b}_1[0] \\ \mathbf{b}_1[1] \end{bmatrix}. \tag{4.6}$$

For the purposes of gaining insight from a geometric point of view, we will restrict these examples to the consideration of k taking on only 3 values: $0 \leq k \leq 2$. Using the above definitions, the least square models corresponding to this range on k are:

$$a)\ \ \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1[0] \\ \mathbf{x}_1[1] \\ \mathbf{x}_1[2] \end{bmatrix} [\,\mathbf{b}_1[0]\,] + \begin{bmatrix} \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix}, \tag{4.7}$$

and

$$b)\ \ \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1[0] & \mathbf{x}_1[-1] \\ \mathbf{x}_1[1] & \mathbf{x}_1[0] \\ \mathbf{x}_1[2] & \mathbf{x}_1[1] \end{bmatrix} \begin{bmatrix} \mathbf{b}_1[0] \\ \mathbf{b}_1[1] \end{bmatrix} + \begin{bmatrix} \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix} \tag{4.8}$$

Filling in the values for $\mathbf{x}_1$, these equations become:

$$a)\ \ \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} [\,\mathbf{b}_1[0]\,] + \begin{bmatrix} \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix}, \tag{4.9}$$

and

$$b)\ \ \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{b}_1[0] \\ \mathbf{b}_1[1] \end{bmatrix} + \begin{bmatrix} \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix} \tag{4.10}$$

For convenience, we will denote the following vectors:

$$\mathbf{v}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{v}_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{v}_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{4.11}$$

so that all of the considered three dimensional vectors can be decomposed into components that lie in the $\mathbf{v}_x$, $\mathbf{v}_y$, and $\mathbf{v}_z$ direction. For instance, the '$\mathbf{v}_x$' component of $\mathbf{y}$ is $\mathbf{y}[0]$, the '$\mathbf{v}_y$' component is $\mathbf{y}[1]$, etc. Using this notation, we illustrate system (a) in Figure 4-1, and system (b) in Figure 4-2.

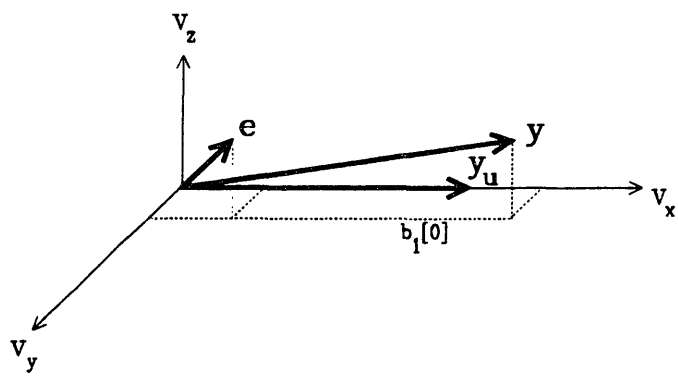


Figure 4-1: Vector Illustration of example system (a)

4.3 Vector Interpretation of White, Gaussian Noise

The assumption that $\mathbf{e}$ is white implies the following:

- $E(\mathbf{e}[i]) = 0$,
- $E(\mathbf{e}^2[i]) = \sigma_e^2$,
- $E(\mathbf{e}[i]\,\mathbf{e}[j]) = E(\mathbf{e}[i])\,E(\mathbf{e}[j]) = 0$, $i \neq j$, implying that the elements of $\mathbf{e}$ are uncorrelated with one another.

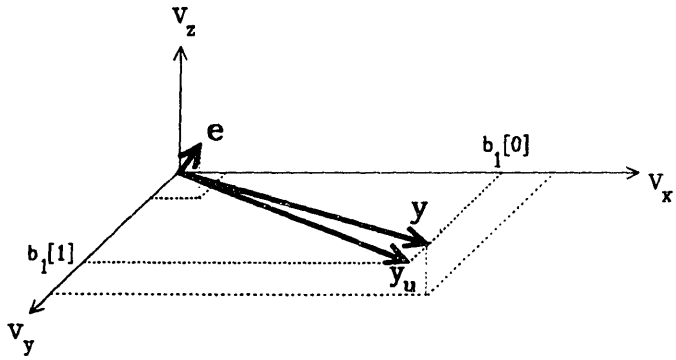


Figure 4-2: Vector illustration of example system (b)

Each of these facts is captured in the 'autocorrelation matrix' of $\mathbf{e}$, $E(\mathbf{e}\mathbf{e}^T)$, which is calculated below for the given examples:

$$E(\mathbf{e}\mathbf{e}^T) = E\left(\begin{bmatrix} \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix} \begin{bmatrix} \mathbf{e}[0] & \mathbf{e}[1] & \mathbf{e}[2] \end{bmatrix}\right) = \begin{bmatrix} E(\mathbf{e}[0]^2) & E(\mathbf{e}[0]\,\mathbf{e}[1]) & E(\mathbf{e}[0]\,\mathbf{e}[2]) \\ E(\mathbf{e}[1]\,\mathbf{e}[0]) & E(\mathbf{e}[1]^2) & E(\mathbf{e}[1]\,\mathbf{e}[2]) \\ E(\mathbf{e}[2]\,\mathbf{e}[0]) & E(\mathbf{e}[2]\,\mathbf{e}[1]) & E(\mathbf{e}[2]^2) \end{bmatrix}$$

$$= \begin{bmatrix} \sigma_e^2 & 0 & 0 \\ 0 & \sigma_e^2 & 0 \\ 0 & 0 & \sigma_e^2 \end{bmatrix} = \sigma_e^2 \mathbf{I}. \tag{4.12}$$

Although the above illustration pertained to our three dimensional examples, it has some very general implications. Namely, a noise vector is white *if and only if* its autocorrelation matrix is diagonal — i.e. $E(\mathbf{e}\mathbf{e}^T) = \sigma_e^2 I$.

On top of specifying that $\mathbf{e}$ is white, we have also associated with it the gaussian probability density function. The implication of this fact is that:

$$\mathbf{e}[i] \rightsquigarrow N(0, \sigma_e^2), \quad 0 \leq i \leq N-1 \tag{4.13}$$

where $\rightsquigarrow$ denotes: "is distributed according to," and $N(m, \sigma^2)$ denotes the gaussian probability density function with mean m and variance σ^2. It should be noted that a property of *uncorrelated* random variables that are gaussian is that they are also *independent*, so that $\mathbf{e}[i]$ is independent of $\mathbf{e}[j]$ for $i \neq j$.

Using the above facts, we now derive the joint probability density function for all of the elements within $\mathbf{e}$, which we denote as $f(\mathbf{e}[0], \mathbf{e}[1], \cdots, \mathbf{e}[N])$. Since these elements are independent and identically distributed, we write:

$$f(\mathbf{e}[0], \mathbf{e}[1], \cdots, \mathbf{e}[N]) = \prod_{i=0}^{N-1} f(\mathbf{e}[i]), \tag{4.14}$$

where:

$$\mathbf{e}[i] \rightsquigarrow N(0, \sigma_e^2) \implies f(\mathbf{e}[i]) = \frac{\exp\left(\frac{-\mathbf{e}[i]^2}{2\sigma_e^2}\right)}{\sqrt{2\pi}\sigma_e}, \quad -\infty < \mathbf{e}[i] < \infty. \tag{4.15}$$

Thus, we express the joint density of the elements within $\mathbf{e}$ as:

$$f(\mathbf{e}[0], \mathbf{e}[1], \cdots, \mathbf{e}[N]) = \frac{\exp\left(\frac{-(\mathbf{e}[0]^2+\mathbf{e}[1]^2+\cdots+\mathbf{e}[N]^2)}{2\sigma_e^2}\right)}{(\sqrt{2\pi}\sigma_e)^N}, \quad -\infty < \mathbf{e}[i] < \infty. \tag{4.16}$$

To gain some intuition about the above function, let us consider the case for which $\mathbf{e}$ has dimension 2 and $\sigma_e^2 = 1$. The joint density is then expressed as:

$$f(\mathbf{e}[0], \mathbf{e}[1]) = \frac{\exp\left(\frac{-(\mathbf{e}[0]^2+\mathbf{e}[1]^2)}{2}\right)}{2\pi}, \quad -\infty < \mathbf{e}[0], \mathbf{e}[1] < \infty. \tag{4.17}$$

We can illustrate this equation with a mesh and a contour plot, as shown in Figure 4-3 and 4-4, respectively. Now, if we consider the l_2 norm of vector $\mathbf{e}$:

$$\|\mathbf{e}\|_2 = \sqrt{\sum_{i=0}^{1} \mathbf{e}[i]^2}, \tag{4.18}$$

along with its angle:

$$\angle\mathbf{e} = \arctan\left(\frac{\mathbf{e}[1]}{\mathbf{e}[0]}\right), \tag{4.19}$$

then we can observe from the given plots that there is a very low probability associated with the norm of $\mathbf{e}$ taking on large values (for instance, exceeding $\sqrt{3^2+3^2} = \sqrt{18}$), but the angle of $\mathbf{e}$ is uniformly random (there are no 'preferential' directions for $\mathbf{e}$ to take).

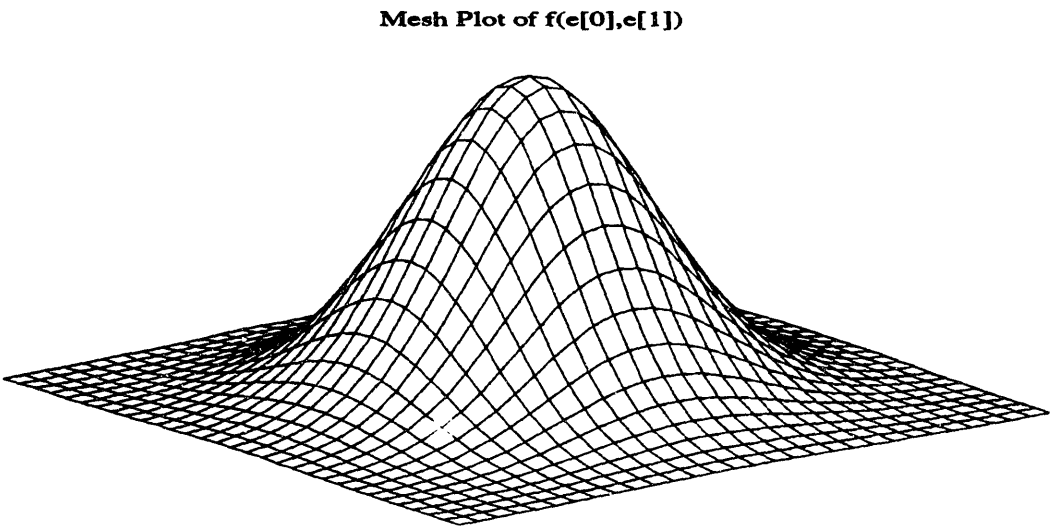


Figure 4-3: Probability density function associated with a two dimensional white noise vector — Mesh plot

Extending the implications drawn by the above example to noise vectors of arbitrary dimension, we note that **e** is a vector whose direction is uniformly random and whose norm follows some probability function that could be derived from the joint density of its individual elements. The point to make out of these facts is that we are helpless in trying to predict the direction this vector will take (every direction is equally likely), but its norm is a different matter — certain norm values are less likely than others. To exploit this fact, we will explicitly determine the probability density function associated with this quantity.

In general, if we define the random variables $z_1, z_2, \ldots, z_\kappa$ such that:

$$z_i \rightsquigarrow N(0,1), \tag{4.20}$$

and also define a random variable, x_0, such that

$$x_0 = \sum_{i=1}^{\kappa} z_i^2, \tag{4.21}$$

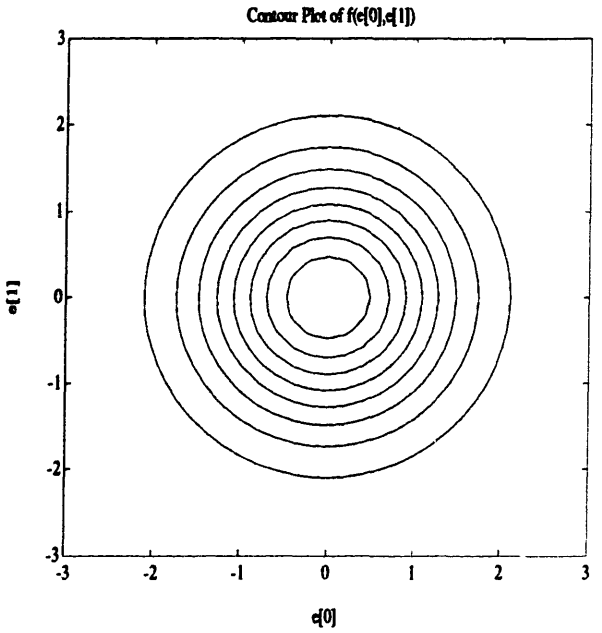


Figure 4-4: Probability density function associated with a two dimensional white noise vector — Contour plot

then x_0 is distributed according to the following density function:

$$\chi^2_\kappa(x_0) = \frac{2^{-\kappa/2}}{(\kappa/2-1)!} x_0^{n/2-1} \exp(-x_0/2), \; x_0 > 0; \; = 0, \text{ otherwise.} \tag{4.22}$$

The above formulation, known as the Chi Square density function with degrees of freedom equal to κ, can be used to quantify the statistical nature of the l_2 norm of $\mathbf{e}$. Specifically, observe the following definition of the energy (squared norm) of $\mathbf{e}$:

$$\|\mathbf{e}\|_2^2 = \sum_{i=0}^{N-1} \mathbf{e}[i]^2, \text{ where: } \mathbf{e}[i] \rightsquigarrow N(0, \sigma_e^2). \tag{4.23}$$

In order to conform the energy of $\mathbf{e}$ to the Chi Square density, we need simply to scale each of its elements such that their variance is 1 as opposed to σ_e^2. To accomplish this, recall the property that scaling a random variable has the following effect on its variance:

$$\text{variance}(az) = a^2 \sigma_z^2. \tag{4.24}$$

(also, note that $E(az) = aE(z)$). Application of this property leads us to state:

$$\text{variance}(\tfrac{\mathbf{e}[i]}{\sigma_e}) = \frac{1}{\sigma_e^2}\sigma_e^2 = 1, \tag{4.25}$$

(with the mean remaining unaltered, i.e. $E(\frac{\mathbf{e}[i]}{\sigma_e}) = 0$) which, in turn, allows us to write that the *normalized energy* of $\mathbf{e}$ is distributed according to the Chi Square density function of N degrees of freedom.

$$\frac{\|\mathbf{e}\|_2^2}{\sigma_e^2} = \sum_{i=0}^{N-1} \left(\frac{\mathbf{e}[i]}{\sigma_e}\right)^2 \rightsquigarrow \chi_N^2. \tag{4.26}$$

The implication of this result is significant — we have effectively quantified, in a statistical sense, the norm of the noise vector $\mathbf{e}$.

To gain some insight into the value of equation 4.26, let us return to the given examples. Suppose we were interested in determining the maximum norm $\mathbf{e}$ would take on at a confidence level of $\eta = 0.95$. In other words, considering a large number of different white, gaussian $\mathbf{e}$ vectors having identical variances, σ_e^2, and dimension ($N = 3$), we would like to determine some value such that 95% of the vectors had a norm less than that value. For our examples, this value can be determined from the Chi Square density function with 3 degrees of freedom (since $\mathbf{e}$ has $N = 3$ elements within it). To illustrate this fact, consider the Chi Square plot shown in Figure 4-5. This diagram indicates a selected maximum value,

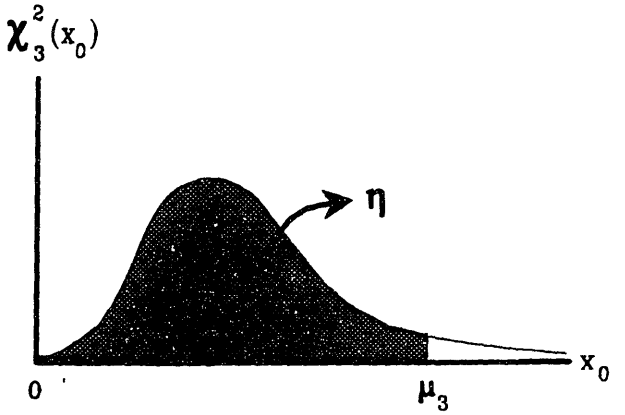


Figure 4-5: Plot of Chi Square probability density function with three degrees of freedom

μ_3, and the probability, η, associated with $x_0 = \|\mathbf{e}\|_2^2/\sigma_e^2$ (the normalized energy of noise

vector **e**) taking on values less than this selected maximum. In adhering to the given confidence level, our desire is to determine the value of μ_3 such that $\text{prob}(x_0 \leq \mu_3) = \eta = 0.95$, thus specifying the area of the shaded region in the above diagram to be equal to 0.95. Equivalently, we state this goal as:

$$\int_0^{\mu_3} \chi_3^2(x_0)dx_0 = \eta = 0.95 \tag{4.27}$$

Examining any statistical table of Chi Square values, we obtain $\mu_3 = 7.82$. Therefore, we can state, at a 0.95 level of confidence, that:

$$\frac{\|\mathbf{e}\|_2^2}{\sigma_e^2} = x_0 \leq \mu_3 = 7.82 \implies \|\mathbf{e}\|_2 \leq \sqrt{7.82}\sigma_e \tag{4.28}$$

As a final observation to be made regarding **e**, we note the fact that its variance, σ_e^2, is not generally known and must therefore be estimated. A straightforward procedure to accomplish this task could be implemented if some, or all, of the components of **e** were given. In particular, suppose κ elements within **e** were known ($1 \leq \kappa \leq N-1$), such that we could compute the following quantity:

$$\|\mathbf{e}_\kappa\|_2^2 = \sum_{i=0}^{\kappa-1} \mathbf{e}_\kappa[i]^2, \tag{4.29}$$

where $\mathbf{e}_\kappa$ denotes the vector that has for its elements the known κ entries in **e**. Examining $\mathbf{e}_\kappa$ from a statistical standpoint, we note that:

$$E(\|\mathbf{e}_\kappa\|_2^2) = \sum_{i=0}^{\kappa-1} E(\mathbf{e}[i]^2) = \kappa\sigma_e^2. \tag{4.30}$$

Combining the previous two equations leads us to an estimate the variance of **e**:

$$\|\mathbf{e}_\kappa\|_2^2 \approx E(\|\mathbf{e}_\kappa\|_2^2) \implies \hat{\sigma}_e^2 = \frac{\|\mathbf{e}_\kappa\|_2^2}{\kappa}. \tag{4.31}$$

4.4 Derivation of Least Squares

We now approach the estimation problem represented by the following equations:

$$\mathbf{y} = \mathbf{\Phi}\boldsymbol{\theta} + \mathbf{e}, \tag{4.32}$$

$$\mathbf{y} = \mathbf{\Phi}\hat{\boldsymbol{\theta}} + \mathbf{error}, \tag{4.33}$$

in which the first equation describes a given system model, and the second equation depicts its associated estimate. To obtain estimates of the parameters, $\hat{\boldsymbol{\theta}}$, our approach had been to use the least squares procedure. Essentially, this method chooses $\hat{\boldsymbol{\theta}}$ such that the l_2 norm of the error sequence, $\|\mathbf{error}\|_2$, is minimized. In order to gain a better understanding of how least squares accomplishes this task, we now present its derivation.

First of all, let us note that since $\mathbf{error} = \mathbf{y} - \mathbf{\Phi}\hat{\boldsymbol{\theta}}$, we can express the energy of this sequence as:

$$\|\mathbf{error}\|_2^2 \equiv \mathbf{error}^T\mathbf{error} = (\mathbf{y} - \mathbf{\Phi}\hat{\boldsymbol{\theta}})^T(\mathbf{y} - \mathbf{\Phi}\hat{\boldsymbol{\theta}}). \tag{4.34}$$

Since minimization of the norm of the **error** vector is equivalent to minimizing its energy, we can express our goal as the determination of $\hat{\boldsymbol{\theta}}$ such that the above quantity is minimized, i.e.:

$$\text{determine } \hat{\boldsymbol{\theta}} \text{ such that: } \min_{\hat{\boldsymbol{\theta}}} \left((\mathbf{y} - \mathbf{\Phi}\hat{\boldsymbol{\theta}})^T(\mathbf{y} - \mathbf{\Phi}\hat{\boldsymbol{\theta}}) \right). \tag{4.35}$$

To find such an estimate, we take the classical calculus approach of setting the partial derivatives of the above quantity, with respect to each element in $\hat{\boldsymbol{\theta}}$, equal to zero. To make this process efficient, we will develop a notation geared toward such an operation.

Consider the general inner product of two vectors:

$$w = \mathbf{u}^T\mathbf{v} = \sum_i \mathbf{u}[i]\mathbf{v}[i], \tag{4.36}$$

where $\mathbf{u}$ and $\mathbf{v}$ are N dimensional vectors, and w is a scalar. Suppose we wanted to determine the net change in w, Δw, for selected changes in the elements of $\mathbf{v}$, $\Delta\mathbf{v}[i]$. As a step toward this goal, let us define:

$$\Delta\mathbf{v} = \begin{bmatrix} \Delta\mathbf{v}[0] \\ \Delta\mathbf{v}[1] \\ \vdots \\ \Delta\mathbf{v}[N] \end{bmatrix}, \tag{4.37}$$

and:

$$\frac{\Delta w}{\Delta \mathbf{v}} = \begin{bmatrix} \frac{\Delta w}{\Delta \mathbf{v}[0]} & \frac{\Delta w}{\Delta \mathbf{v}[1]} & \cdots & \frac{\Delta w}{\Delta \mathbf{v}[N]} \end{bmatrix}. \tag{4.38}$$

Note that, as $\Delta\mathbf{v} \rightarrow 0$, the above quantity becomes:

$$\lim_{\Delta\mathbf{v}\rightarrow 0} \frac{\Delta w}{\Delta \mathbf{v}} = \frac{\delta w}{\delta \mathbf{v}} = \left[\begin{array}{cccc} \frac{\delta w}{\delta \mathbf{v}[0]} & \frac{\delta w}{\delta \mathbf{v}[1]} & \cdots & \frac{\delta w}{\delta \mathbf{v}[N]} \end{array} \right]. \tag{4.39}$$

Using these definitions, we modify equation 4.36 as:

$$w + \Delta w = \mathbf{u}^T(\mathbf{v} + \Delta\mathbf{v}), \tag{4.40}$$

which is immediately rearranged to become:

$$\Delta w \;=\; \mathbf{u}^T\mathbf{v} - w + \mathbf{u}^T\Delta\mathbf{v} \;=\; \mathbf{u}^T\Delta\mathbf{v} \;=\; \sum_i \mathbf{u}[i]\Delta\mathbf{v}[i]. \tag{4.41}$$

By inspection, we can then write:

$$\frac{\Delta w}{\Delta \mathbf{v}} = \mathbf{u}^T \;\;\Longrightarrow\;\; \frac{\delta w}{\delta \mathbf{v}} = \lim_{\Delta\mathbf{v}\rightarrow 0} \frac{\Delta w}{\Delta \mathbf{v}} = \mathbf{u}^T. \tag{4.42}$$

Consider now the product:

$$w = \mathbf{v}^T\mathbf{A}\mathbf{v}, \tag{4.43}$$

where w is again a scalar, $\mathbf{v}$ an N dimensional vector, and $\mathbf{A}$ an N row by N column matrix. To examine the variation in w as elements within $\mathbf{v}$ are varied, we write:

$$w + \Delta w \;=\; (\mathbf{v} + \Delta\mathbf{v})^T\mathbf{A}(\mathbf{v} + \Delta\mathbf{v}) \;=\; \mathbf{v}^T\mathbf{A}\mathbf{v} + \Delta\mathbf{v}^T\mathbf{A}\mathbf{v} + \mathbf{v}^T\mathbf{A}\Delta\mathbf{v} + \Delta\mathbf{v}^T\mathbf{A}\Delta\mathbf{v}. \tag{4.44}$$

Before rearranging the above relationship, let us note that:

$$\Delta\mathbf{v}^T\mathbf{A}\mathbf{v} = \mathbf{v}^T\mathbf{A}^T\Delta\mathbf{v}, \tag{4.45}$$

by the property that the transpose of any scalar is just that scalar (i.e. in general, for the vectors $\mathbf{u}$ and $\mathbf{v}$, and scalar r, $r = \mathbf{u}^T\mathbf{v} \;=\; r^T = \mathbf{v}^T\mathbf{u}$, so that filling in $\mathbf{u} = \mathbf{A}^T\Delta\mathbf{v}$ yields the result shown above). Continuing with our derivation, we write:

$$\Delta w \;=\; \mathbf{v}^T\mathbf{A}\mathbf{v} - w + \mathbf{v}^T(\mathbf{A}^T + \mathbf{A})\Delta\mathbf{v} + \Delta\mathbf{v}^T\mathbf{A}\Delta\mathbf{v} \;=\; \mathbf{v}^T(\mathbf{A}^T + \mathbf{A})\Delta\mathbf{v} + \Delta\mathbf{v}^T\mathbf{A}\Delta\mathbf{v}. \tag{4.46}$$

Examining the above formulation, we note that as $\Delta\mathbf{v}$ is taken to be arbitrarily small, the

rightmost term goes to zero much faster than the others, allowing us to ignore this 'higher order' term if $\Delta\mathbf{v}$ is small enough. This fact becomes clear as we write:

$$\frac{\delta w}{\delta \mathbf{v}} = \lim_{\Delta\mathbf{v}\to 0}\frac{\Delta w}{\Delta \mathbf{v}} = \lim_{\Delta\mathbf{v}\to 0}\mathbf{v}^T(\mathbf{A}^T+\mathbf{A}) + \Delta\mathbf{v}^T\mathbf{A} = \mathbf{v}^T(\mathbf{A}^T+\mathbf{A}). \tag{4.47}$$

We now apply the above results to the least squares problem:

$$\min_{\hat{\boldsymbol{\theta}}}(\mathbf{error}^T\mathbf{error}) = \min_{\hat{\boldsymbol{\theta}}}\left((\mathbf{y}-\boldsymbol{\Phi}\hat{\boldsymbol{\theta}})^T(\mathbf{y}-\boldsymbol{\Phi}\hat{\boldsymbol{\theta}})\right), \tag{4.48}$$

expanding the quantity to be minimized, we obtain:

$$\mathbf{error}^T\mathbf{error} = \mathbf{y}^T\mathbf{y} - \hat{\boldsymbol{\theta}}^T\boldsymbol{\Phi}^T\mathbf{y} - \mathbf{y}^T\boldsymbol{\Phi}\hat{\boldsymbol{\theta}} + \hat{\boldsymbol{\theta}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\hat{\boldsymbol{\theta}} = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\boldsymbol{\Phi}\hat{\boldsymbol{\theta}} + \hat{\boldsymbol{\theta}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\hat{\boldsymbol{\theta}} \tag{4.49}$$

Defining $w = \mathbf{error}^T\mathbf{error}$, we write:

$$w = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\boldsymbol{\Phi}\hat{\boldsymbol{\theta}} + \hat{\boldsymbol{\theta}}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\hat{\boldsymbol{\theta}} \tag{4.50}$$

for which, by the results presented in equations 4.42 and 4.47, we obtain:

$$\frac{\delta w}{\delta \hat{\boldsymbol{\theta}}} = -2\mathbf{y}^T\boldsymbol{\Phi} + \hat{\boldsymbol{\theta}}^T(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\Phi}^T\boldsymbol{\Phi}). \tag{4.51}$$

Therefore, to determine the minimum of w with respect to $\hat{\boldsymbol{\theta}}$, we simply set the above partial derivatives to 0:

$$-2\mathbf{y}^T\boldsymbol{\Phi} + \hat{\boldsymbol{\theta}}^T(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\Phi}^T\boldsymbol{\Phi}) = 0 \implies \hat{\boldsymbol{\theta}}^T(2\boldsymbol{\Phi}^T\boldsymbol{\Phi}) = 2\mathbf{y}^T\boldsymbol{\Phi}. \tag{4.52}$$

Assuming that $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ is invertible, we can then write:

$$\hat{\boldsymbol{\theta}}^T = \mathbf{y}^T\boldsymbol{\Phi}(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}, \tag{4.53}$$

such that we simply transpose the above quantity to obtain the least squares solution:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{y}. \tag{4.54}$$

4.5 Applications of the QR Decomposition

4.5.1 Definition

To gain further understanding into least squares, we now introduce the QR decomposition. Simply put, any rectangular matrix that has at least as many rows as columns can be decomposed into two matrices:

$$\mathbf{\Phi} = \mathbf{QR}, \tag{4.55}$$

where $\mathbf{Q}$ and $\mathbf{R}$ have the following properties: $\mathbf{Q}$ is *orthonormal* and $\mathbf{R}$ is *upper triangular*, i.e.

$$\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = I, \qquad \mathbf{R} = \begin{bmatrix} * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & 0 & * & \cdots & * \\ 0 & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.56}$$

where the '$*$' entries within $\mathbf{R}$ have values that can be nonzero. For convenience in notation, we will partition $\mathbf{R}$ as follows:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{||} \\ \cdots \\ \mathbf{0} \end{bmatrix}, \tag{4.57}$$

where $\mathbf{R}_{||}$ is an *invertible*, square matrix *if and only if* $\mathbf{\Phi}$ has full column rank (i.e. all of the columns in $\mathbf{\Phi}$ are independent). Conforming $\mathbf{Q}$ to the above partitioning, we also write:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{||} & \vdots & \mathbf{Q}_{\perp} \end{bmatrix}, \tag{4.58}$$

for which, applying the fact that $\mathbf{Q}$ is orthonormal, has the useful properties that:

$$\mathbf{Q}\mathbf{Q}^T = \mathbf{I} \implies \mathbf{Q}_{||}\mathbf{Q}_{||}^T + \mathbf{Q}_{\perp}\mathbf{Q}_{\perp}^T = \mathbf{I}, \tag{4.59}$$

and:

$$\mathbf{Q}^T\mathbf{Q} = \mathbf{I} \implies \begin{bmatrix} \mathbf{Q}_{||}^T\mathbf{Q}_{||} & \mathbf{Q}_{||}^T\mathbf{Q}_{\perp} \\ \mathbf{Q}_{\perp}^T\mathbf{Q}_{||} & \mathbf{Q}_{\perp}^T\mathbf{Q}_{\perp} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{4.60}$$

One should note the fact that the latter property implies:

$$\mathbf{Q}_{||}^T\mathbf{Q}_{\perp} = \mathbf{0}, \quad \mathbf{Q}_{\perp}^T\mathbf{Q}_{||} = \mathbf{0}, \tag{4.61}$$

and

$$\mathbf{Q}_{||}^T\mathbf{Q}_{||} = \mathbf{I}, \quad \mathbf{Q}_{\perp}^T\mathbf{Q}_{\perp} = \mathbf{I}, \tag{4.62}$$

Also, using the defined partitions, we note that equation 4.55 can be rewritten as:

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{Q}_{||} & \vdots & \mathbf{Q}_{\perp} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{||} \\ \cdots \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_{||}\mathbf{R}_{||}. \tag{4.63}$$

4.5.2 The Q Matrix as a Decomposition Operator

The property indicated by equation 4.59, namely:

$$\mathbf{Q}_{||}\mathbf{Q}_{||}^T + \mathbf{Q}_{\perp}\mathbf{Q}_{\perp}^T = \mathbf{I}, \tag{4.64}$$

can be viewed in the context of 'decomposing' a vector. This will be shown in a few steps, for which we begin with the trivial fact that, for any vector $\mathbf{v}$, $\mathbf{v} = \mathbf{I}\mathbf{v}$. Making use of equation 4.64, we now write:

$$\mathbf{v} = \mathbf{I}\mathbf{v} = (\mathbf{Q}_{||}\mathbf{Q}_{||}^T + \mathbf{Q}_{\perp}\mathbf{Q}_{\perp}^T)\mathbf{v}. \tag{4.65}$$

Glancing at the above statement, we can form two *orthogonal* components, namely:

$$\mathbf{v}_{||} = \mathbf{Q}_{||}\mathbf{Q}_{||}^T\mathbf{v}, \quad \mathbf{v}_{\perp} = \mathbf{Q}_{\perp}\mathbf{Q}_{\perp}^T\mathbf{v}, \tag{4.66}$$

where $\mathbf{v}_{||}$ lies in the *range* of $\mathbf{Q}_{||}$, and $\mathbf{v}_{\perp}$ in the range of $\mathbf{Q}_{\perp}$. To explain, let us first note that orthogonality implies:

$$\mathbf{v}_{||}^T\mathbf{v}_{\perp} = 0. \tag{4.67}$$

We can prove the above statement by simply filling in the component definitions specified in equation 4.66:

$$(\mathbf{Q}_{||}\mathbf{Q}_{||}^T\mathbf{v})^T\mathbf{Q}_\perp\mathbf{Q}_\perp^T\mathbf{v} = \mathbf{v}^T\mathbf{Q}_{||}(\mathbf{Q}_{||}^T\mathbf{Q}_\perp)\mathbf{Q}_\perp^T\mathbf{v} = \mathbf{0}, \tag{4.68}$$

where we have made use of equation 4.61. As for the *range* of a matrix, this simply refers to the space spanned by linear combinations of its columns. For instance, if we had a matrix with three dimensional columns, as follows:

$$\mathbf{Q}_{||} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{4.69}$$

then the range of $\mathbf{Q}_{||}$ would be the $\mathbf{v}_y - \mathbf{v}_z$ plane. For higher dimension matrices, the geometric interpretation is not as easy to visualize, but the ideas carry through just the same.

Example

To illustrate the decomposition of a vector, let us consider an example. In particular, consider the 3 dimensional **y** vector of our introductory examples:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix}. \tag{4.70}$$

For simplicity, let us choose an example $\mathbf{Q}$ as follows:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{||} & \vdots & \mathbf{Q}_\perp \end{bmatrix} = \left[\begin{array}{cc:c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]. \tag{4.71}$$

(Note that $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$). Implicit in the above expression of $\mathbf{Q}$, we have defined:

$$\mathbf{Q}_{||} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{Q}_\perp = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{4.72}$$

where the range of $\mathbf{Q}_{||}$ is the $\mathbf{v}_x - \mathbf{v}_y$ plane, and the range of $\mathbf{Q}_\perp$ is the $\mathbf{v}_z$ axis.

To decompose the vector $\mathbf{y}$ according to $\mathbf{Q}$, we proceed through the following steps:

$$
\begin{aligned}
\mathbf{y} &= (\mathbf{Q}_{||}\mathbf{Q}_{||}^T + \mathbf{Q}_{\perp}\mathbf{Q}_{\perp}^T)\mathbf{y} \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{y}[2] \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \mathbf{y}[2] \end{bmatrix}.
\end{aligned}
\tag{4.73}
$$

Thus indicating that:

$$
\mathbf{y}_{||} = \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ 0 \end{bmatrix}, \quad \mathbf{y}_{\perp} = \begin{bmatrix} 0 \\ 0 \\ \mathbf{y}[2] \end{bmatrix}. \tag{4.74}
$$

4.6 The QR decomposition

4.6.1 Application to Least Squares Solution

Conversion of Fundamental Equations

The QR decomposition will prove extremely useful as we apply it to the least squares procedure. In particular, we will use this decomposition to observe how the residual error, **error**, and estimated uncorrupted output, $\hat{\mathbf{y}}_u$, are affected by changes in model order. Our first step toward this process is to recall the basic equations involved in the least squares procedure. The first such equation pertains to the system model:

$$
\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{e}, \tag{4.75}
$$

where the uncorrupted output is defined to be $\mathbf{y}_u = \boldsymbol{\Phi}\boldsymbol{\theta}$. In practice, $\boldsymbol{\theta}$ and $\mathbf{e}$ are unknown, and our desire to estimate the parameter vector in order to identify the system leads us to the estimation model equation:

$$
\mathbf{y} = \boldsymbol{\Phi}\hat{\boldsymbol{\theta}} + \mathbf{error}, \tag{4.76}
$$

the estimated uncorrupted output then being $\hat{\mathbf{y}}_u = \boldsymbol{\Phi}\hat{\boldsymbol{\theta}}$. In order to obtain the actual estimates, the least squares procedure is incorporated, namely:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{y}. \tag{4.77}$$

One should note that, in order to write the above equation, we are assuming that $\boldsymbol{\Phi}$ has full column rank so that $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}$ exists.

We now proceed to obtain an alternate form for the above equations by taking the QR decomposition of the data matrix, i.e.:

$$\boldsymbol{\Phi} = \mathbf{QR} = \left[\begin{array}{c:c} \mathbf{Q}_{||} & \mathbf{Q}_{\perp} \end{array}\right] \left[\begin{array}{c} \mathbf{R}_{||} \\ \hdashline \mathbf{0} \end{array}\right] = \mathbf{Q}_{||}\mathbf{R}_{||}. \tag{4.78}$$

In terms of this decomposition, we write equations 4.75 and 4.76 as:

$$\mathbf{y} = \mathbf{Q}_{||}\mathbf{R}_{||}\boldsymbol{\theta} + \mathbf{e}, \tag{4.79}$$

and:

$$\mathbf{y} = \mathbf{Q}_{||}\mathbf{R}_{||}\hat{\boldsymbol{\theta}} + \mathbf{error}. \tag{4.80}$$

The uncorrupted output and its estimate are altered in a similar manner, so that:

$$\mathbf{y}_u = \mathbf{Q}_{||}\mathbf{R}_{||}\boldsymbol{\theta}, \quad \hat{\mathbf{y}}_u = \mathbf{Q}_{||}\mathbf{R}_{||}\hat{\boldsymbol{\theta}}. \tag{4.81}$$

To convert equation 4.77, we write:

$$\hat{\boldsymbol{\theta}} = ((\mathbf{Q}_{||}\mathbf{R}_{||})^T\mathbf{Q}_{||}\mathbf{R}_{||})^{-1}(\mathbf{Q}_{||}\mathbf{R}_{||})^T\mathbf{y} = (\mathbf{R}_{||}^T(\mathbf{Q}_{||}^T\mathbf{Q}_{||})\mathbf{R}_{||})^{-1}\mathbf{R}_{||}^T\mathbf{Q}_{||}^T\mathbf{y}. \tag{4.82}$$

To simplify this formulation, we recall the property expressed in equation 4.62, namely:

$$\mathbf{Q}_{||}^T\mathbf{Q}_{||} = \mathbf{I}, \tag{4.83}$$

so that we obtain:

$$\hat{\boldsymbol{\theta}} = (\mathbf{R}_{||}^T\mathbf{R}_{||})^{-1}\mathbf{R}_{||}^T\mathbf{Q}_{||}^T\mathbf{y} = \mathbf{R}_{||}^{-1}\mathbf{Q}_{||}^T\mathbf{y}. \tag{4.84}$$

(for which we have made use of the fact the $\mathbf{R}_{||}$ is invertible by the assumption that $\boldsymbol{\Phi}$ has

full column rank.)

The above least squares solution can be used to obtain useful expressions for the residual error and uncorrupted output estimate. Beginning with the latter of these quantities, we write:

$$\hat{\mathbf{y}}_u = \mathbf{\Phi}\hat{\boldsymbol{\theta}} = \mathbf{Q}_{||}\mathbf{R}_{||}\mathbf{R}_{||}^{-1}\mathbf{Q}_{||}^T\mathbf{y} = \mathbf{Q}_{||}\mathbf{Q}_{||}^T\mathbf{y}. \tag{4.85}$$

For the residual error, we calculate:

$$\mathbf{error} = \mathbf{y} - \hat{\mathbf{y}}_u = (\mathbf{I} - \mathbf{Q}_{||}\mathbf{Q}_{||}^T)\mathbf{y}, \tag{4.86}$$

and make use of the property expressed in equation 4.59,namely:

$$\mathbf{Q}_{||}\mathbf{Q}_{||}^T + \mathbf{Q}_{\perp}\mathbf{Q}_{\perp}^T = \mathbf{I}, \tag{4.87}$$

to obtain:

$$\mathbf{error} = \mathbf{Q}_{\perp}\mathbf{Q}_{\perp}^T\mathbf{y}. \tag{4.88}$$

Note the fact that **error** and $\hat{\mathbf{y}}_u$ are orthogonal (i.e. $\hat{\mathbf{y}}_u^T\mathbf{error} = 0$, by the fact that $\mathbf{Q}_{||}^T\mathbf{Q}_{\perp} = 0$). Also, it is readily seen that $\hat{\mathbf{y}}_u$ lies in the range of $\mathbf{Q}_{||}$, and **error** lies in the range of $\mathbf{Q}_{\perp}$. This fact becomes more useful when we realize that the range of $\mathbf{\Phi}$ always lies within the range of $\mathbf{Q}_{||}$, and that they are equivalent provided that $\mathbf{\Phi}$ has full column rank. Without providing a rigorous proof for this, we will give the reader an intuitive notion of why this statement is true. To do so, let us define some arbitrary vector $\mathbf{v}$ for use in the following equality:

$$\mathbf{\Phi} = \mathbf{Q}_{||}\mathbf{R}_{||} \implies \mathbf{\Phi}\mathbf{v} = \mathbf{Q}_{||}\mathbf{R}_{||}\mathbf{v}. \tag{4.89}$$

Let us also define the vector $\mathbf{u}$ such that $\mathbf{u} = \mathbf{\Phi}\mathbf{v}$. Designating $\mathbf{p}_i$ as columns within $\mathbf{\Phi}$, we can then write:

$$\mathbf{u} = \begin{bmatrix} | & | & \\ \mathbf{p}_1 & \mathbf{p}_2 & \cdots \\ | & | & \end{bmatrix} \begin{bmatrix} \mathbf{v}[1] \\ \mathbf{v}[2] \\ \vdots \end{bmatrix} = \begin{bmatrix} | \\ \mathbf{p}_1 \\ | \end{bmatrix} \mathbf{v}_1 + \begin{bmatrix} | \\ \mathbf{p}_2 \\ | \end{bmatrix} \mathbf{v}_2 + \cdots, \tag{4.90}$$

so that we observe that $\mathbf{u}$ is formed by summing columns within $\mathbf{\Phi}$ that have been scaled by elements within $\mathbf{v}$. Going back to equation 4.89, we now create a new vector $\mathbf{w}$ such that $\mathbf{w} = \mathbf{R}_{||}\mathbf{v}$. Using this definition, we write:

$$\mathbf{u} = \begin{bmatrix} | & | & \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots \\ | & | & \end{bmatrix} \begin{bmatrix} \mathbf{w}[1] \\ \mathbf{w}[2] \\ \vdots \end{bmatrix} = \begin{bmatrix} | \\ \mathbf{q}_1 \\ | \end{bmatrix} \mathbf{w}_1 + \begin{bmatrix} | \\ \mathbf{q}_2 \\ | \end{bmatrix} \mathbf{w}_2 + \cdots, \tag{4.91}$$

where $\mathbf{q}_i$ denotes the i^{th} column of $\mathbf{Q}_{||}$. The previous equation states that, by the equality given in equation 4.89, a vector $\mathbf{u}$ formed by the columns of matrix $\mathbf{\Phi}$ can also be formed by the columns of $\mathbf{Q}_{||}$. Thus, the range of matrix $\mathbf{\Phi}$ lies within the range of $\mathbf{Q}_{||}$.

In summary, the least squares procedure has the property that it decomposes the vector $\mathbf{y}$ into $\hat{\mathbf{y}}_u$ and **error** such that each of these components are orthogonal. In particular, the vector $\hat{\mathbf{y}}_u$ is confined to lie within the range of matrix $\mathbf{\Phi}$, and the **error** vector is orthogonal to this range.

Example

To get a feel for the above results, let us return to the example portrayed at the beginning of this chapter. We considered two systems described as follows:

$$a) \quad \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} [\, \mathbf{b}_1[0] \,] + \begin{bmatrix} \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix}, \tag{4.92}$$

and

$$b) \quad \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{b}_1[0] \\ \mathbf{b}_1[1] \end{bmatrix} + \begin{bmatrix} \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix} \tag{4.93}$$

Note that the range of $\mathbf{\Phi}$ for system (a) is the $\mathbf{v}_x$ axis, and the range of $\mathbf{\Phi}$ for system (b) is the $\mathbf{v}_x - \mathbf{v}_y$ plane.

The first step in our analysis is to determine the QR decomposition of each system's respective data matrix. Normally, this task would involve performing a numerical procedure. The simplicity of our chosen systems, however, allows us to do this operation by inspection. Thus, we write for system (a):

$$a) \ \mathbf{\Phi} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \implies \mathbf{Q} = \left[\begin{array}{c:cc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right], \ \mathbf{R} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{4.94}$$

so that:

$$a)\ \mathbf{Q}_{||} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{Q}_{\perp} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{and } \mathbf{R}_{||} = [\,1\,]. \tag{4.95}$$

System (b) is described in a similar way, with:

$$b)\ \mathbf{\Phi} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \Longrightarrow \mathbf{Q} = \left[\begin{array}{cc:c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \hdashline 0 & 0 & 1 \end{array}\right], \ \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \tag{4.96}$$

such that:

$$b)\ \mathbf{Q}_{||} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{Q}_{\perp} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \text{and } \mathbf{R}_{||} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.97}$$

The second step of our analysis is to apply equations 4.85 and 4.88 to the results above. This involves the decomposition of the output vector in each system into its respective components: $\hat{\mathbf{y}}_u$ and **error**. Performing this operation on system (a), we obtain:

$$a)\quad \begin{aligned} \hat{\mathbf{y}}_u &= \mathbf{Q}_{||}\mathbf{Q}_{||}^T\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\mathbf{y} = \begin{bmatrix} \mathbf{y}[0] \\ 0 \\ 0 \end{bmatrix} \\ \mathbf{error} &= \mathbf{Q}_{\perp}\mathbf{Q}_{\perp}^T\mathbf{y} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\mathbf{y} = \begin{bmatrix} 0 \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix}. \end{aligned} \tag{4.98}$$

Similarly, for system (b), we obtain:

$$b)\quad \begin{aligned} \hat{\mathbf{y}}_u &= \mathbf{Q}_{||}\mathbf{Q}_{||}^T\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\mathbf{y} = \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ 0 \end{bmatrix} \\ \mathbf{error} &= \mathbf{Q}_{\perp}\mathbf{Q}_{\perp}^T\mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ \mathbf{y}[2] \end{bmatrix}. \end{aligned} \tag{4.99}$$

In order to visualize the above results, we now compare illustrations of each system equation with their associated estimates.

For system (a), we have:

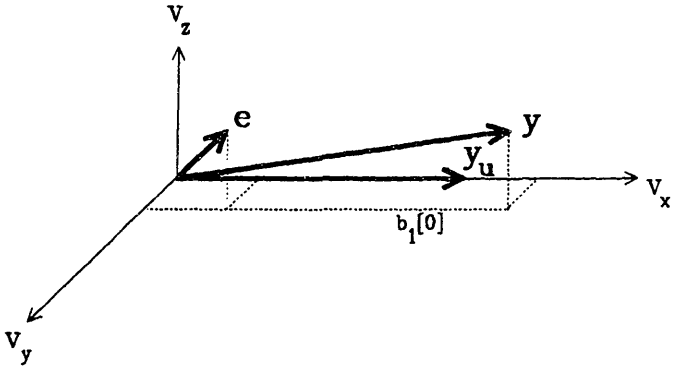


Figure 4-6: Vector illustration of actual system (a)

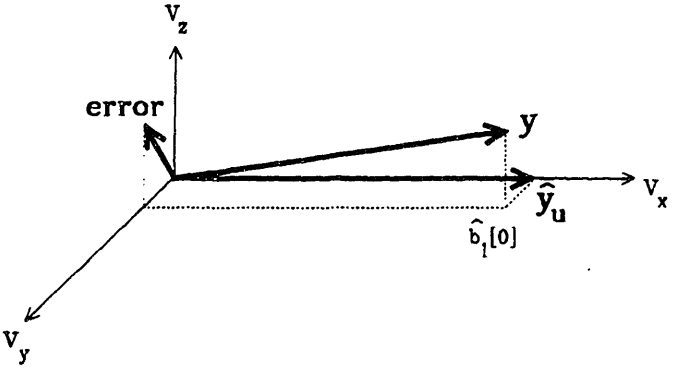


Figure 4-7: Vector illustration of estimated model (a)

For system (b), we have:

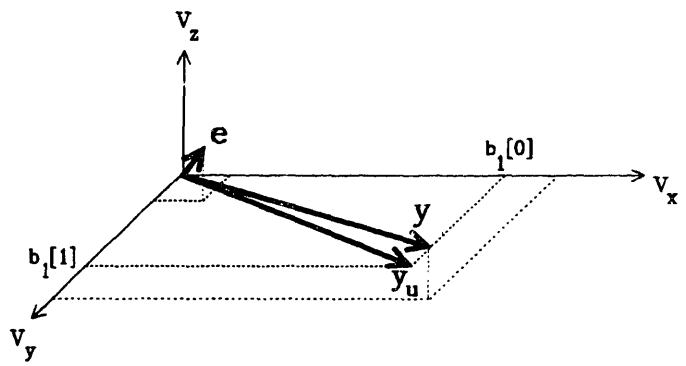


Figure 4-8: Vector illustration of actual system (b)

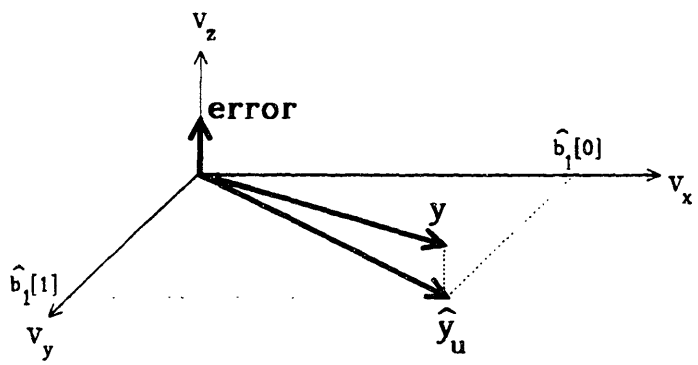


Figure 4-9: Vector illustration of estimated model (b)

Glancing at the above diagrams, we note that the uncorrupted output estimate, $\hat{\mathbf{y}}_u$, lies within the range of Φ for each system, and that the **error** vector is orthogonal to this range. This leads to a very important point — if $\mathbf{y}_u$ is within the range of Φ (i.e. our chosen parameterization includes all the 'true' parameters), then the **error** vector contains only components of the noise vector! Thus, least squares can be viewed as an operation that 'cuts off' part of the noise vector from the output, $\mathbf{y}$, in order to form $\hat{\mathbf{y}}_u$.

Taking the above results one step further, we observe that, in order to get the maximum noise reduction in our estimation procedure, it is desirable to select a Φ whose range is no greater than that needed to include $\mathbf{y}_u$. In other words, the greater the dimension in the range of Φ, the less components of $\mathbf{e}$ are included in the **error** vector (and, therefore, the more components of $\mathbf{e}$ are allowed to corrupt $\hat{\mathbf{y}}_u$). Optimally, we would like to choose Φ such that it corresponds to the 'minimal model' describing the system (i.e. it has only enough columns to include $\mathbf{y}_u$ in its range). Unfortunately, in the context of estimation, the 'minimal model' is unknown. However, we can make use of the above results to empirically *infer* the minimal model of the system. It is with this thought in mind that we approach the next section.

The Overparameterized Model

We now examine the case for which, in the context of estimation, we have assumed a model for the system that has more parameters than necessary. Referring to such a model as being 'overparameterized', we note that the effect on the least square equations is for the parameter vector to contain more elements than necessary. Corresponding to these extra elements, the data matrix, Φ, contains unnecessary columns. For the purposes of analysis, we will assume that the 'true' and 'extra' elements of the parameter vector have been grouped together, which directly imposes the condition that the 'true' and 'extra' columns of Φ are also clumped together. We can formally express these statements by partitioning the parameter vector as:

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_t \\ \cdots \\ \boldsymbol{\theta}_{ex} \end{bmatrix}, \tag{4.100}$$

and the data matrix as:

$$\Phi = \left[\begin{array}{c:c} \Phi_t & \Phi_{ex} \end{array} \right]. \tag{4.101}$$

Note that, by *definition*, $\theta_{ex} = \mathbf{0}$. This fact enables us to rewrite the system equation as:

$$\mathbf{y} = \Phi\theta + \mathbf{e} = \left[\begin{array}{c:c} \Phi_t & \Phi_{ex} \end{array} \right] \left[\begin{array}{c} \theta_t \\ \cdots \\ 0 \end{array} \right] + \mathbf{e} = \Phi_t\theta_t + \mathbf{e}. \tag{4.102}$$

The estimation equation, however, retains the extra parameters, i.e.:

$$\mathbf{y} = \Phi\hat{\theta} + \mathbf{error} = \left[\begin{array}{c:c} \Phi_t & \Phi_{ex} \end{array} \right] \left[\begin{array}{c} \hat{\theta}_t \\ \cdots \\ \hat{\theta}_{ex} \end{array} \right] + \mathbf{error}. \tag{4.103}$$

The QR Decomposition Applied to Changing Model Orders

Our desire is to gain an understanding of how the residual error and the uncorrupted output estimate are affected by overparameterizing, at varying degrees, the estimation model. We will make use of the QR decomposition in order to accomplish this task, but we must further explore this method in order to apply it to the given problem. In particular, we need to determine how the QR decomposition of Φ changes as columns are taken away from this matrix (i.e. as Φ is stripped down to Φ_t).

Recall again the QR decomposition of Φ:

$$\Phi = \mathbf{QR} = \left[\begin{array}{c:c} \mathbf{Q}_{||} & \mathbf{Q}_{\perp} \end{array} \right] \left[\begin{array}{c} \mathbf{R}_{||} \\ \cdots \\ \mathbf{0} \end{array} \right] = \mathbf{Q}_{||}\mathbf{R}_{||}. \tag{4.104}$$

If we now consider the data matrix to be partitioned into 'true' and 'extra' columns, i.e.

$$\Phi = \left[\begin{array}{c:c} \Phi_t & \Phi_{ex} \end{array} \right], \tag{4.105}$$

then we notice that the 'true' data matrix describing the system consists of the first t columns of Φ, with the last ex columns being unnecessary, or 'extra'. A reasonable question to ask is: "How is the QR decomposition of Φ_t related to the QR decomposition of Φ?" To answer this question, we will look at the manner in which Φ is formed from $\mathbf{Q}_{||}$ and $\mathbf{R}_{||}$. Noting the fact that $\mathbf{Q}_{||}$ has N rows and $t + ex$ columns, and $\mathbf{R}_{||}$ has $t + ex$ rows and $t + ex$

columns, we can express Φ in terms of the following matrix multiplication:

$$\Phi[i,j] = \sum_{k=1}^{t+e} \mathbf{Q}_{||}[i,k]\mathbf{R}_{||}[k,j], \;\; 1 \leq i \leq N, \; 1 \leq j \leq t+e, \tag{4.106}$$

where $\Phi[i,j]$ specifies the i^{th} row and j^{th} column of Φ. Now, in order to form Φ_t, we simply limit ourselves to the first t columns of Φ, i.e.

$$\Phi_t[i,j] = \sum_{k=1}^{t+e} \mathbf{Q}_{||}[i,k]\mathbf{R}_{||}[k,j], \;\; 1 \leq i \leq N, \; 1 \leq j \leq t. \tag{4.107}$$

Essentially, the only difference between forming Φ_t and Φ is that the last e columns of $\mathbf{R}_{||}$ are needed to form Φ, but not needed to form Φ_t. We can express this fact by partitioning the $\mathbf{R}_{||}$ matrix accordingly:

$$\Phi = \left[\begin{array}{c:c} \Phi_t & \Phi_{ex} \end{array} \right] = \mathbf{Q}_{||} \left[\begin{array}{c:c} \mathbf{R}_{||_1} & \mathbf{R}_{||_2} \end{array} \right], \tag{4.108}$$

so that we have:

$$\Phi_t = \mathbf{Q}_{||}\mathbf{R}_{||_1}. \tag{4.109}$$

It will be of value to be more explicit about the above partitioning of $\mathbf{R}_{||}$. In particular, let us rewrite equation 4.108 as:

$$\left[\begin{array}{c:c} \Phi_t & \Phi_{ex} \end{array} \right] = \left[\begin{array}{cccc} | & | & & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_{t+e} \\ | & | & & | \end{array} \right] \left[\begin{array}{ccc:ccc} * & \cdots & * & * & \cdots & * \\ 0 & \ddots & \vdots & * & \cdots & * \\ 0 & \ddots & * & * & \cdots & * \\ 0 & & 0 & * & \cdots & * \\ \vdots & & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & * \end{array} \right], \tag{4.110}$$

where $\mathbf{q}_i$ denotes the i^{th} column of $\mathbf{Q}_{||}$, and the $*$ entries have values that may be nonzero. If we also rewrite equation 4.109 in this more explicit format, we obtain:

$$\Phi_t = \begin{bmatrix} | & | & & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_{t+e} \\ | & | & & | \end{bmatrix} \begin{bmatrix} * & \cdots & * \\ 0 & \ddots & \vdots \\ 0 & \ddots & * \\ 0 & & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \tag{4.111}$$

One should notice that the above format lends itself directly to yet one more partitioning, namely:

$$\Phi_t = \left[\begin{array}{ccccc|c|ccc} | & | & & | & \vdots & | & & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_t & \vdots & \mathbf{q}_{t+1} & \cdots & \mathbf{q}_{t+e} \\ | & | & & | & \vdots & | & & | \end{array}\right] \begin{bmatrix} * & \cdots & * \\ 0 & \ddots & \vdots \\ 0 & \ddots & * \\ \cdots & \cdots & \cdots \\ 0 & & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \tag{4.112}$$

As a matter of notation, we rewrite the above equation as:

$$\Phi_t = \begin{bmatrix} \mathbf{Q}_{||_t} & \vdots & \mathbf{Q}_{||_{ex}} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{||_t} \\ \cdots \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_{||_t} \mathbf{R}_{||_t}, \tag{4.113}$$

so that $\mathbf{Q}_{||_t}$ represents the leftmost t columns of the original $\mathbf{Q}$ matrix, and $\mathbf{R}_{||_t}$ is a square matrix formed by the top t rows and leftmost t columns of the original $\mathbf{R}$ matrix. Unfortunately, the matrix $\begin{bmatrix} \mathbf{Q}_{||_t} & \vdots & \mathbf{Q}_{||_{ex}} \end{bmatrix}$ is not orthonormal by the fact that it is not square. However, let us note the fact that the rightmost $N - t$ columns of $\mathbf{Q}$ could be used to form a matrix $\mathbf{Q}_{\perp t}$ such that:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{||_t} & \vdots & \mathbf{Q}_{\perp t} \end{bmatrix}. \tag{4.114}$$

This formulation allows us to express Φ_t as:

$$\Phi_t = \begin{bmatrix} \mathbf{Q}_{||_t} & \vdots & \mathbf{Q}_{\perp t} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{||_t} \\ \cdots \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_{||_t} \mathbf{R}_{||_t} \tag{4.115}$$

(note that the $\mathbf{0}$ matrix in the above formulation has a different number of rows than the

one occurring in equation 4.113). We now have the property that:

$$\left[\mathbf{Q}_{||_t} \ \vdots \ \mathbf{Q}_{\perp t} \right]^T \left[\mathbf{Q}_{||_t} \ \vdots \ \mathbf{Q}_{\perp t} \right] = \left[\mathbf{Q}_{||_t} \ \vdots \ \mathbf{Q}_{\perp t} \right] \left[\mathbf{Q}_{||_t} \ \vdots \ \mathbf{Q}_{\perp t} \right]^T = \mathbf{I} \tag{4.116}$$

The above results indicate an extremely important property of the QR decomposition: we can obtain the QR decomposition of any $\mathbf{\Phi}_t$ matrix that composes the leftmost columns of some larger matrix $\mathbf{\Phi}$ *directly from* the QR decomposition of $\mathbf{\Phi}$! This has tremendous value computationally, as will be shortly developed, and it also leads to some very intuitive insights about least squares.

Example

To get a better feel for the results just covered, let us suppose that we were given the following overparameterized data matrix:

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 7 & 3 & -1 \\ 2 & 4 & -5 & 2 \\ 6 & 10 & 1 & 3 \\ -2 & 4 & 7 & 5 \\ 11 & 4 & 3 & 8 \end{bmatrix}. \tag{4.117}$$

Using a numerical procedure in 'Matlab', we obtained the QR decomposition of the above matrix, $\mathbf{\Phi} = \mathbf{QR}$, as:

$$\mathbf{\Phi} = \begin{bmatrix} -0.078 & -0.571 & 0.100 & 0.536 & 0.608 \\ -0.155 & -0.240 & -0.683 & -0.565 & 0.364 \\ -0.466 & -0.540 & -0.175 & 0.142 & -0.664 \\ 0.155 & -0.482 & 0.623 & -0.597 & -0.008 \\ -0.854 & 0.303 & 0.324 & -0.132 & 0.239 \end{bmatrix} \begin{bmatrix} -12.884 & -8.615 & -1.397 & -7.684 \\ 0 & -11.081 & -3.516 & -1.516 \\ 0 & 0 & 8.870 & 3.713 \\ 0 & 0 & 0 & -5.280 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.118}$$

In describing the given data matrix as:

$$\mathbf{\Phi} = \left[\mathbf{Q}_{||} \ \vdots \ \mathbf{Q}_{\perp} \right] \begin{bmatrix} \mathbf{R}_{||} \\ \cdots \\ \mathbf{0} \end{bmatrix}, \tag{4.119}$$

we can obtain the relevant matrices by inspection of equation 4.118:

$$\mathbf{Q}_{||} = \begin{bmatrix} -0.078 & -0.571 & 0.100 & 0.536 \\ -0.155 & -0.240 & -0.683 & -0.565 \\ -0.466 & -0.540 & -0.175 & 0.142 \\ 0.155 & -0.482 & 0.623 & -0.597 \\ -0.854 & 0.303 & 0.324 & -0.132 \end{bmatrix}, \quad \mathbf{Q}_{\perp} = \begin{bmatrix} 0.608 \\ 0.364 \\ -0.664 \\ -0.008 \\ 0.239 \end{bmatrix}, \tag{4.120}$$

and:

$$\mathbf{R}_{||} = \begin{bmatrix} -12.884 & -8.615 & -1.397 & -7.684 \\ 0 & -11.081 & -3.516 & -1.516 \\ 0 & 0 & 8.370 & 3.713 \\ 0 & 0 & 0 & -5.280 \end{bmatrix} \tag{4.121}$$

Now, suppose that we hypothesized that the true model had a data matrix that was a subset of the above example, so that:

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_t & \vdots & \mathbf{\Phi}_{ex} \end{bmatrix}. \tag{4.122}$$

Specifically, let us specify $t = 2$ (thereby causing ex to be equal to 2), so that $\mathbf{\Phi}_t$ was:

$$\mathbf{\Phi}_t = \begin{bmatrix} 1 & 7 \\ 2 & 4 \\ 6 & 10 \\ -2 & 4 \\ 11 & 4 \end{bmatrix}. \tag{4.123}$$

Using the results of the preceding subsection, we can directly obtain the QR decomposition of $\mathbf{\Phi}_t$, namely:

$$\mathbf{\Phi}_t = \begin{bmatrix} \mathbf{Q}_{||_t} & \vdots & \mathbf{Q}_{\perp t} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{||_t} \\ \cdots \\ \mathbf{0} \end{bmatrix}, \tag{4.124}$$

by inspection of equation 4.118. Proceeding, we write:

$$\mathbf{Q}_{\|_t} = \begin{bmatrix} -0.078 & -0.571 \\ -0.155 & -0.240 \\ -0.466 & -0.540 \\ 0.155 & -0.482 \\ -0.854 & 0.303 \end{bmatrix}, \quad \mathbf{Q}_{\perp t} = \begin{bmatrix} 0.100 & 0.536 & 0.608 \\ -0.683 & -0.565 & 0.364 \\ -0.175 & 0.142 & -0.664 \\ 0.623 & -0.597 & -0.008 \\ 0.324 & -0.132 & 0.239 \end{bmatrix}, \quad \mathbf{R}_{\|_t} = \begin{bmatrix} -12.884 & -8.615 \\ 0 & -11.081 \end{bmatrix} \tag{4.125}$$

One should note that we could have just as easily obtained the QR decomposition for any other Φ_t such that $t \leq 4$.

Residual Error of Models with Varying Order

We are now in a position to examine, in a quantitative sense, the effect of model order on residual error. Proceeding, let us assume that some 'maximal model' has been selected such that all considered models are subsets of it (including the 'true' model of the system). Designating the data matrix of this maximal parameterization as Φ_{max}, we denote its QR decomposition as:

$$\Phi_{max} = \mathbf{Q}_{max}\mathbf{R}_{max} = \begin{bmatrix} \mathbf{Q}_{\|_{max}} & \vdots & \mathbf{Q}_{\perp max} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\|_{max}} \\ \cdots \\ \mathbf{0} \end{bmatrix}. \tag{4.126}$$

Other models considered, Φ_κ, will be restricted to a subset of Φ_{max} such that:

$$\Phi_{max} = \begin{bmatrix} \Phi_\kappa & \vdots & \Phi_{max-\kappa} \end{bmatrix}, \tag{4.127}$$

Their QR decomposition,

$$\Phi_\kappa = \begin{bmatrix} \mathbf{Q}_{\|_\kappa} & \vdots & \mathbf{Q}_{\perp\kappa} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\|_\kappa} \\ \cdots \\ \mathbf{0} \end{bmatrix}, \tag{4.128}$$

can be obtained by appropriately partitioning $\mathbf{Q}_{max}\mathbf{R}_{max}$, as shown in the previous section. It will be important to keep track of the dimensions of each of the above matrices, which are as follows:

- Φ_κ: N rows by κ columns,
- $\mathbf{Q}_{\|_\kappa}$: N rows by κ columns,
- $\mathbf{Q}_{\perp\kappa}$: N rows by $N - \kappa$ columns,

- $\mathbf{R}_{||_\kappa}$: κ rows by κ columns

(where κ equals the number of parameters, such that we substitute in $\kappa = max$ to obtain the dimensions of the appropriate matrices corresponding to the maximal model).

For the purposes of the analysis carried out in this section, we will have need only for the $\mathbf{Q}$ components of $\mathbf{\Phi}_\kappa$. Determination of these matrices is trivial: $\mathbf{Q}_{||_\kappa}$ is simply composed of the leftmost κ columns in $\mathbf{Q}_{max}$, and $\mathbf{Q}_{\perp_\kappa}$ is equivalent to the rightmost $N - \kappa$ columns in $\mathbf{Q}_{max}$. Thus, if we denote:

$$\mathbf{Q}_{max} = \begin{bmatrix} | & & | & & | & & | \\ \mathbf{q}_1 & \cdots & \mathbf{q}_\kappa & \cdots & \mathbf{q}_{max} & \cdots & \mathbf{q}_N \\ | & & | & & | & & | \end{bmatrix} \tag{4.129}$$

(where $\mathbf{q}_i$ corresponds to the i^{th} column of $\mathbf{Q}_{max}$), then:

$$\mathbf{Q}_{||_\kappa} = \begin{bmatrix} | & & | \\ \mathbf{q}_1 & \cdots & \mathbf{q}_\kappa \\ | & & | \end{bmatrix}, \quad \mathbf{Q}_{\perp_\kappa} = \begin{bmatrix} | & & | \\ \mathbf{q}_{\kappa+1} & \cdots & \mathbf{q}_N \\ | & & | \end{bmatrix}. \tag{4.130}$$

It will also be useful to bear in mind that, by the property of $\mathbf{Q}_{max}$ being orthonormal, the columns within this matrix are all of unit length and are orthogonal to one other:

$$\mathbf{q}_i^T \mathbf{q}_j = 0, \;\; i \neq j; \quad \mathbf{q}_i^T \mathbf{q}_j = 1, \;\; i = j. \tag{4.131}$$

As specified earlier, it will be assumed that $\mathbf{\Phi}_{max}$ contains the 'true' data matrix of the system, $\mathbf{\Phi}_t$. It is further assumed that this matrix occupies the leftmost columns of $\mathbf{\Phi}_{max}$ such that choosing $\kappa = t$ defines the true model. Recalling the equation for the 'true' system:

$$\mathbf{y} = \mathbf{Q}_{||_t} \mathbf{R}_{||_t} \boldsymbol{\theta} + \mathbf{e}, \tag{4.132}$$

we note that all of the columns in $\mathbf{Q}_{max}$, namely $\mathbf{q}_i$, for which $i > t$ are orthogonal to each of the columns within $\mathbf{Q}_{||_t}$.

Absolute Error

We now address the quantification of the residual error norm occurring under models of varying order (i.e. κ). Designating the residual error that occurs with the data matrix

defined as Φ_κ as $\mathbf{error}_\kappa$, we now make use of equation 4.88 in writing:

$$\mathbf{error}_\kappa = \mathbf{Q}_{\perp\kappa}\mathbf{Q}_{\perp\kappa}^T\mathbf{y}. \tag{4.133}$$

Our interest lies in describing, in a statistical manner, the characteristics of the l_2 norm of the above vector, $\|\mathbf{error}_\kappa\|_2$. However, it will prove much easier to work with the square of this quantity in proceeding through the derivations, so that our focus will be on the energy of $\mathbf{error}_\kappa$ — $\|\mathbf{error}_\kappa\|_2^2$.

Let us now consider equation 4.133 for the case where $t \leq \kappa \leq max$. Substituting equation 4.132 into the above expression, we obtain:

$$\mathbf{error}_\kappa = \mathbf{Q}_{\perp\kappa}\mathbf{Q}_{\perp\kappa}^T(\mathbf{Q}_{\|_t}\mathbf{R}_{\|_t}\boldsymbol{\theta} + \mathbf{e}). \tag{4.134}$$

We can readily reduce the above quantity by noting that every column within $\mathbf{Q}_{\perp\kappa}$ is orthogonal to every column within $\mathbf{Q}_{\|_t}$ under the assumption that $\kappa \geq t$. This fact implies, therefore, that $\mathbf{Q}_{\perp\kappa}^T\mathbf{Q}_{\|_t} = 0$, so that equation 4.133 is reduced to:

$$\mathbf{error}_\kappa = \mathbf{Q}_{\perp\kappa}\mathbf{Q}_{\perp\kappa}^T\mathbf{e}. \tag{4.135}$$

Essentially, the above equation simply restates the fact that, given that $\mathbf{y}_u$ is within the range of Φ_κ (i.e. $\kappa \geq t$), the **error** vector is composed only of some component of the noise vector, **e**.

The beginning of this chapter sought after a quantification of the l_2 norm of the noise vector **e**, resulting in the conclusion that its 'normalized energy' was distributed according to the Chi Square probability function. Equation 4.135 reveals that the $\mathbf{error}_\kappa$ vector is completely determined by **e** for $\kappa \geq t$, which leads us to the question: "since $\mathbf{error}_\kappa$ is completely determined by **e** for this range on κ, can we quantify $\|\mathbf{error}_\kappa\|_2^2$ as we did $\|\mathbf{e}\|_2^2$?" In pursuit of the answer to this question, a natural first step is to directly calculate $\|\mathbf{error}_\kappa\|_2^2$ using equation 4.135. However, for convenience, let us first define a new vector $\mathbf{w}_\kappa$ such that:

$$\mathbf{w}_\kappa = \mathbf{Q}_{\perp\kappa}^T\mathbf{e} \tag{4.136}$$

(One should observe that $\mathbf{w}_\kappa$ contains $N - \kappa$ elements). Examining the autocorrelation

matrix of this vector, we obtain:

$$E(\mathbf{w}_\kappa \mathbf{w}_\kappa^T) = E(\mathbf{Q}_{\perp\kappa}^T \mathbf{e}\mathbf{e}^T \mathbf{Q}_{\perp\kappa}) = \mathbf{Q}_{\perp\kappa}^T (\sigma_e^2 \mathbf{I}) \mathbf{Q}_{\perp\kappa} = \sigma_e^2 \mathbf{I}. \tag{4.137}$$

Inspection of the above expression reveals that $\mathbf{w}_\kappa$ is, in fact, a white noise vector (i.e. its autocorrelation matrix is diagonal)! Thus, its 'normalized energy' is distributed according to the Chi Square density function with degrees of freedom equal to $N - \kappa$ (the number of elements in $\mathbf{w}_\kappa$):

$$\frac{\|\mathbf{w}_\kappa\|_2^2}{\sigma_e^2} \rightsquigarrow \chi^2_{N-\kappa}. \tag{4.138}$$

We now proceed to examine $\|\mathbf{error}_\kappa\|_2^2$. Rewriting equation 4.135 in terms of $\mathbf{w}_\kappa$, we have:

$$\mathbf{error}_\kappa = \mathbf{Q}_{\perp\kappa} \mathbf{w}_\kappa, \tag{4.139}$$

so that:

$$\|\mathbf{error}_\kappa\|_2^2 = \mathbf{w}_\kappa^T \mathbf{Q}_{\perp\kappa}^T \mathbf{Q}_{\perp\kappa} \mathbf{w}_\kappa = \mathbf{w}_\kappa^T \mathbf{w}_\kappa = \|\mathbf{w}_\kappa\|_2^2. \tag{4.140}$$

The merging of the above expression with equation 4.138 immediately leads to:

$$\frac{\|\mathbf{error}_\kappa\|_2^2}{\sigma_e^2} \rightsquigarrow \chi^2_{N-\kappa}. \tag{4.141}$$

Thus, for $\kappa \geq t$, the normalized residual error norm resulting from the use of $\mathbf{\Phi}_\kappa$ as the estimation data matrix is distributed according the Chi Square density function with $N - \kappa$ degrees of freedom.

Relative Error between Models

We now move into an examination of the residual error difference between models. Specifically, our goal is to investigate the behavior of the squared l_2 norm of the difference in error occurring between the maximal model and some chosen κ model, where $t \leq \kappa \leq max$. This quantity is defined as $\|\mathbf{error}_\kappa - \mathbf{error}_{max}\|_2^2$.

Proceeding with our investigation, we write for the maximal model:

$$\mathbf{error}_{max} = \mathbf{Q}_{\perp max} \mathbf{Q}_{\perp max}^T \mathbf{e}, \tag{4.142}$$

and for the κ model:

$$\mathbf{error}_\kappa = \mathbf{Q}_{\perp\kappa}\mathbf{Q}_{\perp\kappa}^T\mathbf{e}. \tag{4.143}$$

Upon examining the construction of these matrices, one should notice that, given $\kappa \leq max$, $\mathbf{Q}_{\perp max}$ is actually a subset of $\mathbf{Q}_{\perp\kappa}$:

$$\mathbf{Q}_{\perp\kappa} = \left[\begin{array}{cccc:c} | & | & & | & \vdots & \\ \mathbf{q}_\kappa & \mathbf{q}_{\kappa+1} & \cdots & \mathbf{q}_{max-1} & \vdots & \mathbf{Q}_{\perp max} \\ | & | & & | & \vdots & \end{array}\right]. \tag{4.144}$$

For convenience, we will label the additional columns contained within $\mathbf{Q}_{\perp\kappa}$ as $\mathbf{Q}_L$, such that the above equation becomes:

$$\mathbf{Q}_{\perp\kappa} = \left[\begin{array}{c:c} \mathbf{Q}_L & \mathbf{Q}_{\perp max} \end{array}\right], \tag{4.145}$$

where $\mathbf{Q}_L$ has the dimension of N rows by $max - \kappa$ columns, and the columns in $\mathbf{Q}_L$ have the property of being orthogonal to the columns in $\mathbf{Q}_{\perp max}$ (i.e. $\mathbf{Q}_L^T\mathbf{Q}_{\perp max} = \mathbf{0}$). Returning now to equation 4.143, we can rewrite $\mathbf{error}_\kappa$ as:

$$\mathbf{error}_\kappa = \left[\begin{array}{c:c} \mathbf{Q}_L & \mathbf{Q}_{\perp max} \end{array}\right]\left[\begin{array}{c} \mathbf{Q}_L^T \\ \cdots \\ \mathbf{Q}_{\perp max}^T \end{array}\right]\mathbf{e} = \mathbf{Q}_L\mathbf{Q}_L^T\mathbf{e} + \mathbf{Q}_{\perp max}\mathbf{Q}_{\perp max}^T\mathbf{e}. \tag{4.146}$$

Comparing equation 4.142 with the above formulation leads to:

$$\mathbf{error}_\kappa - \mathbf{error}_{max} = \mathbf{Q}_L\mathbf{Q}_L^T\mathbf{e}, \tag{4.147}$$

from which, by the results of the previous section, it becomes clear that:

$$\frac{\|\mathbf{error}_\kappa - \mathbf{error}_{max}\|_2^2}{\sigma_e^2} \rightsquigarrow \chi^2_{max-\kappa}. \tag{4.148}$$

Thus, we observe that, provided $t \leq \kappa \leq max$, the residual error difference between models of different order is distributed according to the Chi Square density function with $max - \kappa$ degrees of freedom.

We will now slightly modify the form of equation 4.148. Before doing so, let us note the

following general property: if we define the vectors $\mathbf{w}$, $\mathbf{u}$, and $\mathbf{v}$ such that:

$$\mathbf{w} = \mathbf{u} + \mathbf{v}, \tag{4.149}$$

and $\mathbf{u}$ is orthogonal to $\mathbf{v}$, then:

$$\|\mathbf{w}\|_2^2 = \|\mathbf{u}\|_2^2 + \|\mathbf{v}\|_2^2. \tag{4.150}$$

This property is easily proved upon writing:

$$\|\mathbf{w}\|_2^2 \;=\; \mathbf{w}^T\mathbf{w} \;=\; (\mathbf{u}+\mathbf{v})^T(\mathbf{u}+\mathbf{v}) \;=\; \mathbf{u}^T\mathbf{u} + \mathbf{v}^T\mathbf{v}, \tag{4.151}$$

where we made use of the fact that $\mathbf{u}^T\mathbf{v} = \mathbf{v}^T\mathbf{u} = 0$ in obtaining the rightmost quantity. Returning now to equation 4.146, we rewrite this expression as:

$$\mathbf{error}_\kappa = \mathbf{Q}_L\mathbf{Q}_L^T\mathbf{e} + \mathbf{error}_{max}, \tag{4.152}$$

where $\mathbf{Q}_L\mathbf{Q}_L^T\mathbf{e}$ and $\mathbf{error}_{max}$ are seen to be orthogonal by the fact that $\mathbf{Q}_L^T\mathbf{Q}_{\perp max} = \mathbf{0}$. Thus, upon taking the squared l_2 norm of the above expression, we obtain:

$$\|\mathbf{error}_\kappa\|_2^2 = \|\mathbf{Q}_L\mathbf{Q}_L^T\mathbf{e}\|_2^2 + \|\mathbf{error}_{max}\|_2^2, \tag{4.153}$$

which leads directly to an alternate expression of equation 4.148:

$$\frac{\|\mathbf{error}_\kappa\|_2^2 - \|\mathbf{error}_{max}\|_2^2}{\sigma_e^2} \rightsquigarrow \chi^2_{max-\kappa}. \tag{4.154}$$

Example

To provide a more concrete understanding of the above results, we will work again with our ongoing example. Recall the difference equation format specified for each of the models:

$$a)\;\; \mathbf{y}[k] = \mathbf{b}_1[0]\mathbf{x}_1[k] + \mathbf{e}[k], \tag{4.155}$$

and

$$b)\;\; \mathbf{y}[k] = \mathbf{b}_1[0]\mathbf{x}_1[k] + \mathbf{b}_1[1]\mathbf{x}_1[k-1] + \mathbf{e}[k], \tag{4.156}$$

along with the specified input sequence:

$$\mathbf{x}_1[k] = \delta[k], \quad \text{where: } \delta[k] = 1,\ k = 0;\ \delta[k] = 0, \text{ otherwise.} \tag{4.157}$$

The difference equation formats are readily converted into the 'true' data matrix and parameter vector for each system:

$$a)\ \ \mathbf{\Phi}_t = [\,\mathbf{x}_1[k]\,] = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\theta}_t = [\,\mathbf{b}_1[0]\,], \tag{4.158}$$

and

$$b)\ \ \mathbf{\Phi}_t = \begin{bmatrix} \mathbf{x}_1[k] & \mathbf{x}_1[k-1] \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \boldsymbol{\theta}_t = \begin{bmatrix} \mathbf{b}_1[0] \\ \mathbf{b}_1[1] \end{bmatrix}, \tag{4.159}$$

for which we have again considered k taking on values: $0 \leq k \leq N = 3$. Thus, we write the system equations of (a) and (b) as:

$$a)\ \ \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} [\,\mathbf{b}_1[0]\,] + \begin{bmatrix} \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1[0] + \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix}, \tag{4.160}$$

and

$$b)\ \ \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{b}_1[0] \\ \mathbf{b}_1[1] \end{bmatrix} + \begin{bmatrix} \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1[0] + \mathbf{e}[0] \\ \mathbf{b}_1[1] + \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix}. \tag{4.161}$$

To illustrate the concepts of the previous section, let us postulate a maximal model for each of the above systems as:

$$\mathbf{y}[k] = \mathbf{b}_1[0]\mathbf{x}_1[k] + \mathbf{b}_1[1]\mathbf{x}_1[k-1] + \mathbf{e}[k], \tag{4.162}$$

so that we have:

$$\mathbf{\Phi}_{max} = \begin{bmatrix} \mathbf{x}_1[k] & \mathbf{x}_1[k-1] \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \boldsymbol{\theta}_{max} = \begin{bmatrix} \mathbf{b}_1[0] \\ \mathbf{b}_1[1] \end{bmatrix}. \tag{4.163}$$

The QR decomposition of $\mathbf{\Phi}_{max}$ follows easily as:

$$\Phi_{max} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \implies \mathbf{Q}_{max} = \left[\begin{array}{cc:c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right], \quad \mathbf{R}_{max} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \tag{4.164}$$

from which we will be able to obtain the QR decomposition of each of the lower order models.

In considering models of varying order for each of the above systems, let us, for this example, take $\kappa = 1$ for our lowest order. Calculating the residual errors that occur for $1 \leq \kappa \leq max = 2$, we obtain for both models:

$$\begin{aligned} \mathbf{error}_2 &= \mathbf{Q}_{\perp 2}\mathbf{Q}_{\perp 2}^T\mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ \mathbf{y}[2] \end{bmatrix} \\ \mathbf{error}_1 &= \mathbf{Q}_{\perp 1}\mathbf{Q}_{\perp 1}^T\mathbf{y} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{y} = \begin{bmatrix} 0 \\ \mathbf{y}[1] \\ \mathbf{y}[2] \end{bmatrix}. \end{aligned} \tag{4.165}$$

so that the residual errors corresponding to system (a) are:

$$\begin{aligned} \mathbf{error}_2 &= \begin{bmatrix} 0 \\ 0 \\ \mathbf{e}[2] \end{bmatrix}, \\ \mathbf{error}_1 &= \begin{bmatrix} 0 \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix}, \end{aligned} \tag{4.166}$$

and for system (b):

$$\begin{aligned} \mathbf{error}_2 &= \begin{bmatrix} 0 \\ 0 \\ \mathbf{e}[2] \end{bmatrix}, \\ \mathbf{error}_1 &= \begin{bmatrix} 0 \\ \mathbf{b}_1[1] + \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix}, \end{aligned} \tag{4.167}$$

As a side comment, note the comparable ease with which estimates of the uncorrupted output are computed:

$$\hat{\mathbf{y}}_{u2} = \mathbf{Q}_{\|_2}\mathbf{Q}_{\|_2}^T\mathbf{y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\mathbf{y} = \begin{bmatrix} \mathbf{y}[0] \\ \mathbf{y}[1] \\ 0 \end{bmatrix}$$
$$\hat{\mathbf{y}}_{u1} = \mathbf{Q}_{\|_1}\mathbf{Q}_{\|_1}^T\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\mathbf{y} = \begin{bmatrix} \mathbf{y}[0] \\ 0 \\ 0 \end{bmatrix} \tag{4.168}$$

so that for system (a) we have:

$$\hat{\mathbf{y}}_{u2} = \begin{bmatrix} \mathbf{b}_1[0] + \mathbf{e}[0] \\ \mathbf{e}[1] \\ 0 \end{bmatrix},$$
$$\hat{\mathbf{y}}_{u1} = \begin{bmatrix} \mathbf{b}_1[0] + \mathbf{e}[0] \\ 0 \\ 0 \end{bmatrix}, \tag{4.169}$$

and for system (b):

$$\hat{\mathbf{y}}_{u2} = \begin{bmatrix} \mathbf{b}_1[0] + \mathbf{e}[0] \\ \mathbf{b}_1[1] + \mathbf{e}[1] \\ 0 \end{bmatrix},$$
$$\hat{\mathbf{y}}_{u1} = \begin{bmatrix} \mathbf{b}_1[0] + \mathbf{e}[0] \\ 0 \\ 0 \end{bmatrix}, \tag{4.170}$$

For illustration purposes, we have plotted each of the above quantities.

For system (a), we have:

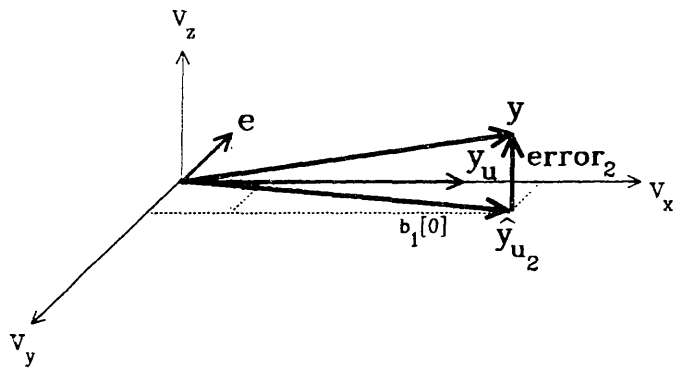


Figure 4-10: Vector illustration of the residual error occuring with the maximal model ($\kappa = 2$) of system (a)

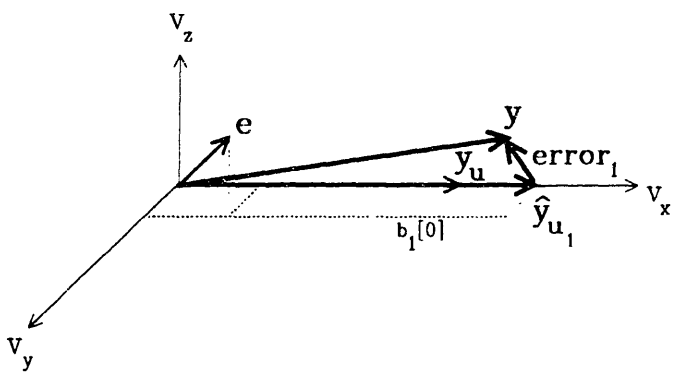


Figure 4-11: Vector illustration of the residual error occuring after removal of one parameter ($\kappa = 1$) from the maximal model in system (a)

For system (b), we have:

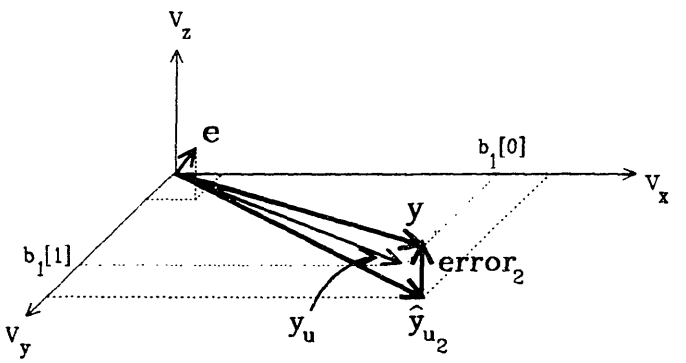


Figure 4-12: Vector illustration of the residual error occuring with the maximal model ($\kappa = 2$) of system (b)

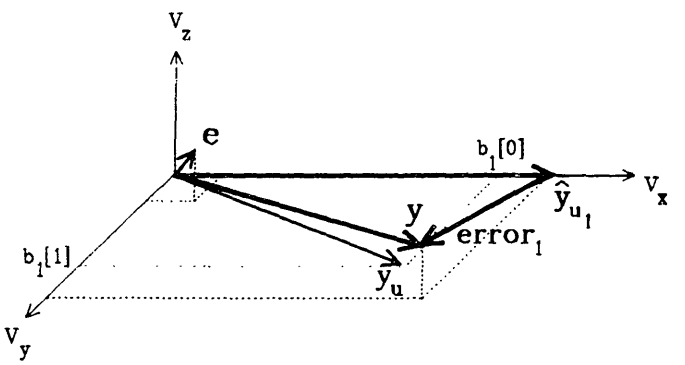


Figure 4-13: Vector illustration of the residual error occuring after removal of one parameter ($\kappa = 1$) from the maximal model in system (b)

One can readily notice from the above plots that, so long as $\kappa \geq t$, the resulting residual error contains only components of the noise. Conversely, as revealed in the case of $\kappa = 1$ for system (b), choosing $\kappa \leq t$ causes part of $\mathbf{y}_u$ to be included in **error**, i.e.

$$\|\mathbf{error}_1\|_2^2 = (\mathbf{e}[1] + \mathbf{b}_1[1])^2 + \mathbf{e}[2]^2. \tag{4.171}$$

Assuming that the signal to noise ratio of our estimation data is acceptable (i.e. $\mathbf{b}_1[1]^2$ is large in comparison to the noise variance), this included portion of $\mathbf{y}_u$ will cause the residual error to take on a larger length than would occur with only noise being included. This fact is directly made use of in the hypothesis testing approach to model selection.

4.7 Hypothesis Testing Revisited

We are finally in a position to explain the rationale behind the hypothesis testing scheme used in chapter 3. Essentially, the l_2 norm of the residual error will follow the Chi Square function so long as its corresponding model includes the 'true' parameters. Assuming this case to be true, the probability function can be used to set maximum limits on how much the error norm should change upon reducing the model order (this limit is determined by the specified confidence level, as covered in the beginning of this chapter). If the error norm, in fact, changes more than allowed by the Chi Square limit, then the interpretation is that part of the $\mathbf{y}_u$ vector has been included in the **error** vector. Such a result then signals that a 'true' column from $\mathbf{\Phi}$ has been removed, so that $\mathbf{\Phi}_t$ is identified.

In order for the above process to work, a good estimate of the variance of $\mathbf{e}$ is required. To obtain such an estimate, we make use of equation 4.140, which states:

$$\|\mathbf{error}_\kappa\|_2^2 = \|\mathbf{w}_\kappa\|_2^2, \quad \kappa \geq t, \tag{4.172}$$

where $\mathbf{w}_\kappa$ is a white noise vector of dimension $N - \kappa$ and variance σ_e^2. Therefore, using the results from the beginning of this chapter, we note that the expected energy of $\mathbf{w}_\kappa$ is:

$$E(\|\mathbf{w}_\kappa\|_2^2) = (N - \kappa)\sigma_e^2, \tag{4.173}$$

so that we make the approximation:

$$\|\mathbf{error}_\kappa\|_2^2 \approx (N - \kappa)\sigma_e^2, \quad \kappa \geq t. \tag{4.174}$$

This last expression leads directly to an estimate of the variance of $\mathbf{e}$. Namely, since it is

assumed that the maximal model has been chosen to include the true model, we can write:

$$\hat{\sigma}_e^2 = \frac{\|\mathbf{error}_{max}\|_2^2}{N - max}. \tag{4.175}$$

One should note that the above estimate improves as the quantity $N - max$ becomes larger (i.e. more samples are used in the estimation process).

With the maximal model being used to characterize the variance of the noise vector, the model order is then decreased and the *amount of change* occurring in residual error is observed. Once the norm of the error difference (between the maximal model and lower order model) exceeds the calculated Chi Square threshold, the true model is identified. In effect, the philosophy behind this approach is that, in order to justify using a lower order model in place of the maximal model, the difference occurring in residual error norm between these two models must be small enough to be ascribable to noise.

4.7.1 Example

Following up on the previous example, we again chose $\mathbf{\Phi}_{max}$ such that $max = 2$ for system (a) and (b). Using exactly the same procedure as followed previously, we calculated residual errors for $0 \leq \kappa \leq max = 2$ (note that we are allowing the case $\kappa = 0$ this time). System (a) errors were:

$$\begin{aligned}
\mathbf{error}_2 &= \begin{bmatrix} 0 \\ 0 \\ \mathbf{e}[2] \end{bmatrix}, \\
\mathbf{error}_1 &= \begin{bmatrix} 0 \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix}, \\
\mathbf{error}_0 &= \begin{bmatrix} \mathbf{b}_1[0] + \mathbf{e}[0] \\ \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix},
\end{aligned} \tag{4.176}$$

system (b) errors were:

$$\mathbf{error}_2 = \begin{bmatrix} 0 \\ 0 \\ \mathbf{e}[2] \end{bmatrix},$$

$$\mathbf{error}_1 = \begin{bmatrix} 0 \\ \mathbf{b}_1[1] + \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix}, \tag{4.177}$$

$$\mathbf{error}_0 = \begin{bmatrix} \mathbf{b}_0[1] + \mathbf{e}[0] \\ \mathbf{b}_1[1] + \mathbf{e}[1] \\ \mathbf{e}[2] \end{bmatrix}.$$

Therefore, for each of the above systems, we would calculate:

$$\hat{\sigma}_e^2 = \frac{\mathbf{e}[2]^2}{3-2} \tag{4.178}$$

for the residual error estimate (not a very good estimate of this quantity for this case — N is too low!).

The residual error norm would then be calculated upon reducing the model order for each system, which would in turn lead to the comparison of these values to their Chi Square limits. The procedure for calculating these limits, $\mu_{max-\kappa}$, was described at the beginning of this chapter. Thus, for system (a), we would have the following:

$$\begin{aligned} \frac{\|\mathbf{error}_1\|_2^2 - \|\mathbf{error}_2\|_2^2}{\sigma_e^2} &= \mathbf{e}[1]^2 \overset{?}{>} \mu_1, \\ \frac{\|\mathbf{error}_0\|_2^2 - \|\mathbf{error}_2\|_2^2}{\sigma_e^2} &= (\mathbf{b}_1[0] + \mathbf{e}[0])^2 + \mathbf{e}[1]^2 \overset{?}{>} \mu_2, \end{aligned} \tag{4.179}$$

and for system (b):

$$\begin{aligned} \frac{\|\mathbf{error}_1\|_2^2 - \|\mathbf{error}_2\|_2^2}{\sigma_e^2} &= (\mathbf{b}_1[1] + \mathbf{e}[1])^2 \overset{?}{>} \mu_1, \\ \frac{\|\mathbf{error}_0\|_2^2 - \|\mathbf{error}_2\|_2^2}{\sigma_e^2} &= (\mathbf{b}_1[0] + \mathbf{e}[0])^2 + (\mathbf{b}_1[1] + \mathbf{e}[1])^2 \overset{?}{>} \mu_2. \end{aligned} \tag{4.180}$$

Inspection of the system (a) results reveals that the lowest model order to have a change in error norm that remained within Chi Square bounds would correspond to $\kappa = 1$ (assuming $|\mathbf{b}_1[0]| \gg 0$), so the $\mathbf{\Phi}_1$ would be correctly picked as the true data matrix. The results of system (b) would suggest that neither of the reduced models would have residual error differences that remained within the set limits (assuming $|\mathbf{b}_1[0]| \gg 0$, and $|\mathbf{b}_1[1]| \gg 0$), so that the maximal model ($\kappa = 2$), would correctly be selected as the true model.

Chapter 5

Confidence Limits for the Estimated ARX model

5.1 Introduction

This chapter focuses on the problem of providing an indicator for the amount of error associated with estimates resulting from the least squares procedure. In particular, we will be concerned with error associated with ARX model parameter estimates and their corresponding impulse responses. Our strategy will lie in first explaining a standard procedure for determining the error bounds at a given 'confidence level' for the estimated parameters, and then extending that same procedure for the calculated impulse response estimates. It should be noted that the latter method is implemented with some approximation, as it otherwise requires the solution of a nonlinear equation.

5.2 Error Associated With ARX Parameter Estimates

Recall the least squares system equation associated with the ARX model:

$$\mathbf{y} = \mathbf{\Phi}\boldsymbol{\theta} + \mathbf{e}, \tag{5.1}$$

along with its corresponding estimation equation:

$$\mathbf{y} = \mathbf{\Phi}\hat{\boldsymbol{\theta}} + \mathbf{error}. \tag{5.2}$$

Using the least squares procedure to determine the parameter estimates in the above equation leads to:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{y}. \tag{5.3}$$

Our goal is to determine some procedure that will provide an indicator of the error associated with the above least square estimates. Proceeding, we first substitute equation 5.1 into equation 5.3 and then simplify to obtain:

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta} + (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{e}. \tag{5.4}$$

Denoting $\Delta\hat{\boldsymbol{\theta}}$ as the estimation error, $\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}$, we then write:

$$\Delta\hat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{e}. \tag{5.5}$$

For convenience, we will define $\mathbf{C} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T$, so that the above equation turns into:

$$\Delta\hat{\boldsymbol{\theta}} = \mathbf{C}\mathbf{e}. \tag{5.6}$$

To gain a better understanding of the above expression, it is useful to recall that, by definition of a white, gaussian noise process, the entries within $\mathbf{e}$ are independent and gaussian. It is a property of gaussian random variables that their addition leads to random variables that are also gaussian and therefore completely characterized by their mean and covariance, which is derived as:

$$\begin{aligned} E(\Delta\hat{\boldsymbol{\theta}}) &= \mathbf{C}E(\mathbf{e}) = 0, \\ E(\Delta\hat{\boldsymbol{\theta}}\Delta\hat{\boldsymbol{\theta}}^T) &= \mathbf{C}E(\mathbf{e}\mathbf{e}^T)\mathbf{C}^T = \sigma_e^2\mathbf{C}\mathbf{C}^T \end{aligned} \tag{5.7}$$

Using the above quantities, we could derive the joint density of each of the elements in $\Delta\hat{\boldsymbol{\theta}}$, $f(\Delta\hat{\boldsymbol{\theta}}[1], \Delta\hat{\boldsymbol{\theta}}[2], \cdots, \Delta\hat{\boldsymbol{\theta}}[\kappa])$. However, for practical purposes, it is often sufficient to simply calculate the probability density associated with each individual parameter error (without taking into account the crosscorrelation between parameters), for which we need only consider the diagonal elements of the autocorrelation matrix, $E(\Delta\hat{\boldsymbol{\theta}}\Delta\hat{\boldsymbol{\theta}}^T)$. In particular, since the mean of each estimate is zero, the estimate of each error's probability density function

becomes:

$$f(\Delta\hat{\boldsymbol{\theta}}[i]) \;=\; N(0, E(\Delta\hat{\boldsymbol{\theta}}[i]^2)) \;=\; N(0, \sigma_e^2 \mathbf{C}[i]\mathbf{C}[i]^T) \;=\; N(0, \sigma_e^2 \|\mathbf{C}[i]\|_2^2) \qquad (5.8)$$

where $\mathbf{C}[i]$ represents the i^{th} row of $\mathbf{C}$.

The above result was actually derived in chapter 3 using a different approach, which we will quickly review here. Namely, we can view each parameter error estimate within equation 5.6 as the sum of scaled, independent, gaussian random variables, as represented by the following:

$$\Delta\hat{\boldsymbol{\theta}}[i] = \sum_{j=1}^{N} \mathbf{C}[i,j]\mathbf{e}[j]. \qquad (5.9)$$

We note that since each $\Delta\hat{\boldsymbol{\theta}}[i]$ is formed by the sum of gaussian random variables, it also gaussian with mean:

$$E(\Delta\hat{\boldsymbol{\theta}}[i]) \;=\; \sum_{j=1}^{N} \mathbf{C}[i,j]E(\mathbf{e}[j]) \;=\; 0, \qquad (5.10)$$

and variance:

$$\mathrm{var}(\Delta\hat{\boldsymbol{\theta}}[i]) \;=\; \sum_{j=1}^{N} \mathbf{C}[i,j]^2\sigma_e^2 \;=\; \|\mathbf{C}[i]\|_2^2\sigma_e^2. \qquad (5.11)$$

5.2.1 Example

To illustrate the above concepts, let us introduce an example least squares estimation equation:

$$\mathbf{y} = \boldsymbol{\Phi}\hat{\boldsymbol{\theta}} + \mathbf{error}, \;\; \text{where: } \mathbf{y} = \begin{bmatrix} 23 \\ 55 \\ 70 \end{bmatrix}, \;\; \boldsymbol{\Phi} = \begin{bmatrix} 1 & 2 \\ 5 & 3 \\ 2 & 7 \end{bmatrix}. \qquad (5.12)$$

We readily obtain the matrix $\mathbf{C}$ as:

$$\mathbf{C} = \left(\begin{bmatrix} 1 & 5 & 2 \\ 2 & 3 & 7 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 5 & 3 \\ 2 & 7 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 5 & 2 \\ 2 & 3 & 7 \end{bmatrix} = \begin{bmatrix} 0 & 0.241 & -0.103 \\ 0.032 & -0.072 & 0.165 \end{bmatrix}, \qquad (5.13)$$

from which we obtain the parameter estimates:

$$\hat{\boldsymbol{\theta}} = \mathbf{C}\mathbf{y} = \begin{bmatrix} 0 & 0.241 & -0.103 \\ 0.032 & -0.072 & 0.165 \end{bmatrix} \begin{bmatrix} 23 \\ 55 \\ 70 \end{bmatrix} = \begin{bmatrix} 6.045 \\ 8.326 \end{bmatrix}. \qquad (5.14)$$

In order to calculate the error probability functions associated with the above estimates, we first obtain the resulting **error** vector as:

$$\mathbf{error} = \mathbf{y} - \mathbf{\Phi}\hat{\boldsymbol{\theta}} = \begin{bmatrix} 23 \\ 55 \\ 70 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 5 & 3 \\ 2 & 7 \end{bmatrix} \begin{bmatrix} 6.045 \\ 8.326 \end{bmatrix} = \begin{bmatrix} 0.303 \\ -0.203 \\ -0.372 \end{bmatrix}. \tag{5.15}$$

From the theory presented in chapter 4, we can use the above values to estimate the noise variance as:

$$\hat{\sigma}_e^2 = \frac{\|\mathbf{error}\|_2^2}{N - \kappa} = \frac{0.303^2 + (-0.203)^2 + (-0.372)^2}{3 - 2} = 0.271. \tag{5.16}$$

If we combine this estimate with the following fact:

$$\|\mathbf{C}[1]\|_2^2 = 0^2 + 0.241^2 + (-0.103)^2 = 0.069, \quad \|\mathbf{C}[2]\|_2^2 = 0.032^2 + (-0.072)^2 + 0.165^2 = 0.033, \tag{5.17}$$

then, based on equation 5.8, we can write:

$$f(\Delta\hat{\boldsymbol{\theta}}[1]) = N(0, 0.069 \cdot 0.271), \quad f(\Delta\hat{\boldsymbol{\theta}}[2]) = N(0, 0.033 \cdot 0.271). \tag{5.18}$$

5.2.2 Confidence Boundaries for Parameter Estimates

Armed with knowledge of the probability densities associated with each parameter estimate error, we now focus on the calculation of boundaries within which we feel, at some level of confidence, that our true parameters lie. As a step toward explaining the classical procedure for determining these boundaries, let us examine Figure 5-1. Designating $-\Delta_c$ to Δ_c as an error boundary on the error $\Delta\hat{\boldsymbol{\theta}}[i]$, we note that the probability of the actual error value occurring within this boundary corresponds to the shaded area in the figure. It is precisely the value of the probability indicated by the shaded region that determines the level of confidence associated with the given bounds. Thus, if we desired an error boundary at a 0.95 level of confidence, we would need to determine the value of Δ_c such that the shaded area equaled 0.95.

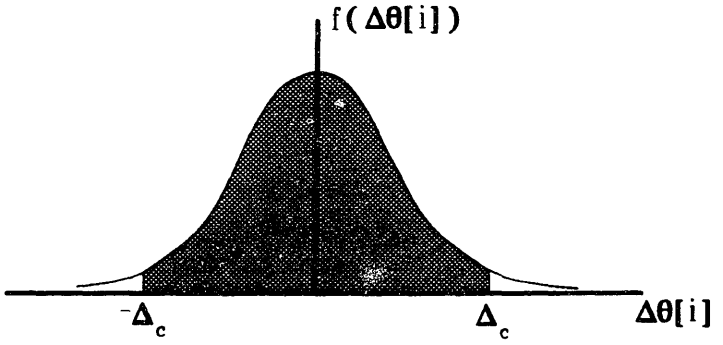


Figure 5-1: Illustration of error bound calculation using the gaussian probability density function

5.2.3 Example

Continuing with the previous example, we now derive boundaries within which we can state the parameter values lie within at a 0.95 level of confidence. By inspection of any statistical table of the gaussian function, we must specify $\Delta_c[i] = 1.96\sqrt{var(\Delta\boldsymbol{\theta}[i])}$ in order to set the probability associated with this boundary to 0.95. Thus, we may state at a 0.95 level of confidence that:

$$\boldsymbol{\theta}[i] = \hat{\boldsymbol{\theta}}[i] \pm \Delta_c[i] \tag{5.19}$$

which we express numerically as:

$$\boldsymbol{\theta}[1] = 6.045 \pm 1.96\sqrt{0.069 \cdot 0.271}, \quad \boldsymbol{\theta}[2] = 8.326 \pm 1.96\sqrt{0.033 \cdot 0.271}. \tag{5.20}$$

5.3 Impulse Response Calculation

For the purposes of this thesis, we are concerned with estimation results for the ARX model. Recall from chapter 2 that the least squares format corresponding to this model is:

$$\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{e}, \tag{5.21}$$

where:

$$\Phi = \left[\begin{array}{cccc} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_M \end{array} \quad \mathbf{Y} \right], \qquad \theta = \left[\begin{array}{c} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_M \\ -\mathbf{a}^* \end{array} \right]. \tag{5.22}$$

Thus, the resulting parameter estimates from the least squares procedure can be classified into one of two groups: 'a' parameters and 'b' parameters.

For the purposes of retrieving information about certain characteristics of a system, it is often useful to calculate the 'impulse responses' associated with the estimated 'a' and 'b' parameters. Since these impulse responses are also estimates, confidence bounds are needed for them in order to provide an indicator of the level of error associated with them. To develop each of these concepts, we return to the chapter 2 definition of the ARX model in its Z transform format:

$$(1 + a^*(z))y(z) \;=\; a(z)y(z) \;=\; b_1(z)x_1(z) + \cdots + b_M(z)x_M(z) + e(z), \tag{5.23}$$

which is immediately restated as:

$$y(z) \;=\; \frac{b_1(z)}{a(z)}x_1(z) + \cdots + \frac{b_M(z)}{a(z)}x_M(z) + \frac{1}{a(z)}e(z). \tag{5.24}$$

If we now recall the original definition of an LTI system:

$$y(z) \;=\; h_1(z)x_1(z) + \cdots + h_M(z)x_M(z) + h_e(z)e(z), \tag{5.25}$$

we note that the ARX model defines the Z transform of its associated impulse responses as:

$$h_1(z) = \frac{b_1(z)}{a(z)}, \;\; \cdots, \;\; h_M(z) = \frac{b_M(z)}{a(z)}, \;\; h_e(z) = \frac{1}{a(z)}. \tag{5.26}$$

For our interests, we have use only for determining the impulse responses from inputs to output, i.e. $h_1(z)$ through $h_M(z)$. Upon setting $e(z)$ to zero in equation 5.25, calculation of each impulse response is performed by incrementing through each input and specifying it as:

$$x_i(z) = 1, \;\; 1 \leq i \leq M; \;\; \text{where: } \; x_j(z) = 0, \;\; \text{for } \; j \neq i, \tag{5.27}$$

which would lead to the resulting $y(z)$ sequence in each case being equivalent to $h_i(z)$.

For computational purposes, it is desirable to actually perform the above operation with the inverse transform of equation 5.23:

$$\mathbf{y}[k] = \sum_{n=1}^{p} -\mathbf{a}^*[n]\mathbf{y}[k-n] + \sum_{n=s_1}^{f_1} \mathbf{b}_1[n]\mathbf{x}_1[k-n] + \cdots + \sum_{n=s_M}^{f_M} \mathbf{b}_M[n]\mathbf{x}_M[k-n] + \mathbf{e}[k], \quad (5.28)$$

for which we set the inverse-transformed inputs, one at a time, to:

$$x_i(z) = \delta[k], \;\; 1 \leq i \leq M; \;\; \text{where: } \; x_j(z) = 0, \;\; \text{for } \; j \neq i. \quad (5.29)$$

Since the resulting **y** sequences from this operation correspond directly to the impulse responses we are seeking, we can directly write:

$$\mathbf{h}_i[k] \;=\; \sum_{n=1}^{p} -\mathbf{a}^*[n]\mathbf{h}_i[k-n] + \sum_{n=s_i}^{f_i} \mathbf{b}_i[n]\delta[k-n] \;=\; \sum_{n=1}^{p} -\mathbf{a}^*[n]\mathbf{h}_i[k-n] + \mathbf{b}_i[k]. \quad (5.30)$$

With slight manipulation, the above equation can also be expressed as:

$$\sum_{n=0}^{p} \mathbf{a}[n]\mathbf{h}_i[k-n] = \mathbf{b}_i[k]. \quad (5.31)$$

which directly relates the fact that each impulse response can be seen in terms of the 'convolution' operation:

$$\mathbf{a} * \mathbf{h}_i = \mathbf{b}_i. \quad (5.32)$$

The above results can be applied to the problem of obtaining *estimates* of the impulse responses associated with an estimated ARX model. In particular, we have geared the estimation procedure for identifying a system as one that estimates the 'a' and 'b' parameters corresponding to that system, so that equation 5.30 provides a means of 'mapping' these parameter estimates to their associated impulse responses. Designating $\hat{\mathbf{h}}_i$ as an estimated impulse response, we write:

$$\hat{\mathbf{h}}_i[k] \;=\; \sum_{n=1}^{p} -\hat{\mathbf{a}}^*[n]\hat{\mathbf{h}}_i[k-n] + \hat{\mathbf{b}}_i[k]. \quad (5.33)$$

5.4 Impulse Response Confidence Bounds

As with the parameter estimates, it is necessary to provide some indicator of error associated with the estimated impulses responses derived by the above procedure. Strictly speaking, the attainment of confidence bounds for these responses presents a very challenging, non-linear problem. However, if we allow for approximation, the procedure to be presented provides a very realistic and straightforward method of mapping 'a' and 'b' error bounds to their impulse response counterparts.

In glancing at equation 5.33, one should note that, so long as the estimates of $\hat{\mathbf{a}}^*$ are nonzero, each impulse response is infinite in duration (i.e. $\hat{\mathbf{h}}_i[k]$ extends from $k = s_i$ to ∞). However, as long as the responses are 'stable', we need only calculate $\hat{\mathbf{h}}_i$ for a finite number of values in order to obtain the information it conveys. Specifically, we will carry out the calculation of each impulse response until the l^{th} value of k, such that $s_i \leq k \leq l$. With this truncation in mind, we can use equation 5.33 to write the matrix expression:

$$\begin{bmatrix} \hat{\mathbf{h}}_i[s_i] & 0 & 0 & 0 & \cdots & 0 \\ \hat{\mathbf{h}}_i[s_i+1] & \hat{\mathbf{h}}_i[s_i] & 0 & 0 & \cdots & 0 \\ \hat{\mathbf{h}}_i[s_i+2] & \hat{\mathbf{h}}_i[s_i+1] & \hat{\mathbf{h}}_i[s_i] & 0 & \cdots & 0 \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & & \ddots & 0 \\ \hat{\mathbf{h}}_i[l] & \hat{\mathbf{h}}_i[l-1] & \hat{\mathbf{h}}_i[l-2] & \hat{\mathbf{h}}_i[l-3] & \cdots & \hat{\mathbf{h}}_i[l-(p+1)] \end{bmatrix} \begin{bmatrix} 1 \\ \hat{\mathbf{a}}^*[1] \\ \hat{\mathbf{a}}^*[2] \\ \vdots \\ \hat{\mathbf{a}}^*[p] \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{b}}_i[s_i] \\ \vdots \\ \hat{\mathbf{b}}_i[f_i] \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.34}$$

It should be easy to verify that it is equivalent to express the above operation as:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & \cdots & 0 \\ \hat{\mathbf{a}}^*[1] & 1 & 0 & 0 & \cdots & \cdots & 0 \\ \hat{\mathbf{a}}^*[2] & \hat{\mathbf{a}}^*[1] & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & & & \ddots & & & \vdots \\ \hat{\mathbf{a}}^*[p] & \hat{\mathbf{a}}^*[p-1] & \cdots & & \ddots & & \vdots \\ 0 & \hat{\mathbf{a}}^*[p] & \cdots & & & \ddots & 0 \\ \vdots & & \ddots & & \cdots & \hat{\mathbf{a}}^*[1] & 1 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{h}}_i[s_i] \\ \hat{\mathbf{h}}_i[s_i+1] \\ \hat{\mathbf{h}}_i[s_i+2] \\ \vdots \\ \hat{\mathbf{h}}_i[l] \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{b}}_i[s_i] \\ \vdots \\ \hat{\mathbf{b}}_i[f_i] \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.35}$$

For convenience, we will use the following notation to express equation 5.34:

$$\hat{\mathbf{H}}_i\hat{\mathbf{a}} = \begin{bmatrix} \hat{\mathbf{b}}_i \\ \mathbf{0} \end{bmatrix}. \tag{5.36}$$

As its counterpart, equation 5.35 will be denoted as:

$$\hat{\mathbf{A}}\hat{\mathbf{h}}_i = \begin{bmatrix} \hat{\mathbf{b}}_i \\ \mathbf{0} \end{bmatrix}, \tag{5.37}$$

so that we have the equality:

$$\hat{\mathbf{H}}_i\hat{\mathbf{a}} = \hat{\mathbf{A}}\hat{\mathbf{h}}_i. \tag{5.38}$$

The above result is consistent with the fact that the convolution $\hat{\mathbf{a}} * \hat{\mathbf{h}}$ is equivalent to $\hat{\mathbf{h}} * \hat{\mathbf{a}}$.

Proceeding now to derive the error associated with each calculated impulse response, we begin by exploring the sensitivity of $\mathbf{h}_i$ to changes in its respective 'a' and 'b' parameters. With the aid of equation 5.36, we write:

$$(\hat{\mathbf{H}}_i + \Delta\hat{\mathbf{H}}_i)(\hat{\mathbf{a}} + \Delta\hat{\mathbf{a}}) = \begin{bmatrix} \hat{\mathbf{b}}_i + \Delta\hat{\mathbf{b}}_i \\ \mathbf{0} \end{bmatrix}, \tag{5.39}$$

where we have defined:

$$\Delta\hat{\mathbf{H}}_i = \begin{bmatrix} \Delta\hat{\mathbf{h}}_i[s_i] & 0 & 0 & 0 & \cdots & 0 \\ \Delta\hat{\mathbf{h}}_i[s_i+1] & \Delta\hat{\mathbf{h}}_i[s_i] & 0 & 0 & \cdots & 0 \\ \Delta\hat{\mathbf{h}}_i[s_i+2] & \Delta\hat{\mathbf{h}}_i[s_i+1] & \Delta\hat{\mathbf{h}}_i[s_i] & 0 & \cdots & 0 \\ \vdots & & & \ddots & & \vdots \\ \vdots & & & & \ddots & 0 \\ \Delta\hat{\mathbf{h}}_i[l] & \Delta\hat{\mathbf{h}}_i[l-1] & \Delta\hat{\mathbf{h}}_i[l-2] & \Delta\hat{\mathbf{h}}_i[l-3] & \cdots & \Delta\hat{\mathbf{h}}_i[l-(p+1)] \end{bmatrix}, \tag{5.40}$$

and:

$$\Delta\hat{\mathbf{a}} = \begin{bmatrix} 0 \\ \Delta\hat{\mathbf{a}}^*[1] \\ \Delta\hat{\mathbf{a}}^*[2] \\ \vdots \\ \Delta\hat{\mathbf{a}}^*[p] \end{bmatrix}. \tag{5.41}$$

Expanding the left side of equation 5.39 leads to:

$$\hat{\mathbf{H}}_i\hat{\mathbf{a}} + \Delta\hat{\mathbf{H}}_i\hat{\mathbf{a}} + \hat{\mathbf{H}}_i\Delta\hat{\mathbf{a}} + \Delta\hat{\mathbf{H}}_i\Delta\hat{\mathbf{a}} = \begin{bmatrix} \hat{\mathbf{b}}_i + \Delta\hat{\mathbf{b}}_i \\ \mathbf{0} \end{bmatrix}. \tag{5.42}$$

By equation 5.38, we note the fact that we can equivalently express $\Delta\hat{\mathbf{H}}_i\hat{\mathbf{a}}$ as $\hat{\mathbf{A}}\Delta\hat{\mathbf{h}}_i$ so that the above expression is rearranged as:

$$\hat{\mathbf{H}}_i\hat{\mathbf{a}} - \begin{bmatrix} \hat{\mathbf{b}}_i \\ \mathbf{0} \end{bmatrix} + \hat{\mathbf{A}}\Delta\hat{\mathbf{h}}_i + \hat{\mathbf{H}}_i\Delta\hat{\mathbf{a}} + \Delta\hat{\mathbf{H}}_i\Delta\hat{\mathbf{a}} = \begin{bmatrix} \Delta\hat{\mathbf{b}}_i \\ \mathbf{0} \end{bmatrix}. \tag{5.43}$$

Cancellation of the leftmost items in the above equation then allows us to write:

$$\hat{\mathbf{A}}\Delta\hat{\mathbf{h}}_i = \begin{bmatrix} \Delta\hat{\mathbf{b}}_i \\ \mathbf{0} \end{bmatrix} + (\hat{\mathbf{H}}_i + \Delta\hat{\mathbf{H}}_i)(-\Delta\hat{\mathbf{a}}). \tag{5.44}$$

One should note that, since the topmost entry of $\Delta\hat{\mathbf{a}}$ is zero by definition, we can restate the above equation as:

$$\hat{\mathbf{A}}\Delta\hat{\mathbf{h}}_i = \begin{bmatrix} \Delta\hat{\mathbf{b}}_i \\ \mathbf{0} \end{bmatrix} + (\hat{\mathbf{H}}_i^* + \Delta\hat{\mathbf{H}}_i^*)(-\Delta\hat{\mathbf{a}}^*), \tag{5.45}$$

where $\hat{\mathbf{H}}_i^*$ denotes the matrix that results from removing the leftmost column from $\hat{\mathbf{H}}_i$, and $\Delta\hat{\mathbf{H}}_i^*$ denotes the matrix that occurs upon removal of the rightmost column within $\Delta\hat{\mathbf{H}}_i$. It is useful to cast this last equation into a matrix format, at the same time premultiplying by the inverse of $\hat{\mathbf{A}}$:

$$\Delta\hat{\mathbf{h}}_i = \hat{\mathbf{A}}^{-1}\begin{bmatrix} \mathbf{I} & \vdots & \hat{\mathbf{H}}_i^* + \Delta\hat{\mathbf{H}}_i^* \\ \mathbf{0} & & \end{bmatrix}\begin{bmatrix} \Delta\hat{\mathbf{b}}_i \\ \cdots \\ -\Delta\hat{\mathbf{a}}^* \end{bmatrix}. \tag{5.46}$$

One should notice that solving for $\Delta\hat{\mathbf{h}}_i$ in the above equation is a nonlinear problem. However, if we specified $\Delta\hat{\mathbf{b}}_i$ and $\Delta\hat{\mathbf{a}}$ as taking on arbitrarily small magnitudes, we would expect that $\Delta\hat{\mathbf{h}}_i$ could be made small enough such that $\hat{\mathbf{H}}_i^* + \Delta\hat{\mathbf{H}}_i^* \approx \hat{\mathbf{H}}_i^*$. Thus, for small errors in the parameter estimates, we could write:

$$\Delta\hat{\mathbf{h}}_i \approx \hat{\mathbf{A}}^{-1}\begin{bmatrix} \mathbf{I} & \vdots & \hat{\mathbf{H}}_i^* \\ \mathbf{0} & & \end{bmatrix}\begin{bmatrix} \Delta\hat{\mathbf{b}}_i \\ \cdots \\ -\Delta\hat{\mathbf{a}}^* \end{bmatrix}, \tag{5.47}$$

which effectively provides a *linearization* of the sensitivity of $\hat{\mathbf{h}}_i$ to parameter changes about the operating point set by $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}_i$.

We now expand the above results to encompass the errors associated with all of the relevant impulse responses. Proceeding, we first rewrite equation 5.47 as:

$$\Delta\hat{\mathbf{h}}_i \approx \begin{bmatrix} \hat{\mathbf{A}}_*^{-1} & \vdots & \hat{\mathbf{A}}^{-1}\hat{\mathbf{H}}_i^* \end{bmatrix}\begin{bmatrix} \Delta\hat{\mathbf{b}}_i \\ \cdots \\ -\Delta\hat{\mathbf{a}}^* \end{bmatrix}, \tag{5.48}$$

where we have defined:

$$\hat{\mathbf{A}}_*^{-1} = \hat{\mathbf{A}}^{-1}\begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \tag{5.49}$$

Now, the consideration of each impulse response leads to the following equations:

$$\begin{aligned}
\Delta\hat{\mathbf{h}}_1 &\approx \left[\begin{array}{c:c} \hat{\mathbf{A}}_*^{-1} & \hat{\mathbf{A}}^{-1}\hat{\mathbf{H}}_1^* \end{array}\right] \left[\begin{array}{c} \Delta\hat{\mathbf{b}}_1 \\ \cdots \\ -\Delta\hat{\mathbf{a}}^* \end{array}\right], \\
\Delta\hat{\mathbf{h}}_2 &\approx \left[\begin{array}{c:c} \hat{\mathbf{A}}_*^{-1} & \hat{\mathbf{A}}^{-1}\hat{\mathbf{H}}_2^* \end{array}\right] \left[\begin{array}{c} \Delta\hat{\mathbf{b}}_2 \\ \cdots \\ -\Delta\hat{\mathbf{a}}^* \end{array}\right], \\
&\vdots \\
\Delta\hat{\mathbf{h}}_M &\approx \left[\begin{array}{c:c} \hat{\mathbf{A}}_*^{-1} & \hat{\mathbf{A}}^{-1}\hat{\mathbf{H}}_M^* \end{array}\right] \left[\begin{array}{c} \Delta\hat{\mathbf{b}}_M \\ \cdots \\ -\Delta\hat{\mathbf{a}}^* \end{array}\right],
\end{aligned} \tag{5.50}$$

which can be combined into a single matrix expression as follows:

$$\begin{bmatrix} \Delta\hat{\mathbf{h}}_1 \\ \Delta\hat{\mathbf{h}}_2 \\ \vdots \\ \Delta\hat{\mathbf{h}}_M \end{bmatrix} \approx \begin{bmatrix} \hat{\mathbf{A}}_*^{-1} & 0 & \cdots & 0 & \hat{\mathbf{A}}^{-1}\hat{\mathbf{H}}_1^* \\ 0 & \hat{\mathbf{A}}_*^{-1} & \cdots & 0 & \hat{\mathbf{A}}^{-1}\hat{\mathbf{H}}_2^* \\ 0 & 0 & \ddots & & \vdots \\ 0 & 0 & \cdots & \hat{\mathbf{A}}_*^{-1} & \hat{\mathbf{A}}^{-1}\hat{\mathbf{H}}_M^* \end{bmatrix} \begin{bmatrix} \Delta\hat{\mathbf{b}}_1 \\ \Delta\hat{\mathbf{b}}_2 \\ \vdots \\ \Delta\hat{\mathbf{b}}_M \\ -\Delta\hat{\mathbf{a}}^* \end{bmatrix}. \tag{5.51}$$

We note that the rightmost vector in the above equation directly corresponds to $\Delta\hat{\boldsymbol{\theta}}$.

As one of the final steps toward calculating the error associated with each impulse response estimate, we define the following quantities from the last written equation:

$$\Delta\hat{\mathbf{h}}_{tot} = \begin{bmatrix} \Delta\hat{\mathbf{h}}_1 \\ \Delta\hat{\mathbf{h}}_2 \\ \vdots \\ \Delta\hat{\mathbf{h}}_M \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} \hat{\mathbf{A}}_*^{-1} & 0 & \cdots & 0 & \hat{\mathbf{A}}^{-1}\hat{\mathbf{H}}_1^* \\ 0 & \hat{\mathbf{A}}_*^{-1} & \cdots & 0 & \hat{\mathbf{A}}^{-1}\hat{\mathbf{H}}_2^* \\ 0 & 0 & \ddots & & \vdots \\ 0 & 0 & \cdots & \hat{\mathbf{A}}_*^{-1} & \hat{\mathbf{A}}^{-1}\hat{\mathbf{H}}_M^* \end{bmatrix}, \quad \Delta\hat{\boldsymbol{\theta}} = \begin{bmatrix} \Delta\hat{\mathbf{b}}_1 \\ \Delta\hat{\mathbf{b}}_2 \\ \vdots \\ \Delta\hat{\mathbf{b}}_M \\ -\Delta\hat{\mathbf{a}}^* \end{bmatrix}. \tag{5.52}$$

Using these definitions, equation 5.51 is rewritten as:

$$\Delta\hat{\mathbf{h}}_{tot} \approx \mathbf{D}\Delta\hat{\boldsymbol{\theta}} = \mathbf{DCe}, \tag{5.53}$$

in which we have also made use of equation 5.6. For convenience, if we now define $\mathbf{F} = \mathbf{DC}$, then we note the approximation

$$\Delta\hat{\mathbf{h}}_{tot} \approx \mathbf{Fe}, \tag{5.54}$$

which indicates, in an approximate sense, that the elements of each impulse response error are simply formed as the sum of scaled elements of the noise vector **e**. Thus, we could now calculate the approximate density function and confidence bounds associated with each response error in complete analogy to the procedure followed for the parameter estimates.

Chapter 6

Verification Using Simulated Data

6.1 Introduction

The focus of this chapter is to present simulation results that will provide verification of the APR algorithm's effectiveness in estimating the impulse response associated with different simulated models. One should note that these simulations will be mimicking ideal conditions (i.e. inputs that are white and noise that is white and gaussian), which is somewhat unrealistic of actual experimental data that the algorithm will normally be applied to. However, we feel that it is important to show that the algorithm does work under the conditions it was designed for. The results will simply be displayed as a series of figures, so that our preliminary task will be to explain the chosen notation.

6.2 Simulation Description

6.2.1 True Model

The simulations to be presented are all based on a two input simulation model that is illustrated in Figure 6-1. The nomenclature associated with the different permutations resulting from this model are discussed in [5]. In every case, the inputs x_1 and x_2 will be created as white noise sequences of unit variance that are uncorrelated (on average) with one another. The noise sequence, **e**, will always be white, gaussian, and uncorrelated (on average) with each of the inputs. The amount of variance in the noise is directly associated with the level of corruption that will take place in the estimation procedure. To provide an indication of the corruption level to be expected in the simulation results, we will form

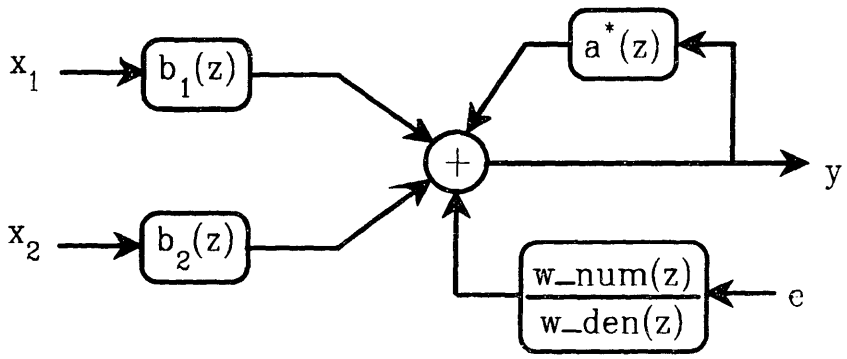


Figure 6-1: A general, two input, LTI simulation model

a ratio that divides the variance of the output that would occur upon the noise sequence being set to zero by the variance of the output that would occur if both of the inputs were set to zero. We will denote this ratio as 's_n' (signal to noise ratio).

One should note that the above model is not actually of the ARX stucture. In particular, the noise sequence goes through a 'shaping filter' that has the transfer function:

$$h_e(z) = \frac{w_{num}(z)}{w_{den}(z)}. \tag{6.1}$$

This flexibility in model structure has been added so that it can be observed how the ARX model can often be used to *approximate* the behavior of systems that do not fall directly into its format. One should refer back to chapter 2 in order to observe the theoretical details of this approximation.

6.2.2 Estimated Model

The APR algorithm is focused on representing the behavior of an investigated system with an ARX model. Thus, the estimated model will always have the structure indicated in Figure 6-2. One should note that the important result to fall out of the estimated parameters in this model is the impulse response they form. In other words, as we introduce different types of noise dynamics in our simulations (i.e. alter $h_e(z)$), one will notice that the

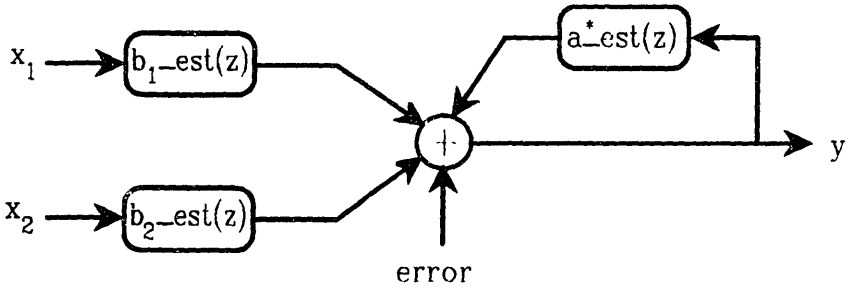


Figure 6-2: The two input, ARX estimation model

estimated ARX parameterization will be dramatically influenced by these noise dynamics. However, the estimated impulse responses that fall out from these parameterizations are what actually characterizes the system, so that the true test of doing a good identification rests on obtaining impulse response estimates that are close in value to the actual responses.

6.3 Results

6.3.1 Discussion of Notation

The following pages will each show two results for every simulated model considered. The first result will correspond to performing a standard least squares procedure on the data set with the maximal model being specified as the ARX parameterization. The second result will display the estimates obtained in running the APR algorithm on the maximal model at a 0.95 level of confidence. To explain the notation used throughout the remaining pages, we will describe Figure 6-3 in detail.

Glancing at Figure 6-3, the left columns specify that we formed a simulation model with the following parameterization:

$$b_1(z) = 10 - 16.4z^{-1} + 8.29z^{-2} - 3.2z^{-4},$$

$$b_2(z) = 8z^{-2} - 6.54z^{-4},$$

$$a(z) = 1.0 - 1.77z^{-1} + 1.45z^{-2} - 0.55z^{-3}.$$

In addition, the noise shaping filter was specified as:

$$h_e(z) = \frac{1}{1} = 1,$$

so that this first simulation model was in the ARX format. The level of noise variance was chosen such that the s_n ratio was equal to 3.0 — this implies that the portion of **y** formed by the inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ had 3 times the value of variance of the portion of **y** formed by the noise sequence.

The right columns specify that the maximal model for the estimation procedure was formed as:

$$\hat{b_1}(z) = \hat{b_1}[0] + \hat{b_1}[1]z^{-1} + \cdots + \hat{b_1}[9]z^{-9},$$

$$\hat{b_2}(z) = \hat{b_2}[0] + \hat{b_2}[1]z^{-1} + \cdots + \hat{b_2}[9]z^{-9},$$

$$\hat{a}(z) = 1 + \hat{a}^*[1]z^{-1} + \cdots + \hat{a}^*[10]z^{-10}$$

with the corresponding estimates for the least squares procedure being shown at the top of the page, and the APR algorithm estimates being shown at the bottom of the page.

Estimates of the impulse responses are plotted below their associated parameter estimates. The leftmost plot corresponds to the impulse response from $\mathbf{x}_1$ to **y** (formed by the 'b_1' and 'a' parameters), while the rightmost plot corresponds to the response from $\mathbf{x}_2$ to **y** (formed by the 'b_2' and 'a' parameters). The solid portion of each plot corresponds to the actual impulse response. The dashed portion corresponds to the estimated impulse response, with dotted lines designating the 95% confidence boundary about this estimate.

6.3.2 Discussion of Individual Plots

The following points should be observed about each of the following figures:

- Figure 6-3: The APR algorithm successfully identifies the 'true' parameters of the simulated system, at a s_n ratio of 3.

- Figure 6-4: The APR algorithm incorrectly removes several 'true' parameters, resulting in an estimate with lowered confidence bounds but inaccurate results. It should be pointed out that the s_n ratio in this case was extremely low (s_n = 0.4), allowing the algorithm to justify the change in residual error that occurred upon removal of the true parameters as being attributable to noise. To guard against such a situation occurring in an actual estimation problem, the user should compare the estimated maximal model impulse response with that found by the APR algorithm. If the shape of the response changes dramatically, the algorithm has probably removed a true parameter. In such case, the user is recommended to lower the confidence level on the APR algorithm until reasonable results are achieved.

- Figure 6-5: The APR algorithm correctly determines the parameterization for a simulated model that has all of its 'a' parameters set to zero.

- Figure 6-6: The APR algorithm correctly determines the parameterization for a simulated model that all of its 'b' parameters set to zero.

- Figure 6-7: The APR algorithm determines a parameterization that leads to estimated impulse responses that closely resemble the actual responses for a simulated model with a nontrivial noise shaping filter . In glancing at the resulting estimates, one might think that the algorithm came up with an overparameterized model. However, if we write out the simulated model description:

$$\mathbf{y}(z) = \frac{b_1(z)}{a(z)}\mathbf{x}_1(z) + \frac{b_2(z)}{a(z)}\mathbf{x}_2(z) + \frac{1}{a(z)}\frac{1}{w_{den}(z)}\mathbf{e}(z),$$

then one should realize that the ARX representation of this model is:

$$\mathbf{y}(z) = \frac{b_1(z)w_{den}(z)}{a(z)w_{den}(z)}\mathbf{x}_1(z) + \frac{b_2(z)w_{den}(z)}{a(z)w_{den}(z)}\mathbf{x}_2(z) + \frac{1}{a(z)w_{den}(z)}\mathbf{e}(z).$$

Thus, the noise shaping filter effectively increases the order of the 'a' and 'b_i' polynomials, but the impulse response estimates are still quite good.

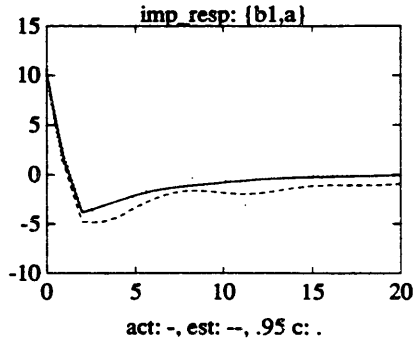
- Figure 6-8: Again, the APR algorithm determines a parameterization that leads to close fitting impulse response estimates for a simulation model with nontrivial noise dynamics.

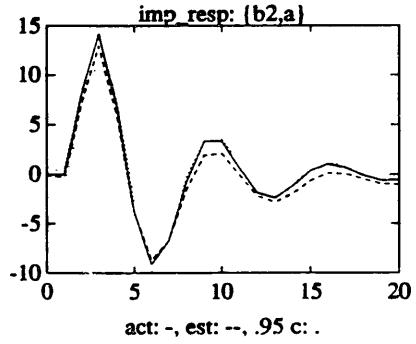
- Figure 6-9: The APR algorithm correcly identifies the fact that the given output sequence is *not* dependent on the given input sequences.

6.3.3 Simulation Estimates

ARX Model (least_squares)

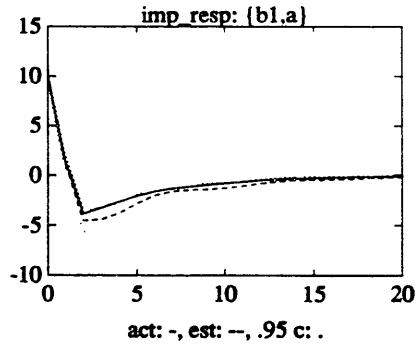
a	b1	b2	a_est	b1_est	b2_est
1	10	0	1	9.79	-0.11
-1.77	-16.4	0	-1.79	-16.6	-0.08
1.45	8.29	8	1.56	8.82	7.36
-0.55	0	0	-0.66	-1.26	-0.05
	-3.2	-6.54	-0.05	-4.19	-6.05
			0.2	1.98	0.89
			-0.23	-1.98	-1.35
w_num	w_den	s_n	0.07	0.09	-0.23
1	1	3	0.02	0.63	0.09
			-0.04	-0.41	-1.15
			0.01		





ARX Model (APR at .95 conf)

a	b1	b2	a_est	b1_est	b2_est
1	10	0	1	9.91	0
-1.77	-16.4	0	-1.79	-16.6	0
1.45	8.29	8	1.43	7.65	7.36
-0.55	0	0	-0.54	0	0
	-3.2	-6.54	0	-2.97	-6.88
			0	0	0
			0	0	0
w_num	w_den	s_n	0	0	0
1	1	3	0	0	0
			0	0	0
			0		



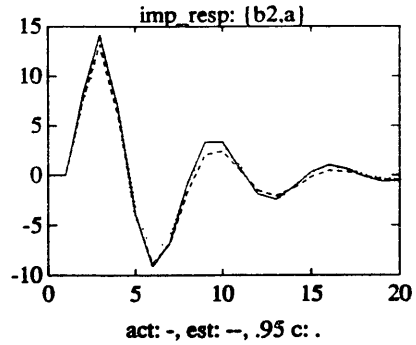
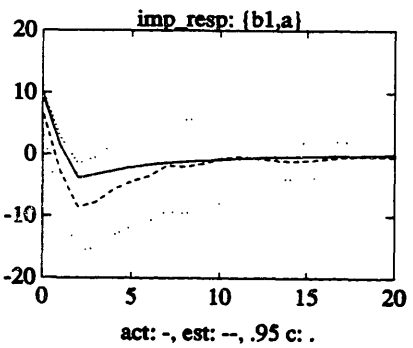
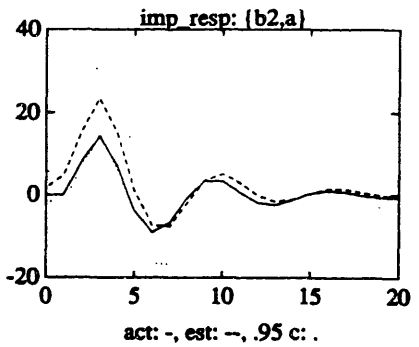


Figure 6-3: Successful simulation results for an ARX model

ARX Model (least_squares)

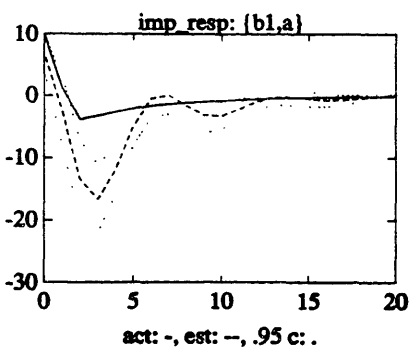
a	b1	b2	a_est	b1_est	b2_est
1	10	0	1	7.12	1.61
-1.77	-16.4	0	-1.78	-15.7	1.91
1.45	8.29	8	1.51	7.52	8.95
-0.55	0	0	-0.69	-1.93	2.74
	-3.2	-6.54	0.15	-1.66	-7.13
			-0.03	-0.66	-0.18
			-0.1	-0.79	-0.94
w_num	w_den	s_n	0.1	1.87	-0.27
1	1	0.4	-0.05	-1.48	-0.14
			0.01	1.46	-0.56
			-0.01		





ARX Model (APR at .95 conf)

a	b1	b2	a_est	b1_est	b2_est
1	10	0	1	6.96	0
-1.77	-16.4	0	-1.73	-14.4	0
1.45	8.29	8	1.34	0	8.73
-0.55	0	0	-0.48	0	0
	-3.2	-6.54	0	0	-8.48
			0	0	0
			0	0	0
w_num	w_den	s_n	0	0	0
1	1	0.4	0	0	0
			0	0	0
			0		



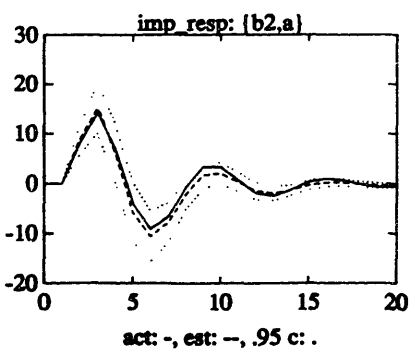
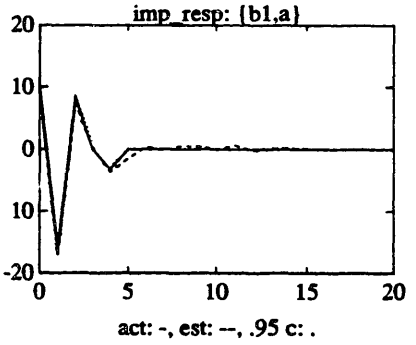
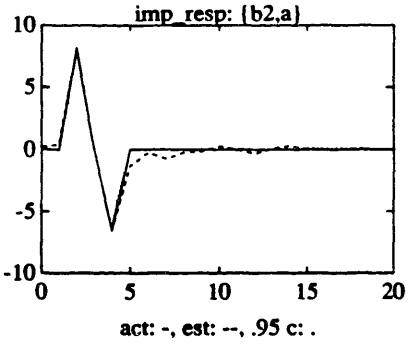


Figure 6-4: Unsuccessful simulation results for an ARX model due to low s_n ratio

X Model (least_squares)

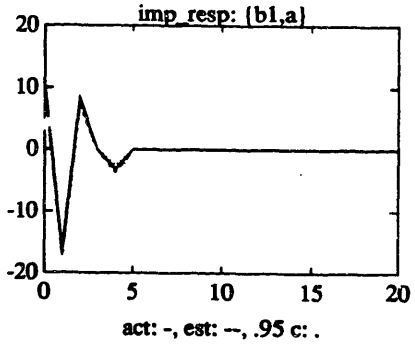
a	b1	b2	a_est	b1_est	b2_est
1	10	0	1	9.7	0.15
	-16.4	0	0	-17.2	0.42
	8.29	8	0	7.34	7.92
	0	0	-0.02	-0.04	0.05
	-3.2	-6.54	-0.03	-3.43	-6.52
			0	-1.21	-1.52
			0.04	0.54	-0.47
w_num	w_den	s_n	0	-0.68	-0.59
1	1	3	0	0.82	0.27
			0.01	0.66	-0.08
			0.04		





X Model (APR at .95 conf)

a	b1	b2	a_est	b1_est	b2_est
1	10	0	1	9.92	0
	-16.4	0	0	-17.1	0
	8.29	8	0	7.34	7.95
	0	0	0	0	0
	-3.2	-6.54	0	-3.66	-6.72
			0	0	0
			0	0	0
w_num	w_den	s_n	0	0	0
1	1	3	0	0	0
			0		



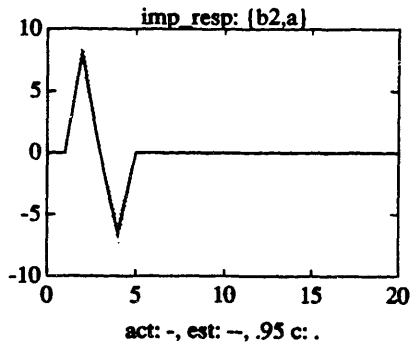
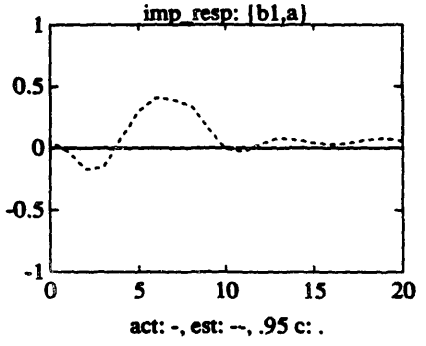
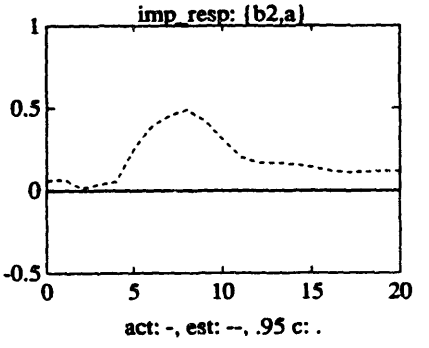


Figure 6-5: Successful simulation results for an X model (all 'a' parameters set to zero)

AR Model (least_squares)

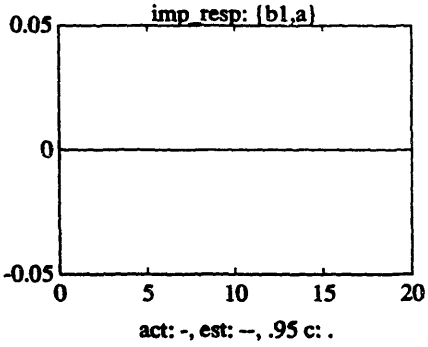
a	b1	b2	a_est	b1_est	b2_est
1	0	0	1	0.05	0.06
-1.77			-1.93	-0.13	-0.05
1.45			1.88	-0.01	-0.01
-0.55			-1.1	0.06	0.07
			0.37	0.11	-0.04
			-0.22	0.03	0.21
			0.25	0.09	-0.02
			-0.36	0.04	0.11
w_num	w_den	s_n	0.41	0.07	0.09
1	1	0	-0.28	-0.13	-0.01
			0.07		





AR Model (APR at .95 conf)

a	b1	b2	a_est	b1_est	b2_est
1	0	0	1	0	0
-1.77			-1.82	0	0
1.45			1.56	0	0
-0.55			-0.66	0	0
			0	0	0
			0	0	0
			0	0	0
			0	0	0
w_num	w_den	s_n	0	0	0
1	1	0	0	0	0
			0		



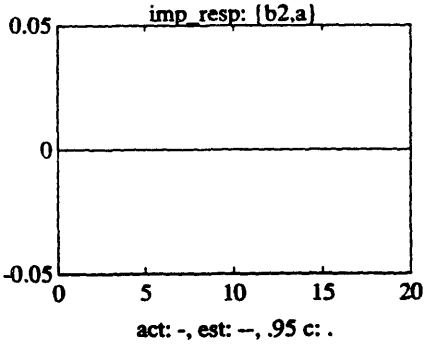
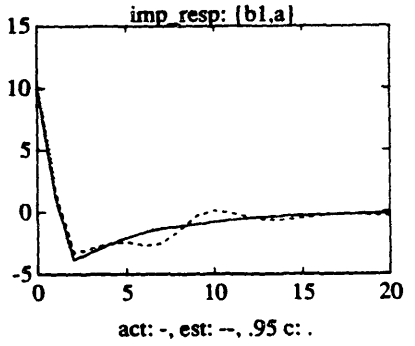
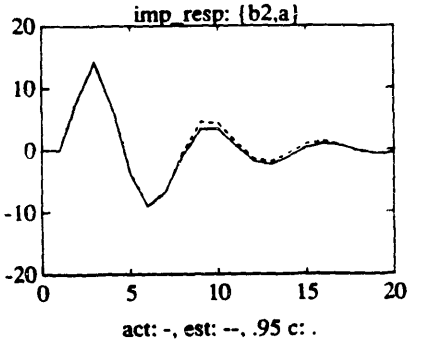


Figure 6-6: Successful simulation results for an AR model (all 'b' parameters set to zero)

ARARX Model (least_squares)

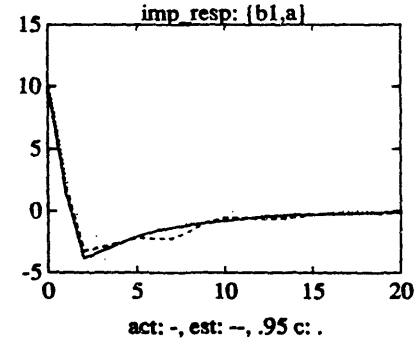
a	b1	b2	a_est	b1_est	b2_est
1	10	0	1	10.1	0.11
-1.77	-16.4	0	-2	-18.4	-0.32
1.45	8.29	8	2.3	16.3	8.25
-0.55	0	0	-1.65	-8.94	-2.12
	-3.2	-6.54	0.69	-0.26	-2.71
			-0.18	0.84	1.44
			0.01	-1.47	-3.5
w_num	w_den	s_n	-0.02	0.03	-0.25
1	1	3	-0.04	-0.37	0.85
	-0.25		0.07	0.75	0.45
	0.5		-0.04		





ARARX Model (APR at .95 conf)

a	b1	b2	a_est	b1_est	b2_est
1	10	0	1	10.1	0
-1.77	-16.4	0	-1.78	-16.2	0
1.45	8.29	8	1.9	12.7	8.25
-0.55	0	0	-1.3	-6.76	0
	-3.2	-6.54	0.58	0	-2.85
			-0.21	0	0
			0	-1.74	-3.02
w_num	w_den	s_n	0	0	0
1	1	3	0	0	0
	-0.25		0	0	0
	0.5		0		



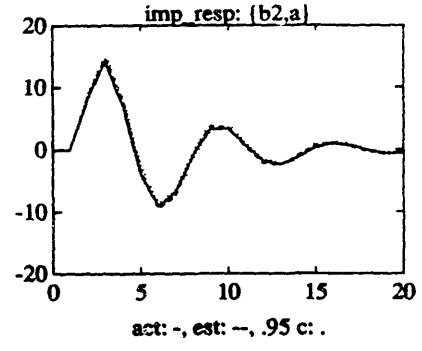
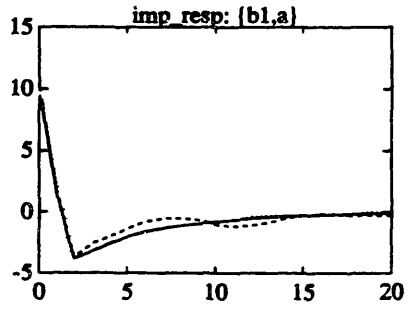
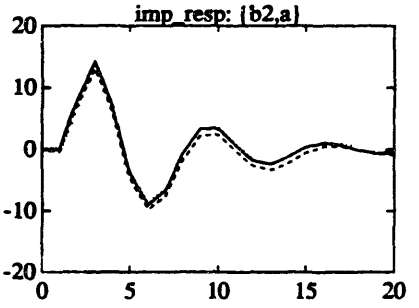


Figure 6-7: Successful simulation results for an ARARX model ($w_{den}(z)$ is nontrivial)

ARMAX Model (least_squares)

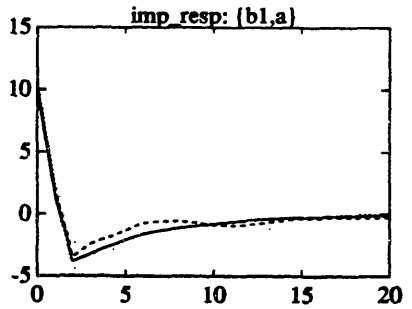
a	b1	b2	a_est	b1_est	b2_est
1	10	0	1	10	-0.09
-1.77	-16.4	0	-1.42	-12.7	-0.33
1.45	8.29	8	0.43	-1.62	7.78
-0.55	0	0	0.39	7.23	2.68
	-3.2	-6.54	-0.07	-0.06	-9.64
			-0.27	-3.84	-4.82
			-0.12	-2.01	4.19
			0.36	3.84	3.87
w_num	w_den	s_n	-0.2	-0.83	-2.16
1	1	3	0.01	-1.09	-2.19
-0.25			0.01		
0.5					





ARMAX Model (APR at .95 conf)

a	b1	b2	a_est	b1_est	b2_est
1	10	0	1	10.1	0
-1.77	-16.4	0	-1.5	-13.5	0
1.45	8.29	8	0.6	0	7.86
-0.55	0	0	0.26	6.54	2.04
	-3.2	-6.54	-0.01	0	-9.47
			-0.41	-5.04	-4.01
			0.12	0	4.5
			0.17	2.64	2.79
w_num	w_den	s_n	-0.13	-0.91	-2.37
1	1	3	0	-0.7	-1.64
-0.25			0		
0.5					



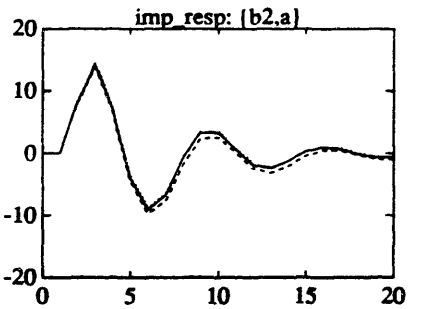
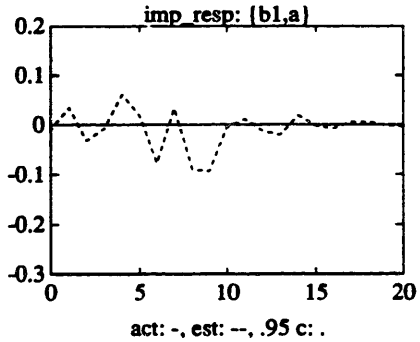
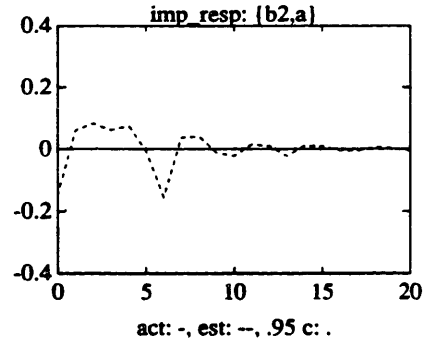


Figure 6-8: Successful simulation results for an ARMAX model ($w_{num}(z)$ is nontrivial)

no Model (least_squares)

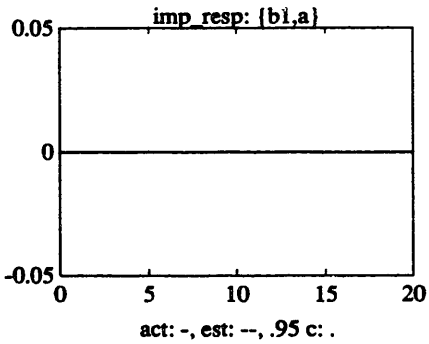
a	b1	b2	a_est	b1_est	b2_est
1	0	0	1	-0.01	-0.15
			0.01	0.03	0.06
			0.05	-0.03	0.08
			-0.06	-0.01	0.07
			-0.11	0.06	0.09
			0.02	0.02	-0.02
			0.07	-0.07	-0.17
w_num	w_den	s_n	-0.1	0.03	0.04
1	1	0	0.07	-0.11	0.02
			0.01	-0.08	0
			-0.01		





no Model (APR at .95 conf)

a	b1	b2	a_est	b1_est	b2_est
1	0	0	1	0	0
			0	0	0
			0	0	0
			0	0	0
			0	0	0
			0	0	0
			0	0	0
w_num	w_den	s_n	0	0	0
1	1	0	0	0	0
			0		



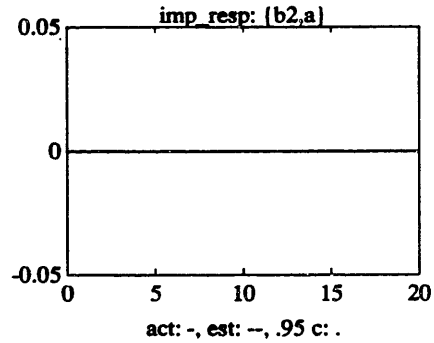


Figure 6-9: Successful simulation results with no model between inputs and output

Chapter 7

Verification using Experimental Data — Application to Autonomic Heart Rate Variability

7.1 Introduction

This chapter is geared toward demonstrating the effectiveness of the APR algorithm in performing transfer function analysis with experimental data. In particular, we will focus on the algorithm's ability to remove 'insignificant' detail from the estimated impulse responses obtained for an overly specified maximal model. Accompanying this removal of unnecessary detail, a significant 'tightening' of the confidence bounds associated with the estimated impulse responses will be observed.

The object of our experimental verification will be the identification of the influence of short term variations in respiration and arterial blood pressure on heart rate variability in two selected experimental data records. A thorough development of the relationship between the examined physiological variables is beyond the scope of this thesis, but we will attempt to provide fundamental background information so that the reader gains a basic handle on the primary issues involved in performing system identification.

7.2 Physiological Background

Under normal conditions, the autonomic nervous system plays the dominant role in regulatory mediation of heart rate. Experiments using pharmacologic blockade of the sympathetic and parasympathetic inputs to the sinoatrial node have demonstrated that virtually all heart rate (HR) fluctuations > 0.03 Hz are caused by changing levels of the efferent activity of these inputs, and that each of the autonomic branches mediate these fluctuations in different frequency bands. Specifically, HR fluctuations at frequencies > 0.15 Hz are mediated solely by changing levels of vagal activity, whereas fluctuations < 0.15 Hz can be affected by changing levels of *both* cardiac vagal and sympathetic activity. HR fluctuations < 0.03 Hz may be also mediated by changing plasma levels of neurohormones. [8]

With the autonomic nervous system taking on the central role in the *mediation* of HR variability, it naturally forms the primary pathway by which other physiological variables gain an influence on heart rate. In particular, everything from respiration (ILV — instantaneous lung volume) to changes in arterial blood pressure (ABP) to extreme emotions can have a dramatic influence on the pace at which the heart pumps. In a very real sense, these physiological signals can be thought of as inputs into a system (corresponding to the autonomic nervous system) that has as its output HR. Therefore, to gain information about this system, one can imagine taking several of its observable inputs, along with the output HR signal, and performing a *system identification* procedure. In the context of this thesis, respiration and arterial blood pressure will be considered as such inputs, so that we will seek to quantify their influence on HR variability.

Taking a closer look at the first of our chosen autonomic inputs, an inverse relationship between arterial blood pressure and heart rate was first described in 1859 by Marey and is often referred to as Marey's law of the heart. It was demonstrated subsequently that the alterations in heart rate evoked by changes in blood pressure depended on baroreceptors located in the aortic arch and carotid sinuses. It has generally been assumed that the changes in heart rate that occur in response to alterations in baroreceptor stimulation are, in fact, mediated by reciprocal changes in activity in the two autonomic divisions. Recent investigations have confirmed that reciprocal reflex changes in sympathetic and vagal activity do occur for small deviations in blood pressure within the normal range of pressure. [3]

The variations in heart rate that occur with respiratory activity, known as the respiratory sinus arrhythmia, have been widely investigated as both a biophysical phenomenon and a window on autonomic regulation. It has been observed that both vagal and cardiac sympathetic efferent activity are modulated with respiration, and that the response of the rate of the sinoatrial pacemaker (i.e. heart rate) as a function of frequency differs markedly for the two autonomic divisions. [9]

7.3 System Modeling

In attempting to quantify the role of the autonomic nervous system in heart rate variability, a linear, time-invariant, closed loop model has been proposed by researchers in our lab as a possible representation of this system's behavior. Although the cardiovascular system may demonstrate nonlinear behavior over large changes in state, our hypothesis is that, for relatively small fluctuations of cardiovascular signals about their mean, the signals of interest are *linearly* coupled. Thus, our proposal is that respiration, blood pressure, and heart rate can be considered to be linearly related about some given operating point. In addition, for the duration of the data segments used in our analysis, we have assumed that the autonomic system is time invariant, so that the observed data records exhibit the property of having *stationary* characteristics. A diagram of the proposed model is shown in Figure 7-1.

Although the coupling between respiration, blood pressure, and heart rate effectively corresponds to a closed loop system, we will seek only to characterize the effect of respiration and blood pressure on heart rate. To achieve this task, we will focus on the identification of a subset of the system blocks shown Figure 7-1. In particular, we will be concerned with the relationship depicted in Figure 7-2, in which we have ignored the effects of respiration and heart rate on blood pressure, focusing only on the influence that ILV and ABP have on HR.

As the APR algorithm will be used to identify the transfer functions depicted in Figure 7-2, we will assume that the system indicated in that figure can be restructured into the ARX format shown in Figure 7-3. In terms of the ARX polynomials, we would therefore express

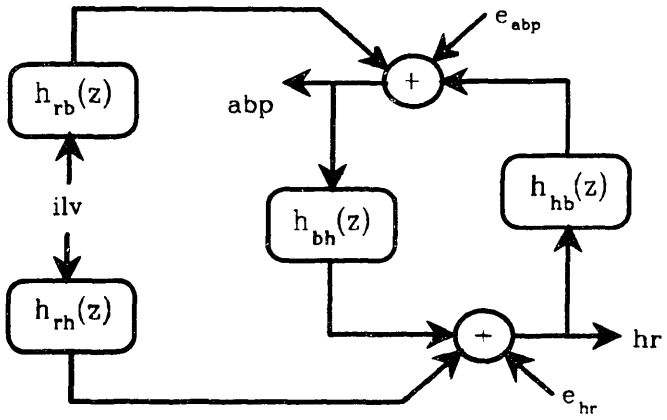


Figure 7-1: Proposed closed loop, LTI model linking variations in respiration (ILV), arterial blood pressure (ABP), and heart rate (HR)

the two transfer functions as:

$$h_{rh}(z) = \frac{b_{rp}(z)}{1 + a^*(z)}, \quad h_{bh}(z) = \frac{b_{bp}(z)}{1 + a^*(z)}, \tag{7.1}$$

and correspondingly describe the HR noise process as:

$$e_{hr}(z) = \frac{1}{1 + a^*(z)} e(z). \tag{7.2}$$

7.4 Experiment

7.4.1 Objective

In order to obtain records of ABP, ILV, and HR in normal subjects for identification purposes, a study was conducted by Saul et al [10]. The experimental procedure, as described below, incorporated a breathing protocol that has been explained by Berger et al [2]. Essentially, this procedure provided a means by which relevant transfer functions could be simultaneously estimated over a broad range of frequencies, as required by the originally intended analysis technique — spectral estimation. The APR algorithm *implicitly* attempts also to simultaneously identify the value of these transfer functions over a broad range of frequencies, so that the indicated breathing protocol was necessary in order that the

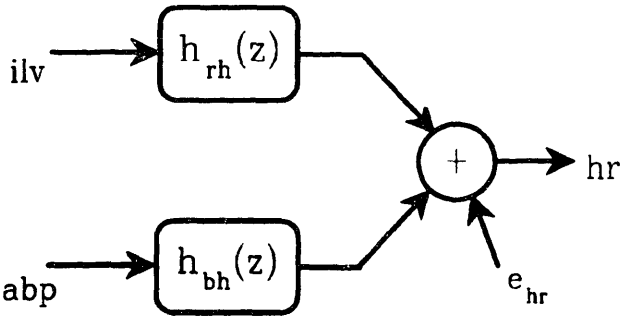


Figure 7-2: Proposed two input, LTI model representing the dependence of heart rate (HR) variations on respiration (ILV) and arterial blood pressure (ABP) changes

algorithm work properly.

7.4.2 Procedure

Fourteen male, non-smoking, adult volunteers (ages 19 to 38 years, median — 21 years) participated in a study conducted by Saul et al [10]. The subjects were instructed to initiate a breath with each tone of a series of auditory cues. The sequence of tones was generated by a computer and recorded on cassette tape for playback during each experiment. The computer was programmed to initially space the tones evenly in time at a preset rate of 12 breaths/minute to allow the subject to find a comfortable depth of inspiration. The program later switched to a mode in which the tones were spaced at irregular intervals, but maintained the same mean rate of 12 breaths/minute. The respiratory intervals in the latter mode were chosen according to a modified Poisson process since the energy of a sequence of Poisson impulses is constant over all frequencies (in a bandlimited sense). Thus, the power spectrum of the respiratory drive chain was significantly broadened using this procedure, which effectively provided data that could be used for transfer function identification over a wide range of frequencies. Minimum and maximum intervals of 1 and 15 seconds were chosen to avoid discomfort. Although the respiratory intervals were controlled, the subjects were allowed to comfortably determine the depth and shape of each breath throughout the

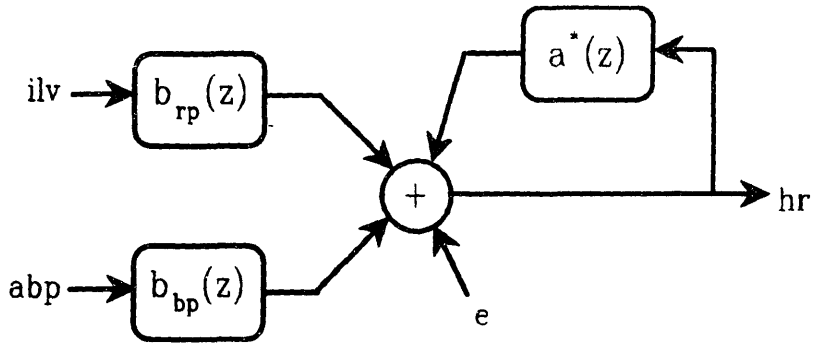


Figure 7-3: A two input ARX model intended to approximate the more general, two input, LTI model relating respiration (ILV) and arterial blood pressure (ABP) variability to heart rate (HR) variability

experiment to preserve normal ventilation.

Throughout the experiment, one lead of the surface electrocardiogram, the instantaneous lung volume (ILV), and radial artery pressure were recorded on an 8-channel FM tape recorder. An intravenous line was also placed for administering normal saline and pharmacologic agents. Data were collected in six 13 minute segments. The first data segment consisted of measurements taken with the subject in the supine position. Following this, the subject was moved to the standing position, and after a minimum of 5 minutes for hemodynamic equalibration, the breathing protocol was repeated. The subject was then returned to the supine position, and given either atropine (0.03 mg/kg, n = 7) or propranolol (0.2mg/kg, n = 7). The doses were chosen for complete parasympathetic blockade (atropine) and complete beta-adrenergic blockade (propranolol) as based on previous studies [4]. Allowing 10 minutes for equilibration, the above breathing protocol was repeated under supine and then standing conditions. Following this procedure, the subject was again returned to the supine position, and given the other autonomic blocking agent. The supine and standing breathing protocols were then repeated under the double blockade conditions for all 14 subjects.

The electrocardiogram, lung volume, and arterial pressure signals were sampled and analyzed off-line on a computer. After anti-alias filtering with 8-pole Butterworth filters,

all the signals were sampled at 360 Hz. R-waves were detected from the ECG, and a smoothed instantaneous HR time series was constructed at 3 Hz according to the procedure discussed in [1]. The ILV and arterial pressure signals were digitally filtered and decimated to 3 Hz.

7.5 Application of the APR Algorithm

The APR algorithm was employed to obtain estimate impulse responses from ILV and ABP to HR for each of the 14 subjects considered in the experiment. For the purposes of brevity, we will limit ourselves to showing only the estimated ILV to HR impulse responses for one of those subjects under two different conditions.

We began our system identification procedure by estimating the impulse responses and their 95% confidence level error boundaries (note that the confidence level of the error boundaries has *nothing* to do with the confidence level associated with the APR algorithm) for each subject with a standard least squares procedure, specifying our parameterization as:

$$\begin{aligned} b_{rp}(z) &= b_{-10}z^{10} + b_{-9}z^{9} + \cdots + b_{14}z^{-14} + b_{15}z^{-15}, \\ b_{bp}(z) &= b_{1}z^{-1} + b_{2}z^{-2} + \cdots + b_{15}z^{-15}, \\ a^{*}(z) &= a_{1}z^{-1} + a_{2}z^{-2} + \cdots + a_{10}z^{-10}. \end{aligned} \tag{7.3}$$

The results for subject 14 under control conditions (and the supine posture) are shown in Figure 7-4, while the results for subject 14 under sympathetic blockade conditions (propranolol) and the upright position are shown in Figure 7-7.

Applying the APR algorithm, we then took the above parameterization as the maximal model, and ran the algorithm at a 0.75 level of confidence, the results of which follow in Figures 7-5 and 7-8. Upon then running the APR algorithm at a 0.95 level of confidence, we obtained the impulse responses shown in Figures 7-6 and 7-9.

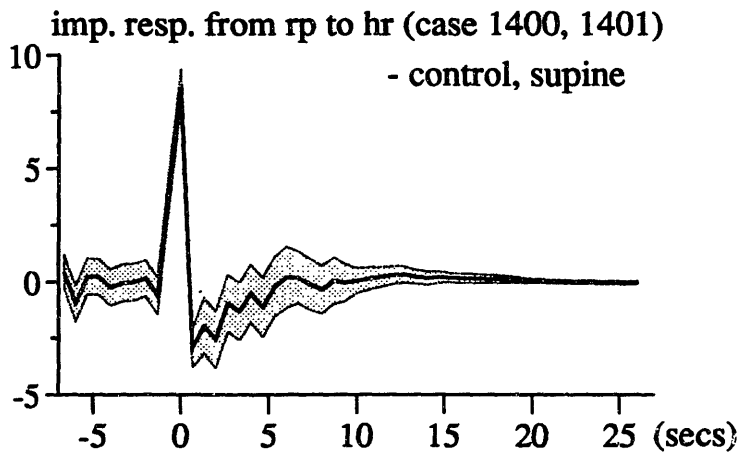


Figure 7-4: Estimation results for maximal model

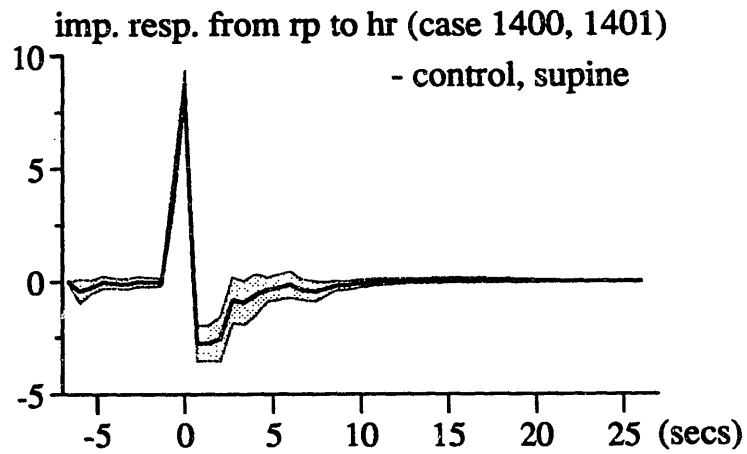


Figure 7-5: Estimation results for APR algorithm run at 0.75 level of confidence

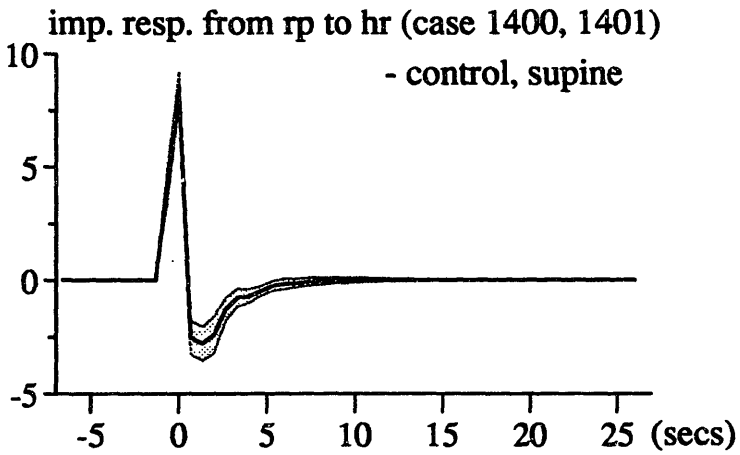


Figure 7-6: Estimation results for APR algorithm run at 0.95 level of confidence

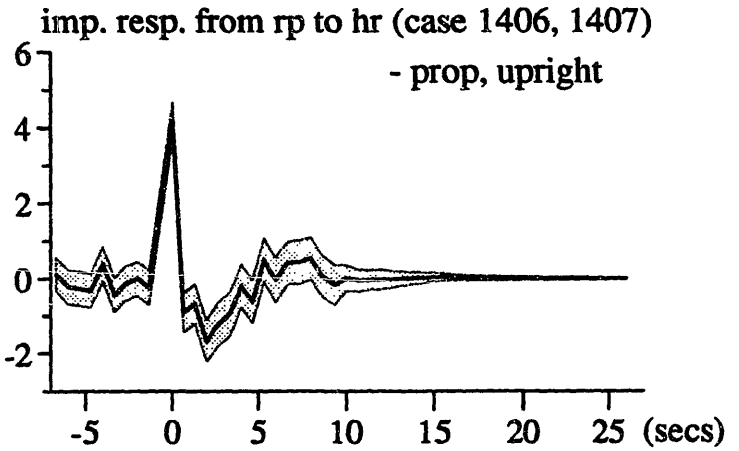


Figure 7-7: Estimation results for maximal model

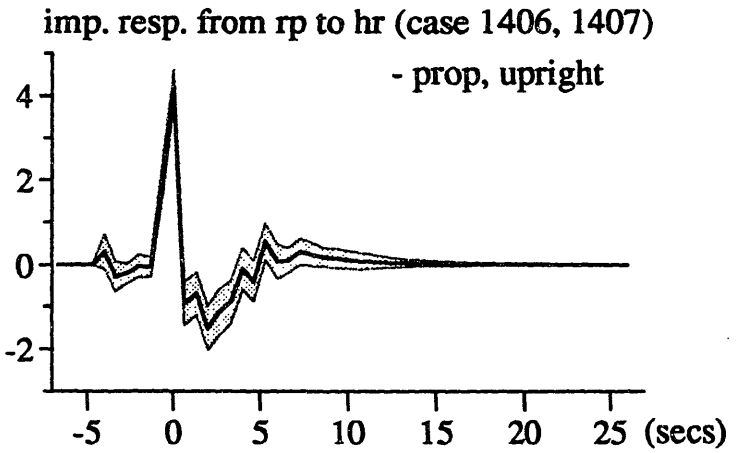


Figure 7-8: Estimation results for APR algorithm run at 0.75 level of confidence

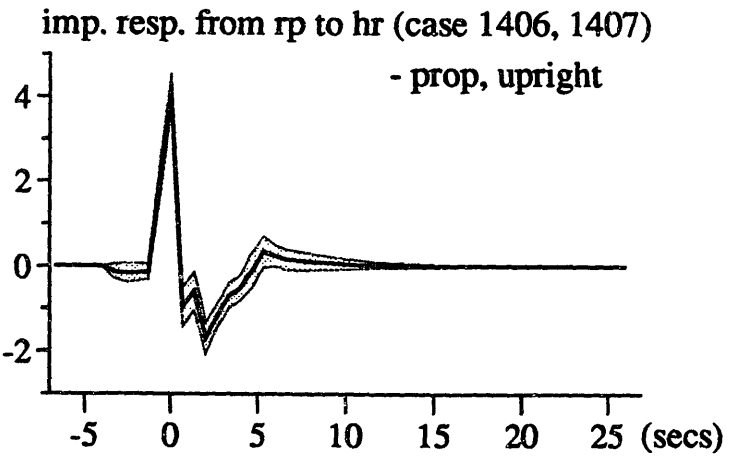


Figure 7-9: Estimation results for APR algorithm run at 0.95 level of confidence

These plots make clear the effectiveness of the APR algorithm in improving the noise reduction of our estimation process. As extra parameters are removed, the impulse response

estimates become 'smoother' and their error confidence bounds become significantly tighter.

7.6 Conclusion

As the results of the previous section make clear, the APR algorithm is an effective tool for performing transfer function analysis on actual experimental data. The primary advantage this technique offers over conventional methods is its ability to objectively remove extra parameters in order to eliminate insignificant detail in the estimated impulse responses associated with each transfer function. The removal of this detail was also shown to lead to vastly reduced error confidence bounds, so that less uncertainty was associated with the given estimates. It should also be noted that the user was not required to specify any details about the system except for a maximal model and a significance level, which allowed for a very simple identification procedure from an implementation standpoint.

As a guide to future applications, one should be careful that the simplicity and general effectiveness of the APR algorithm do not numb the user into unconditionally accepting the results it comes up with. In particular, the person running the algorithm should make him or herself aware of the detail that is removed under higher confidence levels. It is possible, under *very* noisy conditions and high confidence levels, for the algorithm to remove a 'true' parameter of the system and still justify the change in residual error norm by noise. In such a case, the impulse response estimate will change dramatically in shape from what it was calculated with under the maximal model. Thus, a recommended procedure is for the user to mimic the previous section of this chapter and compare the estimated impulse responses under the maximal model and the model chosen by the APR algorithm. The user can then decide if the detail removed is significant or not.

Appendix A

APR_LS Users Guide

A.1 Overview

The APR_LS algorithm is intended to provide a practical analysis tool for the identification of linear, time invariant systems. Two types of optimization procedures are provided — a standard ARX least squares procedure and the APR (ARX Parameter Reduction) algorithm. Upon completion of either algorithms, parametric estimates are presented, along with their associated confidence limits. In addition, impulse response estimates corresponding to the calculated parameters are calculated, along with their associated standard deviation of error.

A.2 Installation

The program 'apr_ls.c' is written in 'non-ansi' C (There is also an ansi version, but the non-ansi is more portable). Hopefully, any unix workstation should be able to compile the given source code. The following files are needed in addition to 'apr_ls.c':

1. matrix_functions.c — a collection of C library functions that perform various matrix operations (many adapted from 'Numerical Recipes in C').
2. matrix_functions.h — header file for matrix_functions.c.
3. apr_makefile — makefile for apr_ls.c

To compile 'apr_ls.c', simple verify that all of the above files are in the same directory, then type (in that same directory):

make -f apr_makefile

at the unix prompt.

A.3 Required User Files

The are two types of files required as input to the APR_LS algorithm:

1. A specification file, an example of which has been included under the name 'apr_ls.par',

2. input and output data files that are in single column (i.e. only one column of data per file), ascii format. The number of samples in each file must be the same (i.e. each file must have the same number of lines). Six example files will be referred to under the names: rp.100, bp.100, hr.100, rp.200, bp.200, and hr.200.

A.3.1 Specification File

The specification file contains directions for the APR_LS algorithm as it runs. Setup for this file is probably best explained through an example. A sample file (from apr_ls.par, in fact) is shown below:

```
goto_section: APR_95
⋮
section: APR_95

source_input: /work/perrott/CRC
source_output: /work/perrott/CRC
direct_results: /work/perrott/APR_95
label_stats: stats
label_imp_resp: H
label_cov: ex_cov
spec_file: spec.dat

input: rp + zero + decimate 2
input: bp + zero + decimate 2
output: hr + zero + decimate 2 + filter lowpass

ma_dim: rp -10 15
ma_dim: bp 1 15 25
auto_dim: 10

length_imp: 50
conf_level: .95

apr_alg

end
```

Let's briefly describe the purpose of each of the preceding lines:

- goto_section: Directs program to the label specified (essentially, this is a jump instruction).

- section: A label or title of the given set of specifications.
- source_input: The directory that contains the input data files.
- source_output: The directory that contains the output data files.
- direct_results: The directory that results will be sent to.
  Results consist of:
  1. A stats file that contains information such as the parameter estimates,
  2. 'Impulse Response' files that contain the calculated impulse responses from each input to output.
  3. Parameter 'covariance' file that contains the covariance matrix of the 'a' and 'b' parameters (note: this file is optional, and will only be created if 'label_cov:' has been included in the specification file).
- label_stats: The prefix label that is attached to the stats file name.
- label_imp_resp: The prefix label that is attached to each impulse response file name.
- label_cov: The prefix label that is attached to the parameter covariance file name (this is an optional command — if this label is left off, the program won't save the covariance information).
- spec_file: an optional file used to specify impulse responses (of any odd length) for the 'filter' option under 'input:' or 'output:'. This file can also be used to specify 'partition' information under 'ma_dim:' or 'auto_dim:'.
  Example file format:

```
lowpass
0.5
1.0
0.5

bp.100
25

bp.300
9
```

- input: The name associated with a given input file (several of these lines are required in the case of multi-input systems — i.e. one line for each input).

  Options: Operations done to each input are specified by using the '+' character (i.e. 'input: rp + zero + decimate 2' loads the file associated with input 'rp', 'zeros' the data in this file (i.e. subtracts the mean data value from each sample so that the overall mean becomes zero), and then decimates the resulting input file by a factor of 2. Options include:

- decimate 'value' — decimates input by a factor of 'value'
- interpolate 'value' — interpolates input by a factor of 'value'
- filter 'label' — filters input with the impulse response 'label' specified in the 'spec_file'.

- output: The name associated with the output file (completely analogous to 'input:', including options).

- ma_dim: Specification of the 'b' parameters associated with the given input. For the given example, 'ma_dim: rp -10 15' associates with input rp the 'b' polynomial:

$$b_{rp}(z) = \mathbf{b}_{rp}[-10]z^{10} + \mathbf{b}_{rp}[-9]z^{9} + \cdots + \mathbf{b}_{rp}[15]z^{-15}.$$

Options:

- 'value' — if a third numerical argument is specified, as in 'ma_dim: bp 1 15 25', the corresponding value is converted to a binary number and used to 'partition' the associated 'b' polynomial. For the given example, 25 is converted to its binary equivalent, 11001, which causes the input bp to be associated with the 'b' polynomial:

$$b_{bp}(z) = \mathbf{b}_{bp}[11]z^{-11} + \mathbf{b}_{bp}[12]z^{-12} + \mathbf{b}_{bp}[15]z^{-15}.$$

To explain this last statement, consider all of the coefficients of the polynomial $b_{bp}(z)$:

$$b_{bp}(z) = \mathbf{b}_{bp}[1]z^{-1} + \mathbf{b}_{bp}[2]z^{-2} + \cdots + \mathbf{b}_{bp}[14]z^{-14} + \mathbf{b}_{bp}[15]z^{-15}.$$

Given that we have specified a polynomial with 15 coefficients, let us now choose a 15 digit binary number such that 1's are associated with coefficients that we want to keep, and 0's are associated with coefficients we want to remove. To obtain a polynomial that had only nonzero coefficients associated with z^{-11}, z^{-12}, and z^{-15}, we would then write our binary number as:

000000000011001

for which the decimal equivalent is 25.

There is also an option to specify 'spec_file' as the third argument, in which case the program looks in the given file for the numerical value (under a label corresponding to the input file name). For this example, had we specified 'ma_dim: bp 1 15 spec_file', then the program would search in file 'spec.dat' for the label 'bp.100' (assuming the input file suffix has been specified as '100' — an issue that will be covered shortly), under which it would look for the appropriate numerical value.

- auto_dim: Specification of the 'order' of the system. This value corresponds to the number of autoregressive coefficients, or the *order* of the 'a' polynomial.For the given example, the autoregressive polynomial would therefore be specified as:

$$a(z) = 1 + \mathbf{a}[1]z^{-1} + \mathbf{a}[2]z^{-2} + \cdots + \mathbf{a}[10]z^{-10}.$$

Note: a third argument can be used to specify partitioning of the 'a' polynomial, in a completely analogous fashion as for 'b' polynomials.

- length_imp: Specification of the length of each calculated impulse response. For the given example, the impulse response from rp to hr would be calculated from -10 to 39 time increments (this translates into time via whatever sampling rate was used). The impulse response from bp to hr would extend from 1 to 50 time increments. It is important to note that each of the impulse responses is calculated directly from the 'a' and 'b' parameter estimates, and that the 'length_imp' specification has *no influence* on the calculated parameter values.

- conf_level: Confidence level used for hypothesis testing done by the APR algorithm (it is not necessary to specify this parameter in the event that standard least squares estimation is done in place of the APR algorithm).

- apr_alg Identifies the chosen optimization procedure as the APR algorithm. Alternatively, the user may replace this line with 'least_squares' to choose standard least squares optimization.

- end Denotes the end of the provided specification group.

A.4 Running APR_LS

We have assumed that the files rp.100, bp.100, hr.100, rp.200, bp.200, hr.200 are contained in the directory /work/perrott/CRC, and that the directory /work/perrott/APR_95 has been created. In the same directory that contains apr_ls (the executable code resulting from compilation of apr_ls.c), the specification file apr_ls.par should also exist. To run 'apr_ls' on the '.100' data, type:

```
apr_ls apr_ls.par 100
```

Notice that at least two arguments are required for apr_ls — the specification file (argument 1), and the data set suffix (argument 2). Alternatively, more than two arguments could be used to combine data sets. For example, suppose we typed:

```
apr_ls apr_ls.par 100 200
```

The result of this operation is that apr_ls would append the '200' data set to the '100' data set before performing its optimization procedure. It is important to note, however, that specifying multiple file sets in this manner has a different effect than manually combining the data sets into one file prior to running the program (for instance, combining file sets '100' and '200' into a new file set, '300', so that you ran: apr_ls apr_ls.par 300). The reason for the difference lies in the fact that the apr_ls program will combine separate data sets in such a manner that 'edge-effects' are removed.

A.5 Program Output

Sample file outputs of the apr_ls program are now shown. The specification file used was identical to the sample listing given above, with the exception of two lines:

the line that read: 'output: hr + zero + decimate 2 + filter lowpass',
was replaced with: 'output: hr + zero + decimate 2',

and the line that read: 'ma_dim: bp 1 15 25',
was replaced with: 'ma_dim: bp 1 15'.

We will assume that the user has typed 'apr_ls apr_ls.par 100 200' to obtain the following results:

- stats file is named 'stats.100', a sample listing of which is:

  STATS FILE for APR_ALG (sig_level = 0.9500000)

  DATA

  100
  200

  REL_AMNT_DATA

  1032

  INPUTS

  rp
  bp

  MA_DIM

  -10 15
  1 15

  AVE_POWER_INPUTS

  0.016369
  14.177050

  OUTPUT

  hr

  AVE_POWER_OUTPUT

  8.942393

  AVE_POWER_ERROR

1.362722

AVE_POWER_ESTIMATED_NOISE

1.345977

rp
0.00000 0.000000
⋮ (actual file contains 8 rows of '0.00000 0.00000' not shown here)
0.00000 0.000000
3.513108 0.320447
3.766129 0.375437
-4.873788 0.358991
0.00000 0.000000
⋮ (actual file contains 12 rows of '0.00000 0.00000' not shown here)
0.00000 0.000000
bp
0.00000 0.000000
0.00000 0.000000
-0.130374 0.023506
0.100898 0.022422
0.00000 0.000000
⋮ (actual file contains 9 rows of '0.00000 0.00000' not shown here)
0.00000 0.000000
AUTO
1.000000 0.000000
-1.168567 0.028423
0.427812 0.041489
-0.153810 0.040425
0.046369 0.026930
-0.000000 0.000000
⋮ (actual file contains 4 rows of '-0.00000 0.00000' not shown here)
-0.00000 0.000000

We will now proceed to explain each of the above entries:

- STATS FILE for APR_ALG — states that the apr algorithm was used, along with the specified significance level. If the least squares procedure had been used instead, this line would read: 'STATS FILE for LEAST_SQUARES procedure'.
- DATA — lists the suffixes of the data files used to perform the estimation procedure.
- REL_AMNT_DATA — specifies the relevant number of data samples used in the estimation procedure. In our example, each of the files (*.100 and *.200) originally had approximately 1,000 samples. Upon decimating by a factor of 2, the number of samples was reduced by half. Some additional samples were also lost from the way the algorithm 'windows' the data.

- INPUTS — lists the inputs used.
- MA_DIM — lists the 'b' polynomial specification for each input (this information is copied directly from the specification file).
- AVE_POWER_INPUTS — equivalent to the sum of the squared sample values of the relevant data associated with each input, divided by the number of relevant samples.
- OUTPUT — lists the output used.
- AVE_POWER_OUTPUT — average power of output, computed in same fashion as AVE_POWER_INPUTS.
- AVE_POWER_ERROR — average power of resulting **error** vector for the selected model, computed in same fashion as AVE_POWER_INPUTS.
- AVE_POWER_ESTIMATED_NOISE — average power of noise vector, computed as the average power of the **error** vector corresponding to the maximal model.
- rp — values of the 'b' polynomial corresponding to input 'rp'. The leftmost values correspond to the coefficient values of $b_{rp}(z)$, with the rightmost values indicating the estimation error standard deviation associated with each coefficient. To interpret the placement of the coefficients, one must refer the associated input's 'ma_dim:' specification. For this example, we have:

  $$b_{rp}(z) = 3.513108z^1 + 3.766129z^0 - 4.873788z^{-1},$$

  and the 95% confidence bounds for each of the parameters would be:

  $$\mathbf{b}_{rp}[-1] : 3.513108 \pm 0.320447 * 1.96,$$

  $$\mathbf{b}_{rp}[0] : 3.766129 \pm 0.375437 * 1.96,$$

  $$\mathbf{b}_{rp}[1] : -4.873788 \pm 0.358991 * 1.96.$$

- bp — values of the 'b' polynomial corresponding to input 'bp'. For this example, we have:

  $$b_{bp}(z) = -0.130374z^{-3} + 0.100898z^{-4},$$

  and the 95% confidence bounds are:

  $$\mathbf{b}_{bp}[3] : -0.130374 \pm 0.023506 * 1.96, \quad \mathbf{b}_{bp}[4] : 0.100898 \pm 0.022422 * 1.96.$$

- AUTO — values of the autoregressive polynomial, $a(z)$. For this example, we have:

  $$a(z) = 1 - 1.168567z^{-1} + 0.427812z^{-2} - 0.153810z^{-3} + 0.046369z^{-4},$$

  with confidence bounds being computed in exactly the same fashion as for each of the 'b' polynomials.

- There are two impulse response files for this example, the first of which is named 'Hrp_hr.100', a sample listing of which is shown below:

```
-10.000000 0.000000 0.000000
```

-9.000000 0.000000 0.000000
⋮ (actual file contains 6 rows not shown here)
-2.00000 0.000000 0.000000
-1.00000 3.513108 0.320447
0.00000 7.871432 0.381468
1.00000 2.821560 0.405504
2.00000 0.470039 0.397396
⋮ (actual file contains 36 rows not shown here)
39.00000 0.000011 0.000075

The values shown above, from left to right, correspond to the time increment, the estimated impulse response value at that increment, and the value of the estimation error standard deviation associated with that impulse response value. Thus, the above data indicates the values taken on by the impulse response from rp to hr from time -10*(sampling period) to 39*(sampling period). Confidence bounds for the impulse response are calculated in precisely the same fashion as for the parameter estimates (i.e. 95% confidence bounds for $h_{\text{rp to hr}}[-1]$ would be $3.513108 \pm 0.320447 * 1.96$).

The second of these files is named 'Hbp_hr.0100', a sample listing of which is shown below:

1.00000 0.00000 0.00000
2.00000 0.00000 0.00000
3.00000 -0.130374 0.023506
4.00000 -0.051453 0.012618
5.00000 -0.004351 0.011959
⋮ (actual file contains 44 rows not shown here)
50.0000 0.000000 0.000000

The above values are in precisely the same format as for 'Hrp_hr.100', with 'Hbp_hr.100' extending in time from 1*(sampling period) to 50*(sampling period).

- Since 'cov_label: ex_cov' was included in the specification file, a covariance file named 'ex_cov.100' was created, a listing of which is shown on the next page.

0.1027	−0.0529	0.0180	−0.0001	−0.0000	0.0003	−0.0013	0.0009	−0.0006
−0.0529	0.1410	−0.0481	0.0003	−0.0005	−0.0035	0.0047	−0.0027	0.0007
0.0180	−0.0481	0.1289	−0.0000	−0.0004	−0.0017	−0.0023	0.0028	−0.0008
−0.0001	−0.0003	−0.0000	0.0006	−0.0005	−0.0001	0.0001	−0.0000	−0.0002
−0.0000	−0.0005	−0.0004	−0.0005	0.0005	0.0001	−0.0001	0.0001	0.0001
0.0003	−0.0035	−0.0017	−0.0001	0.0001	0.0008	−0.0009	0.0004	−0.0001
−0.0013	0.0047	−0.0023	0.0001	−0.0001	−0.0009	0.0017	−0.0012	0.0003
0.0009	−0.0027	0.0028	−0.0002	0.0001	0.0004	−0.0012	0.0016	−0.0008
−0.0006	0.0007	−0.0008	−0.0002	0.0001	−0.0001	0.0003	−0.0008	0.0007

To understand the meaning of the above matrix, you must go through the stats file and note the parameters that have been estimated with nonzero values. You must then place each parameter in a vector in the order found in the stats file. For this example, we would form the parameter vector:

$$\boldsymbol{\theta} = \begin{bmatrix} \mathbf{b}_{rp}[-1] \\ \mathbf{b}_{rp}[0] \\ \mathbf{b}_{rp}[1] \\ \mathbf{b}_{bp}[3] \\ \mathbf{b}_{rp}[4] \\ \mathbf{a}[1] \\ \mathbf{a}[2] \\ \mathbf{a}[3] \\ \mathbf{a}[4] \end{bmatrix}$$

The data values within the example covariance file correspond to the autocorrelation matrix of the above vector, i.e. $E(\boldsymbol{\theta}\boldsymbol{\theta}^T)$.

Bibliography

[1] R. D. Berger, S. Akselrod, D. Gordon, and R. J. Cohen. An efficient algorithm for spectral analysis of heart rate variability. *IEEE Trans. Biomed. Eng., BME-33: 900-904*, 1986.

[2] R. D. Berger, J. P. Saul, and R. J. Cohen. Assessment of autonomic response by broad-band respiration. *IEEE Trans. Biomed. Eng., BME-36: 1061-1065*, 1989.

[3] R.M. Berne and M.N. Levy. *Cardiovascular Physiology, 5th ed.* The C.V. Mosby Company, St. Louis, Missouri, 1986.

[4] A. D. Jose and R. R. Taylor. Autonomic blockade by propranolol and atropine to study intrinsic myocardial function in man. *J. Clin. Invest., 48: 2019*, 1969.

[5] L. Ljung. *System Identification: Theory for the User.* Prentice Hall, Englewood Cliffs, NJ, 1987.

[6] L. Ljung. *System Identification Toolbox User's Guide.* The Mathworks, South Natick, MA, 1988.

[7] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, New York, NY, 1984.

[8] J. P. Saul. Beat-to-beat variations of heart rate reflect modulation of cardiac autonomic outflow. *NIPS,5:32:37*, 1990.

[9] J. P. Saul, R. D. Berger, and R. J. Cohen. Transfer function analysis of autonomic regulation. II. respiratory sinus arrhythmia. *Am. J. Physiol, 256 (Heart Circ. Physiol, 25): H153-H161*, 1989.

[10] J. P. Saul, R.D. Berger, P. Albrecht, and R. J. Cohen. Transfer function analysis of the circulation. *(submitted to Circulation Research)*, 1991.

[11] William M. Siebert. *Circuits, Signals, and Systems.* MIT Press, Cambridge, MA, 1986.