Applications of nonlinear filters with the linear-in-the-parameter structure

Eng Siong CHNG

A thesis submitted for the degree of Doctor of Philosophy.

The University of Edinburgh.

- December 1995 -

Abstract

The subject of this thesis is the application of nonlinear filters, with the linear-in-the-parameter structure, to time series prediction and channel equalisation problems.

In particular, the Volterra and the radial basis function (RBF) expansion techniques are considered to implement the nonlinear filter structures. These approaches, however, will generate filters with very large numbers of parameters. As large filter models require significant implementation complexity, they are undesirable for practical implementations. To reduce the size of the filter, the orthogonal least squares (OLS) algorithm is considered to perform model selection. Simulations were conducted to study the effectiveness of subset models found using this algorithm, and the results indicate that this selection technique is adequate for many practical applications. The other aspect of the OLS algorithm studied is its implementation requirements. Although the OLS algorithm is very efficient, the required computational complexity is still substantial. To reduce the processing requirement, some fast OLS methods are examined.

Two major applications of nonlinear filters are considered in this thesis. The first involves the use of nonlinear filters to predict time series which possess nonlinear dynamics. To study the performance of the nonlinear predictors, simulations were conducted to compare the performance of these predictors with conventional linear predictors. The simulation results confirm that nonlinear predictors normally perform better than linear predictors. Within this study, the application of RBF predictors to time series that exhibit homogeneous nonstationarity is also considered. This type of time series possesses the same characteristic throughout the time sequence apart from local variations of mean and trend.

The second application involves the use of filters for symbol-decision channel equalisation. The decision function of the optimal symbol-decision equaliser is first derived to show that it is nonlinear, and that it may be realised explicitly using a RBF filter. Analysis is then carried out to illustrate the difference between the optimum equaliser's performance and that of the conventional linear equaliser. In particular, the effects of delay order on the equaliser's decision boundaries and bit error rate (BER) performance are studied. The minimum mean square error (MMSE) optimisation criterion for training the linear equaliser is also examined to illustrate the sub-optimum nature of such a criterion. To improve the linear equaliser's performance, a method which adapts the equaliser by minimising the BER is proposed. Our results indicate that the linear equalisers performance is normally improved by using the minimum BER criterion. The decision feedback equaliser (DFE) is also examined. We propose a transformation using the feedback inputs to change the DFE problem to a feedforward equaliser problem. This unifies the treatment of the equaliser structures with and without decision feedback.

Acknowledgements

I would like to thank the following people for their invaluable assistance during the course of this PhD:

- Dr Bernard Mulgrew for his patience and willingness to share his insight throughout my stay in Edinburgh.
- Professor Peter Grant for his support and guidance.
- Dr Sheng Chen for his help in all the papers we wrote together.
- Mr Achilleas Stogillou and Mr John Thompson for their many hours speaking about Mathematics, RBF networks, time series prediction and channel equalisation problems.
- Dr Gavin Gibson for providing guidance to the work performed in Chapters 4 and 5.
- Dr Iain Scott for proofreading several chapters of the thesis.
- Dr Martin Reekie for proofreading the entire thesis.
- Yee Ling for her wonderful company for so many years.
- And lastly, to my parents and family for their love and understanding in letting me study so far away from home.

To my parents, and to Yee Ling.

Contents

List of Figures

List of Tables

Abbreviations

A/D	Analogue to digital
AR	Auto-regressive
ARIMA	Auto-regressive integrated moving average
AWGN	Additive white Gaussian noise
BER	Bit error rate
DFE	Decision feedback equaliser
DSP	Digital signal processing
FIR	Finite impulse response
GRBF	Gradient radial basis function
GS	Gram-Schmidt
IIR	Infinite impulse response
ISI	Inter-symbol interference
LTE	Linear transversal equaliser
LS	Least squares
MARS	Multivariate adaptive regression splines
MBER	Minimum bit error rate
MC	Monte Carlo
MLP	Multi-layer perceptron
MLSE	Maximum likelihood sequence estimator
MLVA	Maximum likelihood Viterbi algorithm
MMSE	Minimum mean square error
MSE	Mean square error
NMSE	Normalised mean square error
NN	Neural networks
OLS	Orthogonal least squares
PAM	Pulse amplitude modulation

PSK	Phase shift keying
QAM	Quadrature amplitude modulation
RBF	Radial basis function
SNR	Signal to noise ratio
SVD	Singular value decomposition
VLSI	Very large scale integration
ZF	Zero-forcing

List of principal symbols

Principal symbols used in chapters 2 and 3

$e(n)$	error in approximation between $\hat{y}(n)$ to desired signal $y(n)$
$y(n)$	desired signal at time n
$\hat{y}(n)$	approximation of desired signal at time n
$\mathbf{e}$	approximation error between the vectors $\mathbf{y}$ and $\hat{\mathbf{y}}$, i.e. $\mathbf{e} = [e(1) \cdots e(N)]^T \in R^N$
$\mathbf{X}$	information matrix
$\tilde{\mathbf{X}}$	transformed information matrix
$\tilde{\mathbf{X}}_R$	rank R approximation of information matrix
$\mathbf{X}^+$	pseudo-inverse of matrix $\mathbf{X}$
$\|\|\mathbf{x}\|\|$	Euclidean norm of vector $\mathbf{x}$, i.e. $\sqrt{\mathbf{x}^T\mathbf{x}}$.
$\|\|\mathbf{X}\|\|_F$	Frobenius norm of matrix $\mathbf{X}$
$\mathbf{y}$	desired vector, i.e. $\mathbf{y} = [y(1) \cdots y(N)]^T \in R^N$
$\hat{\mathbf{y}}$	approximation of the desired vector, i.e. $\mathbf{y} = [\hat{y}(1) \cdots \hat{y}(N)]^T \in R^N$
$\phi_j(.)$	nonlinearity of j^{th} regressor
$\phi_j^{'}(.)$	the 1st order gradient RBF j^{th} nonlinearity
J_{LS}	least squares cost function

Principal symbols used in chapters 4 and 5

d	equaliser's delay order
m	equaliser's feedforward order
n	DFE's feedback order
n_a	length of channel impulse response
s	transmit symbol alphabet
σ_e^2	noise variance
$a(i)$	channel impulse response's coefficients

$n(k)$	additive white Gaussian noise time at time k
$r(k)$	received signal at time k
$\hat{r}(k)$	noiseless received signal at time k
$s(k)$	transmit symbol at time k
$\mathbf{c}_j$	the j^{th} channel state
$\mathbf{v}_l$	translation vector l for DFE
$\mathbf{w}$	linear equaliser's weight
$\mathbf{n}(k)$	the additive Gaussian noise vector, i.e. $\mathbf{n}(k) = [n(k) \ \cdots \ n(k-m)]$
$\mathbf{r}(k)$	the vector of received signal at time k, i.e. $\mathbf{r}(k) = [r(k) \ \cdots \ r(k-m)]^T \in R^m$
$\hat{\mathbf{r}}(k)$	the vector of noiseless received signal at time k, i.e $\hat{\mathbf{r}}(k) = [\hat{r}(k) \ \cdots \ \hat{r}(k-m)]^T \in R^m$ This vector is also called a channel state
$\mathbf{r}'(k)$	transformed received vector of DFE equaliser
$\mathbf{s}(k)$	is the vector of transmitted sequence, i.e. $\mathbf{s}(k) = [s(k) \ \cdots \ s(k-d-n_a+2)]^T$
$\mathbf{s}_a(k)$	is the feedforward vector of the DFE, i.e. $\mathbf{s}_a(k) = [s(k) \ s(k-1) \ \cdots \ s(k-d)]^T$
$\mathbf{s}_b(k)$	is the feedback vector of the DFE, i.e. $\mathbf{s}_b(k) = [s(k-d-1) \ \cdots \ s(k-d-n)]^T$
$p(\mathbf{r}(k)\|\mathbf{c}_j)$	pdf of received signal $\mathbf{r}(k)$ given channel state is $\mathbf{c}_j$
$p(\mathbf{r}(k))$	pdf of received vector
$P(s(k-d)=s\|\mathbf{r}(k))$	*a posteriori* probability of transmit symbol being s conditioned on received vector $\mathbf{r}(k)$
$f_b(\mathbf{r}(k))$	Bayesian equaliser's decision function
$f_b(\mathbf{r}(k), \mathbf{s}_b(k))$	Bayesian DFE's decision function
$f_l(\mathbf{r}(k))$	linear equaliser's decision function
C_d	the set of channel states using delay order d
$C_d^{(+)}$	the set of channel states using delay order d and transmit symbol $s(k-d) = +1$
$C_d^{(-)}$	the set of channel states using delay order d and transmit symbol $s(k-d) = -1$
$C_{d,l}$	the set of channel states associated with DFE feedback vector $\mathbf{s}_b(k) = \mathbf{s}_{b,l}$
$Z_b^{(+)}$	region of $\mathbf{r}(k)$ associated with $f_b(\mathbf{r}(k)) \geq 0$
$Z_b^{(-)}$	region of $\mathbf{r}(k)$ associated with $f_b(\mathbf{r}(k)) < 0$
$Z_l^{(+)}$	region of $\mathbf{r}(k)$ associated with $f_l(\mathbf{r}(k)) \geq 0$
$Z_l^{(-)}$	region of $\mathbf{r}(k)$ associated with $f_l(\mathbf{r}(k)) < 0$
P_b	probability of mis-classification of Bayesian equaliser
$P_{bc}(\mathbf{c}_j)$	probability of mis-classification of Bayesian equaliser given channel state is $\mathbf{c}_j$
P_l	probability of mis-classification of linear equaliser
$P_{lc}(\mathbf{c}_j)$	probability of mis-classification of linear equaliser given channel state is $\mathbf{c}_j$
ζ_j	minimum distance of channel state $\mathbf{c}_j$ to the decision boundary

N_s	number of available channel states
U	the set of channel states nearest to decision boundary
W_d	region of $\mathbf{r}(k)$ where $\text{sgn}(f_b(\mathbf{r}(k))) \neq \text{sgn}(f_l(\mathbf{r}(k)))$
J_{ber}	bit error rate cost function
J_{mmse}	minimum mean square error cost function
∇	vector gradient operator
E[.]	expectation operator

Chapter 1

Introduction

1.1 Theme of thesis

In conventional linear filter's operation, the output value of the filter is simply some linear combination of the input signal, i.e. no nonlinear operation on the input signal is performed. Such filter is therefore very simple to implement, and is very popular in many applications. Although linear filtering techniques have served well in the past few decades [1], the commercial demand for better performance and the requirement to operate in more difficult environments continually drives the need to improve on existing methods. In addition, it is well-known that the applications of linear filters to real world problems like time series prediction [2] and channel equalisation [3] are sub-optimum, and substantial improvements may be achieved by using nonlinear methods.

In this thesis, we consider nonlinear filters which retain the linear-in-the-parameter structure for filter applications. The focus is to examine the Volterra and the radial basis function (RBF) expansion techniques to generate the desired nonlinear filter structure, and to compare the performance of such filters to conventional linear ones. The introduction of nonlinearity into the filter's operation however will lead to an increase in the implementation complexity. Although this makes the nonlinear implementation less attractive than linear methods, this disadvantage is becoming less significant with the recent advances of VLSI technologies which have made the computation of highly complex filtering operation in real time not only feasible but low cost. This however does not mean that nonlinear filters can be implemented without consideration of their implementation complexity. In general, the processing requirements of nonlinear filter operations are still very high and their parameter optimisation difficult to perform. Linear filters, on the other hand, are very simple to implement, their behaviour is well understood, and the techniques for the optimisation of their parameters and model selection process are well tested and documented. Therefore, a natural step towards a nonlinear filter implementation from a linear one is to use a nonlinear filter which is linear in the parameter. This implies that such a nonlinear filter retains a linear structure, i.e. the output is a linear combination of its input regressors. The nonlinearities in the filter's operation are introduced by performing nonlinear transformations on the input signal to generate the regressors' responses. Such a filter structure retains many of the advantages of the conventional linear model and yet is nonlinear.

In this work, the following two restrictions regarding the filter's operation are assumed. Firstly, the filters considered are applied only to stationary problems, unless otherwise expressed, and secondly, the filters operate only on discrete time digital signals.

By restricting the study to stationary problems, adaptive algorithms need not be considered to update the parameters of the filter. This therefore simplifies the study, and allows us to focus on the changes in the filter's performance when different structures are applied.

The second condition, which restricts the nature of the input and output signals to be in discrete time and digital form, focuses our study within the digital signal processing (DSP) framework. In recent years, the vast improvements in computer hardware and software technologies have made DSP techniques much more attractive than classical analogue techniques. Some of the reasons are: (i) DSP system can now be realised by using very cheap VLSI high-speed micro-processors and digital computers. (ii) Their implementations, usually based on software algorithms, may be modified easily, thus allowing quick turn-around time in the development cycle. (iii) They can be used to simulate analogue systems, and perform complicated signal transformation that cannot be implemented using continuous time hardware. (iv) Digital signals may be easily stored on magnetic media without degradation or loss of signal fidelity beyond that introduced in the analogue-to-digital (A/D) conversion. As a consequence of the advantages cited, and many more not listed here, it is not surprising that DSP techniques have become very important and are now being regularly applied in a broad range of practical applications, e.g., speech processing, image processing, digital communication channel equalisation, layered earth modelling, and many more. Although these applications are quite different in nature, they have a common feature, i.e., all of them involve a filtering operation to transform the input signal to a desired response. The difference between the various applications is in the manner the desired response is extracted. To demonstrate this, the filtering operations for the problem of time series prediction and channel equalisation are briefly introduced.

1.2 Modes of operation

The basic task of a filter is to operate on an input sequence of data samples $\mathbf{s} = [s_1 \; s_2 \; \cdots \; s_m]^T \in R^m$ to produce an approximation $\hat{y}$ of the desired output signal y. To illustrate the different operations involved for different filter applications, the applications of the filter to time series prediction and channel equalisation are discussed in the following paragraphs.

Time series prediction

The requirement of time series prediction is to use a finite set of present and past samples of a process to predict samples of the process in the future. The filter designed to perform the prediction is called a predictor. As an example, let us consider the specific case of single-step prediction problem.

Let the samples of a discrete time process be $\{y(k)\}_{k=1}^{N}$, where the index k of $y(k)$ denotes the k^{th} sample of the process. To predict the sample $y(k)$, the predictor uses an input vector $\mathbf{s}(k)$, where $\mathbf{s}(k) = [y(k-1)\ y(k-2)\ \cdots\ y(k-m)]^T$ consists of m past samples of the time series. The model of the prediction process is illustrated in Fig. 1.1a. The approximation of the desired signal is $\hat{y}(k)$, and the difference between the actual sample of the process and the predictor's output is called prediction error and is given by $e(k) = y(k) - \hat{y}(k)$. For time series prediction problems, the most common criterion used to optimise the filter's operation is to minimise the mean square error (MMSE), i.e., $E[e(k)^2]$, where $E[.]$ denotes the expectation operation. More details regarding filter operation for time series prediction will be presented in Chapter 2.

Digital Channel Equalisation

The task of an equaliser in channel equalisation problem is to reconstruct the transmitted digital signals which have been passed through a dispersive medium and corrupted by additive noise [3]. Such a problem is very important in the digital communication discipline. As an example, let us examine the operations of a symbol-decision class equaliser.

The symbol-decision equaliser consists of a filter and a quantiser as illustrated in Fig. 1.1b. Let the transmit signal $y(k)$ be drawn randomly from a binary source with constellation $\{\pm 1\}$. The transmit signal is passed through a channel which introduces noise and inter-symbol interference into the data. Let the signal received at the end of the channel be $s(k)$. Using a vector of m past received signals, i.e. $\mathbf{s}(k) = [s(k)\ s(k-1)\ \cdots\ s(k-m+1)]^T$, the equaliser generates an estimate $\hat{y}(k-d)$ of the transmitted signal $y(k-d)$. The positive integer value d denotes the delay order of the equaliser, and a delay value greater than 0 implies that the equaliser is making delayed decisions for transmitted symbols. The task of the equaliser is to reconstruct the transmitted sequence with the minimum probability of mis-classification. Fig. 1.1b depicts this equalisation process.

1.3 Filter structure

This section briefly introduces the main component of the work, the nonlinear filter which possesses the linear-in-the-parameter structure. By the term filter, we imply a device in the form of physical hardware or computer software applied to a vector of discrete-time input data $\mathbf{s} \in R^m$ to extract a prescribed quantity of interest $y \in R$. We denote this operation as a

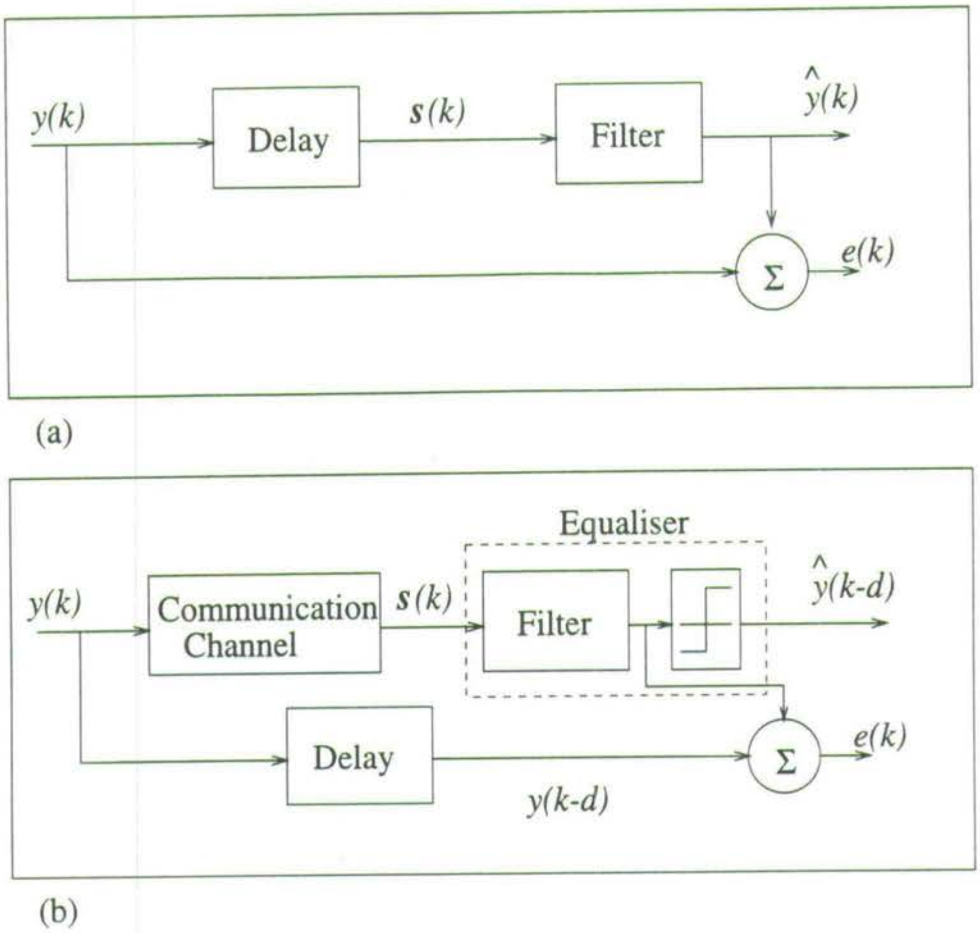

Figure 1.1: Operation of the filter in (a) time series prediction, (b) channel equalisation.

transformation described by the function $f(.)$, where $f(.) : R^m \rightarrow R$, i.e.,

$$\hat{y} = f(\mathbf{s}). \tag{1.1}$$

In particular, we are interested in the following form of $f(.)$,

$$\hat{y} = f(\mathbf{s}) = w_0 + \sum_{i=1}^{K-1} \phi_i(\mathbf{s}) w_i, \tag{1.2}$$

where K denotes the number of parameters, $\phi_i(\mathbf{s})$ is the output of the i^{th} regressor, and w_i, $0 \leq i \leq K-1$, are the filter's weight. From Eq. 1.2 it is observed that the output value y is linearly dependent on the values of the regressors $\phi_i(\mathbf{s})$. This is the reason why such a filter's structure is called linear-in-the-parameter. The nonlinearity in the filter's operation is introduced in the regressor's operation $\phi_i(\mathbf{s})$, where $\phi_i(.)$ is a nonlinear function $R^m \rightarrow R$. The flexibility in the choice of applied nonlinearity allows the function $f(.)$ to realise a host of different nonlinear structures.

1.4 Thesis Layout

The organisation of the thesis is as follows:

Chapter 2 presents much of the necessary background to the work performed in this thesis. The chapter first expands on the discussion of the linear-in-the-parameter filter and then examines two popular techniques of generating such types of filter, namely, the Volterra expansion [4–6] and the RBF expansion [7–9] techniques. Subset model selection using the OLS algorithm [10, 11] is then discussed and, lastly, a short review regarding time series prediction and channel equalisation is given.

Chapter 3 presents some new results regarding the OLS algorithm and the application of RBF filters for time series predictions. The chapter is divided into two parts. The first part discusses results concerning the OLS algorithm. In particular, the chapter studies the sub-optimum nature of the OLS solution and proposes a simple procedure to improve the selection process [12]. In addition, the chapter also investigates the implementation complexity of the algorithm, and introduces fast-OLS methods to reduce processing requirement [13, 14]. The second part of the chapter discusses results concerning the application of RBF predictors to time series that possess homogeneous non-stationarity. Some results are presented to show that RBF predictors perform poorly for this type of time series. To improve the predictive performance, a modification to the RBF network's response behaviour is introduced [15, 16].

Chapters 4 and 5 consider the symbol-decision type equaliser for channel equalisation problems. The focus of Chapter 4 is to study the application of nonlinear techniques to realise the decision function of the equaliser, while that of Chapter 5 is concerned with linear techniques. Chapter 4 begins by deriving the decision function of the optimum symbol-decision equaliser, the Bayesian equaliser [17, 18], to show that the optimum decision function is nonlinear and has an identical structure to the RBF model. The chapter then discusses the effects of delay order on the decision boundaries and equaliser's performance [19]. Also in this chapter, the implementation complexity of the RBF equaliser is examined and two methods to reduce the implementation complexity are discussed. The first method is based on model selection technique [19] and the second method is based on employing decision feedback. In addition, the chapter presents a novel transformation method to reduce the Bayesian decision feedback equaliser (DFE) structure to that of the conventional Bayesian equaliser without feedback [20]. By employing such a transformation, the implementation complexity of the Bayesian DFE is not only reduced, it allows a unified treatment of the DFE problem.

Chapter 5 investigates the performance of linear equalisation techniques with respect to the optimum nonlinear Bayesian equalisation methods. In particular, the chapter derives the degradation of classification performance incurred by the the application of linear techniques with respect to the optimum nonlinear techniques to highlight the sub-optimum nature of the linear implementation. Following the same theme as Chapter 4, a discussion regarding the effects of the delay order parameter on the linear equaliser's performance and decision boundaries is presented. In addition, the chapter also examines the MMSE criterion widely used to optimise the linear equaliser's performance. Some simulations are conducted to show that this criterion is sub-optimum [21], and that the linear equaliser's performance may be improved by applying the minimum BER (MBER) optimisation criterion instead. Lastly, the chapter considers the MBER optimisation procedure for linear combiner DFE [20] problems.

In Chapter 6, the conclusions of this work are summarised and areas in which further work may prove useful are suggested.

Chapter 2

Background

The objective of this chapter is to introduce the necessary background to the work examined in this thesis. The chapter is organised as follows: Sec. 2.1 discusses the theory of regression modelling and Sec. 2.2 presents the application of nonlinear filters which possess the linear-in-the-parameter structure to non-parametric modelling. In particular, the Volterra and the RBF network implementations of the nonlinear models are examined. Sec. 2.3 discusses the OLS algorithm for model selection, and Sec. 2.4 and 2.5 introduce the applications of filters to time series prediction and channel equalisation problems respectively.

2.1 Regression models

Regression models can be used to describe the input and output relationships of a system. A simple example of a regression model is the following,

$$y = g(x) = w_0 + w_1 x, \tag{2.1}$$

where y is the output variable of a system, $g(x)$ is a linear regression model describing the true relationship between the input variable x to y and $\{w_0, w_1\}$ are the parameters of the model. If the functional form of $g(.)$ is known, the problem of identifying the weights $\{w_0, w_1\}$ using a set of N observations, $\{x_i, y_i\}_{i=1}^{N}$, is called parametric modelling [22, 23].

In most real world problems however, the output response of a system does not depend linearly on the input values and the exact relationship between the input and output values is not known. The only data available to describe the process is a collection of N observations between the input vector $\mathbf{x} = [x_1 \; x_2 \; \cdots \; x_m]^T \in R^m$ and the output value y, namely $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$. The task of creating a model $f(.)$ to approximate the true nonlinear model $g(.)$ based on the available data is known as regression analysis [22–25] and is a problem common to many disciplines.

To identify the unknown nonlinear model $g(.)$, we begin with the assumption that it belongs to a general collection of nonlinear functions and use the available data to determine the approximation model $f(.)$ and associated coefficients. In this thesis, we will restrict the structure of $f(.)$ to belong to the class of nonlinear models which possess the linear-in-the-parameter structure. By not assuming the functional form of $g(.)$, the identification process is known as non-parametric modelling [22,23,26,27].

2.2 Linear in the parameter filter

This section considers the linear-in-the-parameter filter and formulates the procedure used to evaluate the weights of such a model using the MMSE criterion. To facilitate the discussion, the example of a linear filter applied to single step time series prediction is examined.

The time series $y(t)$ to be predicted is first sampled and quantised. The discrete value of the signal at sample instant i will be denoted by $y(i)$, where $i = 1 \dots N$, and N is the number of samples. The simplest linear-in-the-parameter model for a time series predictor is the auto-regressive (AR) model [28,29], where the output of the filter is a linear combination of the past values of the time series. The number of parameters used in the model is described as the order of the linear predictor. The following equation describes a linear predictor with order K,

$$\hat{y}(i) = \mathbf{x}(i)^T \mathbf{w} = w_0 + \sum_{j=1}^{K-1} y(i-j) w_j, \tag{2.2}$$

where the vector $\mathbf{x}(i) = [1\ y(i-1)\ y(i-2)\ \cdots\ y(i-K-1)]^T \in R^K$ consists of a constant value 1 and a vector of past values of the time series at time i, and $\mathbf{w} = [w_0\ w_1\ \cdots\ w_{K-1}]^T \in R^K$ are the weights of the filter. The first elements of $\mathbf{w}$ and $\mathbf{x}(i)$, i.e. w_0 and 1, are used to model the mean of the time series. The filter's estimated value of the desired signal $y(i)$ is $\hat{y}(i)$. To describe the regression using matrix notations, we define the followings variables

$$\begin{aligned}
\mathbf{y} &= [y(1)\ y(2)\ \cdots\ y(N)]^T, && (2.3)\\
\hat{\mathbf{y}} &= [\hat{y}(1)\ \hat{y}(2)\ \dots\ \hat{y}(N)]^T, && (2.4)\\
\mathbf{X} &= \begin{bmatrix} \mathbf{x}(1) \\ \mathbf{x}(2) \\ \vdots \\ \mathbf{x}(N) \end{bmatrix} = \begin{bmatrix} 1 & y(0) & \cdots & y(0-(K-1)) \\ 1 & y(1) & \cdots & y(1-(K-1)) \\ \vdots & \vdots & \dots & \vdots \\ 1 & y(N-1) & \cdots & y(N-1-(K-1)) \end{bmatrix}, && (2.5)\\
\mathbf{e} &= [e(1)\ e(2)\ \cdots\ e(N)]^T, && (2.6)
\end{aligned}$$

where the vector $\mathbf{y}$ and $\hat{\mathbf{y}}$ are the actual and estimated values of the time series respectively. The matrix $\mathbf{X}$ is the information matrix, and $e(i) = y(i) - \hat{y}(i)$, $1 \le i \le N$, is the prediction error between the actual value and the model's predicted value. Using the above notations, the regression model for the linear predictor can be expressed in the following matrix form

$$\mathbf{y} = \hat{\mathbf{y}} + \mathbf{e} = \mathbf{X}\mathbf{w} + \mathbf{e}. \tag{2.7}$$

To highlight the linear-in-the-parameter structure, we define the columns of $\mathbf{X}$ as

$$\mathbf{X} = [\mathbf{c}_0 \ \mathbf{c}_1 \ \cdots \ \mathbf{c}_{K-1}], \tag{2.8}$$

and express the model's output $\hat{\mathbf{y}}$ as a linear combination of the columns of $\mathbf{X}$

$$\hat{\mathbf{y}} = \sum_{j=0}^{K-1} \mathbf{c}_j w_j. \tag{2.9}$$

The dimension of the matrix $\mathbf{X}$ is $N \times K$ and the dimension of the vectors $\mathbf{y}$, $\hat{\mathbf{y}}$, $\mathbf{e}$ is N. The most common criterion used to optimise the predictor's performance is the MMSE criterion [1, 29], i.e.,

$$J_{mmse}(\mathbf{w}) = E[(y(i) - \hat{y}(i))^2], \qquad i = 1, \ldots, \infty. \tag{2.10}$$

The MMSE cost function includes the expectation operator $E[.]$ because the processes involved in the estimation problem are random and the criterion operates over ensemble averages. In order to solve for the weights using this criterion, precise knowledge of the second order statistics of the problem are required. When only finite amount of data describing the process are available, e.g., $\{\mathbf{x}(i), y(i)\}_{i=1}^{N}$, only the time average solution for the approximation of the MSE solution is possible. Such an approximation is called the least squares (LS) solution. The LS method finds the parameter vector $\mathbf{w}$ by minimising the Euclidean norm of $\mathbf{e}$ [30, 31], i.e.,

$$J_{LS}(\mathbf{w}) = \sum_{i=1}^{N} (y(i) - \hat{y}(i))^2. \tag{2.11}$$

The LS solution for $\mathbf{w}$ is

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.12}$$

Alternatively, the pseudo-inverse $\mathbf{X}^+$ of $\mathbf{X}$ may be applied to solve for $\mathbf{w}$ [31, 32] i.e.,

$$\mathbf{w} = \mathbf{X}^+\mathbf{y}. \tag{2.13}$$

This avoids ill-conditioning problems when formulating $(\mathbf{X}^T\mathbf{X})^{-1}$.

The approach just outlined for evaluating the weights of the linear predictor using the LS criterion can be extended to nonlinear predictors which possess the linear-in-the-parameter structure. Examples of such types of nonlinear predictors include Volterra filters [4, 5, 33–35], RBF networks [7–9, 36], NARMAX models [37, 38], multivariate adaptive regression splines (MARS) [26, 39] and fuzzy basis function models [40]. The nonlinearity of these predictors' operations are introduced by a nonlinear expansion of the original input vector $\mathbf{s}(i) = [y(i-1)\ y(i-2)\ \cdots\ y(i-m)]^T \in R^m$ to generate an expanded vector

$$\mathbf{x}(i) = [1\ \phi_1(\mathbf{s}(i))\ \phi_2(\mathbf{s}(i))\ \cdots\ \phi_{K-1}(\mathbf{s}(i))] \in R^K. \tag{2.14}$$

The operation $\phi_j(\mathbf{s}(i))$ is the j^{th} regressor's nonlinear transformation $R^m \rightarrow R$ on the input signal. The structure of this nonlinear predictor with the linear-in-the-parameter characteristic is illustrated in Fig. 2.1.

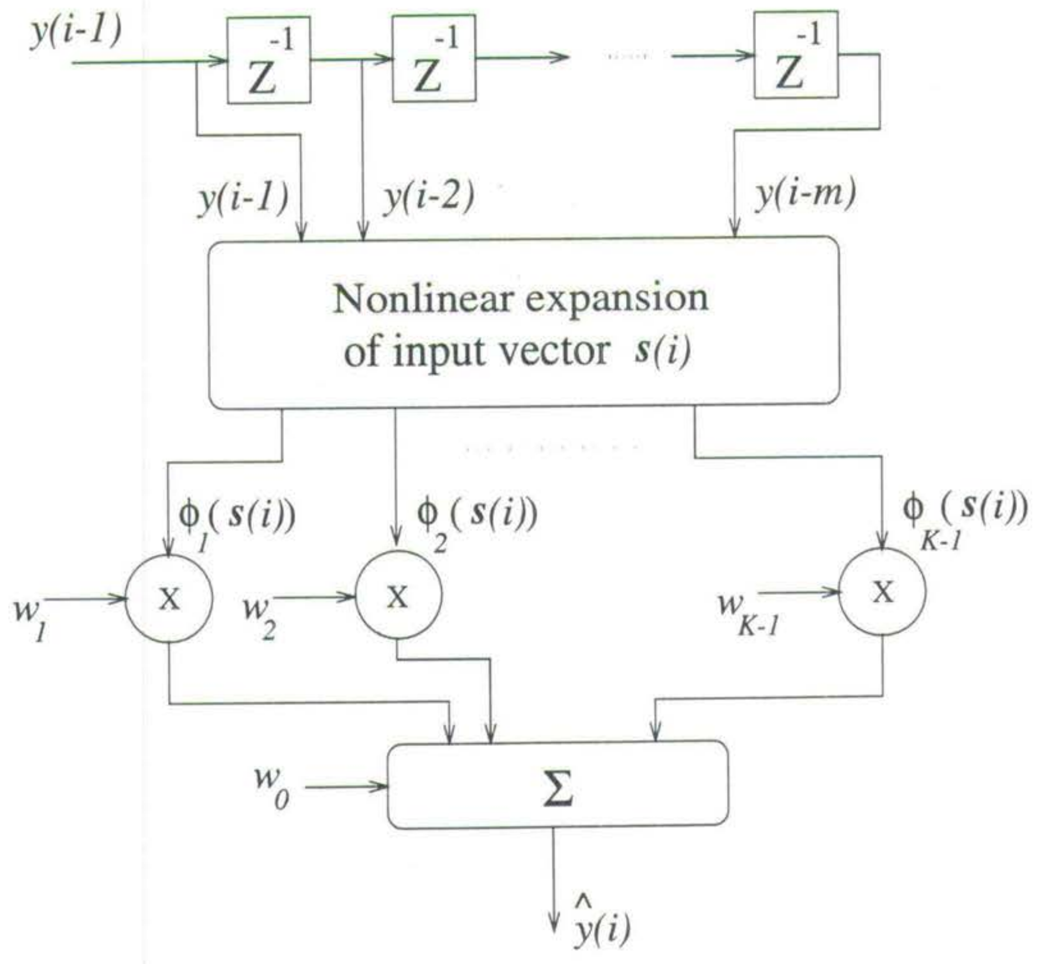

Figure 2.1: Generic model of a filter with the linear-in-the-parameter structure.

By defining

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(1) \\ \mathbf{x}(2) \\ \vdots \\ \mathbf{x}(N) \end{bmatrix} = \begin{bmatrix} 1 & \phi_1(\mathbf{s}(1)) & \phi_2(\mathbf{s}(1)) & \cdots & \phi_{K-1}(\mathbf{s}(1)) \\ 1 & \phi_1(\mathbf{s}(2)) & \phi_2(\mathbf{s}(2)) & \cdots & \phi_{K-1}(\mathbf{s}(2)) \\ \vdots & \vdots & \cdots & \vdots & \\ 1 & \phi_1(\mathbf{s}(N)) & \phi_2(\mathbf{s}(N)) & \cdots & \phi_{K-1}(\mathbf{s}(N)) \end{bmatrix}, \tag{2.15}$$

it is clear that the above information matrix $\mathbf{X}$ for the nonlinear predictor has an identical structure to the information matrix generated for the linear AR model (Eq. 2.5). Therefore the LS weights of the nonlinear model may be evaluated similarly using Eq. 2.12 or Eq. 2.13. The following sections discuss the Volterra and RBF techniques of performing nonlinear expansions.

2.2.1 The Volterra expansion

A classical nonlinear filter is the Volterra model [5,33,41]. The introduction of nonlinearity using the Volterra expansion is based on the application of quadratic, cubic, and higher combinations of input values into the filter's function. Such an expansion technique is very similar to the use of a truncated Taylor series with memory elements to model an analytic function. Like the Taylor series, if the series does not converge, the filter created may only be valid for certain ranges of output. This is however not a significant problem as the Volterra model can characterise a large class of nonlinear functions and systems [6].

However, in spite of its long history and popularity in theoretical studies, relatively few researchers have applied the Volterra filtering techniques to practical applications [4–6, 41, 42]. The main reason for this is that the Volterra expansion method usually results in the formation of very large models which require huge processing complexity for their implementations. To make the Volterra implementation more practical, some researchers [5, 43] have suggested restricting the Volterra expansions to include only quadratic combinations. This, however, may not be a good solution if higher combinations of polynomials are desired in the modelling. Our approach to reducing complexity without sacrificing higher nonlinear expansions is to employ model selection techniques to remove less important regressors from the Volterra expansions.

The Volterra predictor of degree L and order m is created using the expansion shown below,

$$\begin{aligned} \hat{y}(i) &= w_0^{'} + \sum_{j_1=1}^{m} w_{j_1}^{'} y(i-j_1) + \sum_{j_1}^{m} \sum_{j_2 \geq j_1}^{m} w_{j_1 j_2}^{'} y(i-j_1) y(i-j_2) + \cdots \\ &\quad + \sum_{j_1}^{m} \sum_{j_2 \geq j_1}^{m} \cdots \sum_{j_L \geq j_{L-1}}^{m} w_{j_1 j_2 \ldots j_L}^{'} y(i-j_1) y(i-j_2) \ldots y(i-j_L) \end{aligned} \tag{2.16}$$

where

- L is the degree of the Volterra filter. It denotes the highest power of combinations for the past values used in the nonlinear expansion.
- m is the order of the Volterra filter. It denotes the number of past values used in the input vector to the filter.
- $\hat{y}(i)$ = predicted value of $y(i)$.
- $w_0^{'}$ = constant.
- Volterra Kernels
  - $w_{j_1}^{'}$ = coefficients for the linear terms, $j_1 = 1$ to order m.
  - $w_{j_1 j_2}^{'}$ = coefficients for the quadratic terms, $j_2 = 1$ to j_1.
  - $w_{j_1 j_2 \cdots j_L}^{'}$ = coefficients for the degree L combinations of past values.
- Past values
  - $y(i - j_1)$ = samples of past values.
  - $y(i - j_1)y(i - j_2)$ = quadratic combinations of past values.
  - $y(i - j_1)y(i - j_2)\ldots y(i - j_L)$ = degree L combinations of past values.

Fig. 2.2 illustrates a degree L=2, order m=3, Volterra predictor. The application of degree 2 implies that the model is restricted to having only quadratic terms in the expansions. Observe that the expansion not only includes all the original input signals to the predictor but all the possible quadratic combinations as well. By replacing the notation $\mathbf{w}^{'} = [w_0^{'}\ w_1^{'}\ w_2^{'}\ w_3^{'}\ w_{11}^{'}\ \cdots\ w_{33}^{'}] \in R^{10}$ used to define the weights of the Volterra filter as $\mathbf{w} = [w_0\ w_1\ \cdots\ w_9] \in R^{10}$, it is obvious that the realised Volterra structure is identical to the generic model of a nonlinear filter with the linear-in-the-parameter structure illustrated in Fig. 2.1.

From the above example, it is evident that the number of regressors involved in the Volterra expansion can become very large. The formula to calculate the number of regressors for a degree L, order m, Volterra predictor is :

$$\text{No of regressors } K = 1 + \sum_{j=0}^{L} n_j, \tag{2.17}$$

$$\text{where } n_0 = 1, \tag{2.18}$$

$$n_j = n_{j-1}\left(\frac{m+j-1}{j}\right), \qquad j = 1, 2, \ldots, L. \tag{2.19}$$

The equation shows that the number of Volterra regressors increases exponentially with respect to the degree and order, i.e., L and m, of the model. Fig. 2.3 illustrates the resultant size of the Volterra filter with respect to the degree and order specifying the filter. Note that the application of high degree values $L \geq 3$ quickly results in very large models.

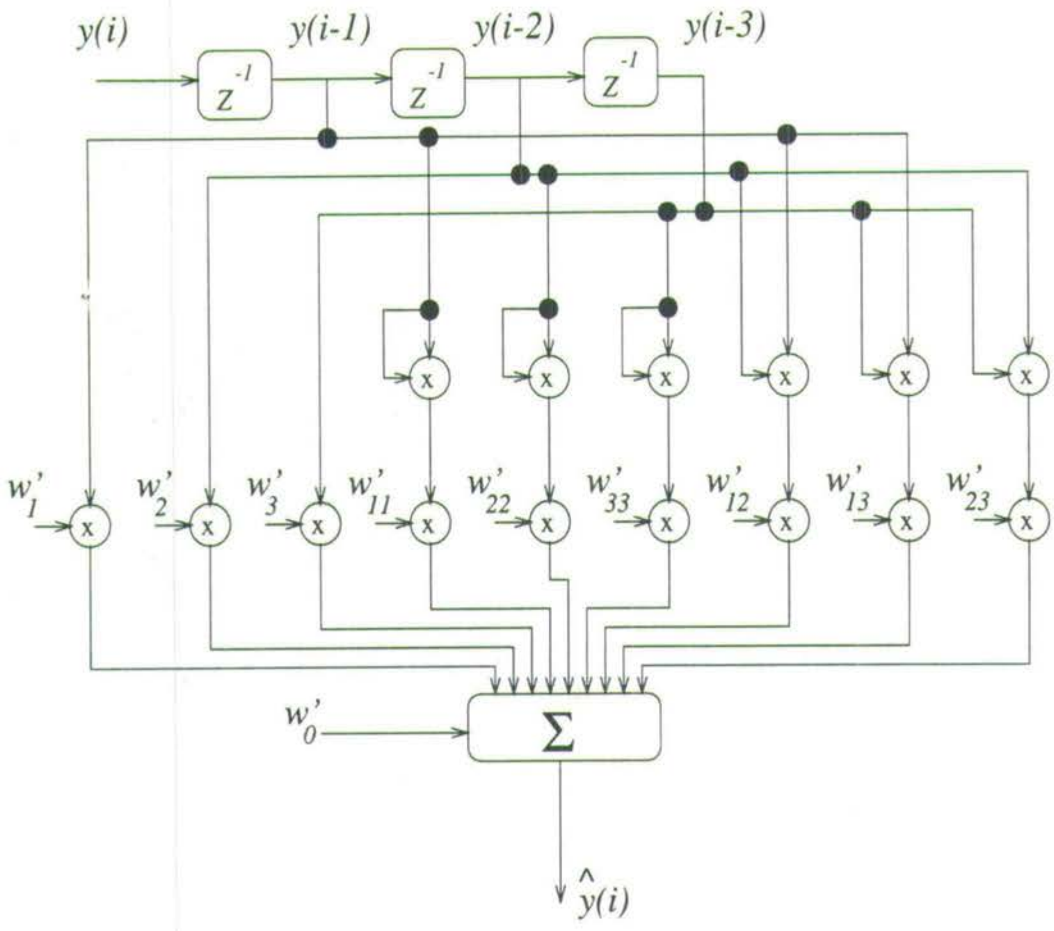

Figure 2.2: Volterra predictor with degree $L = 2$ order $m = 3$.

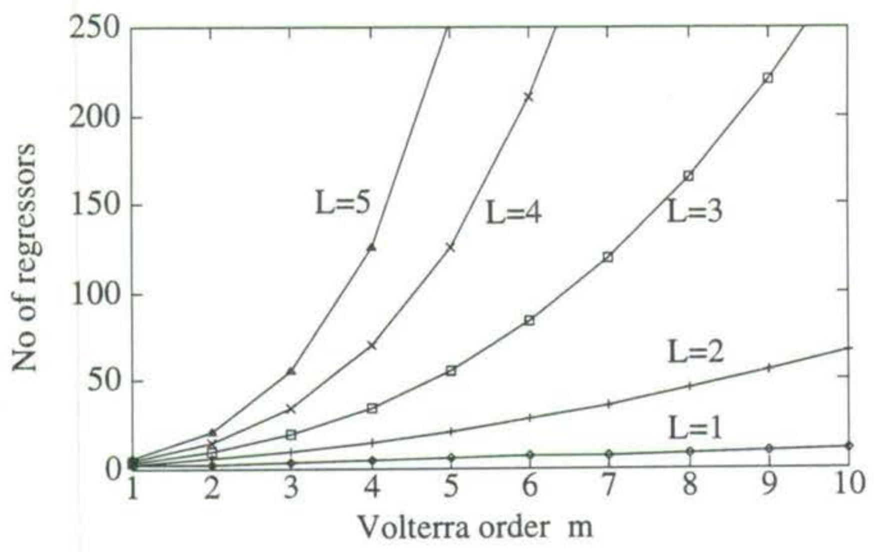

Figure 2.3: Number of terms generated given Volterra parameter degree L, and order m.

2.2.2 The RBF expansion

This section introduces the RBF model [7–9, 36]. As in the previous section, to facilitate the discussion of the RBF model structure, the application of the filter to single step time series prediction is studied. The regression model of a RBF filter to predict the current signal $y(i)$ given a vector of past values $\mathbf{s}(i) = [y(i-1)\ y(i-2)\ \cdots\ y(i-m)]^T \in R^m$ is

$$\hat{y}(i) = f(\mathbf{s}(i)) = w_0 + \sum_{j=1}^{K-1} \phi_j(\mathbf{s}(i))w_j, \tag{2.20}$$

where $\hat{y}(i)$ is the RBF model's approximation of the actual value $y(i)$, $\mathbf{w} = [w_0\ w_1\ \ldots\ w_{K-1}]^T$ are the filter's weights, and $\phi_j(.)$, $1 \leq j \leq K-1$, are the hidden nodes which introduce the nonlinear transformation $R^m \rightarrow R$. These nonlinear expansions $\{\phi_j(.)\}_{j=1}^{K-1}$ are known by the neural networks (NN) community [9, 44] as the hidden layer of a special two-layer network. The structure of the RBF predictor is illustrated in Fig. 2.4.

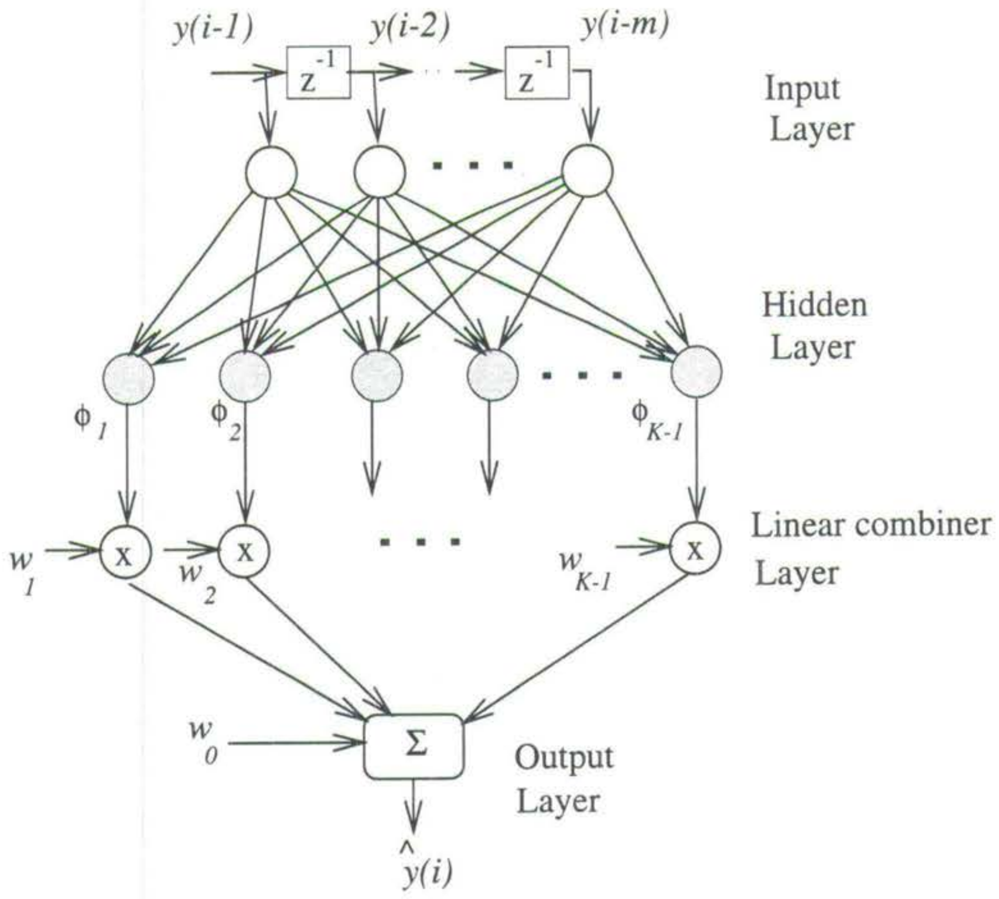

Figure 2.4: Structure of a RBF predictor.

The outputs of the hidden nodes are specified by

$$\phi_j(\mathbf{s}(i)) = \phi(||\mathbf{s}(i) - \mathbf{c}_j||/\alpha_j), \qquad 1 \le j \le K-1, \tag{2.21}$$

where $\mathbf{c}_j \in R^m$ are called the RBF centres, α_j are positive scalers known as widths, and $||.||$ denotes Euclidean norm. The reason for the name radial basis function is the characteristic of nonlinearity $\phi(.)$ which has a response that is radially symmetric shape around the centre in R^m space. For general applications, the nonlinearity $\phi(.)$ can be chosen from a wide class of nonlinear functions. Typical choices include [7–9, 44]

$$\begin{aligned}
\phi_j(r) &= \exp(-r^2/\alpha_j^2), \qquad \alpha_j > 0, && (2.22)\\
\phi_j(r) &= \frac{1}{(\mathbf{c}_j^2 + r^2)^{\alpha_j}}, \qquad \alpha_j > 0, && (2.23)\\
\phi_j(r) &= (\mathbf{c}_j^2 + r^2)^{\alpha_j}, \qquad 0 < \alpha_j < 1, && (2.24)\\
\phi_j(r) &= r^2 \log(r), && (2.25)
\end{aligned}$$

where $r = ||\mathbf{s}(i) - \mathbf{c}_j||$ is the input parameter to the nonlinear function.

Although the listed nonlinear functions exhibit very different properties, theoretical investigations [7, 45] and practical results [10, 36, 46] have shown that the type of nonlinearity is not crucial to the performance of the RBF network and a uniform width α is sufficient for every hidden node for universal approximation [45]. RBF models based on any class of nonlinearity have excellent approximation capabilities.

To cast the RBF response as a linear regression model, we define

$$\begin{aligned}
\mathbf{y} &= [y(1)\ y(2)\ \cdots\ y(N)]^T, && (2.26)\\
\hat{\mathbf{y}} &= [\hat{y}(1)\ \hat{y}(2)\ \cdots\ \hat{y}(N)]^T, && (2.27)\\
\mathbf{e} &= [e(1)\ e(2)\ \cdots\ e(N)]^T, && (2.28)\\
\mathbf{w} &= [w_0\ w_1\ \cdots\ w_{K-1}]^T, && (2.29)\\
\mathbf{x}(i) &= [1\ \phi_1(\mathbf{s}(i))\ \phi_2(\mathbf{s}(i))\ \cdots\ \phi_{K-1}(\mathbf{s}(i))], \qquad 1 \le i \le N, && (2.30)\\
\mathbf{X} &= \begin{bmatrix} \mathbf{x}(1) \\ \mathbf{x}(2) \\ \vdots \\ \mathbf{x}(N) \end{bmatrix} = \begin{bmatrix} 1 & \phi_1(\mathbf{s}(1)) & \cdots & \phi_{K-1}(\mathbf{s}(1)) \\ 1 & \phi_1(\mathbf{s}(2)) & \cdots & \phi_{K-1}(\mathbf{s}(2)) \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \phi_1(\mathbf{s}(N)) & \cdots & \phi_{K-1}(\mathbf{s}(N)) \end{bmatrix}, && (2.31)
\end{aligned}$$

to obtain the following equation

$$\mathbf{y} = \hat{\mathbf{y}} + \mathbf{e} = \mathbf{X}\mathbf{w} + \mathbf{e}, \tag{2.32}$$

again, the weights of the model may be solved by the LS solution,

$$\mathbf{w} = \mathbf{X}^{+}\mathbf{y}. \tag{2.33}$$

The traditional application of the RBF network is for strict-interpolation problems in multi-dimensional spaces [7, 8, 47]. In such application, all the data points will be used to form the centres. As the number of data points available is usually very large, this is rarely practical. Broomhead and Lowe [9] removed the restriction of using all the data as centres, and made use of the RBF network as an estimator. Such an implementation provided a more suitable basis for signal processing applications. In their implementation, the centres were chosen randomly from the available data. Clearly, this is unsatisfactory [48]. A more rational approach would be to consider all the training data $\{\mathbf{s}(i)\}_{i=1}^{N}$ as candidate centres and formulate the problem of centre selection as subset model selection. Since the RBF regression model is linear-in-the-parameter, linear subset modelling techniques may be applied.

2.3 Subset model selection

The initial models formed using Volterra and RBF expansions are often very large. This is necessary in non-parametric modelling techniques as the initial model should contain a sufficient number of nonlinearities to allow for the identification of any unknown nonlinear system.

However in most practical cases, the unknown system only requires a very small subset of nonlinearities from the initial model [48]. To reduce model size, model selection techniques can be applied. Parsimonious models are desirable not only because they avoid ill-conditioning problems during parameter learning, but also because they offer better generalisation ability for applications such as time series prediction [49].

The optimum technique for selecting $R < K$ regressors from an initial model with K regressors is to form all the possible combinations of R subset models and pick the best one [24,50]. This requires a exponential amount of computation and would be impossible to implement for even modest sizes of K and R. A more practical approach is to use selection algorithms based on a step-wise selection technique. There are two major types of implementation of this technique, namely, forward selection and backward-elimination [24, 50, 51]. These methods are based on either adding or removing regressors one at a time to generate the subset model. If $R \ll K$, forward selection is computationally much simpler than backward-elimination. As most of our problems fall into this category, we will concentrate on the forward selection technique.

2.3.1 Forward selection technique : the OLS algorithm

To select R parameters to form the subset model, the forward selection algorithm begins with a subset model with no regressors. The regressors of the original model are then examined to identify which contribute the most to the modelling of the desired signal vector according to some criteria. The regressor from the initial model which satisfies the desired criterion most becomes the first regressor of the subset model. The next regressor considered for inclusion into the subset model is the one that yields the best combination with the currently selected subset regressors to model the desired vector. This selection process is repeated until the required size of the subset model is reached.

However, the forward selection method is known to be sub-optimum [50]. Nevertheless, this is not a serious problem as many researchers have indicated that very good subset models can be found for practical applications using this method [10, 11, 50, 52]. Coupled with its implementation simplicity, the forward selection method has become very popular.

One efficient implementation of the forward selection algorithm is the orthogonal least squares (OLS) method [10, 11, 53, 54]. Let us first review the key to the OLS implementation, the function of orthogonal projection. Fig. 2.5 shows two vectors $\mathbf{u}, \mathbf{v}$ in R^2 space. The orthogonal projection of $\mathbf{u}$ on $\mathbf{v}$, i.e. $Proj_{\mathbf{v}}\mathbf{u}$, can be thought of as the shadow formed by vector $\mathbf{u}$ onto $\mathbf{v}$ when an imaginary light is directed to $\mathbf{u}$ along the normal of $\mathbf{v}$; The vector $Proj_{\mathbf{v}}\mathbf{u}$ is the best approximation of $\mathbf{u}$ using $\mathbf{v}$. The following are the definitions of orthogonal projection and the equation to find the orthogonal residual.

$$Proj_{\mathbf{v}}\mathbf{u} = \mathbf{v}\left(\frac{\mathbf{u}^T\mathbf{v}}{||\mathbf{v}||^2}\right), \tag{2.34}$$

$$\mathbf{w}_{\mathbf{v}} = \mathbf{u} - Proj_{\mathbf{v}}\mathbf{u}, \tag{2.35}$$

where $||\mathbf{v}|| = \sqrt{\mathbf{v}^T\mathbf{v}}$ is the Euclidean norm of $\mathbf{v}$ and the vector $\mathbf{w}_{\mathbf{v}}$ is the residual when $\mathbf{v}$ is used to approximate $\mathbf{u}$.

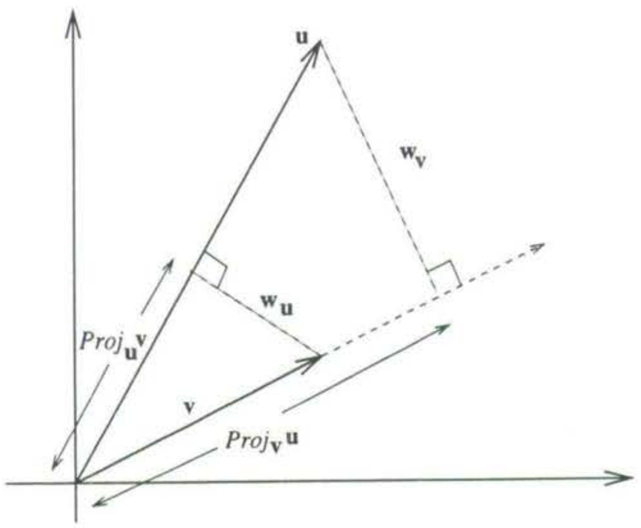

Figure 2.5: Orthogonal projections

In this work, the criterion used in the model selection algorithm is the normalised MSE (NMSE) criterion, i.e.,

$$\text{NMSE} = 10\log_{10}\left(\frac{\sum_{i=1}^{N} e(i)^2}{\sum_{i=1}^{N} y(i)^2}\right), \tag{2.36}$$

where $y(i)$ is the desired signal value at sample i, and $e(i) = y(i) - \hat{y}(i)$. When there is perfect prediction, i.e. $e(i) = 0$ for all i, the NMSE will be $-\infty$ dB. When there is no prediction, i.e. $\hat{y}(i) = 0, e(i) = y_i$ for all i, the NMSE will be 0 dB.

The OLS algorithm is listed in the following paragraphs. The following notation is used in the algorithm,

(i). $\mathbf{X} = [\mathbf{c}_0\ \mathbf{c}_1\ \cdots\ \mathbf{c}_{K-1}] \in^{N\times K}$ is the information matrix of the original model. The vectors $\mathbf{c}_i \in R^N$ denote the i^{th} column of the matrix.

(ii). $\mathbf{y} \in R^N$ is the desired output vector.

(iii). R is the desired size of the subset model.

(iv). Col_Found[$1\ldots R$] is the vector produced by the algorithm which lists the regressors selected from the initial matrix $\mathbf{X}$ by their column number.

Algorithm 2.1: OLS

(i). Initialise

- no_col_found = 0; Col_Found[$1\ldots R$] = 0;
- copy the vector $\mathbf{y}$ to $\mathbf{y}_{original}$.

(ii). Calculate the residual vector

- For columns $i = 0$ to $(K-1)$ of the input matrix $\mathbf{X}$ that are unused,
  - Project $\mathbf{y}$ onto each column $\mathbf{c}_i$
  - $\mathbf{residual}_i = \mathbf{y} - Proj_{\mathbf{c}_i}\mathbf{y}$.

(iii). Select a column

- The column selected $\mathbf{c}_j$ to be included in the subset model is the column that has the smallest Euclidean norm for the residual vector, $\mathbf{residual}_j$.

$$j = \min_i\{\|\mathbf{residual}_i\|\}, \qquad i = \text{unselected columns of } \mathbf{X} \tag{2.37}$$

The NMSE of the selected subset model can be calculated by using $20\log_{10}\left(\frac{\|\mathbf{residual}_j\|}{\|\mathbf{y}_{original}\|}\right)$.

- no_col_found = no_col_found + 1
  Col_Found[no_col_found] = j.
  This array Col_Found stores the indexes of the selected columns of $\mathbf{X}$.
  Mark the selected column as used.
- Update $\mathbf{y}$ as $\mathbf{residual}_j$, i.e. $\mathbf{y} = \mathbf{y} - Proj_{\mathbf{c}_j}\mathbf{y}$. This new $\mathbf{y}$ can be thought of as the unexplained desired output of the present model.

(iv). Update the $\mathbf{X}$ matrix
For columns $i = 0$ to $(K-1)$ of $\mathbf{X}$ that are unselected,
update each column $\mathbf{c}_i$ as follows:

$$\mathbf{c}_i = \mathbf{c}_i - Proj_{\mathbf{c}_j}\mathbf{c}_i. \tag{2.38}$$

This is to remove the contribution already *given* by the selected column $\mathbf{c}_j$.

(v). Repeat steps (ii-iv) until the number of columns found is equal to R, the desired number of coefficients for the subset model.

Remark: The OLS algorithm is very similar to the Gram-Schmidt orthogonalisation scheme [55]. The only difference is that instead of orthogonalising the $\mathbf{X}$ matrix in the order of the first column to the last column, we now first orthogonalise the columns which have bigger contribution to the approximation of the desired output vector $\mathbf{y}$.

The above algorithm returns an array Col_Found[$1 \ldots R$] indicating the selected regressors from the initial model for the subset model. The weights of the subset model $\mathbf{w}_s$ can be found by forming a matrix $\mathbf{X}_s$ which contains the columns indicated by Col_Found[$1 \ldots R$], and evaluating using the LS solution,

$$\mathbf{w}_s = \mathbf{X}_s^+ \mathbf{y}_{original}. \tag{2.39}$$

Comments on the Orthogonal Least Squares Algorithm

The OLS method is a powerful and efficient method of performing model selection. The ability to find good subset regressors with only linearly increasing computational complexity makes this method attractive for practical implementations. Nevertheless, steps (ii-iv) of the algorithm require heavy computation, particularly when the $N \times K$ matrix $\mathbf{X}$ is large. If $N \gg K$, it may be possible to reduce the computational requirement. This is examined in the next chapter.

The OLS algorithm has another useful property. As the new subset model is formed by adding an additional column to the previously selected model, the NMSE performance will always decrease monotonically.

2.4 Time series prediction

The applications of filters to predict the future values of time series are many. Examples include forecasting of the stock market shares index [56,57], economic data [58], river-flow level, weather, sun-spot series [28, 59] and many more. As such, there has been tremendous amount of work done in this field. The reader is urged to consult the following review articles and books for a more extensive treatment of the work [2, 29, 49, 60–62].

The conventional methods for time series prediction were based on linear techniques as summarised in the classic *Time series analysis* by Box and Jenkins [29]. However, towards the end of the 1970's, nonlinear techniques began to be more widely applied. The earlier methods were mainly based on the introduction of simple quadratic [63], higher combinations of polynomials [4, 49], and exponential nonlinearity [64] into the auto-regressive models. In 1983, a major idea was introduced by Tong [60, 65], who proposed using threshold models to characterise different disjoint regions of the input space. In recent years, a special class of nonlinear filters that possess the linear-in-the-parameter structure have gained prominence. These nonlinear filters include structures like the Volterra filters [4, 5, 33], RBF networks [7–9, 36], NARMAX models [37, 38], multivariate adaptive regression splines (MARS) [26, 39] and fuzzy basis function models [40]. Another important class of predictors based on NN techniques is the multi-layer perceptron (MLP) [66] which has also seen wide-spread use.

Time series prediction can also be considered from the viewpoint of dynamic systems [61,62,67]. By that, the time series is assumed to be driven by an underlying deterministic system $g(.)$. If $g(.)$ is known, the entire future evolution of the time series $\{y(i), y(i+1), \ldots, y(\infty)\}$ can be found if the current state $\mathbf{s}(i) = [y(i-1)\ y(i-2) \cdots\ y(i-m)] \in R^m$ is given. The problem, therefore, is to find an approximation of $g(.)$ based on the available training data. To reconstruct the time series given $g(.)$, knowledge of the past $m \geq 2d+1$ samples is required [68]. The integer d is the fractal dimension of the time series [68]. This process is called state-space reconstruction [69].

If the functional structure of $g(.)$ is known, a parametric model can be created, and the problem then is to solve for the parameters of the model based on the available training data. However in most cases, the underlying dynamics are unknown, and the requirement is to find a model $f(.)$ to characterise the actual dynamics $g(.)$ based on available training data. If the approximation of $f(.)$ to $g(.)$ is good, the future values of the time series may be predicted based on the known observations of the current past m observations.

In this thesis, we shall adopt the state-space reconstruction approach, and assume that the underlying dynamics of the time series generator is unknown.

2.4.1 Simulation results : OLS algorithm and the linear and nonlinear predictors

To illustrate the application of the OLS model selection algorithm, and the difference in performance between a linear and nonlinear predictor, we examine the performance of the predictors for two different time series which possess nonlinear dynamics, namely the Duffing's and the Mackey-Glass chaotic time series [70]. The nonlinear predictor used is based on the Volterra expansion created using the parameters degree $L = 3$, embedding-vector length $m = 6$[1];. This expansion results in a 84 parameter model. To give a fair comparison of performance, a linear predictor of order 84 is used. The NMSE criterion is used to measure the performance of the selected subset model.

The chaotic time series were created using the following equations:

- Duffing's time series (Fig. 2.6)

$$\frac{d^2s(t)}{d^2t} + \alpha\frac{ds(t)}{dt} - s(t) + s^3(t) = \beta cos(t) \tag{2.40}$$

  where $\alpha = 0.25$, $\beta = 0.3$, $s(0) = 0$, $\frac{ds(0)}{dt} = 0$ and step size = 0.2 . Gaussian white noise is added to this series to create a signal to noise ratio[2] (SNR) of +50dB.

- Mackey-Glass time series (Fig. 2.7)

$$\frac{ds(t)}{dt} = -bs(t) + \frac{\alpha s(t-\tau)}{1+s^{10}(t-\tau)} \tag{2.41}$$

  where $\tau = -21$, $\alpha = 0.2$, $b = 0.1$, initial conditions $s(t-\tau)$= 0.5 for $0 \le t \le \tau$ and step size = 2. Gaussian noise is added to create a SNR of +50dB.

The OLS algorithm was used to select subset models from the 84-term linear predictor and the 84-term nonlinear Volterra predictor. The NMSE performance of each model for the two time series prediction problems is shown in Figs. 2.8 and 2.9. The performance of a simple linear predictor created by increasing the linear predictor order from K=1 to K=84 is also included. The horizontal axis of Figs. 2.8 and 2.9 shows the number of parameters used to create the subset model, and the vertical axis indicates the NMSE attained by the subset model. Both results demonstrate that

(i). The performance of the linear models selected by the OLS algorithm was better than that of the linear model without selection.

(ii). The nonlinear predictor can improve signal prediction performance.

[1] Size 6 was chosen to satisfy Takens' theorem [68]

[2] SNR is calculated using the following equation,

$$\text{SNR} = 10\log_{10}\frac{\sigma_s^2}{\sigma_e^2}$$

where σ_s^2 is the variance of the time series, and σ_e^2 is the variance of the additive Gaussian noise.

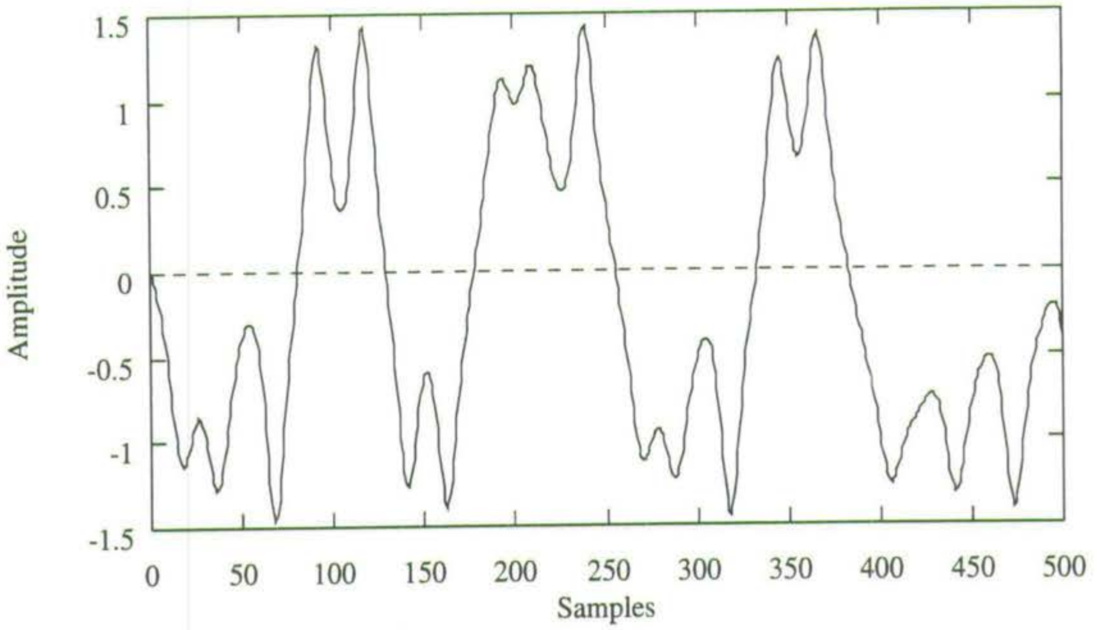

Figure 2.6: Duffing's chaotic time series

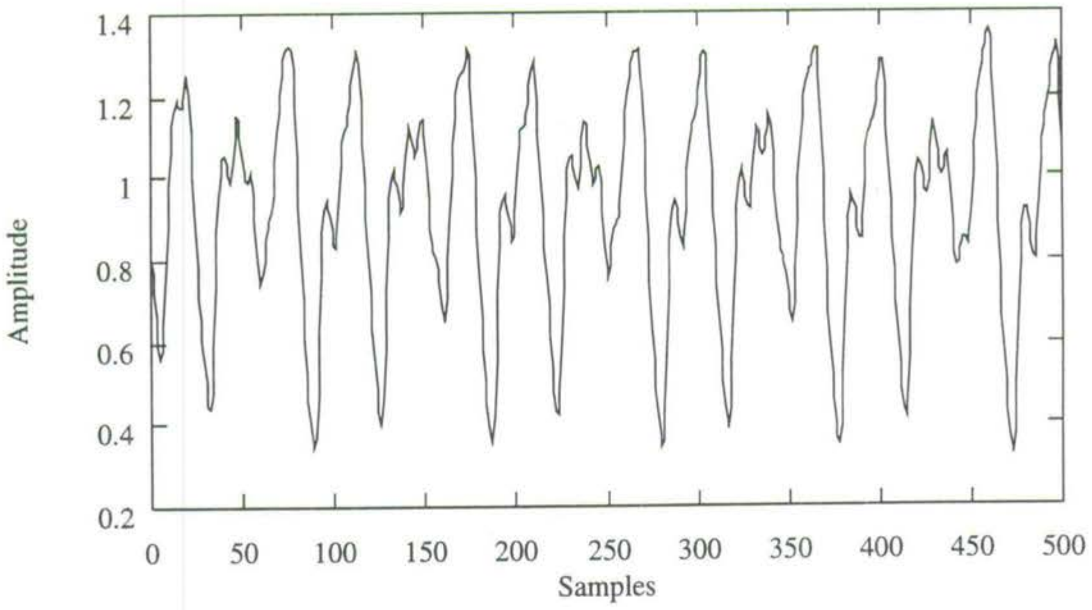

Figure 2.7: Mackey-Glass chaotic time series

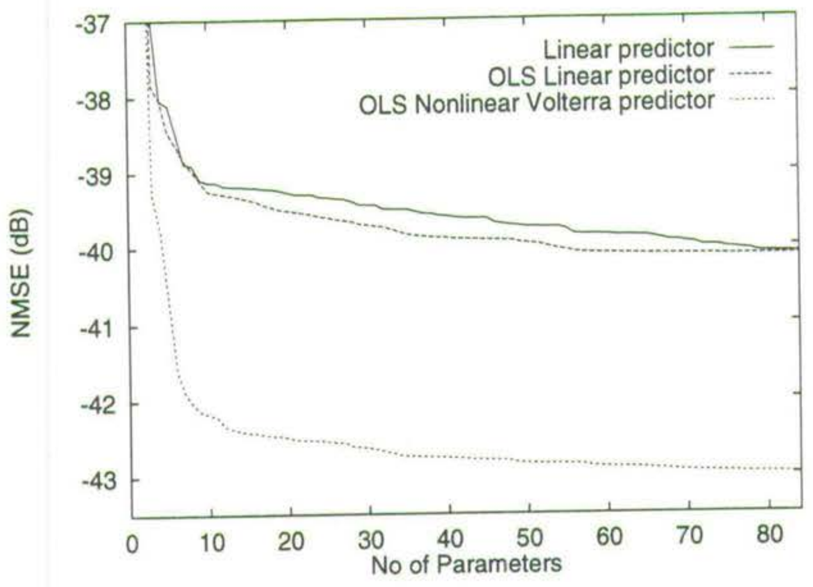

Figure 2.8: Linear vs nonlinear predictors performance (Duffing's series)

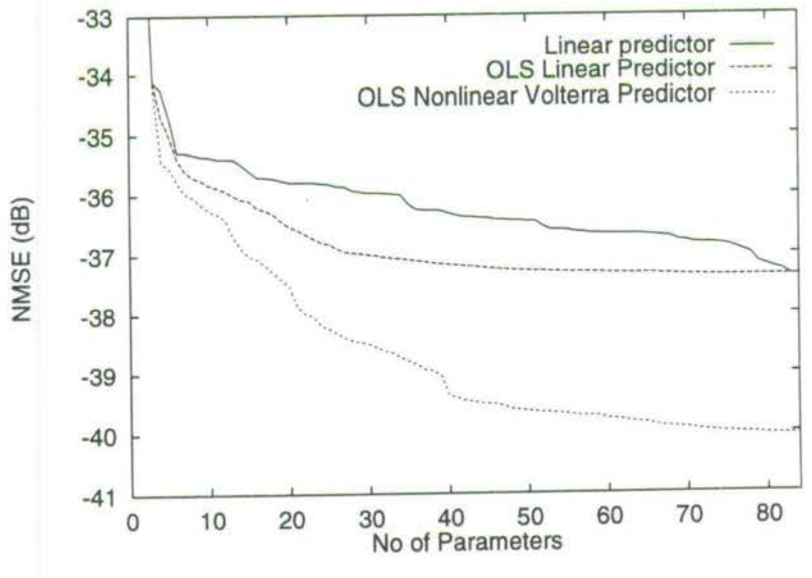

Figure 2.9: Linear vs nonlinear predictors performance (Mackey-Glass series)

2.5 Channel Equalisation

The other application problem which we will consider is that of channel equalisation. Channel equalisation is the technique employed to combat the effects of inter symbol interference (ISI) and noise which corrupt the transmission of signals across a communication channel. The task of the equaliser is to reconstruct the transmitted sequence with the minimum probability of misclassification. The communication system we shall examine in this thesis is a discrete linear baseband system as illustrated in Fig. 2.10, where the channel represents the combination of a transmitter filter, a transmission medium and a receiver filter. This transmission model can represent a wide variety of communication models, e.g. linear baseband channel like the twisted pair wire and coaxial cable, or bandpass channel such as the telephone modem line, and HF radio link [3, 71, 72].

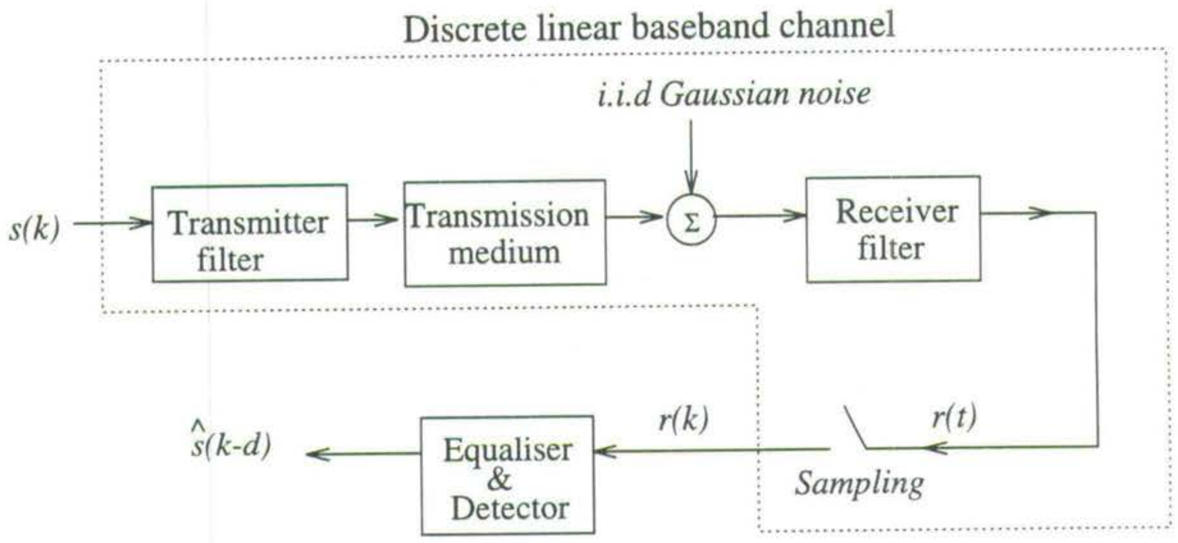

Figure 2.10: Model of a data transmission system

The discrete linear baseband channel is usually modelled by a FIR filter [3, 71, 72] with the following transfer function

$$H(z) = \sum_{i=0}^{n_a-1} a(i) z^{-i}, \tag{2.42}$$

where $a(i)$ are the channel impulse response coefficients and n_a is the length of the channel impulse response. For the study, the channel coefficients are restricted to be real and the transmitted symbol $s(k)$ is taken from the set $\{\pm 1\}$. This corresponds to the use of a 2-ary pulse amplitude modulation (PAM) scheme [71]. Focusing on this simple case allows us to highlight the basic principles and concepts more clearly. Moreover, the techniques studied in the 2-ary PAM scheme can be extended without any difficulties to more complicated modulation techniques e.g. quadrature amplitude modulation (QAM), phase shift keying (PSK) [73–75].

The received signal $r(k)$ to the equaliser (Fig. 2.10) at time k is given by

$$\begin{aligned} r(k) &= \sum_{i=0}^{n_a-1} a(i)s(k-i) + n(k) \\ &= a(0)s(k) + \sum_{i=1}^{n_a-1} a(i)s(k-i) + n(k), \end{aligned} \tag{2.43}$$

where $s(k)$ is the independently identically distributed (i.i.d) transmit signal and $n(k)$ is an additive i.i.d Gaussian noise with zero mean and variance σ_e^2. Eq. 2.43 shows that the received signal $r(k)$ for the transmitted signal $s(k)$ is attenuated by a factor of $a(0)$, and corrupted by additive noise $n(k)$ and the dispersion of energy from neighbouring symbols $\sum_{i=1}^{n_a-1} a(i)s(k-i)$.

The performance of the receiver using $r(k)$ to detect the transmitted signal without compensating for the ISI effects will be very poor. To improve the receiver's performance, various techniques for channel equalisation have been developed during the past three decades [3, 71]. Fig. 2.11 lists some of the available equalisation techniques. The diagram shows two main classes of equalisers; the equalisers listed in the left column are based on symbol-decision detection process, and the equalisers listed in the right column are based on a block or sequence detection process.

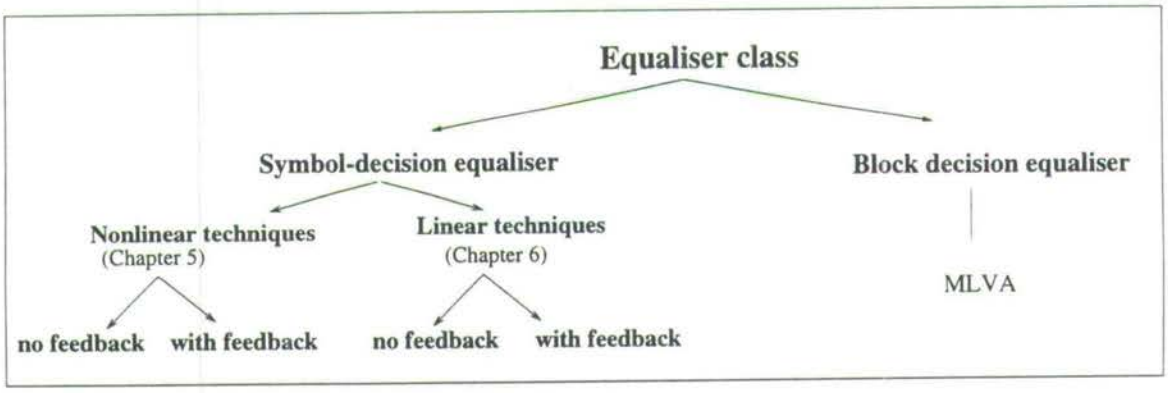

Figure 2.11: General breakdown of equalisation techniques

We first discuss the block detection technique. It is well-known that block detection equalisation based on the principle of maximum likelihood sequence estimator (MLSE) of the entire transmitted sequence will provide the best classification performance when the channel's statistics are completely known [3, 76]. The optimum solution for this class of equaliser is the maximum likelihood Viterbi algorithm (MLVA) [77] which determines the estimated symbol sequence by the following cost function,

$$J_{MLVA} = \sum_{k=1}^{\infty} \left(r(k) - \sum_{i=0}^{n_a-1} a(i)\hat{s}(k-i) \right)^2 \tag{2.44}$$

where $\hat{s}(k)$ is the estimated symbol at time k. Although the MLVA can provide the best performance given perfect knowledge of the channel statistics, its implementation complexity is very large. This is one of the main reasons why the symbol-decision class equaliser which requires significantly simpler implementation is used even though its performance is poorer than that of the block decision class of equaliser. Symbol-decision detectors are also preferred in the situation when the channel is severely non-stationary, e.g., mobile communications. In [74], it was reported that performance of adaptive MLVA degrades seriously in highly non-stationary environments, and better performance may be achieved by a Bayesian symbol-decision equaliser. This is because the MLVA suffers from the drawback of accumulating channel tracking error during decoding of block sequences unlike the symbol decision class equaliser which decodes the transmitted symbols at each symbol period. However, methods of improving the MLSE for fast time varying channel are now receiving wide-spread attention. For more detailed discussion of this type of equalisation technique, refer to [78, 79]. The reader interested in examining block decoding equalisation techniques can refer to the following references [3, 77–82].

For the rest of the thesis, only the symbol-decision detector is considered. The name symbol-decision detector is derived from the fact that this class of equaliser makes a symbol detection at each symbol period. Within this class of detector, there exist two different implementations of the equaliser's decision function, i.e, linear and nonlinear techniques. The linear implementation of the equaliser includes the well-known equaliser structure, the linear transversal equaliser (LTE) which does not utilise past detected symbols in the decision function, and the conventional decision feedback equaliser (DFE) [3,83–85] which uses some past detected symbols in its decision function. The nonlinear implementation similarly includes the two different structures, one with decision feedback and the other without. Several different cost functions are available to optimise the equaliser's performance, namely, the peak distortion criterion [3,86,87], the minimum mean square error (MMSE) criterion [3, 71, 88] and the minimum bit error rate (MBER) criterion. The equaliser optimised using the peak distortion criterion is called the zero-forcing (ZF) equaliser. In recent years however, the ZF equaliser has become less popular [71]. The current implementations of equaliser are normally based on the MMSE and MBER criteria.

To illustrate the difference in performance between linear and nonlinear equalisers, simulations were conducted to compare the classification performance of the MMSE linear and the optimum nonlinear equaliser. The channel transfer function used was $H(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$. For the simulations, the transmit symbol was assumed to be from a binary source with constellation $\{\pm 1\}$, the equaliser's feedforward and delay order were chosen to be $m = 4$ and $d = 1$ respectively, and complete knowledge of the channel statistics was assumed. The simulation results are illustrated in Fig. 2.12. The vertical axis of the graph (Fig. 2.12) indicates the probability of misclassification in $\log_{10}$ scale, i.e BER $= \log_{10}$(probability of misclassification), and the horizontal axis indicates the operating SNR condition. The definition of SNR for channel equalisation applications is based on the ratio of received signal power over the variance of the noise affecting the communication channel, i.e, SNR $= 10\log_{10}\frac{\sigma_s^2 \sum_{i=0}^{n_a-1} a(i)}{\sigma_e^2}$, where σ_s^2 is the transmit signal variance, $a(i)$ are the channel impulse response's coefficients, and σ_e^2 is the noise variance. The simulation results show that the Bayesian equaliser out-performs the linear equaliser significantly. For example, at misclassification probability equal to 10^{-4}, the Bayesian equaliser demonstrated a 4.4dB improvement in SNR over the linear MMSE equaliser.

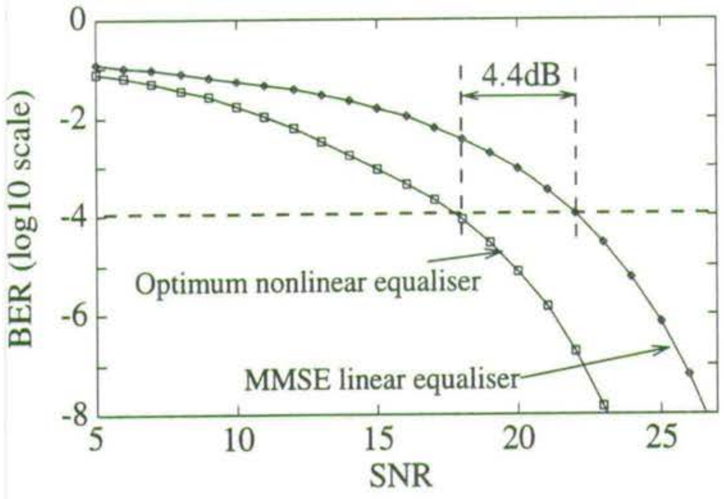

Figure 2.12: Comparison of classification performance between MMSE linear equaliser and the optimum nonlinear symbol-decision equaliser for channel $H(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$. The parameters of the equaliser were $d = 1$ and $m = 4$.

The equaliser structures discussed so far are by no means exhaustive; a considerable number of other structures and adaptive implementations [1, 3] have not been mentioned. In particular, equaliser structures such as fractionally-spaced [89, 90], lattice-filter [1] and infinite impulse response equaliser [91, 92] will not be examined.

To reiterate, this thesis is primarily concerned with nonlinear filter designs. For channel equalisation applications, we will only examine linear and nonlinear filter structures for symbol-decision equalisers.

2.6 Conclusion

In this chapter, we have introduced nonlinear filters which have the linear-in-the-parameter structure for non-parametric modelling. In particular, we considered the Volterra and RBF techniques of generating the nonlinear expansions. Such expansions, however, result in very large initial models which are undesirable due to implementation complexity and poor generalisation ability. To reduce the size of the model, the OLS algorithm of performing model selection was considered. Using simulation results, we showed that nonlinear models, when used to predict nonlinear time series, perform better than linear predictors, and that subset models found using the OLS algorithm can perform adequately. Lastly, a short introduction to time series prediction and channel equalisation was presented.

Chapter 3

Improving the OLS algorithm and RBF model for time series prediction

In Chapter 2, the background to the work performed in the thesis was introduced. In particular, we studied the OLS algorithm for model selection, compared the performance of nonlinear Volterra predictors to that of linear predictors for stationary time series, and introduced the channel equalisation problem. As the emphasis of the previous chapter was to give an overview to the current techniques available, we did not include any of our own findings there. In this chapter, we will present some of our results regarding the OLS algorithm and the application of RBF models to nonstationary time series prediction. The following results are reported in this chapter:

(i). The suboptimum solution of the forward-selection algorithm, and a method of improving the selection process[1].

(ii). The implementation complexity of the OLS algorithm and some methods of reducing the implementation processing requirement[2].

(iii). The RBF predictor's performance on homogeneous nonstationary time series[3].

The organisation of the chapter is as follows:
Sec. 3.1 considers the sub-optimum nature of the forward selection approach and proposes a method of improving the selection process. Sec. 3.2 investigates the implementation complexity

[1]Appeared in - E.S.CHNG, B.MULGREW and S.CHEN, 'Backtracking orthogonal least squares algorithm for model selection', *IEE Colloquium on Mathematical Aspects of Digital Signal Processing*, Feb. 1994, Bristol. Digest No 1994/034, pg 10/1-10/5.

[2]Appeared in - E.S.CHNG, S.CHEN and B.MULGREW, 'Efficient computational schemes for the orthogonal least squares algorithm', *IEEE Trans. Signal Processing*, vol 43, no 1, pg 373-376.

[3]To appear in - E.S.CHNG, S.CHEN and B.MULGREW, 'Gradient radial basis function for nonlinear and nonstationary time series prediction', *IEEE Trans. Neural Networks*, Nov 1995.

of the OLS algorithm and introduces a pre-processing step to reduce the computational requirement. Sec. 3.3 examines the problem of predicting homogeneous nonstationary time series using a RBF model. By the term homogeneous nonstationarity, it is meant that the time series characteristic is similar throughout the time sequence apart frolocal variations of mean and trend. Our simulation results show that the RBF predictor performs relatively poorly for such time series. To improve the performance of the predictor, some modifications to the RBF's node response behaviour are proposed. Some concluding remarks are given in Sec. 3.4.

3.1 Suboptimum nature of the forward selection method

This section investigates the sub-optimum nature of the forward selection method for subset model selection and introduces an implementation which improves the selection process.

It is important to realise that the forward selection method does not guarantee the optimal solution for a R-term subset model taken from an initial K-term model. This method only ensures the best selection of the *next* regressor to be included in the current selected subset model at every step. The term ***greedy algorithm*** [93] can be used to describe this kind of selection technique. The only fool-proof method of selecting an optimal R-term subset model is to try all the possible combinations of R-term models from the full K-term initial model. As there are $K!/(R!(K-R)!)$ possible combinations to form the R-term subset model, it is almost impossible to implement even for modest sizes of R and K [50].

Example: Sub-optimum solution of the forward selection method

The following example is presented to illustrate how the forward selection method can fail in selecting the optimal subset model from the following linear regression model,

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{e} \tag{3.1}$$

where the vector $\mathbf{y} \in R^N$ is the desired signal vector, $\mathbf{X} \in R^{N\times K}$ is the information matrix, $\mathbf{w} \in R^K$ is the weight vector and $\mathbf{e}$ is the approximation error of the regression model to the desired signal vector. Let $\mathbf{X}$ and $\mathbf{y}$ have the following values,

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0.1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix},$$

and the task of the model selection problem be to pick a subset model with two regressors from the initial three regressors model. The columns of $\mathbf{X}$ are numbered as $\mathbf{c}_0, \mathbf{c}_1$ and $\mathbf{c}_2$ respectively. The forward selection algorithm selects columns in the order of $\mathbf{c}_2, \mathbf{c}_0$ and $\mathbf{c}_1$. It is true that the best one-term subset model, i.e. $\mathbf{X}s_1$, is to use column $\mathbf{c}_2$ since it provides the best approximation to $\mathbf{y}$ [4]. The best two-term subset model, $\mathbf{X}s_2$, however, is formed by using columns $\mathbf{c}_0$ and $\mathbf{c}_1$. Since the forward selection type algorithm adds a new column to the previously selected subset, a sub-optimal two-term model using $\mathbf{c}_2$ and $\mathbf{c}_0$ is obtained.

From this example, it is clear that forward selection method is sub-optimal, and the selection process can be improved by not always including the regressors selected in the previous stages. That is, a larger subset model need not necessarily contain all the columns of the smaller subset selected previously.

3.1.1 Backtracking OLS

To improve the selection process of the forward selection algorithm, we examine a backtracking approach [12]. The procedures of the modified selection algorithms are: The OLS algorithm is first applied to select the initial subset model of R terms. As the subset model size is increased from 1 to R, the NMSE, which will decrease monotonically, is calculated and recorded. By examining how each parameter contributes to reducing the approximation error, we can determine if the order of parameter selection should be changed. The idea of the backtrack procedure is to introduce parameters that have provided better performance-gain before those that have provided less significant gains [12]. This idea is very similar to the forward-backward selection method discussed in [50]. The details of the algorithm are as follows,

Algorithm 3.1 Backtrack OLS Algorithm

```
(i).   Use the OLS approach to find the initial $\mathbf{X}s_R$ set. The subscript $R$ of $\mathbf{X}s$ implies that
       the matrix $\mathbf{X}s$ has $R$ columns.

(ii).  For $i = 1$ to $R$
       determine the NMSE values of using $\mathbf{X}s_i$ to approximate $\mathbf{y}$, where $\mathbf{X}s_i$ contains $i$ columns
       selected from $\mathbf{X}$ by OLS algorithm. Store these values into the bench-mark array BM_NMSE,
       i.e., BM_NMSE[$i$] = NMSE attained using the subset model containing $i$ columns. This
       array is used to compare against the performance of subset models selected using the
       backtracking approach. The array drop_NMSE is defined as:
          drop_NMSE[$i$] = BM_NMSE[$i$] - BM_NMSE[$i$-1]

(iii). For $i = 2$ to $R$
       - For $j = i + 1$ to $R$
       if (drop_NMSE[$j$] > drop_NMSE[$i$])
```

[4]The subscript R of $\mathbf{X}s$ imply that the matrix $\mathbf{X}s_R$ has R columns.

(a) Form a $\mathbf{X}_{tmp}$ matrix using the first $i-2$ columns of $\mathbf{X}s_R$ and column j of $\mathbf{X}s_R$. This $\mathbf{X}_{tmp}$ matrix will have $i-1$ columns.
If $i = 2$, $\mathbf{X}_{tmp}$ matrix will be a single column matrix with the column from column j of $\mathbf{X}s_R$.

(b) Re-apply the OLS method to find $\mathbf{X}s'$, where $\mathbf{X}s'$ is the selection of R columns from $\mathbf{X}$ based on the initial set specified in $\mathbf{X}_{tmp}$.

(c) Determine the NMSE values using $\mathbf{X}s'$ to model $\mathbf{y}$ and store it into the array tmp_NMSE. Each element in tmp_NMSE will be compared to its corresponding element in BM_NMSE. Any performance improvement in tmp_NMSE implies that the new subset found using the backtrack procedure is better. The indexes of the columns used to form this subset are stored. The BM_NMSE array is then updated with the new improved values.

(d) Skip the rest of j, i.e. set $j = R+1$.

Remark: Step (iii)a is the key idea to the backtracking algorithm. A parameter that has provided a better contribution to model vector $\mathbf{y}$ albeit in a larger subset model $\mathbf{X}s$ replaces the original selection. Although this exchange is not optimal in forming the i-term subset model $\mathbf{X}s_i$ from the $(i-1)$-term subset model $\mathbf{X}s_{(i-1)}$, it is possible for better combinations of $\mathbf{X}s'$ to be generated when more regressors are selected.

Backtracking OLS simulation results

To illustrate the performance of the selected models using the backtracking OLS algorithm, the experiment of selecting subset models from the 84-term Volterra predictor used on the Duffings and the Mackey-Glass chaotic time series discussed in Sec. 2.4.1 is repeated. The NMSE performance of the selected models for the Duffing's time series using the OLS and backtracking OLS algorithm is depicted in Fig. 3.1, and for the Mackey-Glass time series is depicted in Fig. 3.2. The results in both cases confirmed that the backtracking OLS algorithm can improve on the standard OLS selected subset model's performance. This improvement however is achieved with a significant increase in implementation complexity and it should be noted that in these two examples, the improvement to the predictive performance is only very small. These results are in agreement with many other researchers' observation that forward selection technique normally yield very good subset model selection [10, 11, 50, 52], and that for all practical purpose the OLS solution should suffice.

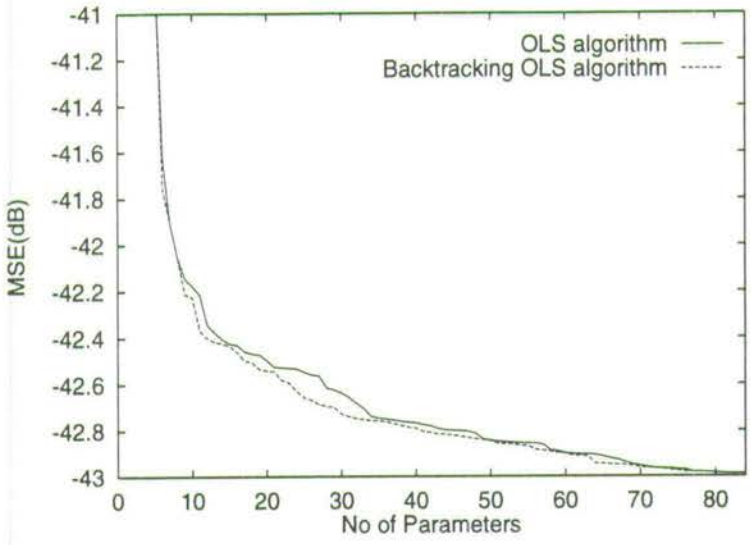

Figure 3.1: OLS and Backtracking OLS on Duffing's series

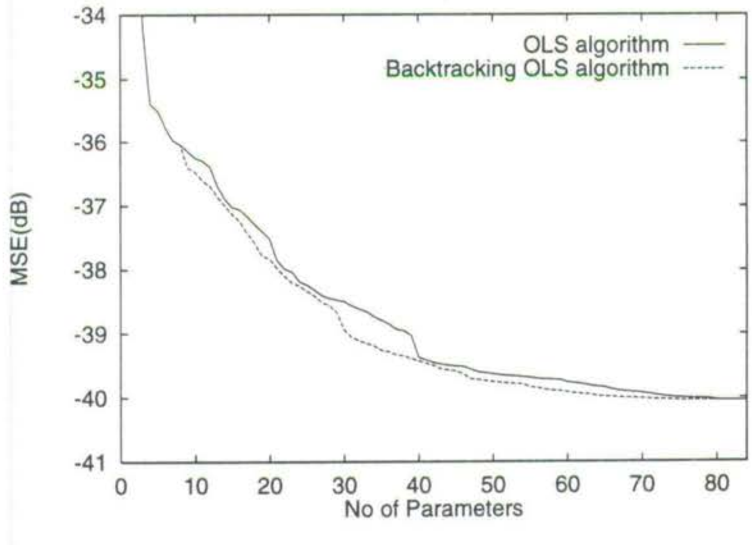

Figure 3.2: OLS and Backtracking OLS on Mackey-Glass series

3.2 Computation reduction of OLS method

Although the OLS algorithm [10,11,53,54] is an efficient implementation of the forward selection method, the implementation complexity of the selection process remains significant when the original model and the number of available data samples is large. In this section, methods of reducing the implementation complexity will be studied. The basic idea is to perform an orthonormal transformation on the $N \times K$ matrix $\mathbf{X}$ and the $N \times 1$ vector $\mathbf{y}$ to change them into a $K \times K$ matrix $\tilde{\mathbf{X}}$ and a $K \times 1$ vector $\tilde{\mathbf{y}}$. The transformation is accomplished by the multiplication process: $\tilde{\mathbf{X}} = \mathbf{Q}^T\mathbf{X}$ and $\tilde{\mathbf{y}} = \mathbf{Q}^T\mathbf{y}$ where the matrix $\mathbf{Q} \in R^{N \times K}$ has the same column space as $\mathbf{X}$. The OLS algorithm is then employed to select significant parameters based on $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{y}}$. Because the size of the transformed information matrix and output vector is smaller than that of the original version, computational complexity is reduced. In normal signal processing problems, the number of data samples, N, is greater than the number of model parameters K. Substantial saving in computation based on $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{y}}$ for subset model selection usually more than offsets the additional computation required for pre-processing. In the case of $N \gg K$, which is certainly not rare, very significant saving in computation can be achieved.

3.2.1 Reduced OLS: Gram Schmidt

One possible way of introducing the orthonormal transformation can be accomplished as follows: Using the Gram-Schmidt (GS) decomposition [31, 94], decompose the $N \times K$ matrix $\mathbf{X}$ into the product of an $N \times K$ matrix $\mathbf{Q}$ satisfying $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ and a $K \times K$ upper triangular matrix $\mathbf{B}$, where $\mathbf{I}$ is the identity matrix of appropriate dimension. That is

$$\mathbf{X} = \mathbf{QB} \tag{3.2}$$

or

$$[\mathbf{c}_0\ \mathbf{c}_1\ \cdots\ \mathbf{c}_{K-1}] = [\mathbf{q}_0\ \mathbf{q}_1\ \cdots\ \mathbf{q}_{K-1}] \begin{bmatrix} b_{00} & b_{01} & \cdots & b_{0(K-1)} \\ 0 & b_{11} & \cdots & b_{1(K-1)} \\ 0 & 0 & \cdots & b_{2(K-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{(K-1)(K-1)} \end{bmatrix} \tag{3.3}$$

where $\mathbf{q}_j$, $0 \le j \le K-1$, are the columns of $\mathbf{Q}$. Pre-multiplying both sides of Eq. 3.1 by $\mathbf{Q}^T$ yields

$$\mathbf{Q}^T\mathbf{y} = \mathbf{Bh} + \mathbf{Q}^T\mathrm{e}. \tag{3.4}$$

By introducing $\tilde{\mathbf{y}} = \mathbf{Q}^T\mathbf{y}$, $\tilde{\mathbf{X}} = \mathbf{B}$ and $\tilde{\mathrm{e}} = \mathbf{Q}^T\mathbf{y}$, Eq. 3.4 can be re-written as

$$\tilde{\mathbf{y}} = \tilde{\mathbf{X}}\mathbf{h} + \tilde{\mathrm{e}}, \tag{3.5}$$

where $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{e}}$ are $K \times 1$ vectors and $\tilde{\mathbf{X}}$ is a $K \times K$ matrix. The OLS algorithm can now be applied using the transformed information matrix $\tilde{\mathbf{X}}$ and transformed desired vector $\tilde{\mathbf{y}}$ for subset selection.

The solution obtained using $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{X}}$ is identical to that using $\mathbf{y}$ and $\mathbf{X}$. This is because $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{X}}$ are created by performing a unitary transformation [30, 31] on $\mathbf{y}$ and $\mathbf{X}$. We can think of this transformation as a rotation and/or reflection operation. As such operations preserve the length of each (column) vector and the angle between two vectors, no new information is created or lost by transforming Eq. 3.1 into Eq. 3.5. Because of this property, we say that $\mathbf{y}$, $\mathbf{X}$ and $\mathbf{e}$ remain *invariant* after the transformation [30, 31].

This transformation of $\tilde{\mathbf{X}} = \mathbf{Q}^T\mathbf{X}$ and $\tilde{\mathbf{y}} = \mathbf{Q}^T\mathbf{y}$ can be viewed as pre-processing. This pre-processing requires approximately $N \times K^2$ multiplications [95] when the GS decomposition procedure is employed to extract $\mathbf{Q}$. If the saving in computation by using $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{y}}$ for subset selection offsets the additional computation of pre-processing, this reduced OLS approach is justified. The computational complexity of this reduced OLS algorithm for subset model selection has been analysed using examples, and the results showing the number of multiplications needed for subset selection on a 500×84 matrix and a 1000×210 matrix are illustrated in Figs. 3.3 and 3.4 respectively. The number of multiplications required to perform a subset selection of R parameters from a $N \times K$ $\mathbf{X}$ matrix using the OLS method can be calculated using:

$$\text{No. multiplications (OLS)} = \sum_{i=1}^{R}(3N(K-i-1)) + \sum_{i=1}^{R-1}(2N(K-i)). \tag{3.6}$$

The number of multiplications required to perform the pre-processing using the Gram-Schmidt decomposition is calculated using:

$$\text{No. multiplications (GS decomposition)} = NK^2 + NK. \tag{3.7}$$

Therefore, the total number of multiplications required to perform a subset selection by the reduced-OLS GS approach is

$$\begin{aligned}\text{No. multiplications (reduced-OLS GS)} &= NK^2 + NK + \\ &\sum_{i=1}^{R}(3K(K-i-1)) + \sum_{i=1}^{R-1}(2K(K-i)).\end{aligned} \tag{3.8}$$

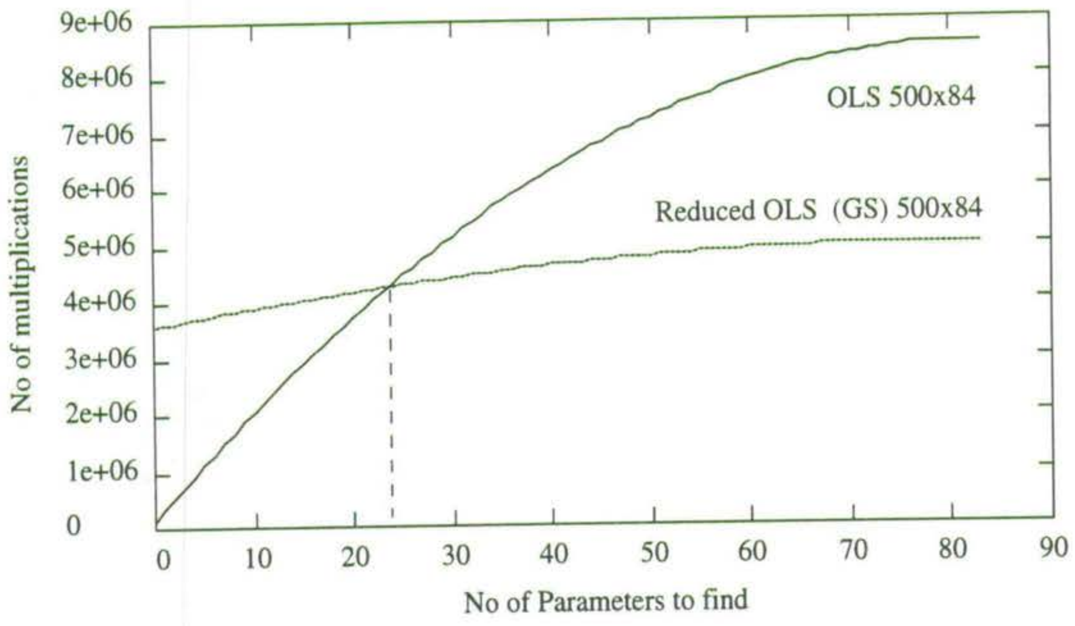

Figure 3.3: Computation requirement for information matrix $\mathbf{X}$ of size 500×84

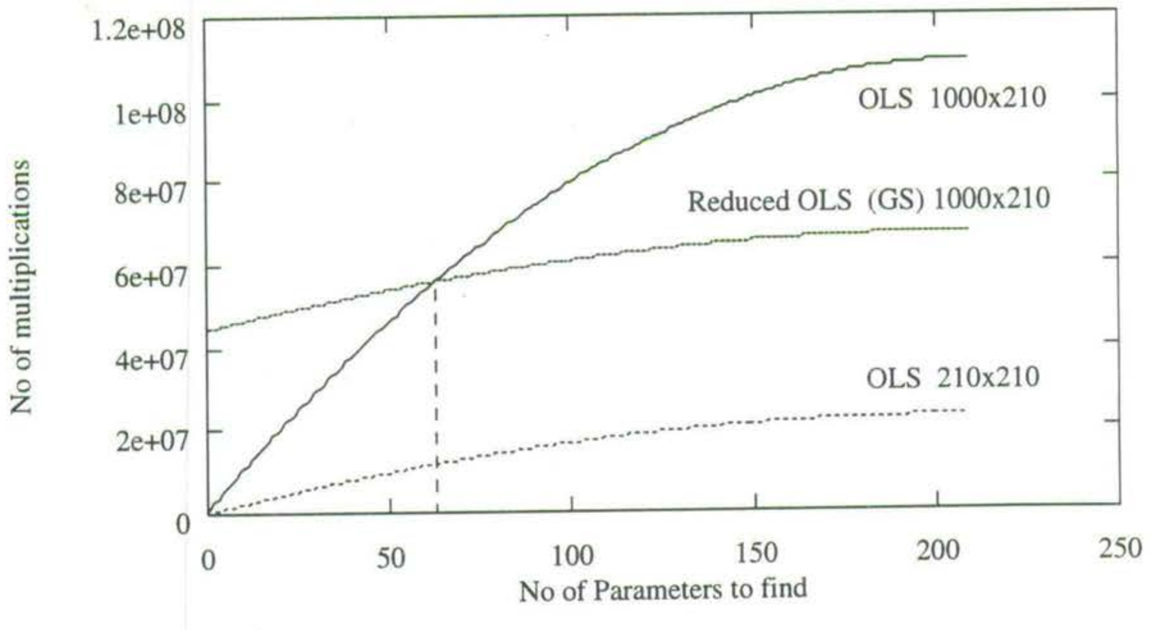

Figure 3.4: Computation requirement for information matrix $\mathbf{X}$ of size 1000×210

3.2.2 Reduced OLS: SVD

The Gram-Schmidt procedure is not the only pre-processing method that can be used to transform Eq. 3.1 into Eq. 3.5. The singular value decomposition (SVD) [32] can be employed as an alternative pre-processing method. The SVD of the $N \times K$ matrix $\mathbf{X}$ is defined by

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = [\mathbf{U}_R|\mathbf{U}_{K-R}]\begin{bmatrix} \mathbf{\Lambda}_R & 0 \\ 0 & \mathbf{\Lambda}_{K-R} \end{bmatrix}\begin{bmatrix} \mathbf{V}_R^T \\ \mathbf{V}_{K-R}^T \end{bmatrix}. \tag{3.9}$$

In the above definition

- The columns of the $N \times K$ matrix $\mathbf{U}$ are the left eigenvectors of $\mathbf{X}$, which are commonly known as the principal components of $\mathbf{X}$. The columns of $\mathbf{U}$ are denoted as $\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{K-1}$. The $N \times R$ matrix $\mathbf{U}_R$ and the $N \times (K-R)$ matrix $\mathbf{U}_{K-R}$ are formed by $[\mathbf{u}_0\ \mathbf{u}_1\ \cdots\ \mathbf{u}_{R-1}]$ and $[\mathbf{u}_R\ \mathbf{u}_{R+1}\ \cdots\ \mathbf{u}_{K-1}]$ respectively.
- $\mathbf{\Lambda}$ is the $K \times K$ diagonal matrix, and the diagonal of $\mathbf{\Lambda}$ contains the singular values of $\mathbf{X}$ arranged in the order of $\lambda_0 \geq \lambda_1 \cdots \geq \lambda_{K-1}$. The diagonal elements of the $R \times R$ diagonal matrix $\mathbf{\Lambda}_R$ are formed by using the $\lambda_0 \geq \lambda_1 \cdots \geq \lambda_{R-1}$.
- The rows of the $K \times K$ matrix $\mathbf{V}^T$ consist of the right eigenvectors of $\mathbf{X}$. The columns of $\mathbf{V}$, the transpose of $\mathbf{V}^T$, are denoted as $\mathbf{v}_0, \mathbf{v}_1, \cdots, \mathbf{v}_{K-1}$. The $R \times K$ matrix $\mathbf{V}_R^T$ is formed by $[\mathbf{v}_0\ \mathbf{v}_1\ \cdots\ \mathbf{v}_{R-1}]^T$.

Notice that $\mathbf{U}$ is an orthonormal matrix. By introducing $\tilde{\mathbf{y}} = \mathbf{U}^T\mathbf{y}$, $\tilde{\mathbf{X}} = \mathbf{\Lambda}\mathbf{V}^T$ and $\tilde{\mathbf{e}} = \mathbf{U}^T\mathbf{e}$, Eq. 3.1 is transformed into Eq. 3.5 as in the previous case of the Gram-Schmidt pre-processing. The only difference is that $\tilde{\mathbf{X}} = \mathbf{\Lambda}\mathbf{V}^T$ is not upper triangular. The SVD however requires significantly more computation [95] (approximately three times more multiplications) than the Gram-Schmidt orthogonalisation scheme. The saving in computation by using $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{X}}$ for subset selection may not always be enough to compensate for the additional computation of the SVD pre-processing.

Further reduction in computation must be sought in order to justify the SVD pre-processing. This can be achieved by further reducing the $K \times K$ problem into an $R \times K$ problem where $R < K$. Pre-multiplying Eq. 3.1 by $\mathbf{U}_R$ yields

$$\mathbf{U}_R^T\mathbf{y} = \mathbf{U}_R^T\mathbf{X}\mathbf{h} + \mathbf{U}_R^T\mathbf{e}. \tag{3.10}$$

We now define an $N \times K$ matrix

$$\hat{\mathbf{X}} = \mathbf{U}_R\mathbf{\Lambda}_R\mathbf{V}_R^T, \qquad R < K, \tag{3.11}$$

$\hat{\mathbf{X}}$ may be thought of as an approximation of $\mathbf{X}$. If we introduce the $R \times 1$ vectors $\tilde{\mathbf{y}}_R = \mathbf{U}_R^T\mathbf{y}$ and $\tilde{\mathbf{e}}_R = \mathbf{U}_R^T\mathbf{e}$, and the $R \times K$ matrix $\tilde{\mathbf{X}}_R = \mathbf{U}_R^T\hat{\mathbf{X}}$, Eq. 3.10 can be rewritten as $\tilde{\mathbf{y}}_R \approx \tilde{\mathbf{X}}_R\mathbf{h} + \tilde{\mathbf{e}}_R$. The OLS algorithm can now be applied to perform subset model selection based on $\tilde{\mathbf{y}}_R$ and

$\tilde{\mathbf{X}}_R$. The maximum rank of the subset model in this case is R. The justification of this reduced OLS selection approach is that if R is sufficiently large, $\hat{\mathbf{X}}$ will be a very good approximation of $\mathbf{X}$ and the solution of the subset selection obtained using $\tilde{\mathbf{y}}_R$ and $\tilde{\mathbf{X}}_R$ will be very similar to that obtained using the original $\mathbf{y}$ and $\mathbf{X}$ (up to the rank R subset model). On the other hand, if R is small enough, the overall computational complexity of this reduced OLS approach will be significantly less than the direct approach using $\mathbf{X}$ and $\mathbf{y}$.

To calculate the error introduced by using $\hat{\mathbf{X}}$ to approximate $\mathbf{X}$, the Frobenius norm [31, 94] of the matrix is needed. The Frobenius norms of $\mathbf{X}$ and $\hat{\mathbf{X}}$ are

$$||\mathbf{X}||_F = \left[\sum_{i=1}^{N}\sum_{j=0}^{K-1}|x_{ij}|^2\right]^{\frac{1}{2}} = \left[\sum_{i=0}^{K-1}\lambda_i^2\right]^{\frac{1}{2}}, \tag{3.12}$$

$$||\hat{\mathbf{X}}||_F = \left[\sum_{i=1}^{N}\sum_{j=0}^{K-1}|\hat{x}_{ij}|^2\right]^{\frac{1}{2}} = \left[\sum_{i=0}^{R-1}\lambda_i^2\right]^{\frac{1}{2}}. \tag{3.13}$$

Let the matrix $\mathbf{E}$ be $\mathbf{E} = \mathbf{X} - \hat{\mathbf{X}}$. The Frobenius norm of $\mathbf{E}$ is

$$||\mathbf{E}||_F = \left[\sum_{i=1}^{N}\sum_{j=0}^{K-1}|x_{ij} - \hat{x}_{ij}|^2\right]^{\frac{1}{2}} = \left[\sum_{i=R}^{K-1}\lambda_i^2\right]^{\frac{1}{2}}. \tag{3.14}$$

The NMSE of using $\hat{\mathbf{X}}$ to approximate $\mathbf{X}$ can be defined as

$$\text{NMSE}_{\hat{\mathbf{X}}} = 20\log_{10}\left(\frac{||\mathbf{E}||_F}{||\mathbf{X}||_F}\right). \tag{3.15}$$

If $\text{NMSE}_{\hat{\mathbf{X}}} = -\infty$, we have $\hat{\mathbf{X}} = \mathbf{X}$. Note that how well $\tilde{\mathbf{y}}_R = \mathbf{U}_R^T\mathbf{y}$ approximates $\mathbf{U}^T\mathbf{y}$ is also important in selecting an appropriate rank R. To measure the error of using $\tilde{\mathbf{y}}_R$ to approximate $\mathbf{U}^T\mathbf{y}$, we define the following NMSE

$$\text{NMSE}_{\tilde{\mathbf{y}}_R} = 10\log_{10}\left(\frac{\sum_{j=R}^{K-1}(\mathbf{u}_j^T\mathbf{y})^2}{\sum_{j=0}^{K-1}(\mathbf{u}_j^T\mathbf{y})^2}\right). \tag{3.16}$$

Let us define the noise floor as the NMSE of approximating $\mathbf{y}$ when the full $\mathbf{X}$ is used. For example, in the two times series previously used for simulations, the noise floor is approximately -43dB for the Duffing's case and approximately -40dB for the Mackey-Glass case (Figs. 3.1 and 3.2). For the reduced OLS based on the SVD pre-processing to work satisfactorily, the rule of thumb is to choose an approximation rank R so that the NMSE measures of Eqs. 3.15 and 3.16 are well below the noise floor. If this criterion is satisfied, the solution of subset model selection up to rank R using $\tilde{\mathbf{y}}_R$ and $\tilde{\mathbf{X}}_R$ is usually comparable to that obtained using the full $\mathbf{y}$ and $\mathbf{X}$. Figs. 3.5 and 3.6 plot the NMSE measures of Eqs. 3.15 and 3.16 as functions of approximation rank R for the two time series examples. The results indicate that an approximation rank $R \geq 40$ may be sufficient for both cases. In practice, the noise floor may not always be known. A compromise between performance of subset selection and computational complexity must be made when choosing R.

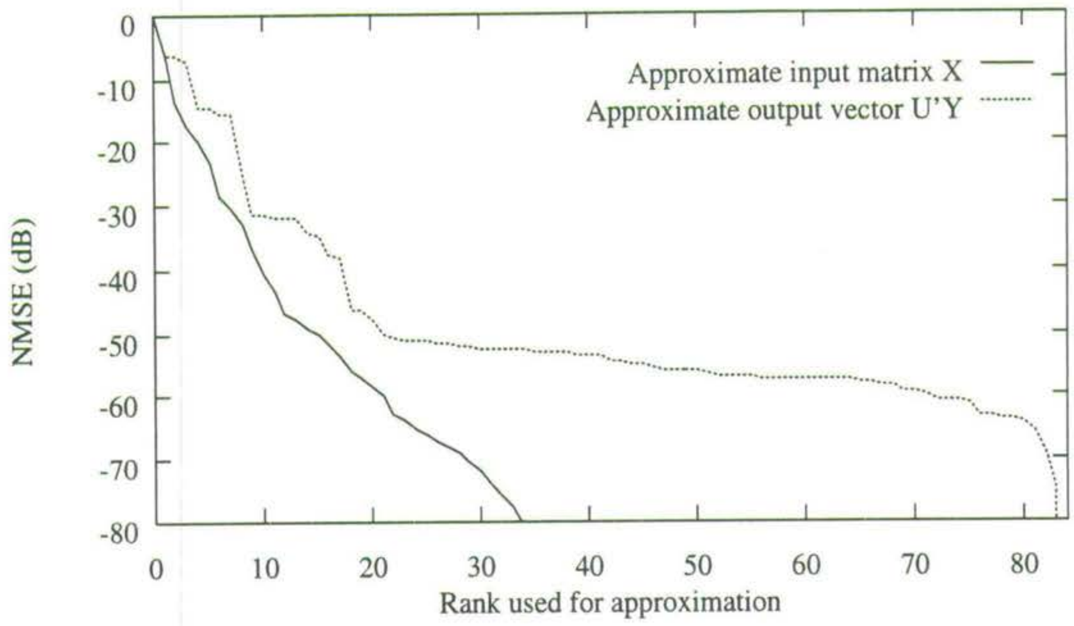

Figure 3.5: NMSE of approximating $\mathbf{X}$ and $\mathbf{U}^T\mathbf{Y}$ (Duffing's series)

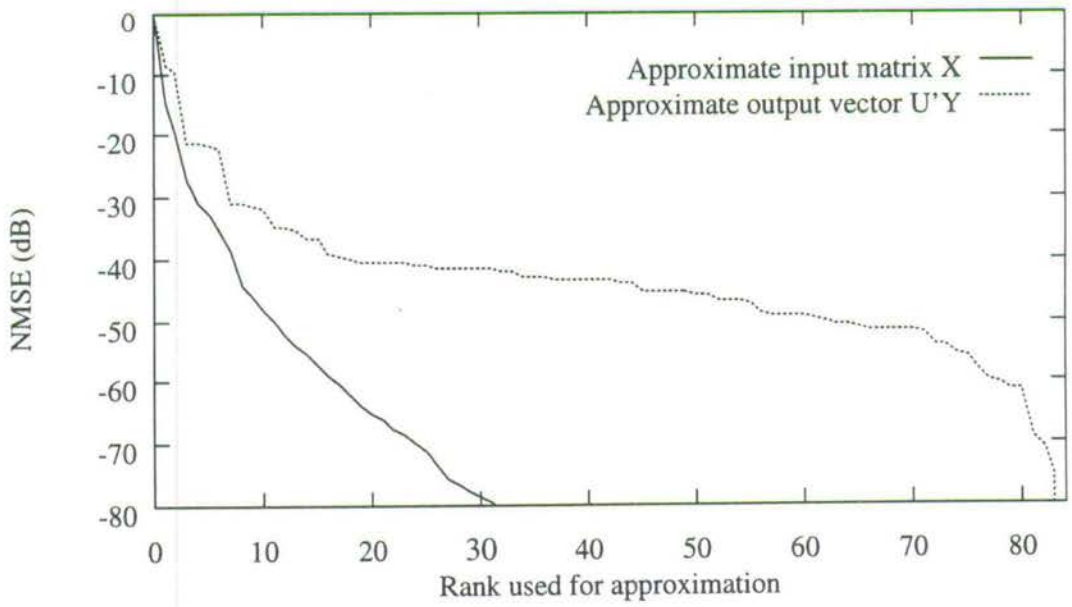

Figure 3.6: NMSE of approximating $\mathbf{X}$ and $\mathbf{U}^T\mathbf{Y}$ (Mackey-Glass series)

3.2.3 Results of reduced OLS methods

Figs. 3.7 and 3.8 show the performance of the predictors, selected from the original 84-term Volterra predictor using the two proposed reduced OLS methods, on the Duffing's and Mackey-Glass chaotic time series. The two time series used were the same as discussed in Sec. 2.4.1.

Since the Gram-Schmidt procedure does not cause any approximation to our original problem but only introduces a change of bases, the reduced OLS method based on Gram-Schmidt pre-processing provides identical performance to the OLS method using the original $\mathbf{X}$ and $\mathbf{y}$.

The reduced OLS SVD approach involves approximating the original $\mathbf{X}$ and $\mathbf{y}$ by $\tilde{\mathbf{X}}_R$ and $\tilde{\mathbf{y}}_R$, and its performance will not generally be identical to the OLS method based on $\mathbf{X}$ and $\mathbf{y}$. For the Duffing's time series, when the approximation rank $R = 40$, subset models with up to 24 regressors selected by the reduced OLS SVD algorithm are the same as those selected by the OLS algorithm using the full $\mathbf{X}$ and $\mathbf{y}$ (Fig. 3.7). This suggests that information regarding the first 24 significant columns of $\mathbf{X}$ is not lost when we approximate the rank 84 matrix $\mathbf{X}$ by the rank 40 matrix $\tilde{\mathbf{X}}_R$. When an approximation rank 60 is used, there is hardly any difference between the subset models selected by the reduced OLS SVD algorithm and those chosen by the OLS algorithm using $\mathbf{X}$ and $\mathbf{y}$. Similar observations can be drawn for the Mackey-Glass time series (Fig. 3.8).

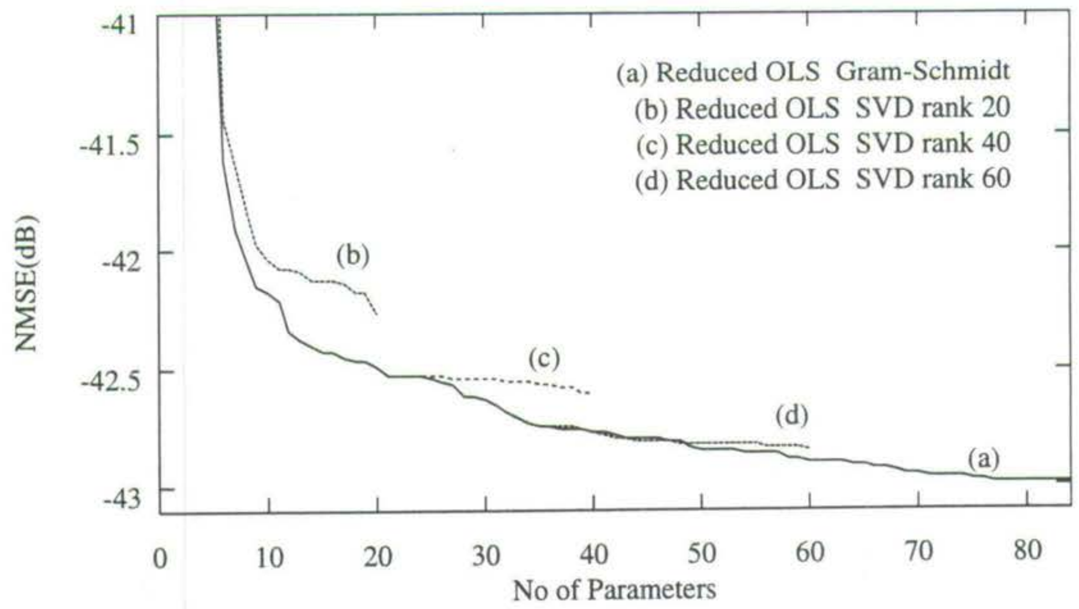

Figure 3.7: Comparison of reduced OLS methods (Duffing's series)

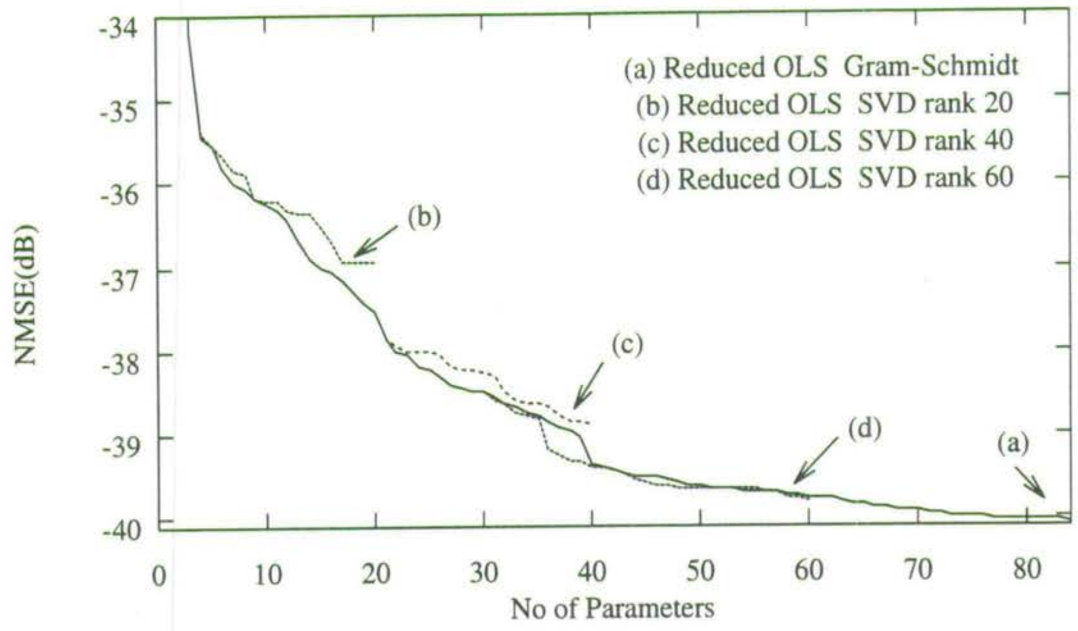

Figure 3.8: Comparison of reduced OLS methods (Mackey-Glass series)

3.3 Nonlinear time series prediction using RBF and GRBF model

In this section, the application of RBF models for nonlinear time series prediction is considered.

The RBF model has enjoyed considerable success in nonlinear time series prediction [9, 10, 44–46, 61]. Most of the successful results, however, are obtained when the model is applied to predict signals that are stationary. The performance of the RBF predictor for nonstationary signal is less satisfactory [96]. This is because the RBF model does not characterise temporal variability well. Since real-world signals are often not only highly nonlinear but also highly nonstationary, it is desired to develop predictors which can handle signals that exhibit both such characteristics.

To improve the predictive performance for nonstationary data, we propose a gradient RBF (GRBF) predictor which is a modification of the classical RBF predictor [15, 16]. In the classical RBF predictor, the centres of the hidden nodes can be interpreted as prototype vectors which are used to sense the presence of the input pattern. That is, if a centre matches the predictor's input vector, the corresponding hidden node will fire strongly. However in the GRBF predictor, a hidden node's function is to sense the presence of a prototype vector's gradient. This significantly improves the predictive capability of the network in the situation where nonstationarity of the signal is due to the variations of mean and trend.

In using this GRBF predictor, we are exploiting the idea that, by performing a suitable difference operation on a nonstationary signal, the resulting signal becomes stationary. This idea is used in the auto-regressive integrated moving average (ARIMA) model for linear prediction of nonstationary signals [29]. By incorporating a similar mechanism into the RBF network, we can create a network model that is capable of dealing with nonlinear and nonstationary signals.

In the following sections, the RBF model is introduced and the procedures required to modify the RBF structure to the gradient RBF (GRBF) model are presented. Some simulation results are also included to compare the performance between RBF, GRBF and linear predictors.

3.3.1 The RBF predictor for nonstationary time series

In this study, the RBF predictor's nonlinearity is chosen to be Gaussian and the width α to be uniform. The centres are initially set to all available training data, and the OLS algorithm is applied to perform model selection.

The input vector to the predictor at time time n is $\mathbf{s}(n) = [y(n-1)\cdots y(n-m)]^T \in R^m$ and the response of the Gaussian j^{th} hidden nodes to the input vector $\mathbf{s}(n)$ is

$$\phi_j(\mathbf{s}(n)) = \exp(-\alpha\|\mathbf{s}(n) - \mathbf{c}_j\|^2) \tag{3.17}$$

where $\mathbf{c}_j$ is the m-dimensional centre vector of the j^{th} node, $\|.\|$ denotes Euclidean norm and α is a positive constant which determines the width of the symmetric response of the hidden node. The predictor's output is

$$\hat{y}(n) = \sum_{j=1}^{K} \phi_j(\mathbf{s}(n))w_j \tag{3.18}$$

where w_j are the network connection weights and K is the number of hidden nodes. The error between the desired network output $y(n)$ and the actual network output $\hat{y}(n)$ is $e(n) = y(n) - \hat{y}(n)$.

3.3.2 The Gradient Radial Basis Function model

The GRBF predictor, like the RBF case, is a nonlinear model with the linear-in-the-parameter structure. In the GRBF model, however, the input vector is generated by differencing the raw data. The order of differencing determines the order of the GRBF predictor. For example, the input vector of the 1st-order GRBF predictor at time n is given by

$$\begin{aligned} \mathbf{s}'(n) &= \mathbf{s}(n) - \mathbf{s}(n-1) \\ &= [y(n-1) - y(n-2), \cdots, y(n-m) - y(n-m-1)]^T \in R^m \end{aligned} \tag{3.19}$$

where $\mathbf{s}(n)$ and $\mathbf{s}(n-1)$ are the original input vectors to the RBF network at time n and $(n-1)$ respectively. The elements of $\mathbf{s}'(n)$ show the rate of change in the trajectory of the time series for the the past m samples.

The function of the hidden node for the GRBF predictor is also slightly different from that of the RBF predictor. Fig. 3.9 depicts the structure of the 1st-order GRBF predictor.

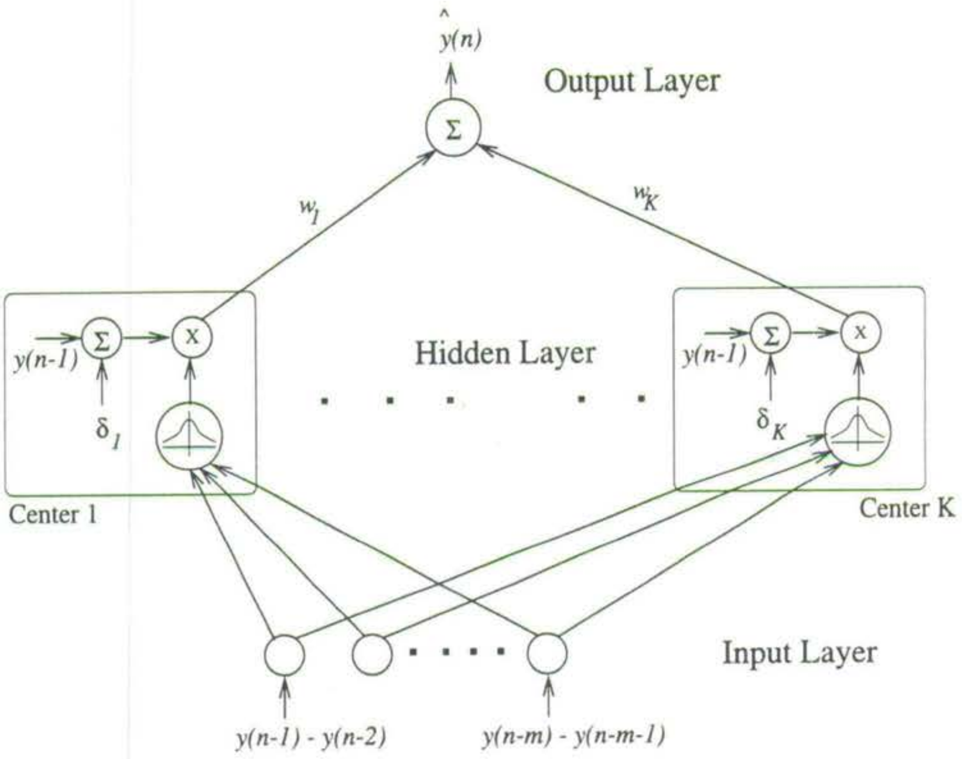

Figure 3.9: Topology of 1st-order GRBF network

Although the Gaussian function still serves as the nonlinear function which compares the similarity of the input vector to the hidden node's centre, the response of the Gaussian function is now multiplied by an additional term $(y(n-1)+\delta)$. The response of the j^{th} hidden node of a 1st-order GRBF network to the input vector $\mathbf{s}'(n)$ is therefore given by

$$\phi'_j(\mathbf{s}(n)) = \exp(-\alpha||\mathbf{s}'(n) - \mathbf{c}'_j||^2) \times (y(n-1) + \delta_j) \tag{3.20}$$

where $\mathbf{c}'_j$ is the m-dimensional centre vector of the j^{th} hidden node and δ_j is a constant value associated with the centre. The term $(y(n-1)+\delta_j)$ can be interpreted as a local single-step prediction of $y(n)$ by the j^{th} hidden node. From Eq. 3.20, if the input vector is very similar to the j^{th} centre, the value of the Gaussian function will be close to 1.0 and the predictor $(y(n-1)+\delta_j)$ becomes fully active. As in the case of the RBF network, the output layer is a linear combiner with weights w_j, $1 \leq j \leq K$.

The centres $\mathbf{c}'_j$ and the scalers δ_j, $1 \leq j \leq K$, can be chosen during the training from the training data $\{\mathbf{s}'(k)\}_{k=1}^{N}$. For each training input vector $\mathbf{s}'(k)$, define

$$d(k) = y(k) - y(k-1). \tag{3.21}$$

If $\mathbf{s}'(k)$ is selected as the j-th centre $\mathbf{c}'_j$, we set $\delta_j = d(k)$ to ensure that the j^{th} hidden node is a perfect predictor of $y(k)$. In this way, the problem of constructing a network is equivalent to the task of selecting a K-term subset model $\{\mathbf{c}'_j, \delta_j\}_{j=1}^{K}$ from the full N-term model $\{\mathbf{s}'(k), d(k)\}_{k=1}^{N}$. The OLS algorithm can readily be applied to perform this subset selection task.

The rationale behind the GRBF model becomes obvious when the network performs predictive operation. Each hidden node compares the network input vector $\mathbf{s}'(n)$ to its centre $\mathbf{c}'_j$. The Gaussian response of each hidden node indicates the degree of matching between $\mathbf{s}'(n)$ and $\mathbf{c}'_j$. The hidden nodes thus sense the gradient of the time series rather than the series itself, as would be in the case of the RBF model. The term $(y(n-1)+\delta_j)$ also has a clear geometric meaning. Referring to Fig. 3.10, if the j^{th} centre $\mathbf{c}'_j$ matches the gradient $\mathbf{s}'(n)$ of the series, $(y(n-1)+\delta_j)$ is likely to be a very good prediction of $y(n)$.

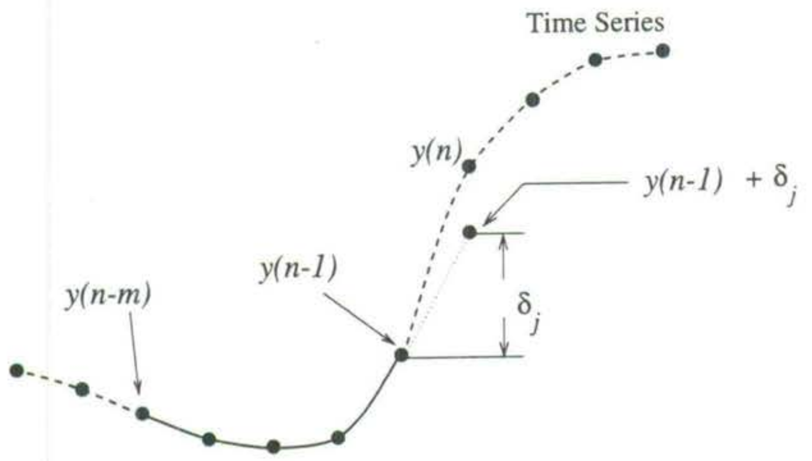

Figure 3.10: Predictive function of j^{th} hidden node. If the centre $\mathbf{c}'_j$ matches $\mathbf{s}'(n)$, $(y(n-1)+\delta_j)$ is a good approximation of $y(n)$.

Fig. 3.11 illustrates the response behaviour of the RBF and GRBF nodes for a given task of sensing the peaks of a sinusoid signal with a time varying mean. To perform the required operation, the RBF and GRBF centres were set using a segment of the signal containing a peak. The results clearly show that the GRBF node was able to respond to each peaks while the RBF node was unable to track the time series.

Although the complexity of a GRBF hidden node is greater than that of a RBF hidden node, the GRBF has better generalisation properties, particularly in predicting homogeneous non-stationary time series. This often results in a smaller GRBF network. Therefore, the overall complexity of the GRBF network may not necessarily be greater than that of the RBF network in practical applications.

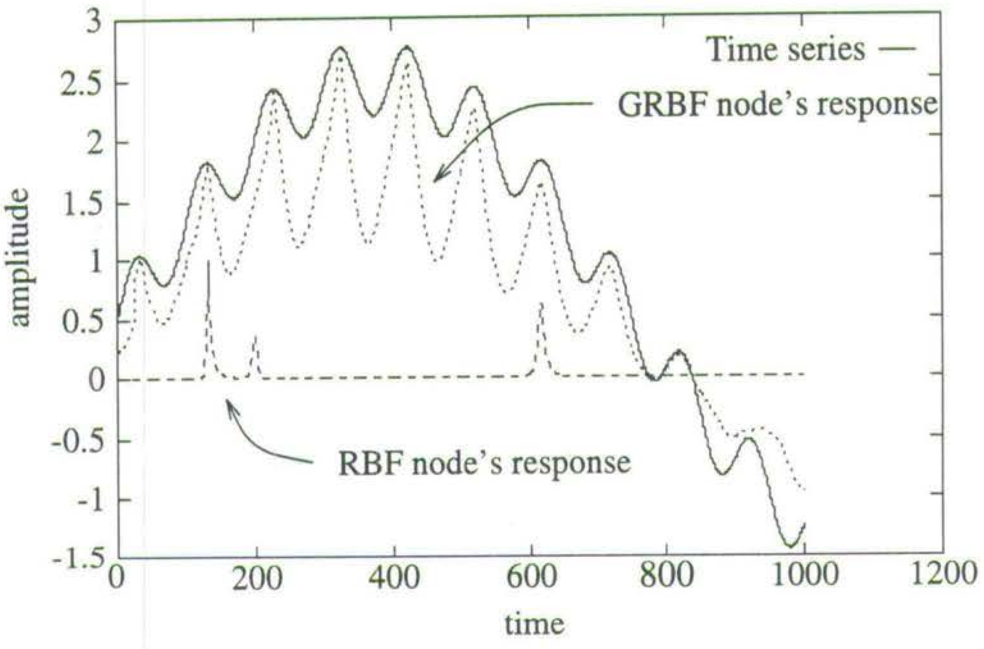

Figure 3.11: RBF and GRBF node's response.

3.3.3 Higher-Order GRBF Networks

We can extend the idea of mapping the data's gradient by the 1st-order GRBF network to that of matching higher-order gradients in a higher order GRBF network. For instance, the input vector to the 2nd order GRBF network at time n can be defined as

$$\begin{aligned}\mathrm{s}''(n) &= \mathrm{s}'(n) - \mathrm{s}'(n-1) \\ &= [(y(n-1) - y(n-2)) - (y(n-2) - y(n-3)), \cdots, \\ &\quad (y(n-m) - y(n-m-1)) - (y(n-m-1) - y(n-m-2))]^T \in R^m. \quad (3.22)\end{aligned}$$

The response of the j^{th} hidden node of the 2nd order GRBF network to $\mathrm{s}''(n)$ is calculated according to

$$\phi_j''(\mathbf{s}''(n)) = \exp(-\alpha\|\mathbf{s}''(n) - \mathbf{c}_j''\|^2) \times ((y(n-1) - y(n-2)) + \delta_j'). \tag{3.23}$$

The m-dimensional centre vectors $\mathbf{c}_j''$ and the scalers δ_j', $1 \leq j \leq K$, can similarly be selected from the training data $\{\mathbf{s}''(k)\}_{k=1}^{N}$. For each training input vector $\mathbf{s}''(k)$, define

$$\begin{aligned} d'(k) &= d(k) - d(k-1) \\ &= (y(k) - y(k-1)) - (y(k-1) - y(k-2)). \end{aligned} \tag{3.24}$$

If $\mathbf{s}''(k)$ is selected as the j^{th} centre $\mathbf{c}_j''$, the value of δ_j' is set to $d'(k)$. The OLS algorithm is well suited to this subset selection problem.

The geometric properties of the 1^{st}-order GRBF network can similarly be extended to a higher-order GRBF network. If we view the 1^{st}-order GRBF network as using a matching of gradient to predict the next value of the time series, then the 2^{nd} order GRBF network predicts the next rate of change using a matching of the 2^{nd} order gradient. This interpretation can be generalised to higher-order GRBF networks.

3.3.4 Simulation Results

We present some simulation results of time series prediction using the RBF and GRBF predictors in this section. Initial full models were created by using all the available data in the training set as RBF and/or GRBF centres. Some linear terms were also included into the full models. Subset models were then selected from these large full models using the OLS scheme, and used to evaluate single-step and multi-step prediction performance.

The same two chaotic series, the Mackey-Glass (Fig. 3.12) and the Duffings (Fig. 3.13) [5], as used in Sec. 2.4.1 were used to evaluate model predictive performance.

Data samples of point 100-600 were used as the training set ($N = 500$) and samples 601 to 1100 were used as the validation set. The embedding vector's dimension was chosen to be $m = 6$ and the width of Gaussian function was set to $\alpha = 1.0$. The following types of models were used:

L-model	Linear model of order 50.
L0-model	Combinations of the linear mode and the classical RBF model.
L01-model	Combinations of the linear model, the classical RBF and 1^{st}-order GRBF models.

[5]The step size used to generate this Duffing's time series is step size = 0.4 , the rest of the parameters used to generate the series is as given in Sec. 2.4.1

L012-model Combinations of the linear model, the classical RBF model, the 1st and 2nd order GRBF models.

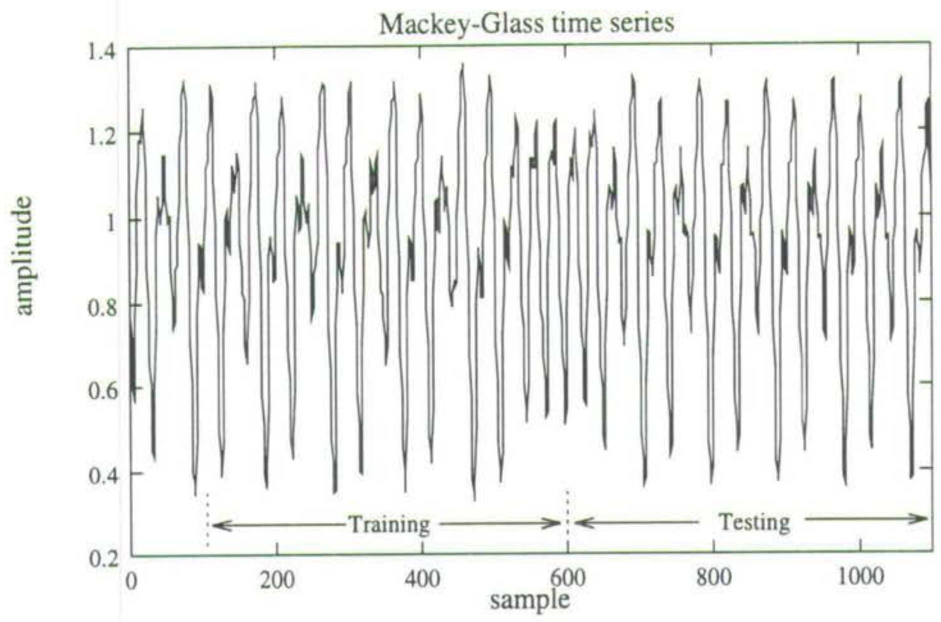

Figure 3.12: Mackey-Glass time series

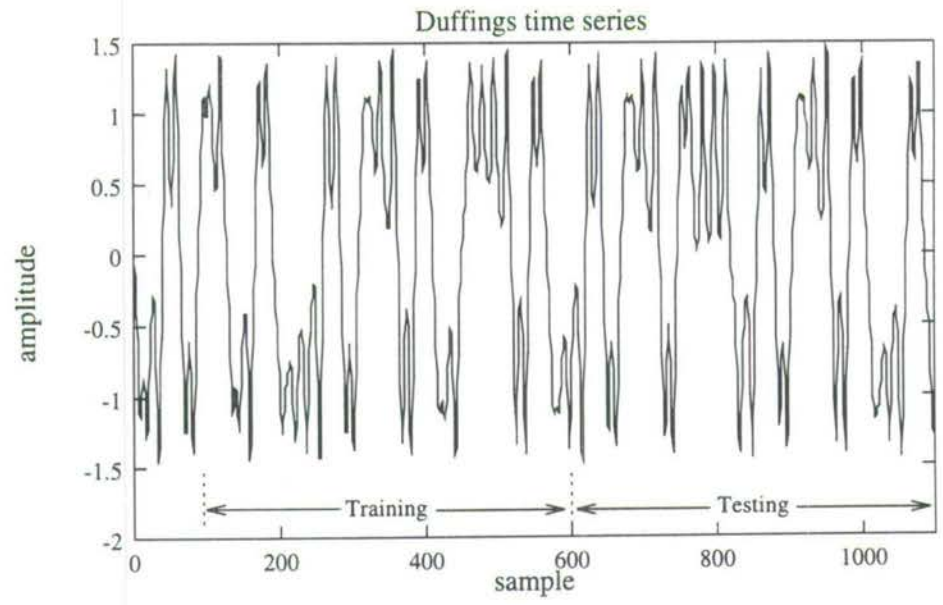

Figure 3.13: Duffing's time series

3.3.5 Simulation results for stationary nonlinear time series predictions

For the Mackey-Glass time series, the results of single-step performance for the predictors in the training phase are shown in Fig. 3.14, where the vertical axis indicates the NMSE in dB. As expected, as the size of each selected subset model increases, the accuracy of the model continued to improve. However, the rate of improvement was not the same for each model. The predictors with GRBF expansion, i.e. **L01** and **L012-models**, achieved better error reduction with a smaller model size. These two GRBF subset models also performed better on the validation set when compared to the linear and classical RBF models, as can be seen in Fig. 3.15. The multi-step prediction performance on the validation set for each of the models was tested using a model size of 25 (Fig. 3.16), and the results show that the two GRBF models had better multi-step predictive accuracy.

The same experiment was repeated for the Duffing's time series with the three models, namely **L-model, L0-model** and **L01-model** [6]. The results of single-step prediction in training and testing phases (Fig. 3.17 and 3.18) and the multi-step predictions using a model size of 25 (Fig. 3.19) again show that the GRBF predictor possesses better generalisation property.

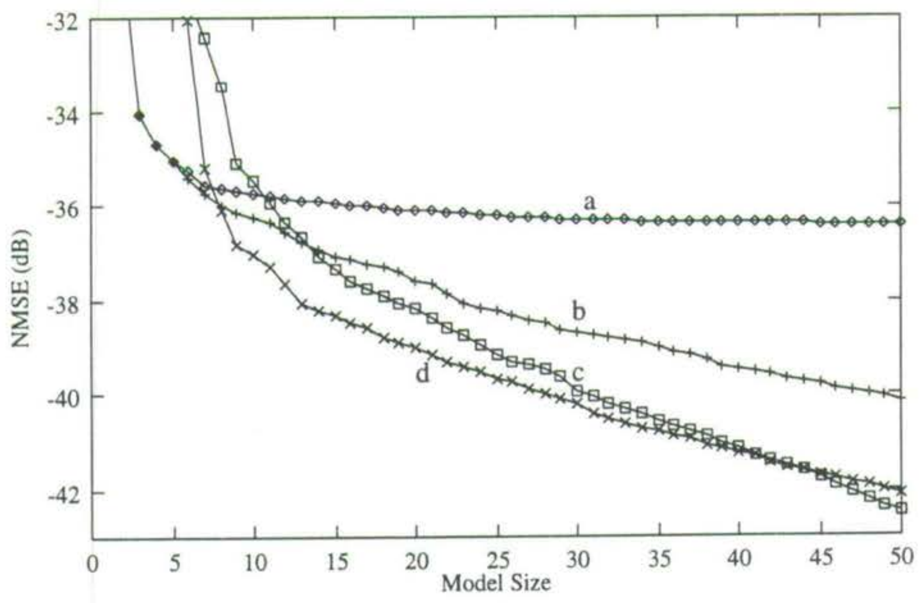

Figure 3.14: Performance of predictors in training phase for stationary Mackey-Glass series : a) Linear model, b) Linear & RBF model, c) Linear, RBF & 1st order GRBF model, d) Linear, RBF, 1st & 2nd order GRBF model.

[6] The results for the **L012-models** are very similar to **L01-model**. They are therefore not included in the report.

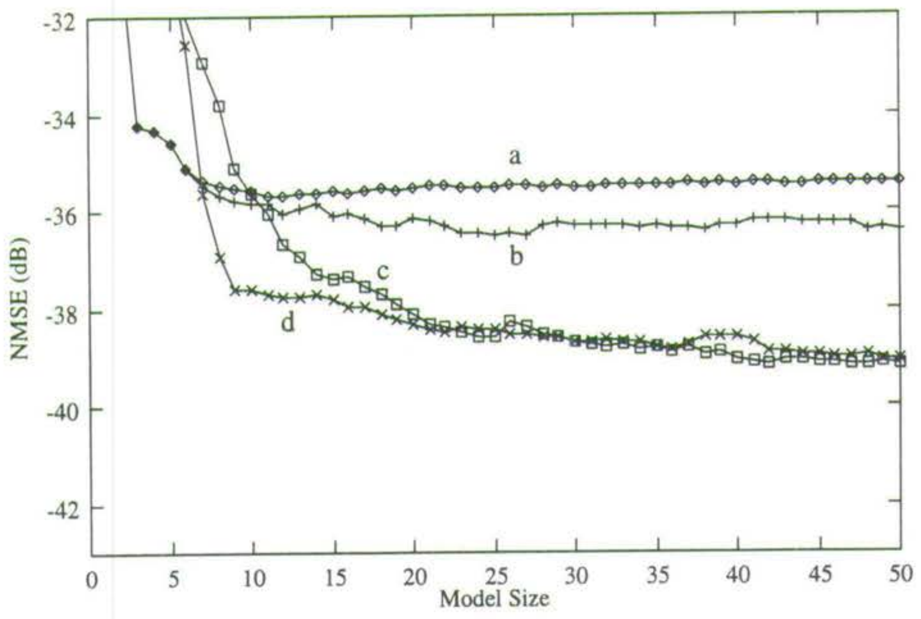

Figure 3.15: Performance of predictors in testing phase for stationary Mackey-Glass series : a) Linear model, b) Linear & RBF model, c) Linear, RBF & 1st order GRBF model, d) Linear, RBF, 1st & 2nd order GRBF model.

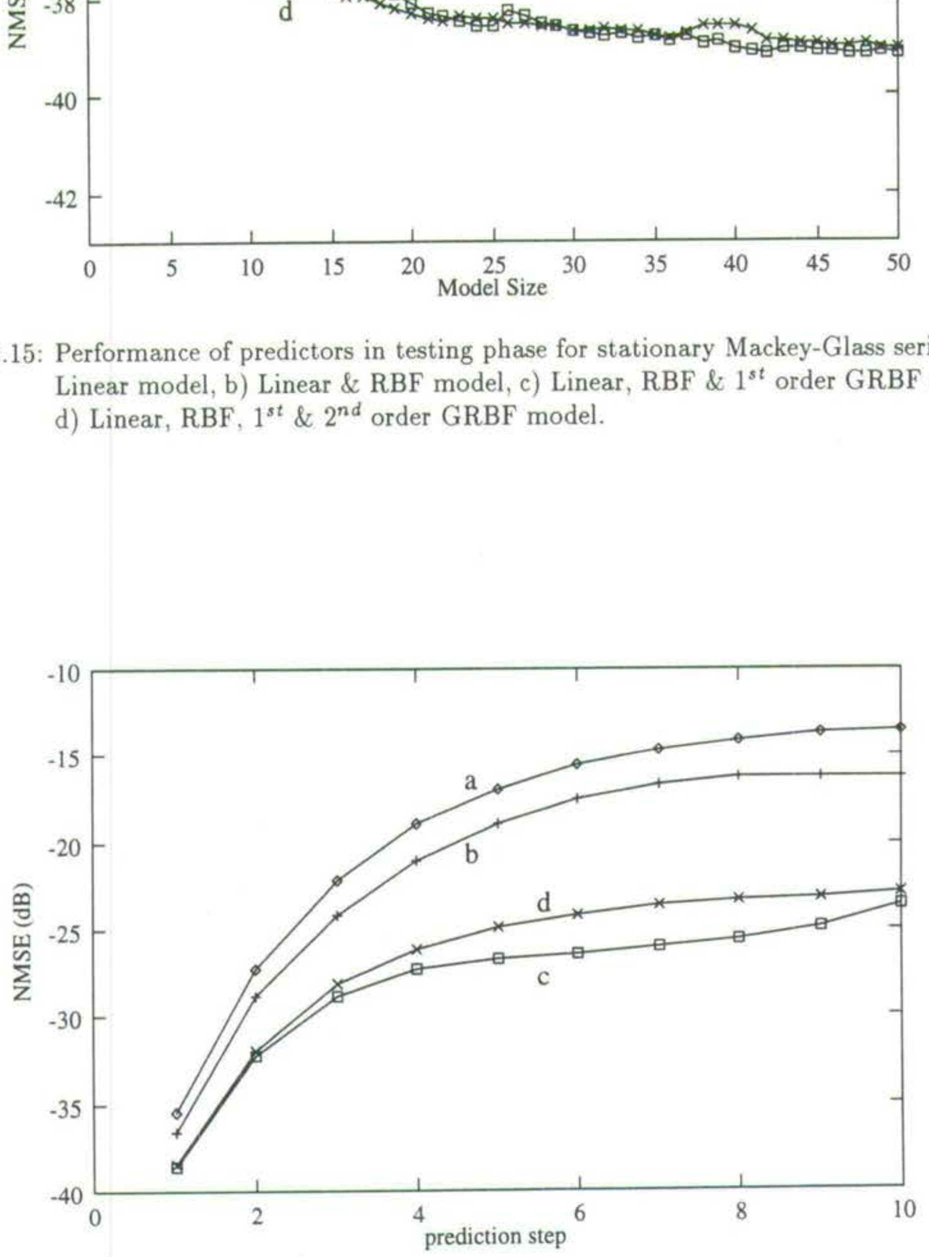

Figure 3.16: Multi-step performance of predictors with a model size of 25 for stationary Mackey-Glass series : a) Linear model, b) Linear & RBF model, c) Linear, RBF & 1st order GRBF model, d) Linear, RBF, 1st & 2nd order GRBF model.

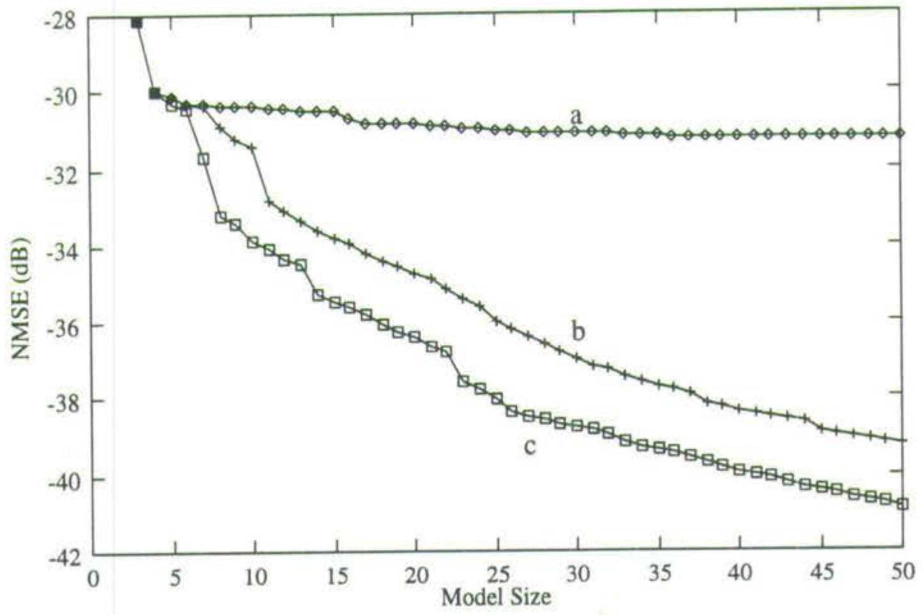

Figure 3.17: Performance of predictors in training phase for stationary Duffing's series : a) Linear model, b) Linear & RBF model, c) Linear, RBF & 1st order GRBF model.

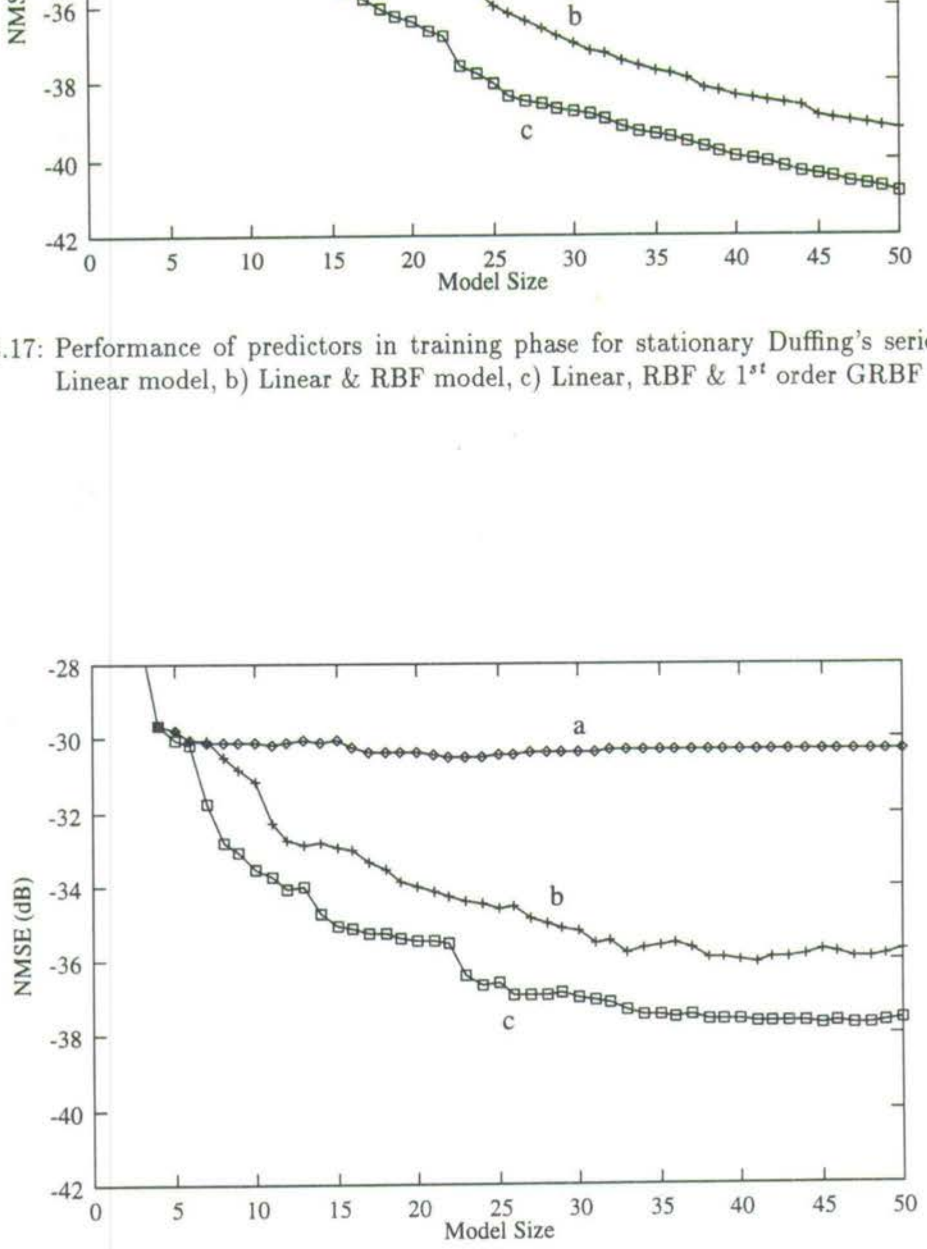

Figure 3.18: Performance of predictors in testing phase for stationary Duffing's series : a) Linear model, b) Linear & RBF model, c) Linear, RBF & 1st order GRBF model.

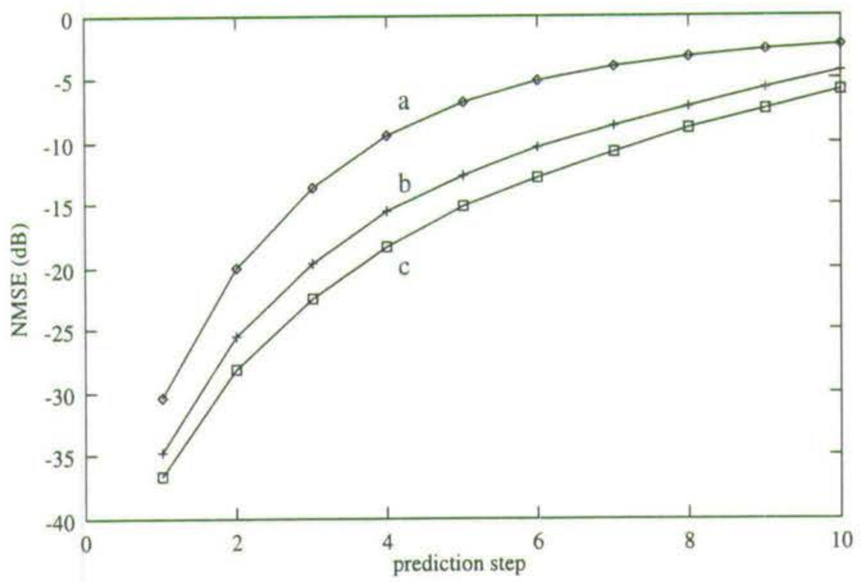

Figure 3.19: Multi-step performance of predictors with a model size of 25 for stationary Duffing's series : a) Linear model, b) Linear & RBF model, c) Linear, RBF & 1^{st} order GRBF model.

3.3.6 Simulation results for homogeneous nonstationary nonlinear time series predictions

To examine how the predictors behave for nonstationary series, we used a modified nonstationary Mackey-Glass time series. This new series, illustrated in Fig. 3.20, was formed by adding a sinusoid with amplitude 0.3 and a period of 3000 samples to the Mackey-Glass time series used in the previous example. As the training data were formed from samples 100 to 600 and the validation data consisted of samples 601-1100, the predictors were trained without being exposed to the change in the level and trend of the test data. The results for the single-step prediction in the validation phase (Fig. 3.21) and the multi-step predictive performance on the validation set using a model size of 35 (Fig. 3.22) suggest that the GRBF network can perform better than the classical RBF network in a homogeneous nonstationary environment.

3.4 Conclusions

The following three main results reported in this chapter are summarised below:

(i). **Improving the forward selection algorithm**

In this chapter, we demonstrated the sub-optimum nature of the forward selection algorithm for model selection. To improve the selection process, a backtracking OLS approach was introduced. Although the simulation results showed that the forward selection process could be improved, the results also indicated that the subset models found are

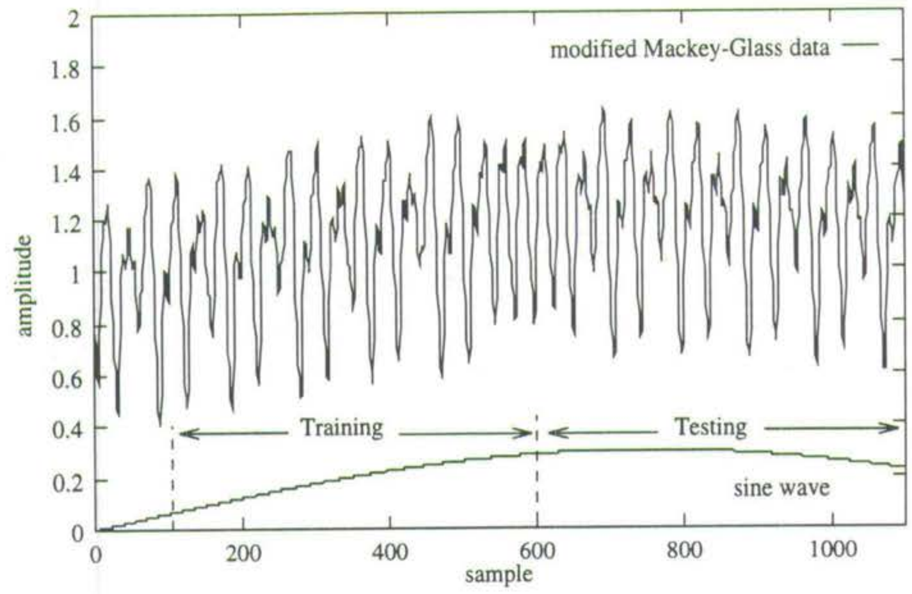

Figure 3.20: Modified nonstationary Mackey-Glass time series

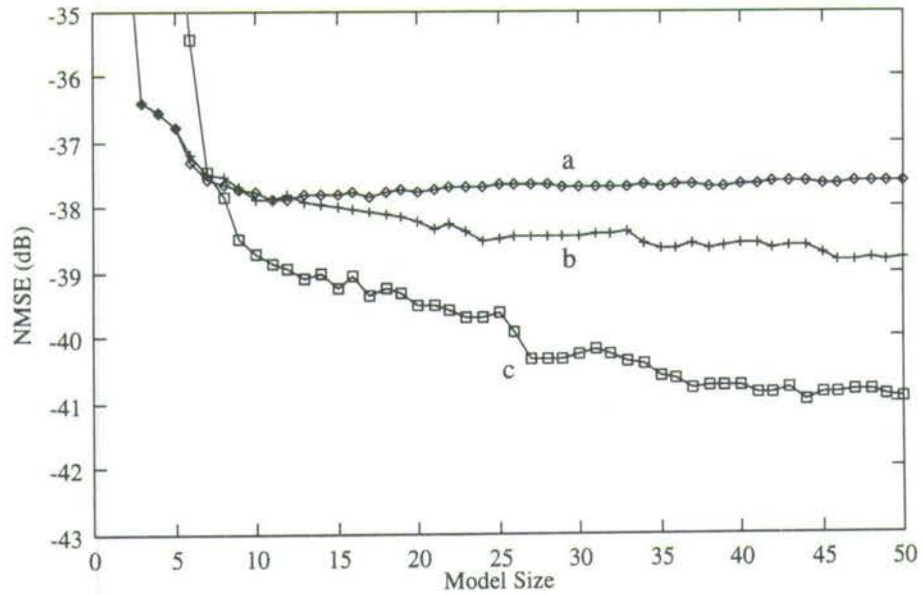

Figure 3.21: Performance of predictors in testing phase for modified nonstationary Mackey-Glass series : a) Linear model, b) Linear & RBF model, c) Linear, RBF & 1st order GRBF model.

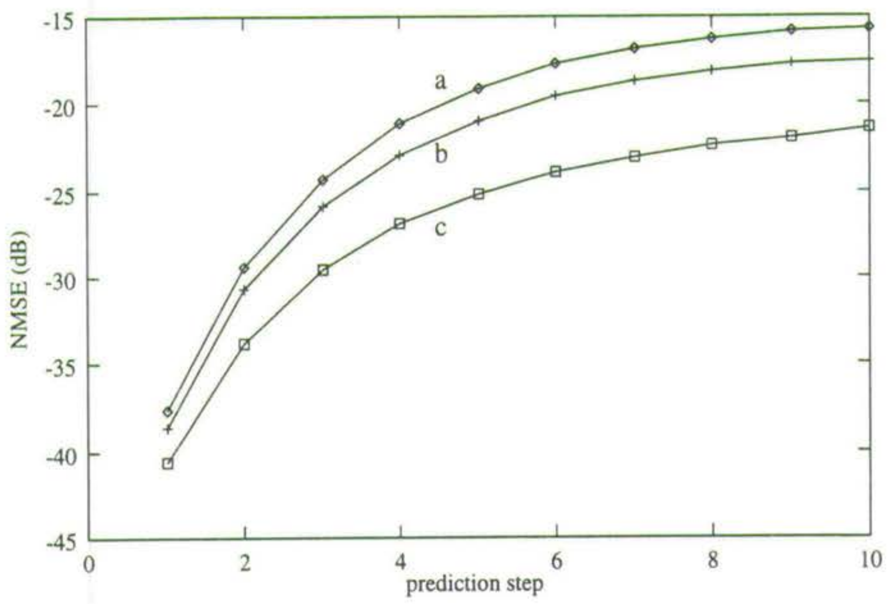

Figure 3.22: Multi-step performance of predictors with a model size of 35 for modified non-stationary Mackey-Glass series : a) Linear model, b) Linear & RBF model, c) Linear, RBF & 1st order GRBF model.

in general very good, and it is normally not necessary to employ more complex methods than forward selection method for practical applications.

(ii). **Reducing the implementation complexity of the OLS algorithm**

The computational requirement of the OLS algorithm was also examined and a method of reducing the processing requirement was presented. The reduction in processing was shown to be significant when the number of rows in the information matrix $\mathbf{X}$ is significantly larger than the number of its columns. Two schemes of the reduced-complexity OLS method were proposed. The first scheme was based on a Gram-Schmidt pre-processing and would provide identical results to those obtained using the original input matrix and the desired output vector. The second scheme was based on a SVD pre-processing and it was shown that subset selection performance can always be traded for computational complexity.

(iii). **Improving the RBF predictor for nonlinear time series prediction**

This chapter also examined the RBF predictor for time series prediction. We showed that the predictive performance of the RBF network can degrade significantly when the predictor is applied to time series which possess homogenous nonstationary behaviour. To improve the predictor's performance, the GRBF model was proposed. The hidden layer of this GRBF network was designed to respond to the gradient of the time series rather than the trajectory itself. This can usually improve predictive accuracy, particularly for homogeneous nonstationary time series as demonstrated in the simulation results. Although the discussion of the GRBF model was restricted to time series prediction, it can be applied to other signal processing applications.

Chapter 4

Digital communication channel equalisation using nonlinear techniques

In this chapter and the next, we will study the application of filters to channel equalisation problems. As stated in the beginning of the thesis, our study is restricted to the symbol-by-symbol detection type equaliser. The structure of such an equaliser consists of a decision function and a quantiser to partition the output of the decision function into the various classes of possible transmit symbol. The task of the equaliser is to classify the unknown received sequence with the minimum probability of misclassification. In this chapter, we will examine nonlinear techniques to realise the equaliser's decision function, and in the next chapter, we will consider linear implementation techniques.

The main objective of this chapter is to examine the application of nonlinear techniques to realise the decision function of the equaliser. In particular, we will study the effects of delay order parameter on the performance of the equaliser and decision boundary. In addition, we will also examine the implementation of the symbol-decision equaliser using RBF techniques.

The following results are reported in this chapter:

(i). The effects of the delay order parameter on the BER performance of the symbol-decision equaliser[1].

(ii). An algorithm to reduce the RBF equaliser's implementation complexity to model the Bayesian equaliser[1].

(iii). A transformation to convert the DFE structure into a feedforward structure[2].

[1]To appear in - E.S.CHNG, B.MULGREW, S.CHEN and G.GIBSON, 'Optimum Lag and Subset Selection for Radial Basis Function Equaliser', *IEEE Workshop on Neural Networks for Signal Processing (NNSP'95)*, Sep 95, BOSTON.

[2]Submitted to - S.CHEN, E.S.CHNG, B.MULGREW and G.GIBSON, 'On decision feedback equaliser', *IEEE Trans. Communications.*

The outline of this chapter is as follows: Sec. 4.1 introduces the channel equalisation problem. Sec. 4.2 derives the optimum symbol-decision nonlinear equaliser, the Bayesian equaliser. Sec. 4.3 considers the application of the RBF network to the realisation of the Bayesian equaliser, and discusses the effects of delay order parameter on the decision boundary. In addition, the section also examines the probability of misclassification and introduces a simple BER estimator to evaluate the equaliser's performance. Sec. 4.4 studies the implementation complexity of the RBF equaliser and proposes a subset model selection method to reduce the complexity. Sec. 4.5 examines the Bayesian DFE and Sec. 4.6 concludes the chapter.

4.1 Channel equalisation

The generic structure of the digital communication channel model and symbol-decision equaliser we will be studying is depicted in Fig. 4.1. It is shown that the transmitted symbol $s(k)$ is passed through a linear dispersive channel and corrupted by additive white Gaussian noise (AWGN) before being received by the equaliser.

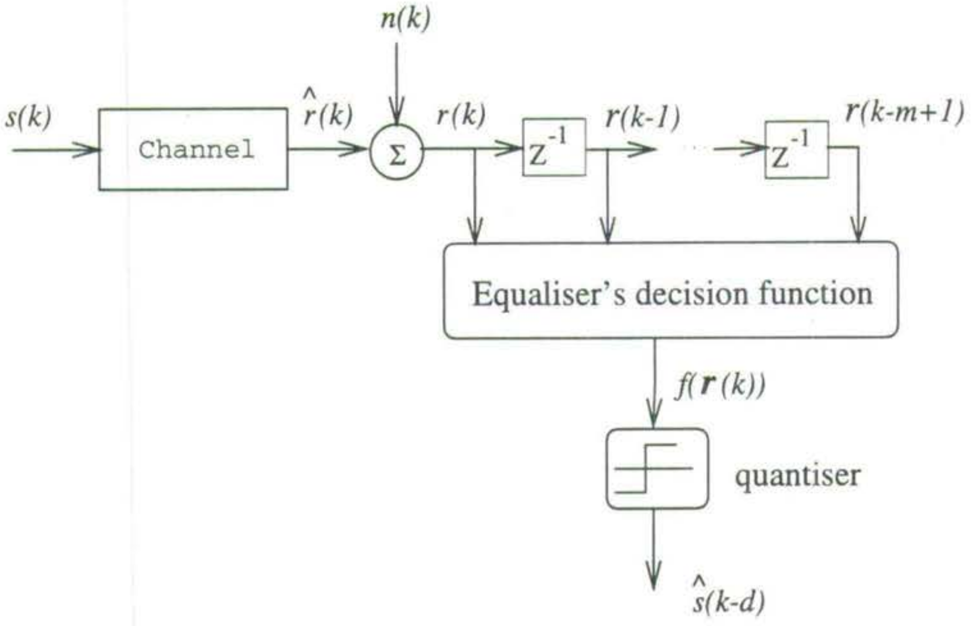

Figure 4.1: Discrete time model of data transmission system

The transmitted signal $s(k)$ is chosen randomly from a binary source with the constellation $\{\pm 1\}$ and the channel is modelled by a FIR filter with the response function $a(0) + a(1)z^{-1} + \cdots + a(n_a - 1)z^{n_a - 1}$. The noise source is assumed to have a Gaussian distribution with mean zero and variance σ_e^2. The signal received by the equaliser at time k is $r(k)$, i.e.,

$$r(k) = \hat{r}(k) + n(k) = \sum_{i=0}^{n_a-1} s(k-i)a(i) + n(k), \tag{4.1}$$

where $\hat{r}(k)$ is the noise-free part of the channel output, and $n(k)$ the additive Gaussian noise sample. The measure of the equaliser's performance is the probability of misclassification with respect to the signal to noise ratio (SNR). The SNR is defined by

$$\mathrm{SNR} = \frac{E[\hat{r}^2(k)]}{E[n^2(k)]} = \frac{\sigma_s^2(\sum_{i=0}^{i=n_a-1} a(i)^2)}{\sigma_e^2} = \frac{\sum_{i=0}^{i=n_a-1} a(i)^2}{\sigma_e^2}. \tag{4.2}$$

The equaliser uses a vector of received signal samples $\mathbf{r}(k) = [r(k) \ \cdots \ r(k-m+1)]^T \in R^m$ to estimate the transmitted symbol $s(k-d)$ where the integers m and d are known as the equaliser's feedforward order and delay order respectively. The estimated value of $s(k-d)$ is denoted by $\hat{s}(k-d)$. The equaliser consists of a filter to implement the decision function and a slicer to quantise the real output value of the decision function $f(\mathbf{r}(k))$ into one of the possible transmitted symbols. In our case when the transmit alphabet is from the set $\{\pm 1\}$, the quantiser can be implemented using the $sgn(.)$ function, i.e.,

$$sgn(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}. \tag{4.3}$$

The equaliser's performance is therefore determined by the decision function. In the following sections, it will be shown that the optimum decision function, the Bayesian decision function [17,18], is nonlinear and to achieve this solution, the equaliser's decision function must contain nonlinearity. This is one of the reasons why many researchers have examined nonlinear equaliser structures, e.g., the Volterra equalisers [34,35,97,98], multi-layer perceptron (MLP) equalisers [98–102] and RBF equalisers [17,18,73,103]. The RBF equaliser however has been shown to be able to realise the Bayesian solution given the channel and noise statistics [17,18,73]. We will therefore concentrate on the RBF implementation for the rest of the chapter.

4.2 The decision function of the Bayesian symbol-decision equaliser

The equalisation process can be viewed as a classification problem in which the equaliser's task is to partition the input space $\mathbf{r}(k) \in R^m$ into two distinct regions [17,18,99] given that the transmitted symbol is binary. The boundary points which separate these two regions are referred to as the decision boundary. The partitioning which results in the minimum probability of misclassification is known as the Bayesian solution and the decision boundary which realises this solution is nonlinear [17,18,99]. To derive the Bayesian decision function, we first introduce the definition of channel states.

4.2.1 Channel states

The input vector to the equaliser is $\mathbf{r}(k) = \hat{\mathbf{r}}(k) + \mathbf{n}(k)$. The vector $\hat{\mathbf{r}}(k)$ is known as the channel state and is the vector of noise-free received signals $\hat{\mathbf{r}}(k) = [\hat{r}(k) \ \cdots \ \hat{r}(k-m+1)]^T \in R^m$. The values of the channel state are determined by the values of the transmitted sequence vector

$$\mathbf{s}(k) = [s(k) \ \cdots \ s(k-m-n_a+2)]^T \in R^{m+n_a-1}, \tag{4.4}$$

and are related to $\mathbf{s}(k)$ by the following expression [99]

$$\hat{\mathbf{r}}(k) = F[\mathbf{s}(k)], \tag{4.5}$$

where the matrix $F \in R^{m\times(m+n_a-1)}$ is

$$F = \begin{bmatrix} a(0) & a(1) & \dots & a(n_a-1) & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & a(0) & a(1) & \dots & a(n_a-1) & 0 & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots & a(0) & a(1) & \dots & a(n_a-1) \end{bmatrix} . \tag{4.6}$$

Since $\mathbf{s}(k)$ has $N_s = 2^{m+n_a-1}$ combinations [17, 18], $\hat{\mathbf{r}}(k)$ has N_s states. Let the N_s sequences of $\mathbf{s}(k)$ be

$$\mathbf{s}_j(k) = [s_j(k) \ s_j(k-1) \ \cdots \ s_j(k-m-n_a+2)]^T, \qquad 1 \le j \le N_s. \tag{4.7}$$

The corresponding states of $\hat{\mathbf{r}}(k)$, denoted as $\mathbf{c}_j$, are given by

$$\mathbf{c}_j = \hat{\mathbf{r}}(k) = F[\mathbf{s}_j(k)], \qquad 1 \le j \le N_s. \tag{4.8}$$

The set of N_s channel states $C_d = \{\mathbf{c}_j\}, 1 \le j \le N_s$, can be partitioned into two subsets according to the values of the transmitted symbol $s(k-d)$, i.e.,

$$C_d = C_d^{(+)} \cup C_d^{(-)} \tag{4.9}$$

where

$$C_d^{(+)} = \{\hat{\mathbf{r}}(k) | s(k-d) = +1\}, \tag{4.10}$$
$$C_d^{(-)} = \{\hat{\mathbf{r}}(k) | s(k-d) = -1\}. \tag{4.11}$$

Each set $C_d^{(+)}$ and $C_d^{(-)}$ contains $N_s/2$ channel states.

4.2.2 The Bayesian decision function

Because of additive noise, the observation vector $\mathbf{r}(k) = \hat{\mathbf{r}}(k) + \mathbf{n}(k)$ is a random process having conditional Gaussian density function centred at each channel state. Given that the channel state $\hat{\mathbf{r}}(k) = \mathbf{c}_j$, the conditional probability distribution of the observation vector is

$$p(\mathbf{r}(k)|\mathbf{c}_j) = (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r}(k) - \mathbf{c}_j\|^2/(2\sigma_e^2)). \tag{4.12}$$

Due to the additive noise, the received vector $\mathbf{r}(k)$ may be perturbed sufficiently to cross the decision boundary, resulting in a misclassification. To minimise the probability of misclassification given that the received vector is $\mathbf{r}(k)$ [17, 104, 105], the estimated symbol should be chosen by determining which transmitted signal $s \in \{\pm 1\}$ has the maximum *a posteriori* probability $P(s(k-d) = s|\mathbf{r}(k))$. This leads to the following decision rule:

$$\hat{s}(k-d) = sgn(f_b(\mathbf{r}(k))) = \begin{cases} 1, & f_b(\mathbf{r}(k)) \geq 0, \\ -1, & f_b(\mathbf{r}(k)) < 0, \end{cases} \tag{4.13}$$

where $f_b(\mathbf{r}(k))$ compares the *a posteriori* probabilities of the binary transmitted symbols[3], i.e.,

$$f_b(\mathbf{r}(k)) = P(s(k-d) = +1|\mathbf{r}(k)) - P(s(k-d) = -1|\mathbf{r}(k)), \tag{4.14}$$

where $P(s(k-d) = +1|\mathbf{r}(k))$ and $P(s(k-d) = -1|\mathbf{r}(k))$ are the *a posteriori* probabilities that the transmitted signal is $+1$ and -1 respectively given $\mathbf{r}(k)$. The function $f_b(\mathbf{r}(k))$ is also known as the Bayesian decision function because the Bayes rule [104] is applied to express the *a posteriori* probability into the product of the *a priori* probability $P(s(k-d) = s)$ and the state-conditional probability density function (pdf) $p(\mathbf{r}(k)|s(k-d) = s)$ over the pdf of $\mathbf{r}(k)$, i.e.,

$$P(s(k-d) = s)|\mathbf{r}(k)) = \frac{p(\mathbf{r}(k)|s(k-d) = s)P(s(k-d) = s)}{p(\mathbf{r}(k))}. \tag{4.15}$$

The reason for expressing the *a posteriori* probabilities in terms of the *a priori* and state-conditional probabilities is that these *a priori* and state-conditional probabilities can be calculated given the channel and noise statistics. If the transmit symbol is i.i.d, the *a priori* probabilities $P(s(k-d) = +1)$ and $P(s(k-d) = -1)$ have the same value $\frac{1}{2}$ and the state-conditional pdf $p(\mathbf{r}|s(k-d) = +1)$ is the sum of the pdfs for each channel state $\mathbf{c}_j \in C_d^{(+)}$,

[3]The notation upper-case $P(.)$ denotes a probability mass function, and a lower-case $p(.)$ denotes a probability density function.

i.e.,

$$\begin{aligned} p(\mathbf{r}(k)|s(k-d)=+1) &= \frac{1}{N_s}\sum_{\mathbf{c}_j\in C_d^{(+)}} p(\mathbf{r}(k)|\mathbf{c}_j) \\ &= \frac{1}{N_s}\sum_{\mathbf{c}_j\in C_d^{(+)}} (2\pi\sigma_e^2)^{-m/2}\exp(-\|\mathbf{r}(k)-\mathbf{c}_j\|^2/(2\sigma_e^2)), \end{aligned} \tag{4.16}$$

where the constant $\frac{1}{N_s}$ is the *a priori* probability of $\mathbf{c}_j$. Similarly the conditional pdf $p(\mathbf{r}|s(k-d)=-1)$ is the sum of the pdfs for each channel state $\mathbf{c}_j \in C_d^{(-)}$, i.e.,

$$\begin{aligned} p(\mathbf{r}(k)|s(k-d)=-1) &= \frac{1}{N_s}\sum_{\mathbf{c}_k\in C_d^{(-)}} p(\mathbf{r}(k)|\mathbf{c}_k) \\ &= \frac{1}{N_s}\sum_{\mathbf{c}_k\in C_d^{(-)}} (2\pi\sigma_e^2)^{-m/2}\exp(-\|\mathbf{r}(k)-\mathbf{c}_k\|^2/(2\sigma_e^2)). \end{aligned} \tag{4.17}$$

Substituting Eq. 4.15 into Eq. 4.14, the Bayesian decision function becomes

$$\begin{aligned} f_b(\mathbf{r}(k)) &= \frac{p(\mathbf{r}(k)|s(k-d)=+1)P(s(k-d)=+1)}{p(\mathbf{r}(k))} \\ &- \frac{p(\mathbf{r}(k)|s(k-d)=-1)P(s(k-d)=-1)}{p(\mathbf{r}(k))}. \end{aligned} \tag{4.18}$$

Since the *a priori* probabilities are the same for the transmitted symbols, and the two denominator terms $p(\mathbf{r}(k))$ in the right hand side of Eq. 4.18 are common, they do not affect the sign of the equation and may be removed. The equation simplifies to,

$$\begin{aligned} f_b(\mathbf{r}(k)) &= p(\mathbf{r}(k)|s(k-d)=+1) - p(\mathbf{r}(k)|s(k-d)=-1) \\ &= \frac{1}{N_s}\sum_{\mathbf{c}_j\in C_d^{(+)}} p(\mathbf{r}(k)|\mathbf{c}_j) - \frac{1}{N_s}\sum_{\mathbf{c}_k\in C_d^{(-)}} p(\mathbf{r}(k)|\mathbf{c}_k) \\ &= \frac{1}{N_s}\sum_{\mathbf{c}_j\in C_d^{(+)}} (2\pi\sigma_e^2)^{-m/2}\exp(-\|\mathbf{r}(k)-\mathbf{c}_j\|^2/(2\sigma_e^2)) \\ &- \frac{1}{N_s}\sum_{\mathbf{c}_k\in C_d^{(-)}} (2\pi\sigma_e^2)^{-m/2}\exp(-\|\mathbf{r}(k)-\mathbf{c}_k\|^2/(2\sigma_e^2)). \end{aligned} \tag{4.19}$$

Similarly, the coefficients $\frac{1}{N_s}$ and $(2\pi\sigma_e^2)^{-m/2}$ do not affect the sign of the decision function and Eq. 4.19 can be further simplified to

$$f_b(\mathbf{r}(k)) = \sum_{\mathbf{c}_j\in C_d^{(+)}} \exp(-\|\mathbf{r}(k)-\mathbf{c}_j\|^2/(2\sigma_e^2)) - \sum_{\mathbf{c}_k\in C_d^{(-)}} \exp(-\|\mathbf{r}(k)-\mathbf{c}_k\|^2/(2\sigma_e^2)). \tag{4.20}$$

This concludes the derivation of the Bayesian equaliser's decision function. It is observed from the above equation that the Bayesian decision function is nonlinear and is completely specified by the channel state locations and noise variance σ_e^2. Therefore, if the channel is known completely, the channel states (Eq. 4.8) and noise statistics (Eq. 4.2) can be calculated to generate the Bayesian decision function.

In the following section, the RBF network is briefly introduced to show that the RBF model structure is identical to that of the Bayesian decision function (Eq. 4.20).

4.2.3 Implementing the Bayesian decision function using the RBF model

The response of the RBF network is [7–9, 17, 36]

$$f_r(\mathbf{r}) = \sum_{j=1}^{n} w_j \phi(||\mathbf{r} - \mathbf{c}_j||^2/\alpha) \tag{4.21}$$

where n denotes the number of computing units or hidden nodes, w_j are the weights, $\phi(.)$ are the nonlinearities of the hidden nodes, $\mathbf{c}_j$ are the centres of the hidden nodes, and α is a positive constant known as the width.

It is obvious by comparing Eq. 4.21 to Eq. 4.20, that the structure of the RBF network realises exactly the Bayesian decision function when the weights, centres, α, and the nonlinearity $\phi(.)$ are set according to Eq. 4.20 [17, 18]. The RBF network is therefore ideal to model the Bayesian equaliser.

To realise the Bayesian equaliser using the RBF network, the channel states $\mathbf{c}_j, 1 \leq j \leq N_s$, are first calculated from an estimate of the channel model. These channel states become the RBF centres, and the nonlinearity of $\phi(.)$ is chosen to be

$$\phi(x) = \exp(-x). \tag{4.22}$$

In addition, the constant term α of the nonlinearity is set to the value of $2\sigma_e^2$, the weights w_j associated with the channel states belonging to $C_d^{(+)}$ are set to $+1$, and the weights associated with the channel states belonging to $C_d^{(-)}$ are set to -1.

The performance of the RBF model depends critically on the position of the estimated channel states and less critically on the estimated noise variance. It has been shown that given a good estimate of the channel state locations and noise statistics, the RBF model will result in BER performance very close to the Bayesian equaliser [17].

4.2.4 Example of Bayesian decision function response

To illustrate the Bayesian decision function response, we examine the response behaviour of the Bayesian equaliser when applied to the channel

$$H(z) = 0.5 + 1.0z^{-1} \tag{4.23}$$

with delay order $d = 0$, feedforward order $m = 2$ at SNR = 15dB. Table 4.1 lists the set of channel states. Given delay $d = 0$, the set of channel states associated with transmit symbols $s = +1$ and $s = -1$ are $C_d^{(+)} = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4\}$ and $C_d^{(-)} = \{\mathbf{c}_5, \mathbf{c}_6, \mathbf{c}_7, \mathbf{c}_8\}$ respectively.

S/No j	Transmitted sequence $\mathbf{s}_j$ [$s(k)$	$s(k-1)$	$s(k-2)$]	Channel state $\mathbf{c}_j$ [$\hat{r}(k)$	$\hat{r}(k-1)$]
1	1	1	1	1.5	1.5
2	1	1	-1	1.5	-0.5
3	1	-1	1	-0.5	0.5
4	1	-1	-1	-0.5	-1.5
5	-1	1	1	0.5	1.5
6	-1	1	-1	0.5	-0.5
7	-1	-1	1	-1.5	0.5
8	-1	-1	-1	-1.5	-1.5

Table 4.1: Transmitted sequence and received channel states for channel $H(z) = 0.5 + 1.0z^{-1}$

Figs. 4.2a,b depict the conditional probability distribution $p(\mathbf{r}(k)|s(k-d) = +1)$ and $p(\mathbf{r}(k)|s(k-d) = -1)$ (Eqs. 4.16 and 4.17) respectively. Note that the pdf $p(\mathbf{r}(k)|s(k-d) = s)$ is a mixture of multi-variable Gaussian functions whose means correspond to the respective channel states in $C_d^{(+)}$ and $C_d^{(-)}$. Fig. 4.3a illustrates the mapping of the Bayesian decision function $f_b(\mathbf{r}(k))$. The positive peaks in the graph are attributed to $p(\mathbf{r}(k)|s(k-d) = +1)$ and the negative peaks attributed to $p(\mathbf{r}(k)|s(k-d) = -1)$. The set of points $\{\mathbf{r}_0\}$ for $f_b(\mathbf{r}_0) = 0$ forms the decision boundary

This example clearly illustrates two characteristics of the Bayesian decision boundary; firstly, the decision boundary can be highly nonlinear and secondly, the set of $C_d^{(+)}$ and $C_d^{(-)}$ may not be linearly separable.

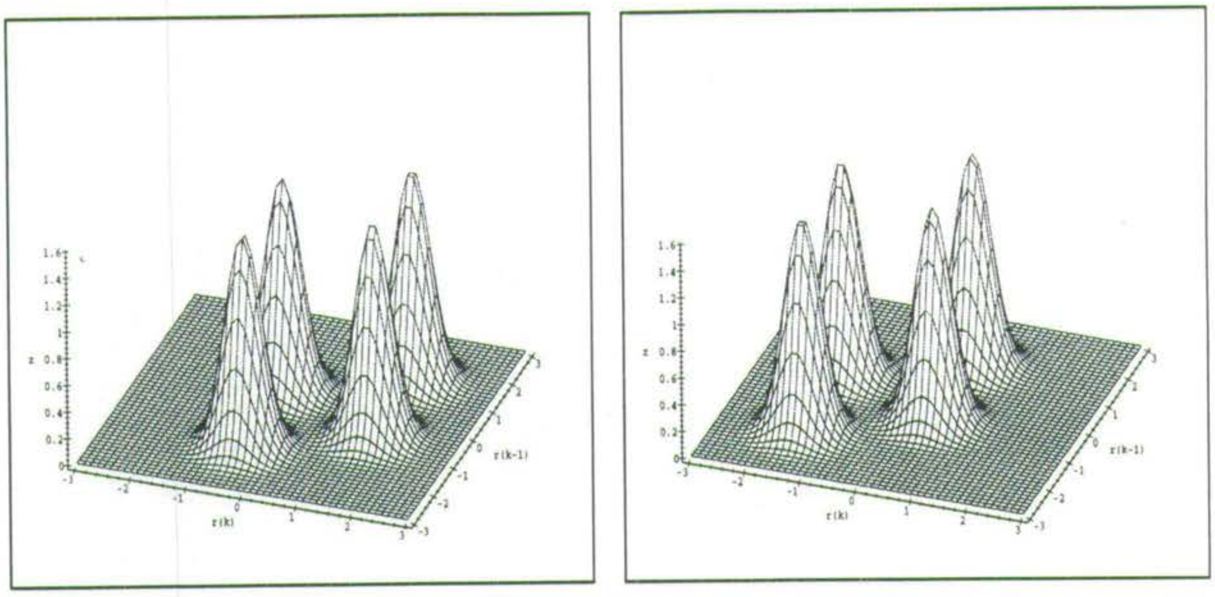

Figure 4.2: Conditional pdf of the received signal vector $\mathbf{r}(k)$: (a) $z = p(\mathbf{r}(k)|s(k-d) = +1)$, (b) $z = p(\mathbf{r}(k)|s(k-d) = -1)$ at SNR=15dB.

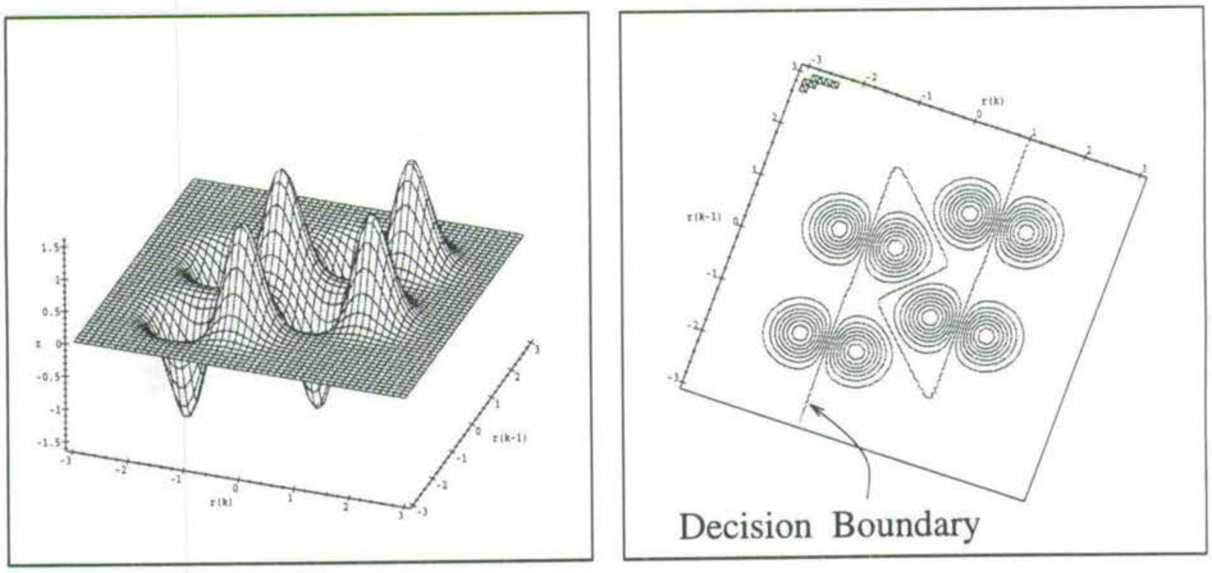

Figure 4.3: (a) Bayesian decision function $z = f_b(\mathbf{r}(k))$, (b) Contours of the decision function mapping and the Bayesian decision boundary at SNR=15dB.

4.2.5 Effects of error variance and delay order on decision boundaries

The Bayesian decision boundary is affected by the locations of the channel states, delay order parameter and the noise statistics [17, 18]. In the case when the noise is i.i.d Gaussian, the effects of noise variance on the decision boundary is not significant [17, 18]. To illustrate this, Fig. 4.4 plots the decision boundaries of the Bayesian equaliser for different SNR. The results show that the shape of the decision boundaries do not vary significantly for wide ranges of SNR and in the cases at high SNR, e.g. for the case when the SNR equals to or greater than 20dB, the decision boundary becomes a set of linear-partitions joined at each end. Such decision boundaries are also called the asymptotic decision boundaries as these boundaries are realised when SNR tends to ∞.

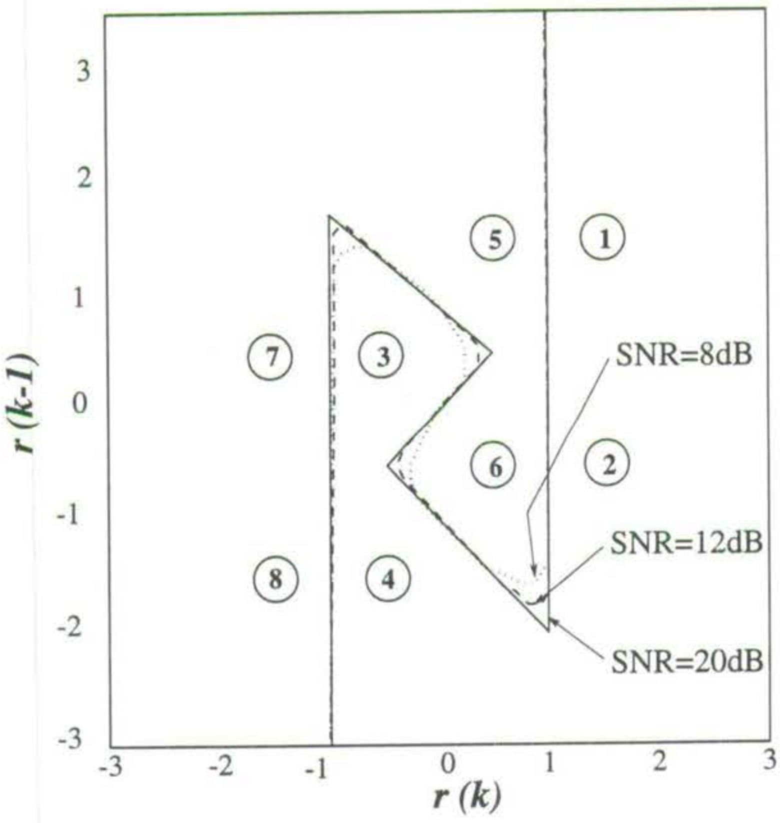

Figure 4.4: Bayesian decision boundaries for channel $H(z) = 0.5 + 1.0z^{-1}$ with delay $d = 0$ at SNR = 8dB, 12dB and 20dB

The channel state locations and SNR operating conditions do not uniquely define the decision boundary. Given a set of channel states and a fixed SNR value, the decision boundary can also be changed by using different delay orders. As an example, we illustrate the Bayesian decision boundaries for the channel $H(z)$ (Eq. 4.23) using delay order $d = 0, 1$ and 2, feedforward order $m = 2$, and SNR = 30dB. The set of channel states for this problem is listed in Table 4.1 and the corresponding Bayesian decision boundaries for the various delay orders are illustrated in Fig. 4.5.

Note that the channel state positions remain unchanged for different delay orders. It is the changes in the delay order parameter value that modify the shape of the decision boundaries. It is observed that all three decision boundaries are nonlinear, and in the case of using delay order $d = 0$, the set of channel states $C_d^{(+)}$ and $C_d^{(-)}$ become not linearly separable and hence cannot be partitioned correctly using linear boundaries. In the cases of using delay order $d = 1$ and 2, the figure shows that the decision boundaries are less nonlinear and that it is possible to employ linear boundaries to partition the channel states into the two classes correctly. In addition, our studies (Sec. 4.3.2) reveal that the probability of misclassifications performance of the equaliser is significantly affected by the delay order parameter. In general, delay orders resulting in the channel states $C_d^{(+)}$ and $C_d^{(-)}$ being linearly separable have better classification performance than those that result in these two sets of channel states being non-linearly separable.

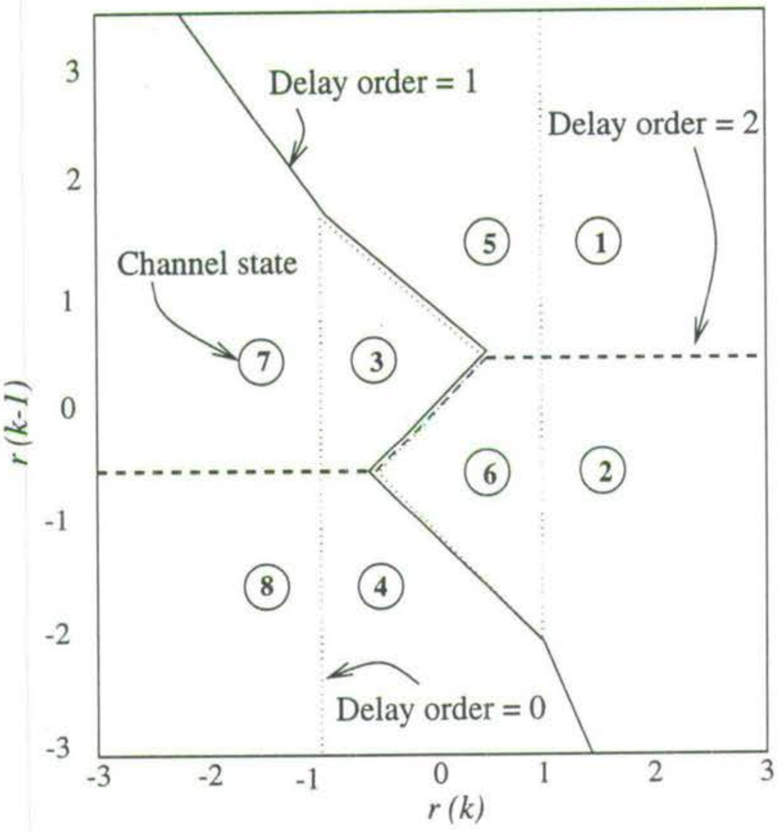

Figure 4.5: Bayesian decision boundaries for channel $H(z) = 0.5 + 1.0z^{-1}$ with delay $d = 0, 1$ and 2 at SNR = 30dB.

4.3 Probability of misclassification

This section presents the analysis of probability of misclassification by the Bayesian equaliser.

Due to noise, the received sequence $\mathbf{r} = \mathbf{r}(k)$ may be perturbed into regions defined for a different classification, e.g. given $\mathbf{c}_j \in C_d^{(+)}$, the misclassification region for the Bayesian equaliser is $Z_b^{(-)}$, and given $\mathbf{c}_j \in C_d^{(-)}$, the misclassification region is $Z_b^{(+)}$. The regions $Z_b^{(+)}$ and $Z_b^{(-)}$ are defined by the following equations,

$$Z_b^{(+)} = \{\mathbf{r} \mid f_b(\mathbf{r}) \geq 0\}, \tag{4.24}$$

$$Z_b^{(-)} = \{\mathbf{r} \mid f_b(\mathbf{r}) < 0\}. \tag{4.25}$$

To evaluate the probability of misclassification, we first evaluate the probability of misclassific-ation conditioned on the received sequence for each channel state, i.e.,

$$P_{bc}(\mathbf{c}_j) = \int_{\mathbf{r}\in Z_b^{(-)}} p(\mathbf{r}|\mathbf{c}_j)d\mathbf{r}, \qquad \text{for } \mathbf{c}_j \in C_d^{(+)}, \tag{4.26}$$

$$P_{bc}(\mathbf{c}_k) = \int_{\mathbf{r}\in Z_b^{(+)}} p(\mathbf{r}|\mathbf{c}_k)d\mathbf{r}, \qquad \text{for } \mathbf{c}_k \in C_d^{(-)}. \tag{4.27}$$

The above equations state that the probability of misclassification of a channel state is equal to the integral of the pdf $p(\mathbf{r}|\mathbf{c})$ over the misclassification region.

The total probability of misclassification P_b by the Bayesian equaliser is therefore the mean of the probability of misclassification for all channel states, i.e.,

$$P_b = \sum_{\mathbf{c}_j\in C_d^{(+)}} \frac{1}{N_s} \int_{\mathbf{r}\in Z_b^{(-)}} p(\mathbf{r}|\mathbf{c}_j)d\mathbf{r} + \sum_{\mathbf{c}_k\in C_d^{(-)}} \frac{1}{N_s} \int_{\mathbf{r}\in Z_b^{(+)}} p(\mathbf{r}|\mathbf{c}_k)d\mathbf{r} \tag{4.28}$$

where the constant $1/N_s$ is the *a priori* probability of channel state being $\mathbf{c}_j$ or $\mathbf{c}_k$. Because the channel states constellation is symmetric, the probability of misclassification can be reduced to,

$$P_b = 2\sum_{\mathbf{c}_j\in C_d^{(+)}} \frac{1}{N_s} \int_{\mathbf{r}\in Z_b^{(-)}} p(\mathbf{r}|\mathbf{c}_j)d\mathbf{r}. \tag{4.29}$$

4.3.1 Estimating the probability of misclassification

The evaluation of probability of misclassification using Eq. 4.29 involves evaluating m-dimensional integrals over the error region $Z_b^{(-)}$. As a closed-form solution for the expression does not exist, one must resort to numerical methods. This option, however, is un-attractive for large m. If our requirement of finding classification performance is only one of comparing relative performance for equalisers using different delay orders, a simple approximation may be used to estimate this measure.

The probability of misclassifications P_b (Eq. 4.29) can be re-written as

$$P_b = \frac{2}{N_s} \sum_{\mathbf{c}_j\in C_d^{(+)}} P_{bc}(\mathbf{c}_j). \tag{4.30}$$

The problem therefore is to calculate $P_{bc}(\mathbf{c}_j)$. To simplify, if the channel state $\mathbf{c}_j$ is wrongly classified, we assigned $P_{bc}(\mathbf{c}_j) = 1.0$. For the case when the channel state is correctly classified, it can be shown that when SNR $\rightarrow \infty$, $P_{bc}(\mathbf{c}_j)$ is reduced to the minimum distance bound [106] i.e.,

$$P_{bc}(\mathbf{c}_j) = Q(|\zeta_j|/\sigma_e) = \int_{|\zeta_j|}^{\infty} (2\pi\sigma_e^2)^{-1/2} \exp(-x^2/(2\sigma_e^2))dx \tag{4.31}$$

where $|\zeta_j|$ is the absolute minimum Euclidean distance of $\mathbf{c}_j$ to the decision boundary. To reiterate, $P_{bc}(\mathbf{c}_j)$ is the probability of misclassification conditional on channel state being $\mathbf{c}_j$ and can be evaluated approximately using the following equations,

$$P_{bc}(\mathbf{c}_j) = \begin{cases} 1, & \text{if } \mathbf{c}_j \text{ is wrongly classified,} \\ Q(|\zeta_j|/\sigma_e), & \text{if } \mathbf{c}_j \text{ is correctly classified.} \end{cases} \tag{4.32}$$

Therefore to evaluate $P_{bc}(\mathbf{c}_j)$, we only need to find the minimum distances of the channel states to the decision boundary. This can be performed using the following algorithm.

Algorithm 4.1 : Evaluating the minimum distance ζ_j

```
For $\mathbf{c}_j \in C_d^{(+)}$
    $\mathbf{c}_{k(nearest)} = \min_{\mathbf{c}_k}\{||\mathbf{c}_j - \mathbf{c}_k||\}, \mathbf{c}_k \in C_d^{(-)}$
    We assume that the closest decision boundary point to $\mathbf{c}_j$ lies between $\mathbf{c}_j$ and $\mathbf{c}_{k(nearest)}$.
    To find the location of the boundary point, evenly spaced points along the vector from $\mathbf{c}_j$
    to $\mathbf{c}_{k(nearest)}$ are evaluated for the occurrence of a sign change in the decision function
    $f_b(.)$. A sign change indicates the change in the classification region. We denote the
    boundary point thus found using $\mathbf{r}_0^*$, therefore,
        $\zeta_{j(appr)} = ||\mathbf{c}_j - \mathbf{r}_0^*||$
next $\mathbf{c}_j$
```

The algorithm is based on the assumption that the closest boundary point to a channel state $\mathbf{c}_j \in C_d^{(+)}$ is between $\mathbf{c}_j$ and the nearest channel state $\mathbf{c}_k \in C_d^{(-)}$. This is however only true for $\mathbf{c}_j \in U$ and in high SNR cases. The set U is defined as the set of centres nearest to the boundary points, i.e.

$$\{U\} \leftarrow \min_{\mathbf{c}_j}\{||\mathbf{r}_0 - \mathbf{c}_j||\}, \qquad 1 \leq j \leq N_s, \tag{4.33}$$

where $\mathbf{r}_0$ is the set of all boundary points. In the cases when $\mathbf{c}_j \notin U$, the algorithm will produce an over-estimate of the minimum distance $\zeta_{j(appr)}$. The following paragraph illustrates with a geometric example how an over-estimate of ζ may occur.

As an example, we examine the approximation of ζ for the channel $H(z) = 0.5 + 1.0z^{-1}$ using delay order $d = 1$ and $m = 2$; the channel state locations and decision boundary are illustrated in Fig. 4.5. In this case, the set of centres nearest to the decision boundary are $U = \{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$, and the set of centres not in U, i.e. $\overline{U} = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_7, \mathbf{c}_8\}$. Fig. 4.6 illustrates why the algorithm will produce an over-estimate of ζ_1 as $\mathbf{c}_1$ is not a member of U. The nearest channel state to $\mathbf{c}_1$ from the set $C_d^{(-)}$ is $\mathbf{c}_3$. The boundary point $\mathbf{r}_0^*$ between $\mathbf{c}_1$ and $\mathbf{c}_3$ is found by examining intervals along the vector $\mathbf{c}_1$ to $\mathbf{c}_3$. As the evaluated points along the intervals are not normal to the decision boundary, the estimated $\zeta_{1(appr)}$ would therefore be greater than the actual $\zeta_{1(act)}$.

In the case of estimating ζ for centres belonging to U, e.g. ζ_6, the boundary point is found by inspecting intervals along the normal to the boundary[4]. Therefore, the estimated $\zeta_{6(appr)} = \zeta_{6(act)}$.

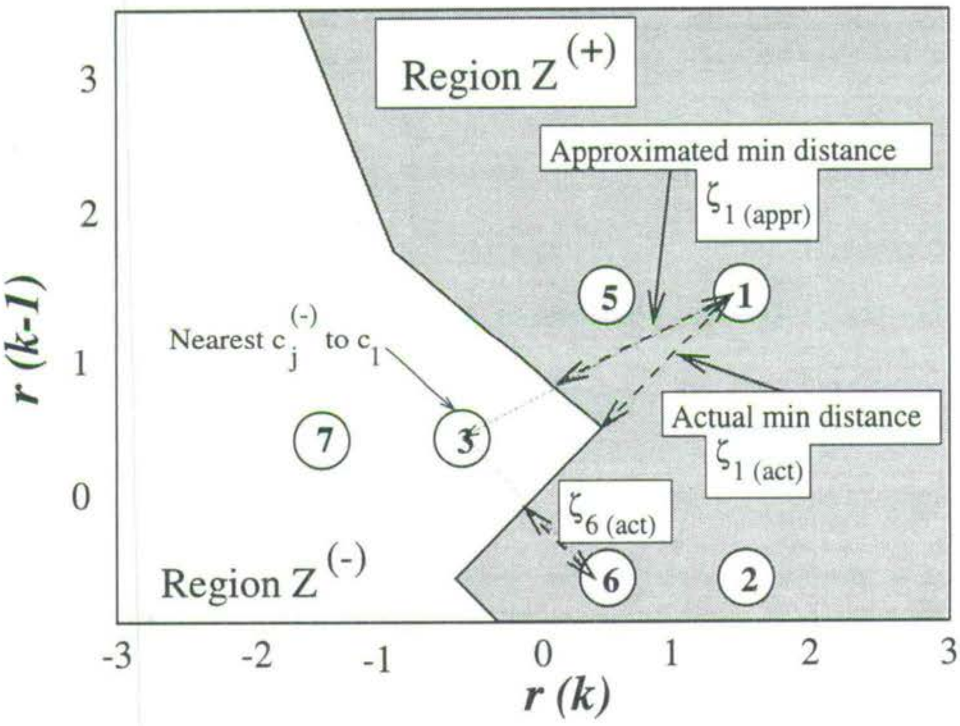

Figure 4.6: Minimum distance of channel states $\mathbf{c}_i$ to decision boundary.

The over-estimation of ζ however does not seriously affect the evaluated probability of misclassification. To show why this is so, let us examine the expression of P_b (Eq. 4.30) when it is decomposed into two parts,

[4] It will be shown in Sec. 4.4.1 that the sets of centres belonging to U defines the decision boundary and that the decision boundary is formed by hyperplanes partitioning at the normal of closest pairs of centres of different classes at high SNR.

$$P_b = \frac{1}{N_s}\left(\sum_{\mathbf{c}_j \in U} P_{bc}(\mathbf{c}_j) + \sum_{\mathbf{c}_k \notin U} P_{bc}(\mathbf{c}_k)\right), \tag{4.34}$$

where one part of the above expression accounts for the probability of misclassification for the set of channel states belonging to U and the other part of the expression accounts for the probability of misclassification for the set of channel states not belonging to U. As a direct result of Eq. 4.33, the distances to the decision boundary for channel states $\mathbf{c}_k$ not belonging to U are greater than those from the channel states $\mathbf{c}_j$ belonging to U. As the probability of misclassification for each channel state is a function of these distances ζ (Eq. 4.31), it is obvious that the channel states which are further away from the boundary have a significantly smaller misclassification probability than those that are nearer to the boundary at high SNR. Hence, at sufficiently high SNR, the over-estimate of minimum distances for $\mathbf{c}_k \notin U$ will not seriously affect the estimate of P_b. These is confirmed by simulation results in the following section.

4.3.2 BER estimate : Some simulation results

The measure of an equaliser's performance in communication literature is usually expressed in terms of the bit error rate (BER) and is evaluated using the following equation,

$$\text{BER} = \log_{10} P_b. \tag{4.35}$$

Simulations were conducted to compare the BER results calculated using Eqs. 4.30 and 4.31 with those obtained by Monte Carlo (MC) simulations for the following channels,

$$\begin{aligned} H1(z) &= 0.8745 + 0.4372z^{-1} - 0.2098z^{-2}, & (4.36)\\ H2(z) &= 0.2620 - 0.6647z^{-1} - 0.6995z^{-2}. & (4.37) \end{aligned}$$

These two channels exhibit the same magnitude but different phase response. For the experiment, the equaliser's feedforward order m was chosen to be 4 with the transmit symbol alphabet $\{\pm 1\}$. Fig. 4.7 compares the BER estimates of Eqs. 4.30 and 4.31 with those of MC simulations for the two channels using different delay orders. The results show that the proposed BER estimates are very close to the MC simulations, and that the application of different delay order parameter values will result in very different lower limits of BER performance for a given SNR condition. To illustrate this strong dependence of the equaliser's performance with respect to the delay order, the performance of the equaliser using the delay parameter as the horizontal axis is illustrated in Figs. 4.8 for the SNR condition of 12dB, 14dB and 16dB. The results show that given the same operating SNR conditions, the difference in BER performance can be very significant. For example, the BER performance of the Bayesian equalisers on channel $H1(z)$ (Fig. 4.8a) using delay order $d = 3$ is -4.1 and the BER performance using $d = 5$ is just -0.97.

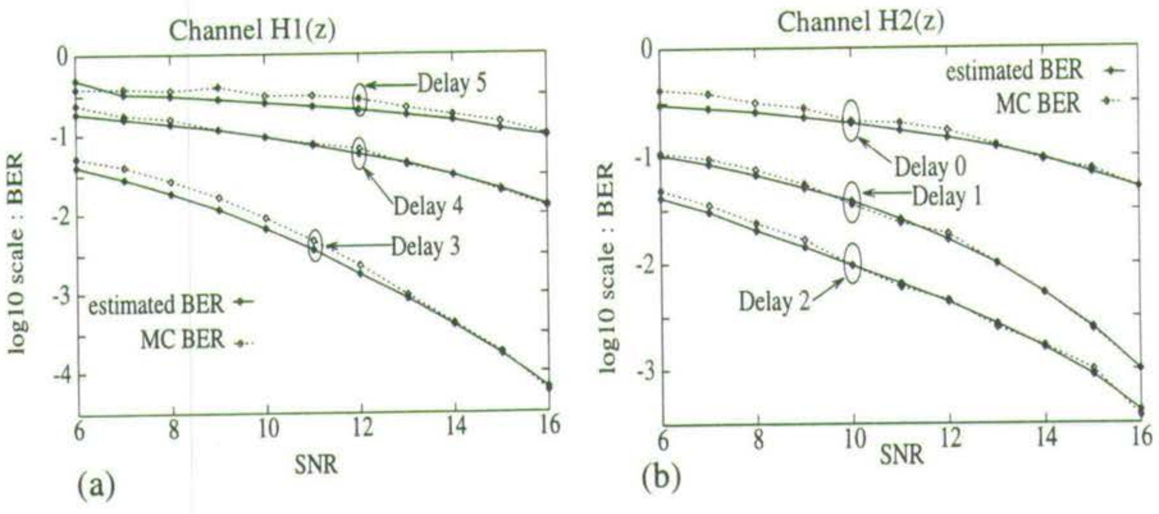

Figure 4.7: Estimated and MC simulations of BER vs SNR for (a) channel H1(z), (b) channel H2(z).

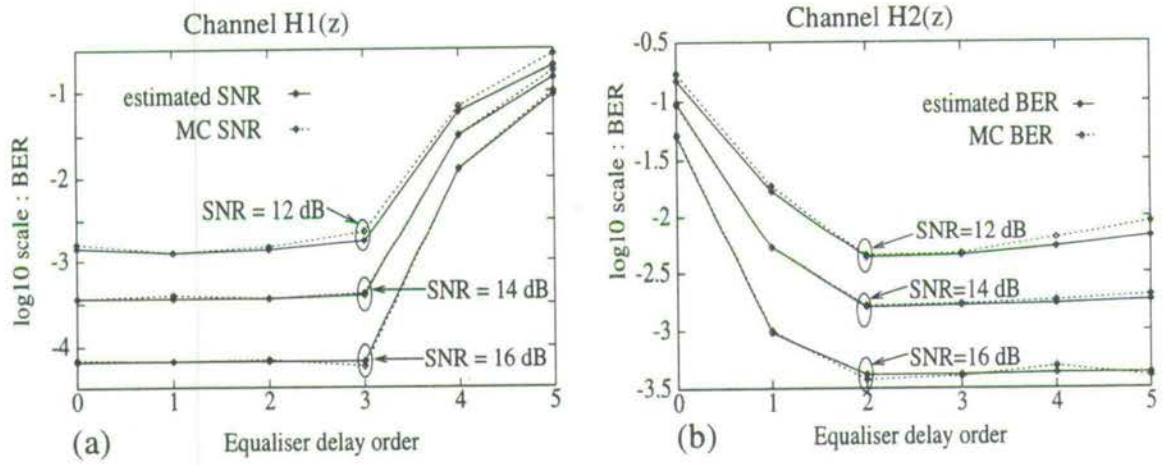

Figure 4.8: Estimated and MC simulations of BER vs delay order at SNR 12dB, 14dB and 16dB for (a) channel $H1(z)$, (b) channel $H2(z)$.

4.4 RBF equaliser implementation complexity

This section discusses the implementation complexity of the RBF equaliser. In Sec. 4.2.3, it was shown that the RBF network can realise the Bayesian equaliser. The computational requirement of implementing the full Bayesian solution using the RBF equaliser is however considerable. This is because the implementation complexity of the RBF equaliser grows proportionally to the number of channel states N_s, and N_s grows exponentially with the number of feedforward terms m and channel length $n_a - 1$, i.e. $N_s = 2^{m+n_a-1}$. The following list the numbers of exp(.) function evaluations, additions and multiplications required to make a decision [18]:

$$\begin{aligned}\text{Number of exp(.)} &= N_s \\ \text{Number of additions} &= (N_s \times m) + N_s \\ \text{Number of multiplications} &= N_s \times m \end{aligned} \tag{4.38}$$

In the case when m or n_a is large, the full implementation using all the N_s channel states as centres may be impractical. One method of reducing the implementation complexity is to reduce the number of channel states required to implement the decision function using decision feedback [18]. This method will be discussed in detail in Section 4.5. The other more direct method of reducing the number of channel states is to perform selection from the N_s channel states and to pick a subset of these states to approximate the optimum decision boundary [19]. The implementation complexity is reduced by not including those channel states which do not contribute to the 'position' of the decision boundary. The following section presents an algorithm to perform channel state selection to generate the subset RBF equaliser.

4.4.1 Selecting subset RBF model

This section examines subset model selection algorithms to reduce the implementation complexity of the RBF equaliser. The objective is to find a smaller-sized, in terms of number of centres, RBF model to realise or to approximate the same Bayesian decision function as the full model.

Before examining the subset model selection algorithm, we have to first understand how centre (i.e., channel state) locations affect decision boundaries. To do so, we analyse the Bayesian decision function $f_b(.)$ (Eq. 4.20) for $\sigma_e \to 0$. Although the boundary thus determined is only exact in infinite SNR, experimental results [107] and our example in Sec. 4.2.5 have shown that the true decision boundary closely approximates the asymptotic decision boundary in high SNR condition. Following the work performed in [107], we state the proposition which establishes the asymptotic decision boundaries.

Proposition 4.1 : The following conditions are sufficient for a point $\mathbf{r}_0$ to lie on the asymptotic decision boundary.

(i). $||\mathbf{r}_0 - \mathbf{c}^+||^2 < ||\mathbf{r}_0 - \mathbf{c}_j||^2$, where $\mathbf{c}^+ \in C_d^{(+)}$, and $\mathbf{c}_j = \{\mathbf{c}_i \in C_d^{(+)} \cap \mathbf{c}_i \notin \mathbf{c}^+\}$.

(ii). $||\mathbf{r}_0 - \mathbf{c}^-||^2 < ||\mathbf{r}_0 - \mathbf{c}_k||^2$, where $\mathbf{c}^- \in C_d^{(-)}$, and $\mathbf{c}_k = \{\mathbf{c}_i \in C_d^{(-)} \cap \mathbf{c}_i \notin \mathbf{c}^-\}$.

(iii). $||\mathbf{r}_0 - \mathbf{c}^+||^2 = ||\mathbf{r}_0 - \mathbf{c}^-||^2$,

where the channel states $\mathbf{c}^+ \in C_d^{(+)}$ and $\mathbf{c}^- \in C_d^{(-)}$ are a pair of channel states closest in

Euclidean distance to $\mathbf{r}_0$. To prove the above proposition, let us consider the Bayesian decision function $f_b(.)$ (Eq. 4.20) for the input vector $\mathbf{r} = \mathbf{r}_0$. This implies $f_b(\mathbf{r}_0) = 0$ and Eq. 4.20 becomes

$$\sum_{\mathbf{c}_j \in C_d^{(+)}} \exp(-\|\mathbf{r}_0 - \mathbf{c}_j\|^2/(2\sigma_e^2)) - \sum_{\mathbf{c}_k \in C_d^{(-)}} \exp(-\|\mathbf{r}_0 - \mathbf{c}_k\|^2/(2\sigma_e^2)) = 0, \tag{4.39}$$

and therefore,

$$\sum_{\mathbf{c}_j \in C_d^{(+)}} \exp(-\|\mathbf{r}_0 - \mathbf{c}_j\|^2/(2\sigma_e^2)) = \sum_{\mathbf{c}_k \in C_d^{(-)}} \exp(-\|\mathbf{r}_0 - \mathbf{c}_k\|^2/(2\sigma_e^2)). \tag{4.40}$$

Dividing both sides of the Eq. 4.40 by the summand corresponding to $\mathbf{c}^+$, i.e. $\exp(-\|\mathbf{r}_0 - \mathbf{c}^+\|^2/(2\sigma_e^2))$, and assuming that condition (3) of the proposition is true, we have

$$\begin{aligned} 1 + \sum_{(\mathbf{c}_j \in C_d^{(+)}) \cap (\mathbf{c}_j \notin \mathbf{c}+)} \exp(-\|\mathbf{r}_0 - \mathbf{c}_j\|^2/(2\sigma_e^2)) &= \\ 1 + \sum_{(\mathbf{c}_k \in C_d^{(-)}) \cap (\mathbf{c}_k \notin \mathbf{c}-)} \exp(-\|\mathbf{r}_0 - \mathbf{c}_k\|^2/(2\sigma_e^2)). \end{aligned} \tag{4.41}$$

As σ_e^2 tends to 0, the summation term on the left hand side of Eq. 4.41 will vanish if condition (i) of the proposition is satisfied, to show that the dominating energy from the channel states in $C_d^{(+)}$ is from $\mathbf{c}^+$. Similarly, the summation term on the right hand side of Eq. 4.41 will vanish, if conditions (ii and iii) of the proposition are satisfied, to show that the dominating energy is contributed by $\mathbf{c}^-$. It is therefore obvious that the decision boundary in the regions between $\{\mathbf{c}^+, \mathbf{c}^-\}$ is formed by a hyperplane bisecting the space between the two channel states. For example, Fig. 4.9 shows that the decision boundary partitioning the pair of channel states $\{\mathbf{c}_3, \mathbf{c}_5\}$, which belongs to U, is a line passing perpendicularly between the space of the two channel states at the mid-point.

Following the above argument, it is obvious that the asymptotic decision boundary is defined by the set of all channel states nearest to the decision boundary [107]. We denote the set of channel states that satisfy this condition as U^+ and U^-, i.e.,

$$\{U^+\} \leftarrow \min_{\mathbf{c}_j}\{\|\mathbf{r}_0 - \mathbf{c}_j\|^2\}, \qquad \mathbf{c}_j \in C_d^+, \tag{4.42}$$

$$\{U^-\} \leftarrow \min_{\mathbf{c}_k}\{\|\mathbf{r}_0 - \mathbf{c}_k\|^2\}, \qquad \mathbf{c}_k \in C_d^-, \tag{4.43}$$

and the union of U^+ and U^- as U. The channel states that are members of the set U must satisfy the following two conditions,

(i). The mid point $\mathbf{r}^*_{j,k}$ between the two states $\mathbf{c}_j \in C_d^+$ and $\mathbf{c}_k \in C_d^-$ would be a boundary point. To test if $\mathbf{r}^*_{j,k}$ is a boundary point, the Bayesian decision function at that point is

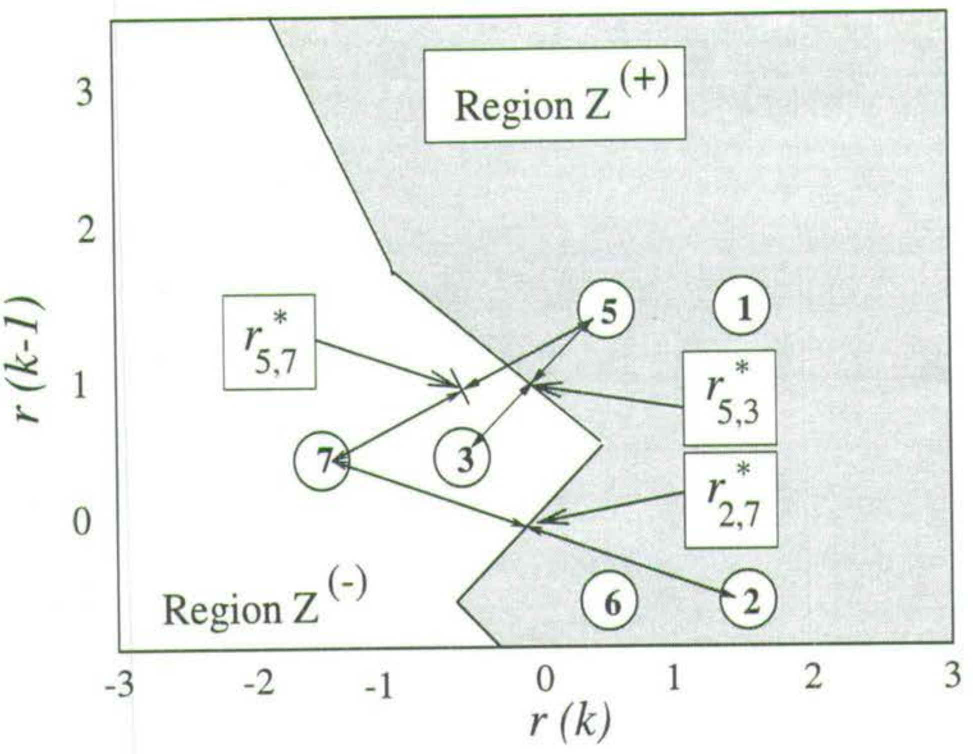

Figure 4.9: Channel $H(z) = 0.5 + 1.0z^{-1}$ using $d = 1$ and $m = 2$. The mid point $\mathbf{r}^*_{j,k}$ between channel states of different transmit class.

evaluated, i.e. $f_b(\mathbf{r}^*_{j,k})$ [5], to test if the value is 0.

(ii). In addition, the channel states $\mathbf{c}_j$ and $\mathbf{c}_k$ must be the closest in the Euclidean norm sense to the mid point $\mathbf{r}^*_{j,k}$.

Fig. 4.9 illustrates three different types of mid points, namely $\mathbf{r}^*_{5,3}$, $\mathbf{r}^*_{5,7}$, and $\mathbf{r}^*_{2,7}$. It is shown that the mid point for the pair of channel states $\{\mathbf{c}_5, \mathbf{c}_3\}$ satisfies the above two conditions and these two channel states are therefore members of the set U. In the case of the mid point for the pair of channel states $\{\mathbf{c}_2, \mathbf{c}_7\}$, although condition (i) is satisfied, i.e. the mid point $\mathbf{r}^*_{2,7}$ falls into the boundary, condition (ii) is not satisfied as $\{\mathbf{c}_6, \mathbf{c}_3\}$ are nearer to $\mathbf{r}^*_{2,7}$. Therefore these two channel states are not accepted as members of the set U. And for the case of the mid point for the pair of channel states $\{\mathbf{c}_5, \mathbf{c}_7\}$, the mid point $\mathbf{r}^*_{5,7}$ does not fall into the decision boundary and these two channel states therefore do not qualify as members of U.

[5]In practical cases, we evaluate $f_b(\mathbf{r}^*_{j,k} + \mathbf{q})$ and $f_b(\mathbf{r}^*_{j,k} - \mathbf{q})$ to see if there is a sign change in the evaluated values. The vector $\mathbf{q}$ is a scaled vector having a very small Euclidean norm with a direction along the two channel states.

The following algorithm summarises the steps to find U,

Algorithm 4.2 : Finding $\{U\}$

```
For $\mathbf{c}_j \in C_d^{(+)}$
    For $\mathbf{c}_k \in C_d^{(-)}$
        { $\mathbf{r}^*_{j,k} = \mathbf{c}_j + (\frac{\mathbf{c}_k - \mathbf{c}_j}{2})$

            if $\left[ \begin{array}{l} f_b(\mathbf{r}^*_{j,k}) \equiv 0 \text{ and} \\ \mathbf{c}_j \equiv \min_{\mathbf{c}_i \in C_d^{(+)}} \{||\mathbf{r}^*_{j,k} - \mathbf{c}_i||\} \end{array} \right]$        (4.44)

            $\mathbf{c}_j \rightarrow \{U^+\}, \mathbf{c}_k \rightarrow \{U^-\}$
        }
    next $\mathbf{c}_k$,
next $\mathbf{c}_j$.
```

Algorithm 4.2 was tested to find the the set U to form the subset RBF equalisers with feedforward order $m = 2$ for channel [6] $H(z) = 0.5 + 1.0z^{-1}$. When delay order $d = 0$ was used, all the centres $\{\mathbf{c}_1, \ldots, \mathbf{c}_8\}$ were picked to form the subset model (Fig. 4.5). This is because all the centres were required to define the decision boundary. For the case of using delay order $d = 1$, the algorithm picked the subset centres $\{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$ to form the subset model. Fig. 4.10a compares the subset equaliser's decision boundary to the Bayesian solution. The results show that the approximated boundary is very similar to the actual optimum decision boundary, and the regions of the decision boundary which are different are very 'far' away from the channel state locations. This implies that the classification performance of the subset equaliser will be very similar to the full Bayesian solution.

Although the above example shows that Algorithm 4.2 can pick a good solution for the subset equaliser, the algorithm is not optimum in the sense that redundant channel states may also be included in the selection. To illustrate this, consider the selected subset model when delay order 2 is used. By visual inspection of Fig. 4.5 and Fig. 4.10b, it is clear that the subset model with channel states $\{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$ is sufficient to approximate the Bayesian boundary. Algorithm 4.2, however, picked all the centres to form the subset model. The reason for including $\{\mathbf{c}_1, \mathbf{c}_2\}$ and $\{\mathbf{c}_7, \mathbf{c}_8\}$ is that these two pairs of channel states also satisfy Eq. 4.44 and thus were accepted into the set of selected centres. They are, however, unnecessary because the decision boundary formed using centres $\{\mathbf{c}_3, \mathbf{c}_4\}$ and $\{\mathbf{c}_5, \mathbf{c}_6\}$ defines the same decision boundary.

To minimise the inclusion of redundant centres, an additional condition is introduced in Algorithm 4.2 to verify whether the new channel states under consideration would affect the decision boundary realised by the currently selected subset RBF equaliser. The function $f_s(.)$

[6] The list of channel states for this problem can be found in Sec. 4.2.4

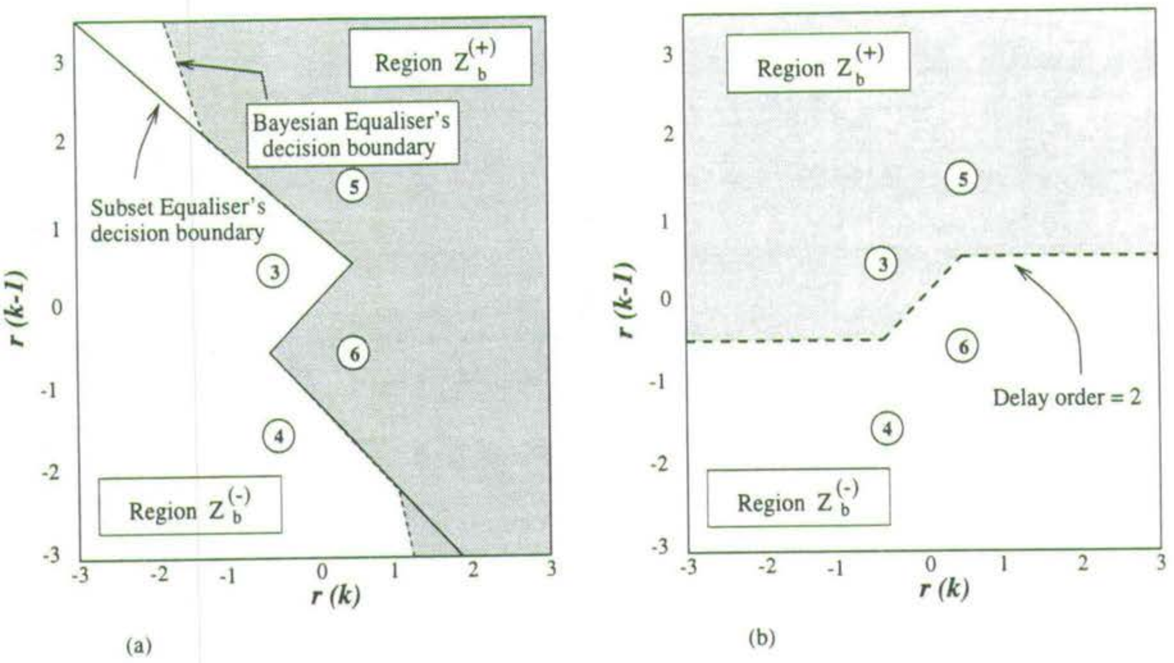

Figure 4.10: Realisation of decision boundary using subset RBF model. (a) : decision boundary for delay $d = 1$, (b) : decision boundary for delay $d = 2$

denotes the selected subset equaliser's decision function and is formed by applying the current selected channel states in U. If the decision boundary changes with the inclusion of the new centres, they will be accepted, otherwise they will be ignored. The algorithm for the improved version is as follows:

```
Algorithm 4.3 : Finding {U}
        ⋮
$\mathbf{r}^*_{j,k} = \mathbf{c}_j + (\frac{\mathbf{c}_k - \mathbf{c}_j}{2})$

if $\begin{bmatrix} f_b(\mathbf{r}^*_{j,k}) \equiv 0 & \text{and} \\ \mathbf{c}_j = \arg\min_{\mathbf{c}_i \in C_d^{(+)}} \{\|\mathbf{r}^*_{j,k} - \mathbf{c}_i\|^2\} & \text{and} \\ f_s(\mathbf{r}^*_{j,k}) \neq 0 & \end{bmatrix}$        (4.45)

    $\mathbf{c}_j \rightarrow \{U^+\}, \mathbf{c}_k \rightarrow \{U^-\}$
    $f_s$ = subset RBF model formed using current $U^+, U^-$ as centres.
        ⋮
```

Although algorithm 4.3 can reduce the number of redundant states in the selection process, not all the redundant states are removed. This can be observed when the algorithm is applied to select U for the previous problem (Fig. 4.5) using $d = 2$. The selection process examined the channel states in $C_d^{(+)}$ in ascending order, i.e. from $\mathbf{c}_1$,$\mathbf{c}_3$, $\mathbf{c}_5$ to $\mathbf{c}_7$. When the channel state $\mathbf{c}_1$ was considered in the selection process, the algorithm accepted the pair of channel states $\{\mathbf{c}_1, \mathbf{c}_2\}$ into the set U since the selection criteria Eq. 4.45 were satisfied. The selection process continued with the second channel state belonging to $C_d^{(+)}$, i.e. $\mathbf{c}_3$. These two pairs of channel states, i.e. $\{\mathbf{c}_3, \mathbf{c}_4\}$ and $\{\mathbf{c}_3, \mathbf{c}_6\}$ were accepted as the selection criteria were also met. In this case, the two pairs of channel states were included because $\mathbf{c}_3$ had two hyperplane decision boundaries surrounding it. For the case of channel state $\mathbf{c}_5$, the pair of channel states $\{\mathbf{c}_5, \mathbf{c}_6\}$ was rejected as the current selected set U already contained $\{\mathbf{c}_1, \mathbf{c}_2\}$ which defined the same boundary. Similarly, for $\mathbf{c}_7$, the algorithm also rejected the pair $\{\mathbf{c}_7, \mathbf{c}_8\}$ as redundant since the selected set U already contained $\{\mathbf{c}_3, \mathbf{c}_4\}$ which defined the same boundary. Therefore, the selected subset U for this example is $U = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_6\}$. In this case, the selection algorithm did not find the optimum solution as the same decision boundary could have been generated by using a smaller set of channel states, namely $\{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$.

4.4.2 Subset model selection : some simulation results

Simulations were conducted using Algorithm 4.3 to select subset RBF equalisers from the full model applied to channels $H1(z)$ and $H2(z)$ (Eqs. 4.36 and 4.37). The feedforward order used was $m = 4$, resulting in a full model with 64 centres. Using the SNR condition of 16dB, simulations were conducted to evaluate the performance of the subset RBF, full RBF and the linear MMSE [7] equalisers operating on the two channels. The results are listed in Table 4.2 and 4.3; the first column of each table indicates the delay order parameter, the second column shows the size of the subset model used while the third, fourth and fifth columns list the BER performance of the respective equalisers and the last column indicates whether the two subsets channel states $C_d^{(+)}$ and $C_d^{(-)}$ are linearly or not linearly separable.

Our results indicate that the full RBF model's BER performance for the cases when the equalisation problem is linearly separable is normally better than the cases when the problem is not linearly separable. This is not surprising since decision boundaries which are not linearly separable tend to be much more complicated and have more channel states with different decision outputs near to each other. The results also indicate that a smaller-sized RBF subset equaliser with classification performance very similar to the full solution can usually be found for the case when the equalisation problem is linearly separable.

[7]The linear MMSE equaliser is discussed in Chapter 5. It is the linear equaliser with weights found by minimising the MSE.

Delay	Subset Size	Subset BER	Full-model BER	Linear-Eq BER	Decision Boundary
0	56	-4.09	-4.09	-3.44	Linear Sep.
1	57	-4.14	-4.14	-3.07	Linear Sep.
2	32	-4.11	-4.12	-2.36	Linear Sep.
3	32	-4.11	-4.12	-1.84	Linear Sep.
4	48	-1.91	-1.91	-0.59	Not-Linear Sep.
5	64	-0.97	-0.97	-0.37	Not-Linear Sep.

Table 4.2: Channel $H1(z)$: Performance of full-model (64 centres), subset RBF model and linear MMSE equaliser at SNR = 16dB.

Delay	Subset Size	Subset BER	Full-model BER	Linear-Eq BER	Decision Boundary
0	56	-0.80	-1.30	-0.37	Not-Linear Sep.
1	46	-2.99	-2.99	-1.61	Linear Sep.
2	38	-3.38	-3.38	-2.67	Linear Sep.
3	56	-3.43	-3.43	-1.94	Linear Sep.
4	55	-3.32	-3.32	-1.16	Not-Linear Sep.
5	54	-3.41	-3.41	-0.80	Not-Linear Sep.

Table 4.3: Channel $H2(z)$:Performance of full-model (64 centres), subset RBF model and linear MMSE equaliser at SNR = 16dB.

4.5 Bayesian Decision Feedback Equaliser

This section deals with the Bayesian decision feedback equaliser (DFE) [17, 18]. The Bayesian DFE is very similar to the Bayesian equaliser without feedback as described in the previous section. The difference lies in the additional application of past detected symbols to the decision function. The reasons for introducing decision feedback are to reduce implementation complexity and to improve classification performance. There is however one disadvantage when applying decision feedback, namely, it can introduce error propagation [3, 71]. Error propagation occurs when the equaliser makes wrong decisions and passes these wrong decisions to the feedback vector. Because of the wrong information being introduced to the equaliser, more error decisions may occur as a direct result. This event is known as error propagation or error burst [108–111]. Error propagation, however, is not catastrophic and the equaliser normally recovers with only a small performance loss. In almost all cases, the equaliser's performance is improved by the introduction of feedback.

In the following sections, the operations of the DFE is discussed, and the formulation of the Bayesian DFE is examined. In addition, a method of transforming the DFE problem into a feedforward structure is considered.

4.5.1 Structure of the Bayesian DFE

The generic structure of a DFE is illustrated in Fig. 4.11. There are two input vectors to the equaliser. The first input vector is the observation vector $\mathbf{r}(k)$, i.e.,

$$\mathbf{r}(k) = [r(k) \ \cdots \ r(k-m+1)]^T \in R^m, \tag{4.46}$$

and this vector is the same as the input vector to the Bayesian equaliser without feedback discussed in the previous sections. The second input vector to the DFE is a vector of past detected symbols, i.e.,

$$\hat{\mathbf{s}}_b(k) = [\hat{s}(k-d-1) \ \cdots \ \hat{s}(k-d-n)]^T \in R^n. \tag{4.47}$$

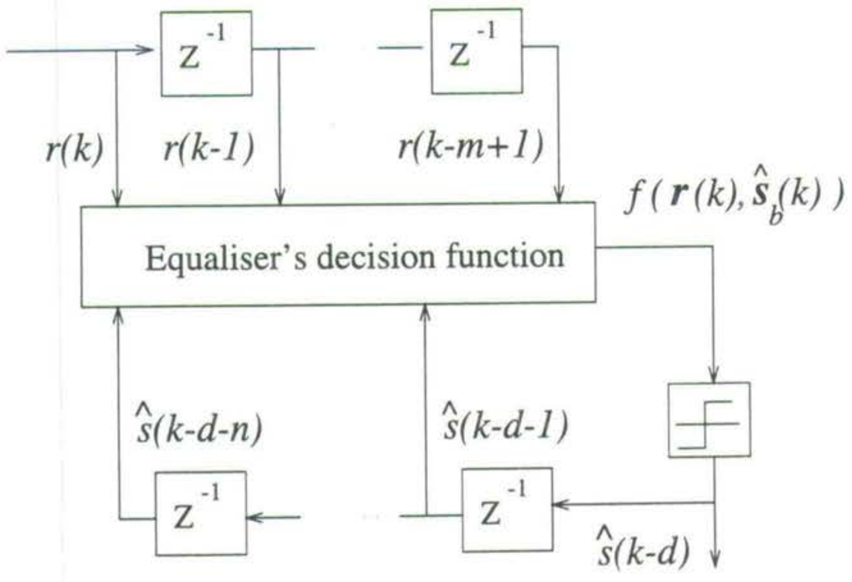

Figure 4.11: Schematic of a generic decision feedback equaliser.

The function of the Bayesian DFE is to use these two vectors to estimate the transmitted symbol $s(k-d)$, where the variables d, m and n are the decision delay order, feedforward order and feedback order parameters respectively. The feedforward order parameter is usually set to [17,18]

$$m = d + 1, \tag{4.48}$$

and the feedback order set to [17, 71, 112]

$$n = n_a - 1. \tag{4.49}$$

The remaining parameter d is usually set to

$$d = n_a - 1. \tag{4.50}$$

Note that the use of feedforward parameter $m = d + 1$ is optimum and the use of larger feedforward order $m > d + 1$ will not improve the performance of the equaliser [18].

4.5.2 Formulating the Bayesian DFE decision function

As in the feedforward Bayesian equaliser's case, the Bayesian DFE's decision function is to model the *a posteriori* probabilities $P(s(k-d) = s|\mathbf{r}(k))$ in terms of the conditional probabilities $p(\mathbf{r}(k)|s(k-d) = s)$ (Eq. 4.15) where $s \in \{\pm 1\}$. In the Bayesian equaliser's case without feedback, all the N_s channel states in $C_d = C_d^{(+)} \cup C_d^{(-)}$ are used to implement the decision function $f_b(\mathbf{r}(k))$ (Sec. 4.2). If the knowledge of past detected symbols is applied, only a subset of the channel states from C_d is required to implement the conditional Bayesian decision function $f_b(\mathbf{r}(k), \hat{\mathbf{s}}_b(k))$.

As discussed in Sec. 4.2.1, the set of channel states in C_d can be evaluated using

$$C_d = \{\hat{\mathbf{r}}(k)\} = \{F[\mathbf{s}_j(k)]\}, \qquad 1 \le j \le N_s. \tag{4.51}$$

For the DFE's case, the transmit sequence $\mathbf{s}(k)$ can be decomposed into two vectors

$$\mathbf{s}(k) = [\mathbf{s}_a^T(k), \mathbf{s}_b^T(k)]^T, \tag{4.52}$$

where

$$\begin{aligned} \mathbf{s}_a(k) &= [s_a(k) \cdots s_a(k-d)]^T \in R^{d+1}, & (4.53)\\ \mathbf{s}_b(k) &= [s_b(k-d-1) \cdots s_b(k-d-n)]^T \in R^n. & (4.54) \end{aligned}$$

There are $N_a = 2^{d+1} = 2^m$ combinations of $\mathbf{s}_a(k)$ and $N_b = 2^n$ combinations of $\mathbf{s}_b(k)$, i.e.

$$\begin{aligned} \mathbf{s}_{a,j} &= [s_{a,j}(k) \ \cdots \ s_{a,j}(k-d)]^T \in R^{d+1}, & 1 \le j \le N_a = 2^{d+1}, \quad (4.55)\\ \mathbf{s}_{b,l} &= [s_{b,l}(k-d-1) \ \cdots \ s_{b,l}(k-d-n)]^T \in R^{n}, & 1 \le l \le N_b = 2^{n}. \quad (4.56) \end{aligned}$$

Under the assumption that the given feedback $\mathbf{s}_b(k) = \mathbf{s}_{b,l}$ is correct, the set of possible channel states $C_{d,l}$ that can occur is determined by the combinations of $\mathbf{s}_a(k)$, i.e.,

$$C_{d,l} = \{F[\mathbf{s}_{a,j}^T, \ \mathbf{s}_{b,l}^T]^T\}, \qquad 1 \le j \le N_a. \tag{4.57}$$

The set of all channel states C_d is the union of these subsets $C_{d,l}$, i.e.

$$C_d = \bigcup_{1 \le l \le N_b} C_{d,l} \tag{4.58}$$

and

$$C_{d,l} \cap C_{d,k} = \emptyset \qquad \text{if } l \ne k. \tag{4.59}$$

The conditional Bayesian decision function $f_b(\mathbf{r}(k), \hat{\mathbf{s}}_b(k))$ for $\mathbf{s}_b(k) = \mathbf{s}_{b,l}$ is obtained by applying the set of channel states in $C_{d,l}$ to model the conditional probabilities $p(\mathbf{r}(k)|s(k-d)=s)$, i.e.

$$p(\mathbf{r}(k)|s(k-d) = +1 \cap \mathbf{s}_b(k) = \mathbf{s}_{b,l}) = \frac{1}{N_s} \sum_{\mathbf{c}_j \in C_{d,l}^{(+)}} p(\mathbf{r}(k)|\mathbf{c}_j), \tag{4.60}$$

$$p(\mathbf{r}(k)|s(k-d) = -1 \cap \mathbf{s}_b(k) = \mathbf{s}_{b,l}) = \frac{1}{N_s} \sum_{\mathbf{c}_k \in C_{d,l}^{(-)}} p(\mathbf{r}(k)|\mathbf{c}_k), \tag{4.61}$$

where the set $C_{d,l}^{(+)}$ and $C_{d,l}^{(-)}$ are

$$\begin{aligned} C_{d,l}^{(+)} &= \{\hat{\mathbf{r}}(k)|\mathbf{s}_b(k) = \mathbf{s}_{b,l} \cap s(k-d) = +1)\}, & (4.62)\\ C_{d,l}^{(-)} &= \{\hat{\mathbf{r}}(k)|\mathbf{s}_b(k) = \mathbf{s}_{b,l} \cap s(k-d) = -1)\}. & (4.63) \end{aligned}$$

Therefore the conditional Bayesian decision function is

$$\begin{aligned} f_b(\mathbf{r}(k), \hat{\mathbf{s}}_b(k) = \mathbf{s}_{b,l}) &= p(\mathbf{r}(k)|s(k-d) = +1 \cap \mathbf{s}_b(k) = \mathbf{s}_{b,l}) \\ &\qquad - p(\mathbf{r}(k)|s(k-d) = -1 \cap \mathbf{s}_b(k) = \mathbf{s}_{b,l}) \\ &= \frac{1}{N_s} \sum_{\mathbf{c}_j \in C_{d,l}^{(+)}} p(\mathbf{r}(k)|\mathbf{c}_j) - \frac{1}{N_s} \sum_{\mathbf{c}_k \in C_{d,l}^{(-)}} p(\mathbf{r}(k)|\mathbf{c}_k). \qquad (4.64) \end{aligned}$$

With the introduction of feedback, only $N_a = 2^m$ channel states are required in the conditional decision function $f_b(\mathbf{r}(k)|\hat{\mathbf{s}}_b(k))$. In the case when there is no feedback, all the $N_s = 2^{m+n}$ states would be required in the original decision function $f_b(\mathbf{r})$. As a result of feedback, implementation complexity is reduced. In addition, the reduced number of possible channel state occurrences also normally results in less nonlinear conditional Bayesian decision boundaries, and increased minimum distances of channel states of $C_d^{(+)}$ and $C_d^{(-)}$ to the decision boundaries [18]. Because of this increase in minimum distances to the boundary, the misclassification probability of the DFE is lower than those without feedback.

Example : Reduction in implementation complexity

This section discusses the reduction in implementation complexity of the DFE using the equalisation problem on channel $H(z) = 0.5 + 1.0z^{-1}$ using the following equaliser's parameters, delay $d = 1$, feedforward order $m = 2$ and feedback order $n = 1$. Referring to Table 4.1, the set of channel states $C_{d,l} = C_{d,l}^{(+)} \cup C_{d,l}^{(-)}$ for $\mathbf{s}_b(k) = \mathbf{s}_{b,l} = [+1]$ is $\{\mathbf{c}_1, \mathbf{c}_3, \mathbf{c}_5, \mathbf{c}_7\}$ and the set of channel states $C_{d,l}$ for $\mathbf{s}_b(k) = \mathbf{s}_{b,l} = [-1]$ is $\{\mathbf{c}_2, \mathbf{c}_4, \mathbf{c}_6, \mathbf{c}_8\}$. Fig. 4.12 illustrates the conditional Bayesian DFE boundary formed.

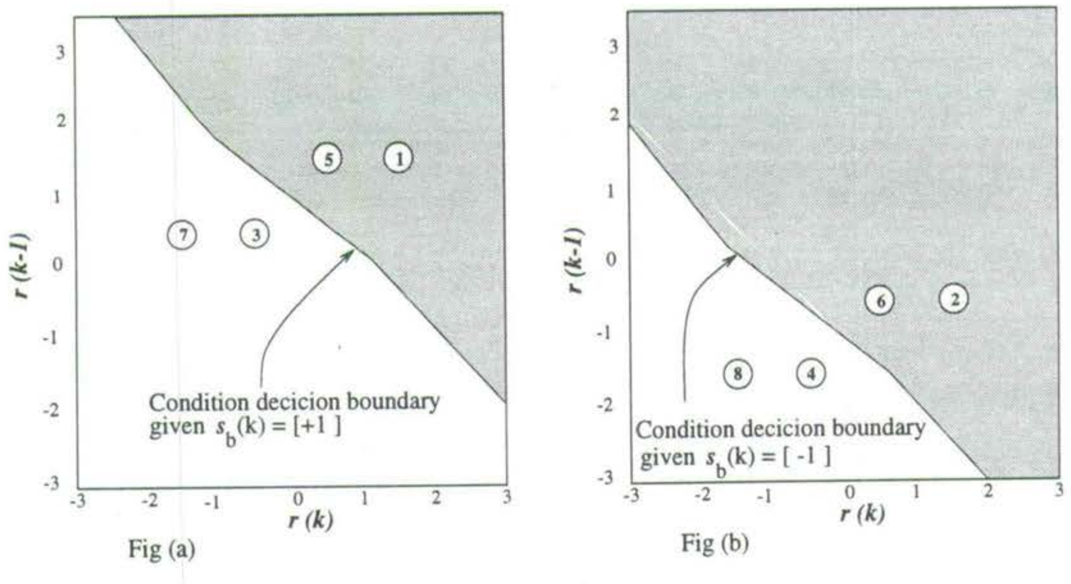

Figure 4.12: Conditional Bayesian decision boundary and set of channel states $C_{d,j}$, Fig (a) $f_b(\mathbf{r}(k)|\hat{\mathbf{s}}_b(k) = [1])$, Fig (b) $f_b(\mathbf{r}(k)|\hat{\mathbf{s}}_b(k) = [-1])$.

In this example, $2^m = 4$ channel states are required to implement each conditional Bayesian function $f_b(\mathbf{r}(k)|\hat{\mathbf{s}}_b(k) = \mathbf{s}_{b,l})$. Implementation complexity is therefore reduced as compared to the full model's decision function $f_b(\mathbf{r}(k))$ which uses $N_s = 2^{m+n} = 8$ channel states (Fig. 4.5).

4.5.3 Channel state translation

The current Bayesian DFE implementation requires $N_b = 2^n$ different conditional Bayesian decision functions $f_b(\mathbf{r}(k)|\hat{\mathbf{s}}_b(k) = \mathbf{s}_{b,l})$, i.e., one decision function for each possible feedback vector sequence $\mathbf{s}_{b,l}, 1 \leq l \leq 2^n$. This therefore implies that significant implementation complexity is required if the number of possible feedback states $N_b = 2^n$ is large. It is, however, possible to remove this requirement of having multiple conditional Bayesian decision function by introducing a transformation of the input vector $\mathbf{r}(k) \in R^m \rightarrow \mathbf{r}'(k) \in R^m$ [20].

The description of the transformation is as follows: Assuming that the feedback vector $\hat{\mathbf{s}}_b(k) = \mathbf{s}_{b,l}$ is correct, the original input vector $\mathbf{r}(k)$ is transformed into the new input vector $\mathbf{r}'(k)$ by the following translation,

$$\mathbf{r}'(k) = \mathbf{r}(k) - \mathbf{v}_l \tag{4.65}$$

where

$$\mathbf{v}_l = F[\,0\;,\;\mathbf{s}_{b,l}]. \tag{4.66}$$

The translation performed on $\mathbf{r}(k)$ removes the contribution of past detected symbols $\hat{\mathbf{s}}_b(k)$ from the input vector, i.e., the new input vector

$$\begin{aligned}\mathbf{r}'(k) &= \mathbf{r}(k) + \mathbf{n}(k) - \mathbf{v}_l \\ &= F[\mathbf{s}_a(k)\;,\;\hat{\mathbf{s}}_b(k)] + \mathbf{n}(k) - F[\,0\;,\;\hat{\mathbf{s}}_b(k)] \\ &= F[\mathbf{s}_a(k)\;,0] + \mathbf{n}(k).\end{aligned} \tag{4.67}$$

We denote the translated noiseless channel states $F[\mathbf{s}_a(k)\;,0]$ as

$$\hat{\mathbf{r}}'(k) = F[\mathbf{s}_a(k)\;,\;0\,] = [\hat{r}'(k)\;\cdots\;\hat{r}'(k-d+1)]^T \in R^{d+1}. \tag{4.68}$$

As there are only $N_a = 2^{d+1}$ combinations of $\mathbf{s}_a(k)$, there are only N_a channel states $\hat{\mathbf{r}}'(k)$ in the set

$$C_d' = \{\hat{\mathbf{r}}'(k)\} = \{F[\mathbf{s}_{a,j}\;,\;0]\}, \qquad 1 \leq j \leq N_a = 2^m. \tag{4.69}$$

Therefore after performing the transformation, only one Bayesian decision function with the set of channel states $C_d^{'}$ as centres is needed,

$$f_b(\mathbf{r}(k), \hat{\mathbf{s}}_b(k)) = \frac{1}{N_s} \sum_{\mathbf{c}_j \in C_d^{'(+)}} p(\mathbf{r}(k)|\mathbf{c}_j) - \frac{1}{N_s} \sum_{\mathbf{c}_k \in C_d^{'(-)}} p(\mathbf{r}(k)|\mathbf{c}_k), \tag{4.70}$$

where the two sets of channel states $C_d^{'(+)}$ and $C_d^{'(-)}$ denote the channel states in $C_d^{'}$ (Eq. 4.69) associated with transmitted symbol $s(k-d) = +1$ and -1 respectively. The translation therefore removes the requirement of having multiple conditional decision functions for each feedback pattern.

4.5.4 Recursive implementation to generate $\mathbf{r}^{'}(k)$

The transformation as described in the previous section has an additional advantage that it can be implemented effectively using recursive implementation. We demonstrate how this may be achieved using the following DFE equalisation problem. The channel used is a 3-tap channel $H(z) = a_0 + a_1 z^{-1} + a_2 z^{-2}$ and the equaliser's parameters are $d = 2$, $m = 3$, and $n = 2$. The original input vector $\mathbf{r}(k)$ is

$$\mathbf{r}(k) = \begin{bmatrix} r(k) \\ r(k-1) \\ r(k-2) \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 & 0 & 0 \\ 0 & a_0 & a_1 & a_2 & 0 \\ 0 & 0 & a_0 & a_1 & a_2 \end{bmatrix} \begin{bmatrix} s(k) \\ s(k-1) \\ s(k-2) \\ s(k-3) \\ s(k-4) \end{bmatrix} + \begin{bmatrix} n(k) \\ n(k-1) \\ n(k-2) \\ n(k-3) \\ n(k-4) \end{bmatrix} \tag{4.71}$$

and the translated vector $\mathbf{r}^{'}(k) = F[\mathbf{s}_a(k)\ ,\ \mathbf{s}_b(k)] - F[\,0\ ,\ \mathbf{s}_b(k)] + \mathbf{n}(k)$ is

$$\mathbf{r}^{'}(k) = \begin{bmatrix} r^{'}(k) \\ r^{'}(k-1) \\ r^{'}(k-2) \end{bmatrix} = \begin{bmatrix} r(k) \\ r(k-1) - a_2 s(k-3) \\ r(k-2) - a_1 s(k-3) - a_2 s(k-4) \end{bmatrix}. \tag{4.72}$$

Assuming that the past detected symbols are detected correctly, i.e. $\hat{\mathbf{s}}_b(k) = \mathbf{s}_b(k)$, the transformation can be implemented recursively as illustrated in Fig. 4.13.

In the general case, it can be shown that the translated elements of vector $\mathbf{r}^{'}(k)$ can be found recursively using the following expression:

$$\left.\begin{aligned} r^{'}(k-i) &= z^{-1} r^{'}(k-i+1) - a_{n_a - i} s(k-d-1), \qquad i = m-1, \cdots, 1. \\ r^{'}(k) &= r(k). \end{aligned}\right\} \tag{4.73}$$

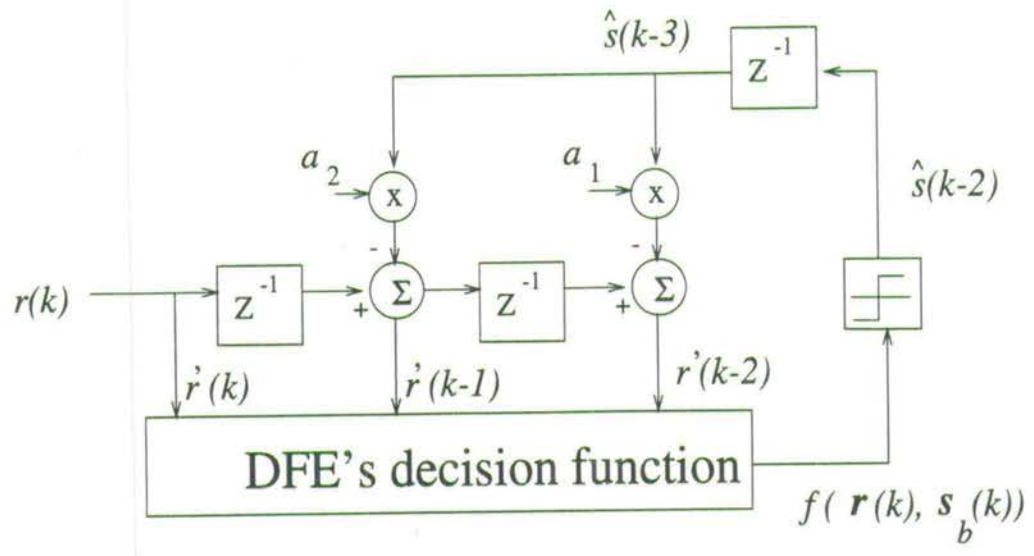

Figure 4.13: Recursive implementation of DFE for $\mathbf{r}'(k)$ with channel $H(z) = a_0 + a_1 z^{-1} + a_2 z^{-2}$.

Example

Let us consider the problem of applying the transformation to the DFE on channel $H(z) = 0.5 + 1.0z^{-1}$ using $d = 1$, $m = 2$ and $n = 1$. The matrix F is

$$F = \begin{bmatrix} 0.5 & 1.0 & 0 \\ 0 & 0.5 & 1.0 \end{bmatrix}, \tag{4.74}$$

and the set of $\mathbf{s}_a(k)$ is $\{\mathbf{s}_{a,j}\}, 1 \le j \le 2^m$. The values of these vectors are listed below:

$$\begin{aligned} \mathbf{s}_{a,1} &= [1,1]^T, & \mathbf{s}_{a,2} &= [1,-1]^T, \\ \mathbf{s}_{a,3} &= [-1,1]^T, & \mathbf{s}_{a,4} &= [-1,-1]^T. \end{aligned}$$

The channel states of $C_d' = \{\mathbf{c}_1', \mathbf{c}_2', \mathbf{c}_3', \mathbf{c}_4'\}$ are

$$\begin{aligned} \mathbf{c}_1' &= F[\mathbf{s}_{a,1}, 0]^T = [1.5, 0.5]^T, & \mathbf{c}_2' &= F[\mathbf{s}_{a,2}, 0]^T = [-0.5, -0.5]^T, \\ \mathbf{c}_3' &= F[\mathbf{s}_{a,3}, 0]^T = [0.5, 0.5]^T, & \mathbf{c}_4' &= F[\mathbf{s}_{a,4}, 0]^T = [-1.5, -0.5]^T. \end{aligned}$$

The translation vectors $\mathbf{v}_l = F[0, \mathbf{s}_{b,l}]$ are

$$\mathbf{v}_1 = F[0\ 0\ -1]^T = [0\ -1]^T, \tag{4.75}$$

$$\mathbf{v}_2 = F[0\ 0\ +1]^T = [0\ +1]^T. \tag{4.76}$$

The translation vectors $\mathbf{v}_1, \mathbf{v}_2$, the translated set of channel states C_d', and the translated Bayesian decision boundary are illustrated in Fig. 4.14.

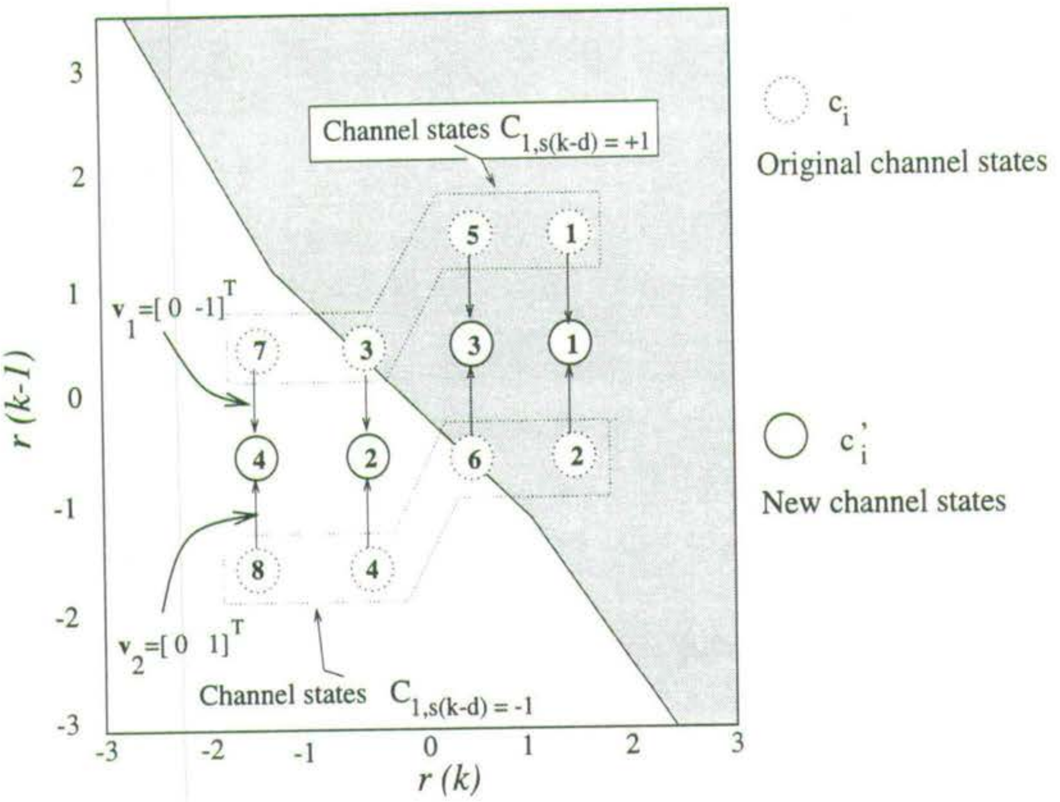

Figure 4.14: Translated Bayesian decision boundary and translated channel states using delay $d = 1$ and $m = 2$ for channel $H(z) = 0.5 + 1.0z^{-1}$.

4.5.5 Bayesian DFE : some simulation results

Computer simulations were used to compare the performance of the Bayesian feedforward equaliser with that of the Bayesian DFE. The same two channels $H1(z)$ and $H2(z)$ as discussed in Sec. 4.3.2 were used. The feedforward and delay parameters of both the feedforward and DFE were set to $m = 4$ and $d = 2$. The feedback parameter was set to $n = 2$ for the DFE. The results of both the experiments are illustrated in Figs. 4.15 and 4.16. Both results show that the degradation in performance due to error propagation resulted in only a very small loss of performance and that the application of decision feedback improves the performance of the equaliser.

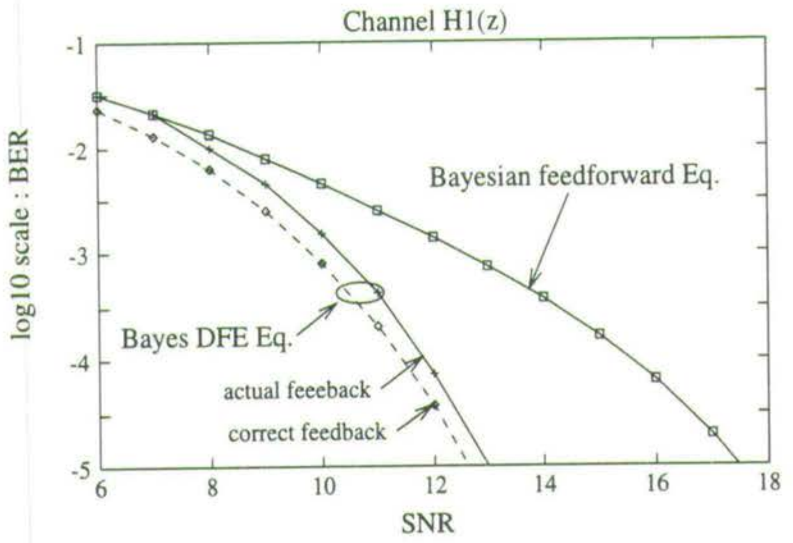

Figure 4.15: Comparing the performance of the Bayesian feedforward equaliser and Bayesian DFE for channel H1(z).

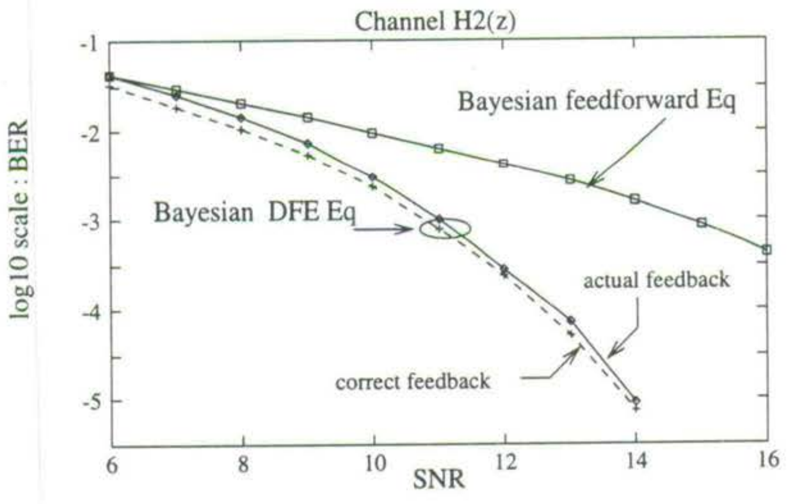

Figure 4.16: Comparing the performance of the Bayesian feedforward equaliser and Bayesian DFE for channel H2(z).

4.6 Conclusions

This chapter studies the symbol-decision equaliser. In particular, we show that the optimum symbol-decision equaliser is nonlinear and that it is possible to realise the optimum Bayesian decision function using the RBF model. The effects of the delay order parameter on decision boundaries and BER performance were highlighted and our results show that the attainable BER performance depends strongly on the delay order parameter and can be significantly different for various values of the delay order. To determine the optimum operating delay parameter, a simple BER estimator for the Bayesian equaliser was proposed.

The implementation complexity of the RBF equaliser required to realise the Bayesian solution was also discussed. To reduce the implementation complexity, an algorithm was proposed to select a subset model from the full RBF equaliser implementation. Our results showed that the number of centres, and hence the computation complexity, can be reduced with no significant effects on the BER performance of the equaliser.

Lastly, the Bayesian DFE was examined. We introduced a geometric transformation on the input vector to reduce the DFE to an equivalent equaliser without feedback. The geometric translation not only provided deeper understanding of the DFE mechanism but also facilitated more efficient practical implementation. The results also indicated that significant improvement of classification performance may be achieved by introducing feedback.

Chapter 5

Digital communication channel equalisation using linear techniques

The previous chapter considered the optimum symbol-decision equaliser, the Bayesian equaliser. It was shown that the Bayesian equaliser's decision function is nonlinear, and hence can only be realised using nonlinear techniques. Although nonlinear equalisers provide better performance than linear equalisers, they are not as widely used due to their significantly more demanding implementation complexity. Linear equalisation techniques, which have much simpler implementation requirement, are very popular despite having poorer performance.

This chapter is concerned with the application of linear techniques to channel equalisation. We first consider the most common type of linear equaliser, the minimum MSE (MMSE) linear equaliser. Although the MMSE linear equaliser is very popular, the classification performance of such an equaliser is sub-optimum in two ways; firstly, the approximation of nonlinear decision boundaries using linear methods will yield poor results, and secondly, the MMSE cost function is not the optimum criterion to be applied for finding the weights of the linear equaliser. In this chapter, we will demonstrate these two short-comings and examine an optimimsation criterion based on minimising the BER to improve the linear equaliser's performance. Also in this chapter we will consider the application of this minimum BER (MBER) criterion to a linear-combiner DFE.

The following results are reported in this chapter:

(i). Optimisation techniques to minimise the BER of the linear feedforward equaliser are introduced [1].

[1] Submitted to - S.CHEN, E.S.CHNG, B.MULGREW and G.GIBSON, 'On decision feedback equaliser', *IEEE Trans. Communication*

(ii). Optimising the feedforward weights of the DFE using the MBER criterion [2].

The outline of the chapter is as follows: The MMSE linear equaliser is introduced in Sec 5.1. Sec 5.2 considers the difference in functional mapping capability between the linear and nonlinear equaliser and examines the effects of delay order on the decision boundaries. Sec 5.3 studies the probability of misclassification of the linear equaliser and introduces a method to evaluate the BER. Sec 5.4 proposes a method for optimising the equaliser's performance for MBER. Sec 5.5 examines the transformation introduced in Chapter 4 to convert the linear-combiner DFE to a feedforward equaliser structure and considers the optimisation of the DFE's performance based on the MBER criterion. Sec 5.6 concludes the chapter.

5.1 The MMSE Linear equaliser

This section introduces the MMSE linear equaliser. The aim of channel equalisation using linear techniques is the same as channel equalisation using nonlinear techniques, that is, to re-construct the transmitted symbol given a vector of received signal which has been corrupted by additive noise and ISI effects [1, 3, 71, 72]. The difference between the linear and nonlinear techniques is in the implementation of the equaliser's decision function, i.e., no nonlinearity is present in the linear equaliser's decision function.

As in Chapter 4, the received signal $r(k)$ for the linear equaliser at time k is

$$r(k) = \hat{r}(k) + n(k) \tag{5.1}$$

where $\hat{r}(k)$ is the noise-free received signal, and $n(k)$ is the additive white Gaussian noise with zero mean and variance σ_e^2 [3, 17]. The noise-free received signal $\hat{r}(k)$ is

$$\hat{r}(k) = \sum_{i=0}^{n_a-1} s(k-i)a(i) \tag{5.2}$$

where $a(i)$ are the channel impulse response, n_a the channel impulse length, and $s(k)$ the i.i.d transmitted signal from a 2-ary PAM source with symbol constellation $\{\pm 1\}$. The equaliser uses a vector $\mathbf{r}(k)$ of consecutive received samples, i.e.,

$$\begin{aligned}\mathbf{r}(k) &= \hat{\mathbf{r}}(k) + \mathbf{n}(k) \\ &= [\hat{r}(k) \ \cdots \ \hat{r}(k-m+1)]^T + [n(k) \ \cdots \ n(k-m+1)]^T \\ &= [r(k) \ \cdots \ r(k-m+1)]^T \end{aligned} \tag{5.3}$$

[2]Submitted to - S.CHEN, E.S.CHNG, B.MULGREW and G.GIBSON, 'On decision feedback equaliser', *IEEE Trans. Communication*

to form an estimate $\hat{s}(k-d)$ of $s(k-d)$. The vector $\hat{\mathbf{r}}(k)$ containing the noiseless received signal is known as a channel state and the vector $\mathbf{n}(k)$ is zero-mean Gaussian i.i.d noise. The integers m and d are known as the equaliser's feedforward order and delay order respectively.

The linear equaliser's decision function $f_l(\mathbf{r}(k))$ is [1,3,71,72]

$$f_l(\mathbf{r}(k)) = \mathbf{w}^T\mathbf{r}(k) = \sum_{i=1}^{m} w_i r(k-i+1) \tag{5.4}$$

where $\mathbf{w} = [w_1 \cdots w_m]^T$ are the weights of the linear equaliser. To determine the transmitted symbol $s(k-d)$, a decision slicer is applied to quantise the decision function output value $f_l(\mathbf{r}(k))$ into one of the possible symbol constellation. In the case of the 2-ary PAM source, the decision slicer can be implemented using the $sgn(.)$ function.

The most common criterion to optimise the linear equaliser is the MMSE criterion [1,3,71,72]. This criterion is popular because the equaliser thus optimised can usually perform well and the implementation requirement is simple [1,3,71,72]. The MMSE criterion is

$$J_{MMSE} = E[(s(k-d) - f_l(\mathbf{r}(k)))^2] \tag{5.5}$$

and the MMSE solution of $\mathbf{w}$, also known as the Wiener solution, is

$$\mathbf{w} = \{E[\mathbf{r}(k)\mathbf{r}(k)^T]\}^{-1}E[s(k-d)\mathbf{r}(k)] \tag{5.6}$$

where $\{E[\mathbf{r}(k)\mathbf{r}(k)^T]\}$ is the autocorrelation matrix and $E[s(k-d)\mathbf{r}(k)]$ is the crosscorrelation vector between the input of the equaliser and the desired response $s(k-d)$. If the statistics of the channel are known, the optimum Weiner solution can be solved using Eq. 5.6 [1,71,87].

Although the MMSE is very popular, it is not the best criterion for optimising the weights of the linear equaliser. A better choice would be the MBER criterion. The MBER criterion however is not popular as it was thought to be difficult to implement [71]. Our experience, however, disagrees with this observation. We have found that MBER optimisation of the linear equaliser can be carried out quite easily for cases when the channel is stationary using un-constrained gradient descent optimisation techniques [113]. The details of the MBER optimisation procedure for the linear equaliser is reported in Sec. 5.4 .

5.2 Approximating the Bayesian decision boundary using linear boundary

This section examines the application of the linear boundary to realise the Bayesian boundary. As in the case of the Bayesian equaliser (Sec. 5.1), the linear equaliser's decision boundary is similarly affected by the channel state locations and delay order. A linear equaliser, however, may only realise linear boundaries, and hence cannot be applied to equalisation problems which have channel states belonging to different transmitted signal that are not linearly separable. The sets of channel states belonging to different transmit symbols, i.e. $C_d^{(+)}$ and $C_d^{(-)}$, as defined in Sec. 4.2.1, are given by

$$C_d^{(+)} = \{\hat{\mathbf{r}}(k) | s(k-d) = +1\}, \quad (5.7)$$
$$C_d^{(-)} = \{\hat{\mathbf{r}}(k) | s(k-d) = -1\}, \quad (5.8)$$

and the set of all channel states is

$$C_d = C_d^{(+)} \cup C_d^{(-)}. \quad (5.9)$$

In the case when $C_d^{(+)}$ and $C_d^{(-)}$ are not linearly separable, linear equalisation techniques will result in misclassifications even when noise is not affecting the communication system. To illustrate some of the problems that arise when applying linear equalisation techniques, we compare the realised MMSE linear equaliser's decision boundaries with the Bayesian equaliser's decision boundaries for the channel $H(z) = 0.5 + 1.0z^{-1}$.

Examples of decision boundaries

The following paragraphs describe the realised decision boundaries for the channel $H(z) = 0.5 + 1.0z^{-1}$ using a linear MMSE equaliser and a Bayesian equaliser with feedforward order $m = 2$ and delay order $d = 0, 1$ and 2, operating under SNR condition = 15dB.

The results depicted in Figs. 5.1a,b and 5.2a,b illustrate the realised linear and Bayesian decision functions mapping and decision boundaries for delay order $d = 1$ and 2 respectively. The plots clearly highlight the nonlinear nature of the Bayesian function which allows hyper-surfaces to be formed in the $\mathbf{r}(k) \in R^2$ space to generate nonlinear boundaries. This is in contrast to the linear decision functions which can only realise hyper-planes and hence only generate linear boundaries. It is also observed that the nonlinear nature of the Bayesian function allows the Bayesian boundary to be 'bent' for optimum partitioning of channel states belonging to different transmit symbols. This is unlike the linear boundary which could not be 'bent' and hence has poorer partitioning capability. This implies that the channel states are nearer to the linear boundary than the nonlinear optimum boundary. This 'nearness' to the linear boundary has the undesirable effect that when noise is present in the communication system, a higher probability of misclassification will occur. A formal treatment of the linear equaliser's performance with respect to the Bayesian equaliser is given in Sec. 5.3.

In both examples, using delay $d = 1$ and 2, the channel states of $C_d^{(+)}$ and $C_d^{(-)}$ were linearly separable, and hence linear equalisation techniques could be applied. However, in the case

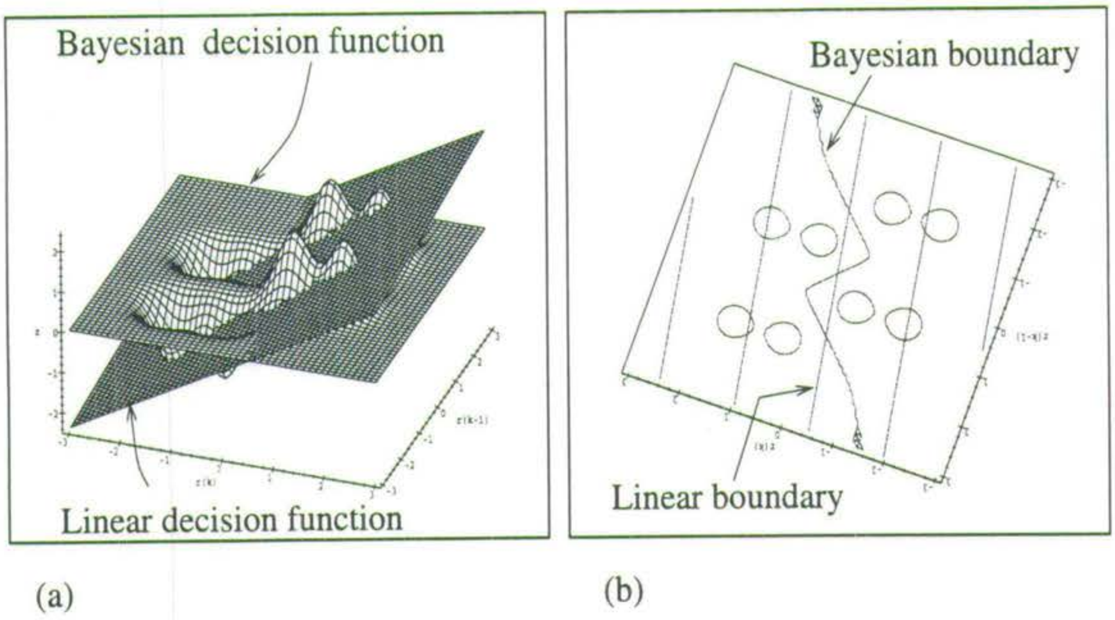

Figure 5.1: Effects of delay order on the decision function of the Bayesian and linear equaliser for channel $H(z) = 0.5 + 1.0z^{-1}$, delay order $d = 1$, $m = 2$ and SNR = 15dB: (a) Mapping of the Bayesian decision function (vertical axis) for $f_b(\mathbf{r}(\mathbf{k}))$ and MMSE linear decision function $f_l(\mathbf{r}(k))$, (b) The corresponding Bayesian and linear decision boundaries.

when the equalisation problem is not linearly separable, applying linear equalisation techniques will result in misclassification for certain channel states even in the absence of additive noise. An example of an equalisation problem which is not linearly separable is illustrated in Fig. 5.3 where the equaliser's delay order is chosen to be $d = 0$ for channel $H(z)$. From the diagram, it is observed that the Bayesian decision boundary was able to partition the channel states belonging to different transmit symbols successfully, while the linear partitioning resulted in channel states $\{\mathbf{c}_3, \mathbf{c}_6\}$ being misclassified even in the absence of noise.

From the above examples, the results clearly indicate two major short-comings of linear equalisation techniques; firstly, the poorer partitioning capability of the linear equaliser as compared to the Bayesian equaliser, and secondly the in-ability of the linear equaliser to operate on non linearly separable problems. These are the primary reasons why nonlinear equalisers which do not possess these disadvantages have been so actively pursued [17, 97, 99, 101, 102].

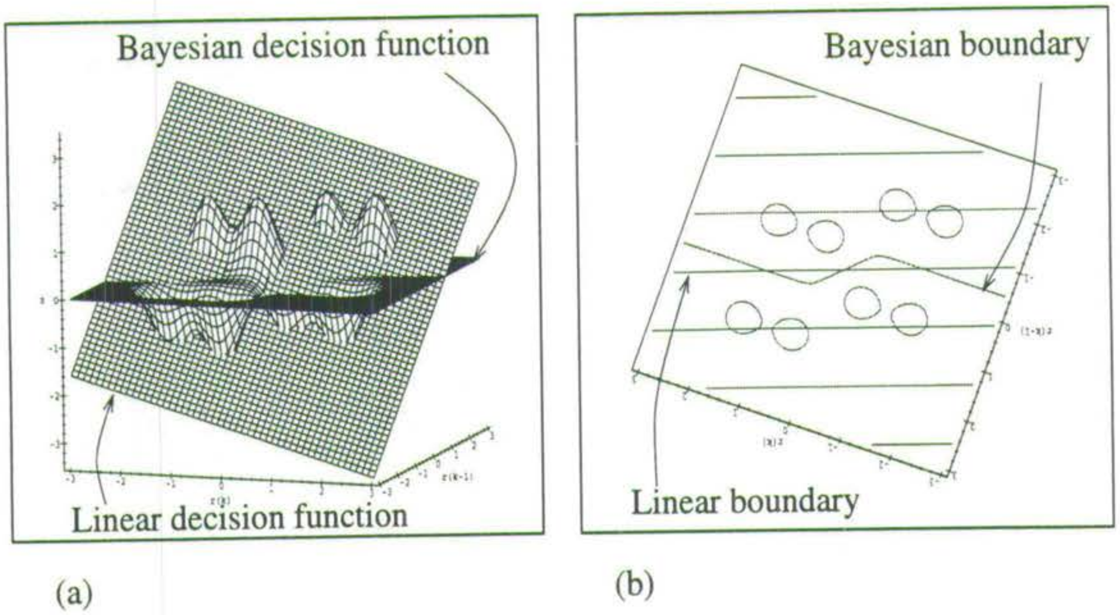

Figure 5.2: Effects of delay order on the decision function of the Bayesian and linear equaliser for channel $H(z) = 0.5 + 1.0z^{-1}$, delay order $d = 2$, $m = 2$ and SNR = 15dB: (a) Mapping of the Bayesian decision function (vertical axis) $f_b(\mathbf{r(k)})$ and MMSE linear decision function $f_l(\mathbf{r}(k))$, (b) The corresponding Bayesian and linear decision boundary.

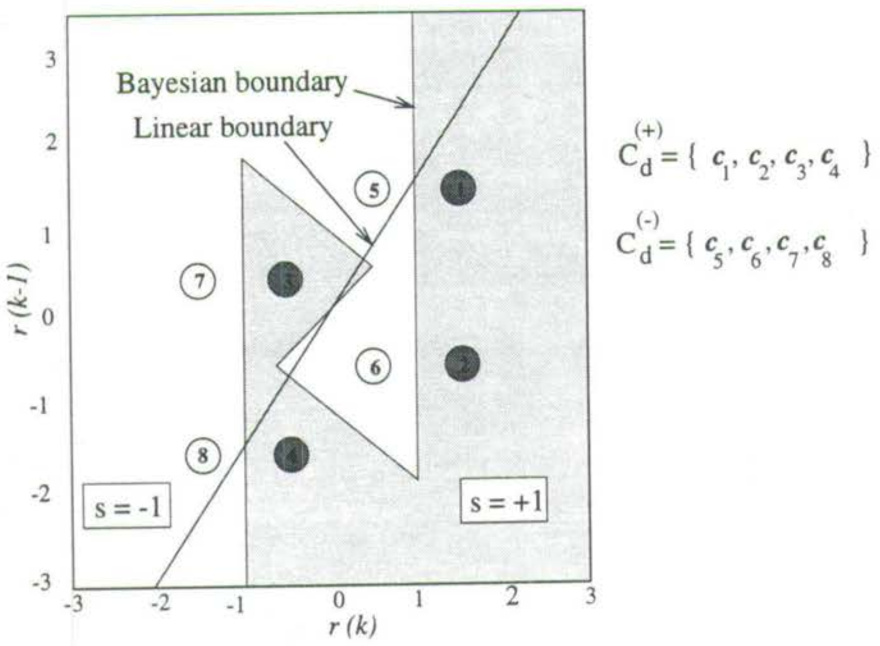

Figure 5.3: Effects of delay order on the decision function of the Bayesian and linear equaliser for channel $H(z) = 0.5 + 1.0z^{-1}$, delay order $d = 0$, $m = 2$ and SNR = 15dB: The Bayesian and linear decision boundaries.

5.3 Probability of misclassification

This section presents the analysis of probability of misclassification by the linear equaliser.

Following the analysis presented in Sec. 4.3 for the Bayesian equaliser's case, the received vector $\mathbf{r}(k)$ to the equaliser is a noise-perturbed version of the channel state $\hat{\mathbf{r}}(k)$. Due to noise, the received sequence $\mathbf{r} = \mathbf{r}(k)$ may be perturbed into regions defined for a different classification. Given $\mathbf{c}_j \in C_d^{(+)}$, the misclassification region for a linear boundary is $Z_l^{(-)}$, and given $\mathbf{c}_j \in C_d^{(-)}$, the misclassification region is $Z_l^{(+)}$. The regions $Z_l^{(+)}$ and $Z_l^{(-)}$ are defined by the following equations and illustrated in Fig. 5.4a,

$$Z_l^{(+)} = \{\mathbf{r} \mid f_l(\mathbf{r}) \geq 0\}, \tag{5.10}$$
$$Z_l^{(-)} = \{\mathbf{r} \mid f_l(\mathbf{r}) < 0\}. \tag{5.11}$$

Following the same analysis as in Sec. 4.3, the total probability of misclassification P_l by the linear equaliser is therefore,

$$P_l = \sum_{\mathbf{c}_j \in C_d^{(+)}} \frac{1}{N_s} \int_{\mathbf{r} \in Z_l^{(-)}} p(\mathbf{r}|\mathbf{c}_j) d\mathbf{r} + \sum_{\mathbf{c}_k \in C_d^{(-)}} \frac{1}{N_s} \int_{\mathbf{r} \in Z_l^{(+)}} p(\mathbf{r}|\mathbf{c}_k) d\mathbf{r}. \tag{5.12}$$

To highlight the difference in performance between the linear and the Bayesian equaliser, we introduce the following equation to express the linear equaliser's misclassification probability with respect to the Bayesian equaliser's misclassification probability [3] P_b,

$$P_l = P_b + P_W \tag{5.13}$$

where P_W is the difference in probability of misclassification between the linear and Bayesian equaliser. The difference in probability of misclassification exists because the regions of integration for P_l and P_b are different. We denote these difference regions using the notations $W_d^{(-)}$ and $W_d^{(+)}$, and they are defined by the following equations and illustrated in Fig. 5.4b,

$$W_d = W_d^{(-)} \cup W_d^{(+)} \tag{5.14}$$
$$W_d^{(-)} = \{\mathbf{r} \mid f_b(\mathbf{r}) < 0 \text{ and } f_l(\mathbf{r}) > 0\}$$
$$W_d^{(+)} = \{\mathbf{r} \mid f_b(\mathbf{r}) > 0 \text{ and } f_l(\mathbf{r}) < 0\} \tag{5.15}$$

[3] The Bayesian equaliser's probability of misclassification P_b (Eq. 4.28) is,

$$P_b = \sum_{\mathbf{c}_j \in C_d^{(+)}} \frac{1}{N_s} \int_{\mathbf{r} \in Z_b^{(-)}} p(\mathbf{r}|\mathbf{c}_j) d\mathbf{r} + \sum_{\mathbf{c}_k \in C_d^{(-)}} \frac{1}{N_s} \int_{\mathbf{r} \in Z_b^{(+)}} p(\mathbf{r}|\mathbf{c}_k) d\mathbf{r}.$$

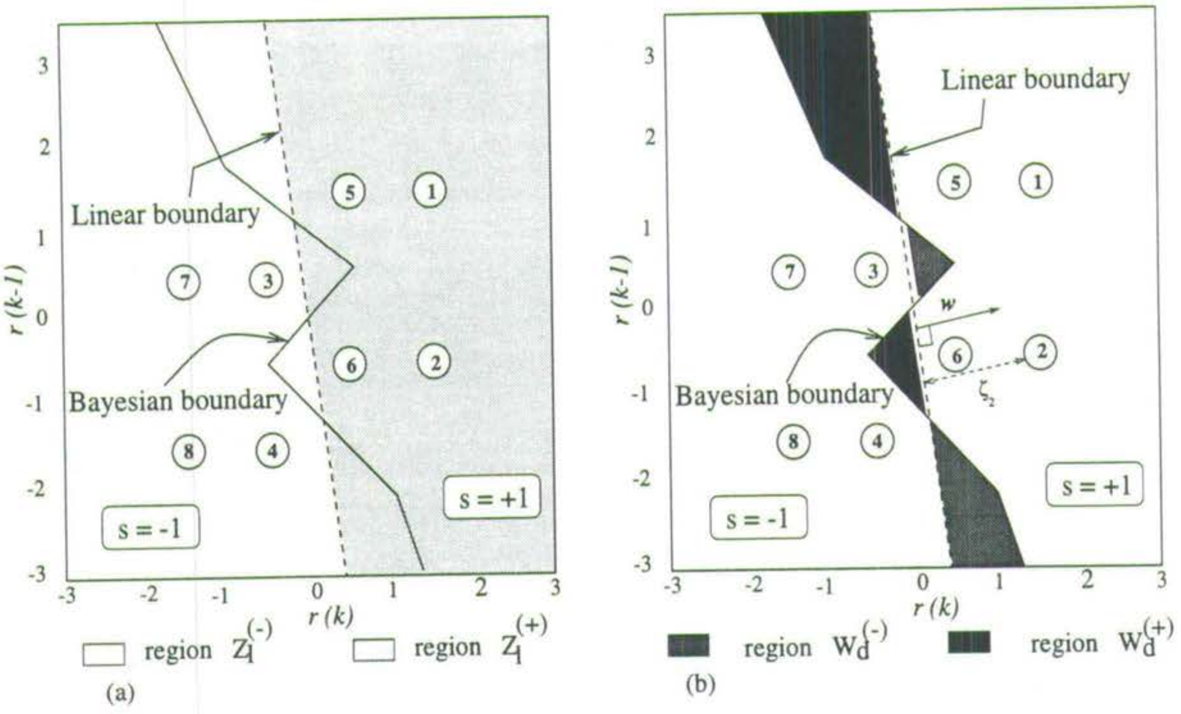

Figure 5.4: Bayesian and linear classification regions. Channel $H(z) = 0.5 + 1.0z^{-1}$, $m = 2$, $d = 1$: (a) RBF and linear boundaries . (b) Misclassification region due to linear approximation of Bayesian solution

To derive P_W, let us first consider the case of evaluating P_W for channel states $\mathbf{c}_j \in C_d^{(+)}$,

$$P_W(\mathbf{c}_j \in C_d^{(+)}) = \sum_{\mathbf{c}_j \in C_d^{(+)}} \frac{1}{N_s} \int_{\mathbf{r} \in W_d^{(+)}} p(\mathbf{r}|\mathbf{c}_j) d\mathbf{r} - \sum_{\mathbf{c}_j \in C_d^{(+)}} \frac{1}{N_s} \int_{\mathbf{r} \in W_d^{(-)}} p(\mathbf{r}|\mathbf{c}_j) d\mathbf{r} \qquad (5.16)$$

where the first term on the right hand side of the equation is to perform integration in the additional area included by the linear decision boundary and the second term is to remove the integration for the region not defined in the linear boundary but was included in the region specified for the Bayesian boundary.

Similarly, the following equation specifies the difference in probability of misclassification in the region W_d for $\mathbf{c}_k \in C_d^{(-)}$,

$$P_W(\mathbf{c}_k \in C_d^{(-)}) = \sum_{\mathbf{c}_k \in C_d^{(-)}} \frac{1}{N_s} \int_{\mathbf{r} \in W_d^{(-)}} p(\mathbf{r}|\mathbf{c}_k) d\mathbf{r} - \sum_{\mathbf{c}_k \in C_d^{(-)}} \frac{1}{N_s} \int_{\mathbf{r} \in W_d^{(+)}} p(\mathbf{r}|\mathbf{c}_k) d\mathbf{r}. \qquad (5.17)$$

Therefore, the total P_W is,

$$\begin{aligned} P_W &= P_W(\mathbf{c}_j \in C_d^{(+)}) + P_W(\mathbf{c}_j \in C_d^{(-)}) \\ &= \sum_{\mathbf{c}_j \in C_d^{(+)}} \frac{1}{N_s} \int_{\mathbf{r} \in W_d^{(+)}} p(\mathbf{r}|\mathbf{c}_j) d\mathbf{r} - \sum_{\mathbf{c}_j \in C_d^{(+)}} \frac{1}{N_s} \int_{\mathbf{r} \in W_d^{(-)}} p(\mathbf{r}|\mathbf{c}_j) d\mathbf{r} + \\ & \quad \sum_{\mathbf{c}_k \in C_d^{(-)}} \frac{1}{N_s} \int_{\mathbf{r} \in W_d^{(-)}} p(\mathbf{r}|\mathbf{c}_k) d\mathbf{r} - \sum_{\mathbf{c}_k \in C_d^{(-)}} \frac{1}{N_s} \int_{\mathbf{r} \in W_d^{(+)}} p(\mathbf{r}|\mathbf{c}_k) d\mathbf{r}. \end{aligned} \tag{5.18}$$

Substituting the Bayesian decision function $f_b(\mathbf{r})$ [4] into Eq. 5.18, we get

$$P_W = \int_{\mathbf{r} \in W_d^{(+)}} f_b(\mathbf{r}) d\mathbf{r} - \int_{\mathbf{r} \in W_d^{(-)}} f_b(\mathbf{r}) d\mathbf{r}. \tag{5.19}$$

By the definition of $W_d^{(+)}$ and $W_d^{(-)}$ (Eq. 5.15), the Bayesian function $f_b(\mathbf{r})$ is positive over the region $W_d^{(+)}$ and negative over the region $W_d^{(-)}$. Therefore, Eq. 5.19 can be written as

$$P_W = \int_{\mathbf{r} \in W_d^{(+)}} |f_b(\mathbf{r})| d\mathbf{r} + \int_{\mathbf{r} \in W_d^{(-)}} |f_b(\mathbf{r})| d\mathbf{r}. \tag{5.20}$$

Since the function $|f_b(\mathbf{r})|$ is non-negative, the above equation shows that if the region $W_d^{(-)}$ or $W_d^{(+)}$ is not empty, the linear equaliser will have a degradation in performance P_W with respect to the Bayesian solution.

5.3.1 Evaluating the probability of misclassification

The previous section considered the evaluation of P_l in terms of m-dimensional surface integrals. As numerical integration is undesirable, a much simpler method of evaluating P_l is introduced in the section. We first re-write Eq. 5.12 as

$$P_l = \frac{1}{N_s} \sum_{\mathbf{c}_j \in C_d^{(+)}} P_{lc}(\mathbf{c}_j) + \frac{1}{N_s} \sum_{\mathbf{c}_k \in C_d^{(-)}} P_{lc}(\mathbf{c}_k). \tag{5.21}$$

The problem therefore lies in evaluating $P_{lc}(\mathbf{c})$. To simplify we shall assign $P_{lc}(\mathbf{c}) = 1$ if the channel state $\mathbf{c}$ is wrongly classified. If the channel state is correctly classified, the following expression should be evaluated,

[4] The Bayesian decision function (Eq. 4.19) $f_b(\mathbf{r}(k))$ is,

$$f_b(\mathbf{r}) = \frac{1}{N_s} \sum_{\mathbf{c}_j \in C_d^{(+)}} p(\mathbf{r}|\mathbf{c}_j) - \frac{1}{N_s} \sum_{\mathbf{c}_k \in C_d^{(-)}} p(\mathbf{r}|\mathbf{c}_k).$$

$$P_{lc}(\mathbf{c}) = \begin{cases} \int_{\mathbf{r} \in Z_l^{(-)}} p(\mathbf{r}|\mathbf{c}) d\mathbf{r}, & \text{for } \mathbf{c} \in C_d^{(+)} \\ \int_{\mathbf{r} \in Z_l^{(+)}} p(\mathbf{r}|\mathbf{c}) d\mathbf{r}, & \text{for } \mathbf{c} \in C_d^{(-)} \end{cases} . \tag{5.22}$$

These m-dimensional surface integral equations can be simplified in the linear boundary case by introducing an orthonormal transformation $T(.)$ to rotate $\mathbf{r}$ so that one of the transformed basis vector x_1 becomes parallel to the normal of the linear boundary. The following expression shows the transformation of $T(.)$ on Eq. 5.22,

$$P_{lc}(\mathbf{c}) = \begin{cases} \int_{\mathbf{x} = T(\mathbf{r} \in Z_l^{(-)})} p(\mathbf{x}|\mathbf{c}) |\det T'| d\mathbf{x}, & \mathbf{c} \in C_d^{(+)} \\ \int_{\mathbf{x} = T(\mathbf{r} \in Z_l^{(+)})} p(\mathbf{x}|\mathbf{c}) |\det T'| d\mathbf{x}, & \mathbf{c} \in C_d^{(+)} \end{cases} . \tag{5.23}$$

where $\det T'$ is the Jacobian determinant of T [114]. Since a rotation is an orthonormal transformation, and the function $p(\mathbf{r}|\mathbf{c}_j)$ is symmetrical, the equation reduces to

$$P_{lc}(\mathbf{c}_j) = \int_{\zeta_j}^{\infty} p(x_1) dx_1 \int_{-\infty}^{\infty} p(x_2) dx_2 \cdots \int_{-\infty}^{\infty} p(x_m) dx_m, \tag{5.24}$$

where ζ_j is the Euclidean distance of $\mathbf{c}_j$ to the linear boundary along the normal [5] of the linear boundary $\mathbf{w}$ and $p(x_j)$ is a one dimension Gaussian variable,

$$p(x_j) = (2\pi\sigma_e^2)^{-1/2} \exp(-x^2/(2\sigma_e^2)\ ). \tag{5.25}$$

As the integration of $\int_{-\infty}^{\infty} p(x_m) dx_m$ has the value unity, Eq. 5.24 reduces to,

$$P_{lc}(\mathbf{c}_j) = \int_{\zeta_j}^{\infty} p(x_1) dx_1 = \int_{\zeta_j}^{\infty} (2\pi\sigma_e^2)^{-1/2} \exp(-x_1^2/(2\sigma_e^2)\ ) dx_1. \tag{5.26}$$

The distance of $\mathbf{c}_j$ to the linear boundary along $\mathbf{w}$ can be easily found using orthogonal projection [115], i.e.,

$$\zeta_j = \frac{|(\mathbf{c}_j - \mathbf{p})^T \mathbf{w}|}{||\mathbf{w}||}, \tag{5.27}$$

where $\mathbf{p}$ is any point on the decision boundary, and $(\mathbf{c}_j - \mathbf{p})$ is a vector from $\mathbf{p}$ to $\mathbf{c}_j$. Fig. 5.4b illustrates the distance of channel state $\mathbf{c}_2$ to the linear decision boundary along the normal $\mathbf{w}$.

[5] The normal of the linear boundary is the weight $\mathbf{w}$ of the linear equaliser.

Applying the $Q(.)$ function [71],

$$Q(\zeta) \overset{\text{def}}{=} \frac{1}{\sqrt{2\pi}} \int_{\zeta}^{\infty} \exp(-x^2/2)dx \tag{5.28}$$

for Eq. 5.26, the probability of misclassification using the linear equaliser (Eq. 5.21) becomes

$$P_l = \frac{1}{N_s} \sum_{\mathbf{c}_j \in C_d^{(+)}} P_{lc}(\mathbf{c}_j) + \frac{1}{N_s} \sum_{\mathbf{c}_k \in C_d^{(-)}} P_{lc}(\mathbf{c}_k), \tag{5.29}$$

where

$$P_{lc}(\mathbf{c}_i) = \begin{cases} P_{lc}(\mathbf{c}_i) = Q(|\zeta_i|/\sigma_e) \text{ if channel state } \mathbf{c}_i \text{ is correctly classified} \\ P_{lc}(\mathbf{c}_i) = 1 \text{ if channel state } \mathbf{c}_i \text{ is wrongly classified} \end{cases} \tag{5.30}$$

5.3.2 BER estimate : some simulation results

Simulations were conducted to compare the theoretical BER using Eqs. 5.29, 5.30 and BER results from Monte Carlo (MC) simulations. For the experiment, the linear MMSE equaliser's feedforward order m was chosen to be 4. The following four channels were used, these channel exhibiting the same magnitude but different phase responses,

$$\begin{aligned} H1(z) &= 0.8745 + 0.4372z^{-1} - 0.2098z^{-2}, & (5.31)\\ H2(z) &= 0.2620 - 0.6647z^{-1} - 0.6995z^{-2}, & (5.32)\\ H3(z) &= 0.6996 + 0.6646z^{-1} - 0.2623z^{-2}, & (5.33)\\ H4(z) &= 0.2098 - 0.4370z^{-1} - 0.8750z^{-2}. & (5.34) \end{aligned}$$

Simulations were conducted using a range of SNR with delay order $d = 0, \ldots, 5$ to study the effects of delay order on BER performance. The simulation results are illustrated in Fig. 5.5. The experiment shows that i) the theoretical BER are very close to the Monte Carlo simulation results, and ii) the delay order parameter can seriously affect the performance of the linear equaliser. The results also show that the optimum delay order which result in the best BER performance is different for each of the channel models even though these channels exhibit the same magnitude response.

For channel equalisation under non-stationary environment, the optimum operating delay order should be re-calculated and applied. By allowing the delay order to change dynamically, it is obvious that significant improvement to the equaliser's performance can be achieved over an equaliser which operates with a fixed delay order.

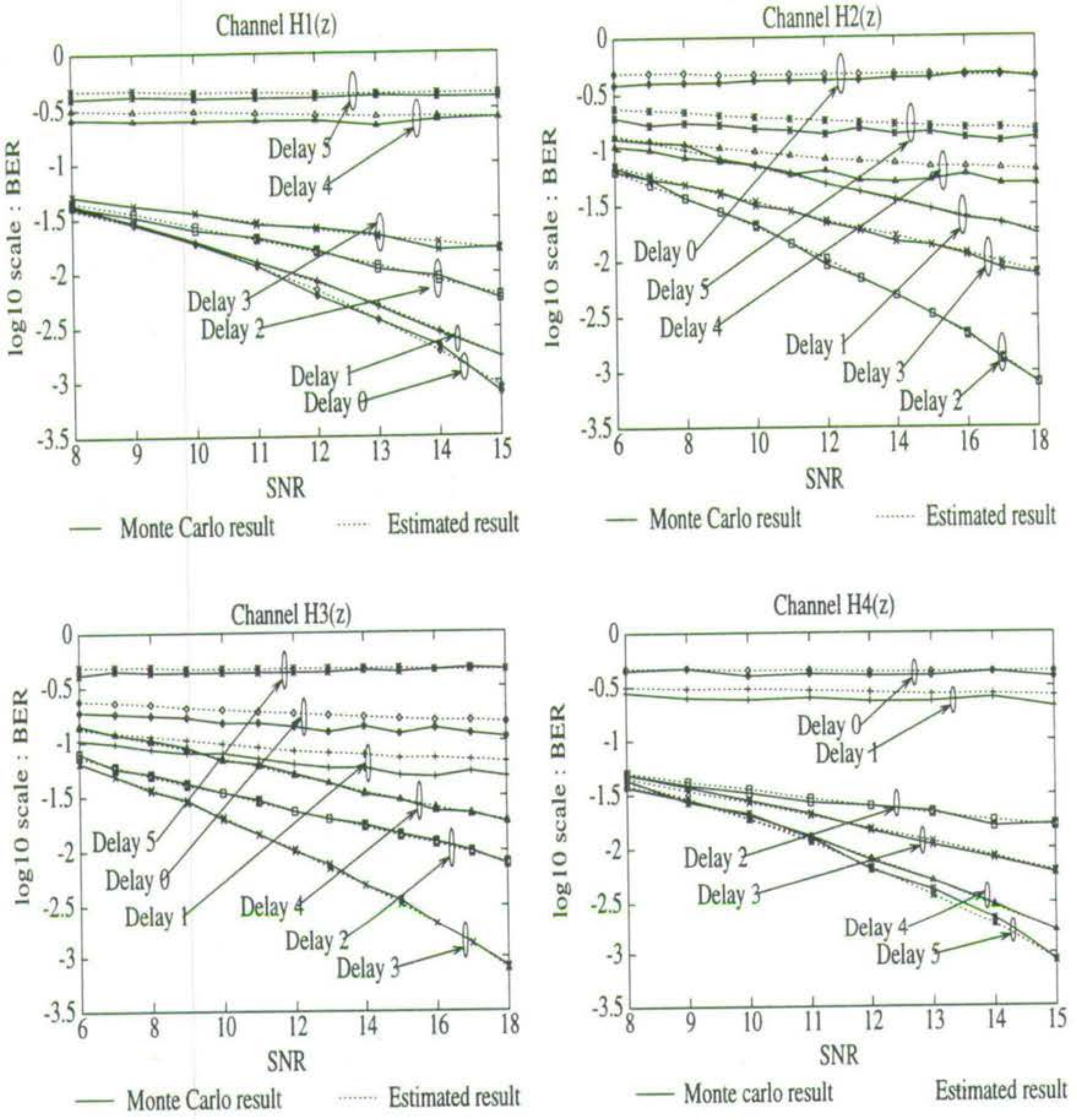

Figure 5.5: Estimated and Monte Carlo BER for channel $H1(z), H2(z), H3(z)$ and $H4(z)$

5.4 The minimum BER (MBER) linear equaliser

This section compares the difference between the classification performance of a linear equaliser with weights found by optimising the MSE versus weights found by optimising the BER.

To show that there can be a difference in performance, we illustrate the MSE and BER surface plots of the performance of a linear equaliser applied to the channel $H(z) = 0.5 + 1.0z^{-1}$ using delay order $d = 1$ with feedforward order $m = 2$ and SNR=20dB. The MSE surface and corresponding contours plot with respect to the weights $\mathbf{w} = [w_1 \; w_2]^T$ are illustrated in Figs. 5.6a,b respectively. It is observed from Fig. 5.6b that the locus of weights for a fixed MSE is in the form of an ellipse and the elliptical locus shrinks in size as the mean-square error J_{mmse} approaches the minimum value. When $\mathbf{w} = \mathbf{w}_{mmse}$, the locus reduces to a point. The

MMSE solution for this example is at $\mathbf{w}_{mmse} = [7.534e^{-01}, 9.766e^{-02}]^T$.

The BER surface and contour plots for the same equalisation problem are illustrated in Fig 5.7 to highlight the difference between the MSE surface and the BER surface. The plots show that the BER surface is a valley and that the MBER solution is along the weight value $w_2 \approx 0.01$.

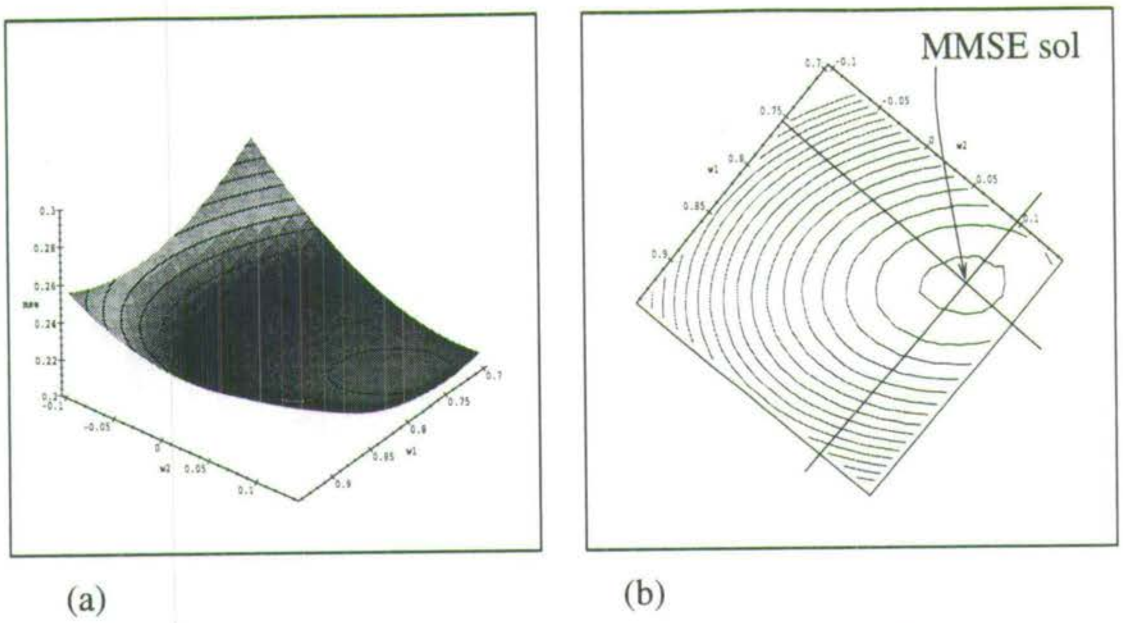

Figure 5.6: Linear equaliser's MSE performance for channel $H(z) = 0.5 + 1.0z^{-1}$, $m = 2$, $d = 1$ and SNR=20dB: (a) MSE surface with respect to weights. (b) Contour of MSE with respect to weights.

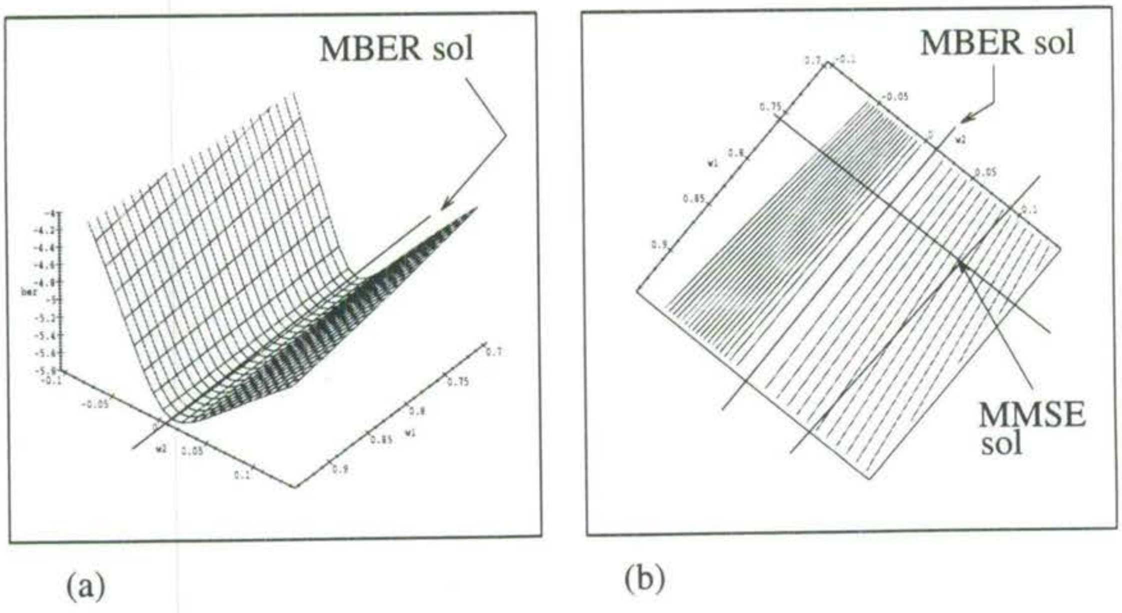

Figure 5.7: Linear equaliser's BER performance for channel $H(z) = 0.5 + 1.0z^{-1}$, $m = 2$, $d = 1$ and SNR=20dB: (a) BER surface with respect to weights. (b) Contour of BER with respect to weights.

It is obvious by comparing Figs. 5.6b and 5.7b that the weights satisfying the optimum MSE solution do not correspond to the optimum BER solution. For this example, it is obvious the BER performance may be improved by optimising the weights to minimise the BER cost function rather than the MSE cost function. An algorithm to perform BER optimisation is presented in the next section.

5.4.1 Optimising the weights of the linear equaliser with respect to BER

It was shown in the previous section that the MMSE criterion to find the weights of the linear equaliser may not result in the optimum BER performance. To optimise the weights for MBER, we examine a gradient descent method [113] to optimise the linear equaliser's probability of misclassification cost function (Eq. 5.29),

$$J_{ber}(\mathbf{w}) = P_l = \frac{1}{N_s} \sum_{\mathbf{c}_j \in C_d^{(+)}} P_{lc}(\mathbf{c}_j) + \frac{1}{N_s} \sum_{\mathbf{c}_k \in C_d^{(-)}} P_{lc}(\mathbf{c}_k). \tag{5.35}$$

To find the weights which will satisfy the minimum BER solution, we proceed as follows. We begin with an initial weight value $\mathbf{w}(0)$ using the MMSE solution. Using this initial guess, we compute the gradient vector, which is defined as the derivative of the $J_{ber}(\mathbf{w})$ cost function evaluated with respect to the weight $\mathbf{w}(n)$ at time n (i.e., the n^{th} iteration). We compute the next guess of the weight vector by making a change of the present guess in a direction scaled by a step size value opposite to that of the gradient. If the updated weight improves the BER performance, the new weights are accepted and the value of the step size doubled to accelerate the optimisation process. If the updated weight vector degrades the performance of the equaliser, the weight vector is not accepted and the step size parameter value is reduced to allow for finer update. The recursive process of calculating the gradient and modifying the weights is repeated until the maximum number of allowed iteration is reached. The details of the algorithm are as follows:

Algorithm 5.1 : Optimising the weights of linear equaliser to minimise BER

```
(i). Estimate the channel model $a(i)$.
(ii). Solve for the MMSE solution $\mathbf{w}_{mmse}$.
      Let $n = 1, \mathbf{w}(n) = \mathbf{w}_{mmse}, \mu(n) = \nabla(J_{ber}(\mathbf{w}(n)))/||\nabla(J_{ber}(\mathbf{w}(n)))||$.
(iii). Optimise w with respect to minimising $P_l$ (Eq. 5.35) using gradient descent [113], i.e.
      (a) $\mathbf{w}_{tmp} = \mathbf{w}(n) + \mu(n)[-\nabla(J_{ber}(\mathbf{w}(n)))]$
      (b) if $J_{ber}(\mathbf{w}_{tmp}) \leq J_{ber}(\mathbf{w}(n))$
            { $\mathbf{w}(n+1) = \mathbf{w}_{tmp}/||\mathbf{w}_{tmp}||$;        /* Update new weights */
```

```
        $\mu(n+1) = \mu(n) * 2;$              /* Increase step size */
    }
    else
    { $\mathbf{w}(n+1) = \mathbf{w}(n);$          /* Retain old weights */
        $\mu(n+1) = \mu(n) * 0.3;$  /* Reduce step size */
    }
(iv). n = n+1;
(v). if ($n \leq$ max iterations) Goto (iii).
```

The vector $\mathbf{w}(n)$ is the weight vector of the equaliser at iteration n, $\mu(n)$ is a positive real-value and $\nabla(J_{ber}(\mathbf{w}(n)))$ is the gradient vector of P_l using $\mathbf{w}(n)$, i.e.

$$\nabla(J_{ber}(\mathbf{w}(n))) = [\partial P_l/\partial w_1 \; \partial P_l/\partial w_2 \; \cdots \; \partial P_l/\partial w_m]^T, \tag{5.36}$$

where $\partial P_l/\partial w_k$, $1 \leq k \leq m$, is

$$\partial P_l/\partial w_k = \frac{1}{N_s}\sum_{j=1}^{N_s} \frac{\partial Q(|\zeta_j|/\sigma_e)}{\partial w_k}. \tag{5.37}$$

By the chain rule, each $\partial Q(|\zeta_j|/\sigma_e)/\partial w_k$ can be expressed as

$$\frac{\partial Q(|\zeta_j|/\sigma_e)}{\partial w_k} = \frac{\partial Q(|\zeta_j|/\sigma_e)}{\partial \zeta_j}\frac{\partial \zeta_j}{\partial w_k}. \tag{5.38}$$

After some algebra [6] it can be shown that

$$\frac{\partial Q(|\zeta_j|/\sigma_e)}{\partial w_k} = \frac{1}{\sqrt{2}} e^{-\frac{1}{2\sigma_e^2}\left(\frac{\mathbf{c}_j^T\mathbf{w}}{\mathbf{W}^T\mathbf{W}}\right)} \left(\frac{w_k(\mathbf{c}_j^T\mathbf{w}) - (sgn(\mathbf{c}_i^T\mathbf{w})c_{jk}\sum_{i=1}^m w_i^2)}{\sigma_e\sqrt{\pi}(\sum_{i=1}^m w_i^2)^{3/2}}\right). \tag{5.39}$$

In the above derivation, we have used $\mathbf{p} = 0$, i.e. the origin is a point in the decision boundary. We will prove later in the chapter that $\mathbf{p} = 0$ is a valid boundary point for our equalisation problems.

[6] Such partial differentiation can be easily solved using symbolic-mathematical packages like Maple. Maple is a registered trademark of *Waterloo Maple Software*. The Maple code used to derived $\partial Q(|\zeta_j|/\sigma_e)/\partial w_k$ is listed in Appendix B.

5.4.2 MBER linear equaliser : some simulation results

Example : Channel $H(z) = 0.5 + 1.0z^{-1}$

To examine the operations of Algorithm 5.1, it is applied to optimise the weights of a linear equaliser operating on the channel $H(z) = 0.5+1.0z^{-1}$ with delay order $d = 1$, feedforward order $m = 2$ and SNR=20dB. As illustrated in Figs. 5.6 and 5.7, the MMSE and MBER solutions for this problem are significantly different. To start the optimisation, we first evaluate the MMSE weights assuming perfect knowledge of the channel model and noise statistics. The weights are then iteratively updated if the modification of the weights improves the BER performance.

Figs. 5.8a-e depict the changes in the BER, MSE, weight values, and the step size μ during optimisation. The results show that there were no change in the weights value in the first two iterations. This is because the updating vector $\mu(n)[-\nabla(J_{ber}(\mathbf{w}(n)))]$ is too large, and the weights $\mathbf{w}_{tmp}$, if accepted, will actually cause a degradation in performance. When the μ value is reduced to a sufficiently small value in $n = 2$, the updated weights were accepted and the corresponding BER plot showed a jump in improvement from a value of -4.84 to -5.78, while the MSE value showed an increased from 0.217 to 0.264. From iterations $n = 3, \ldots, 20$, some updates of the weights occurred as indicated by the positive increase in the step size values. The changes to the weights are however very small, and little changes in the values of MSE and BER were actually observed. After 20 iterations, the step size plot (Fig. 5.8e) shows that μ converges to 0 indicating that the weights have converged to the MBER solution.

Fig. 5.9a compares the BER performance of the Bayesian equaliser and the linear equaliser with weights minimised by MMSE and MBER on channel $H(z) = 0.5 + 1.0z^{-1}$ using delay order $d = 1$ for various SNRs. The results indicate that the linear equaliser's performance was improved by applying the MBER criterion and that the best equaliser solution is the optimum nonlinear Bayesian equaliser.

Fig. 5.9b compares the same three equalisers' performance on the same channel using delay order $d = 2$. In this case, only a small improvement was gained by using the MBER linear equaliser over the MMSE equaliser. However, it is observed that the Bayesian equaliser's performance was only slightly better than the MMSE and MBER solution.

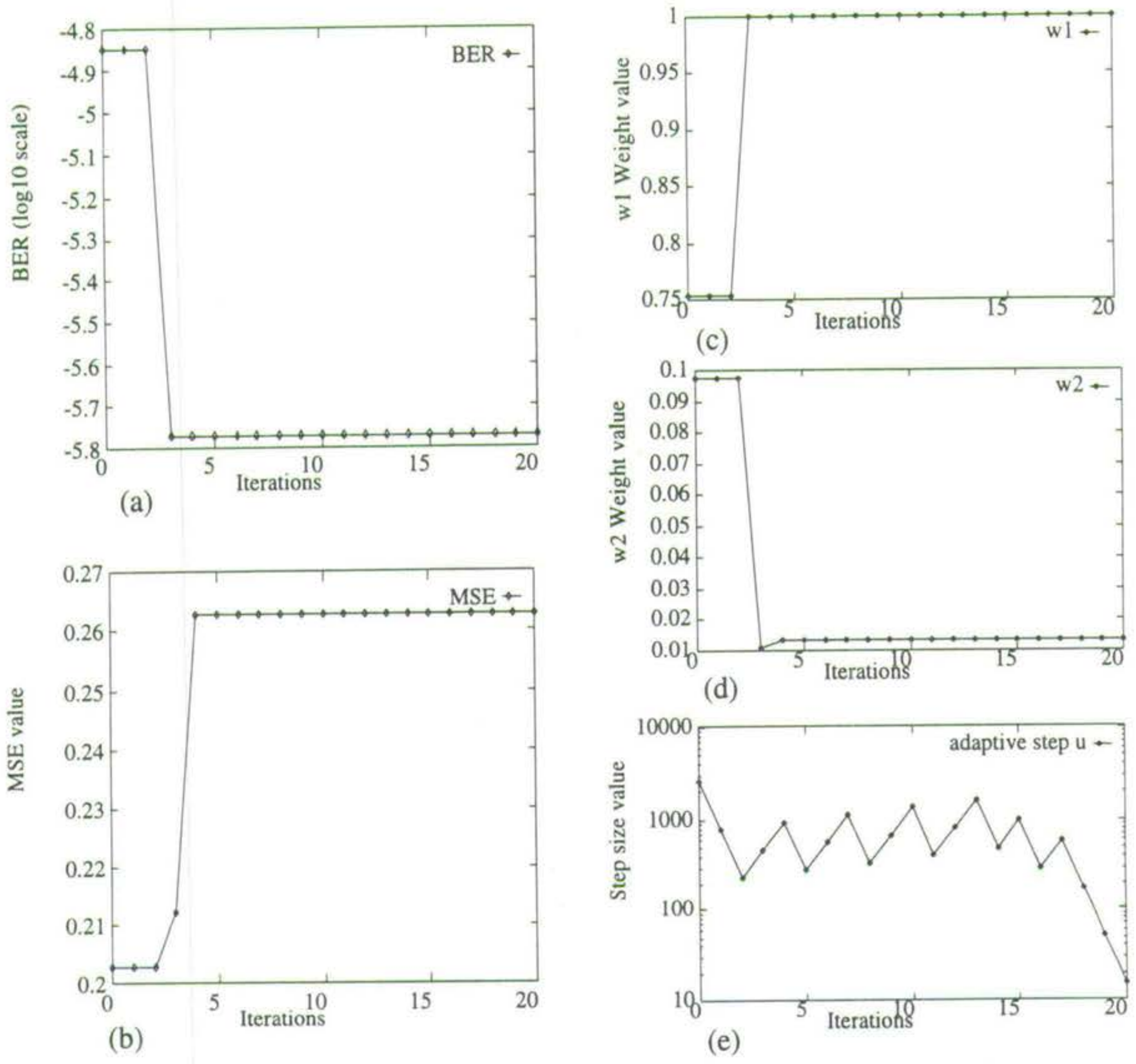

Figure 5.8: Optimisation behaviour of the equaliser for channel $H(z) = 0.5 + 1.0z^{-1}$, $m = 2$, $d = 1$, SNR=20dB: (a) BER plot, (b) MSE plot, (c,d) w1, w2 weight value plot, (e) Step-size μ plot.

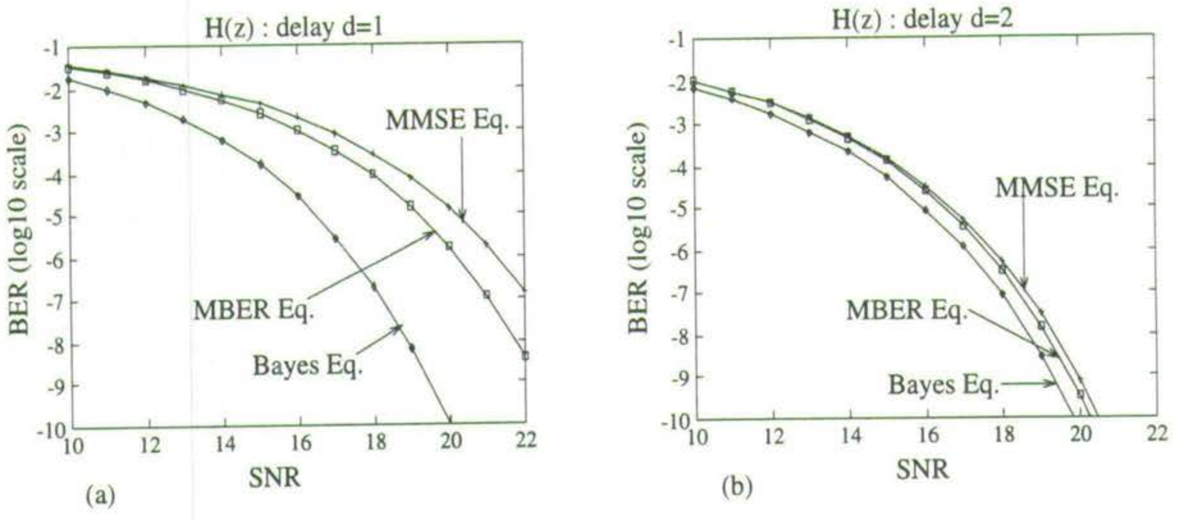

Figure 5.9: BER performance of the Bayesian equaliser, the MBER and MMSE linear equaliser for channel $H(z) = 0.5 + 1.0z^{-1}$ using $m = 2$ and (a) delay $d = 1$, (b) delay $d = 2$.

Example : Channel $H5(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$

Another channel $H5(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$ was used to study the effects of optimisation using the MBER cost function. The parameters used for the equaliser were delay order $d = 1$ and feedforward order $m = 4$ operating under a SNR condition of 20dB. The initial MMSE weights of the equaliser were evaluated by assuming perfect knowledge of the channel model and noise statistics. Fig. 5.10 illustrates the changes in the values of BER, MSE, weights and μ during optimisation. For this example, convergence is reached approximately after 20 iterations and the results showed that the BER was improved from -3.05 to -3.37 while the MSE value showed a degradation from 0.198 to 0.231. The weights showed little changes after convergence (20 iterations), although the step size μ parameter exhibited some oscillation behaviour indicating that updates to the weights were still being carried out.

The BER performance of the MMSE linear equaliser, MBER linear equaliser and the Bayesian equaliser for the channel $H5(z)$ for various SNR is depicted in Fig. 5.11. It is observed that the Bayesian equaliser achieves the best BER performance followed by the MBER linear equaliser, and the worst performance by the MMSE linear equaliser.

From the simulation results, we have seen that for some cases, significant improvement in misclassification performance may be achieved by optimising the weights of the linear equaliser using the MBER cost function instead of the MMSE cost function, and for other cases, the results show that there is no significant gain to be achieved by optimising the weights to minimise the BER. It is, however, not possible to know in advance if BER optimisation would result in a much better BER performance over the MMSE linear equaliser.

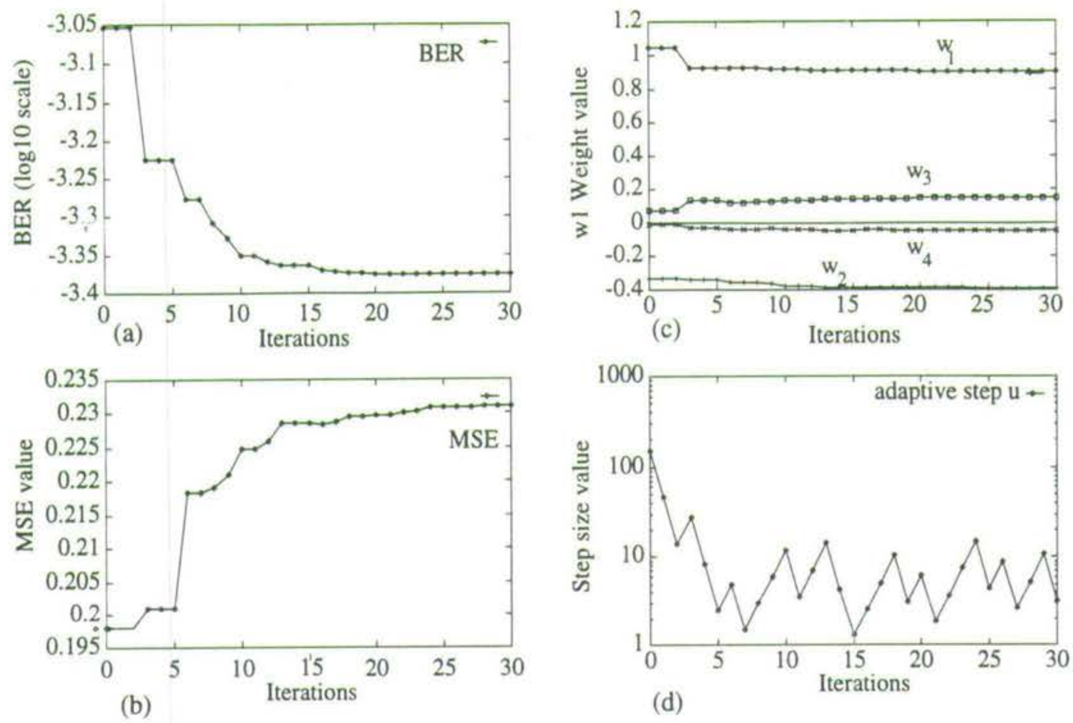

Figure 5.10: Optimisation behaviour of the equaliser for channel $H5(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$ operating at SNR = $20dB$: (a) BER plot, (b) MSE plot, (c) $\{w_1, w_2, w_3, w_4\}$ weight values plot, (d) Step-size μ plot.

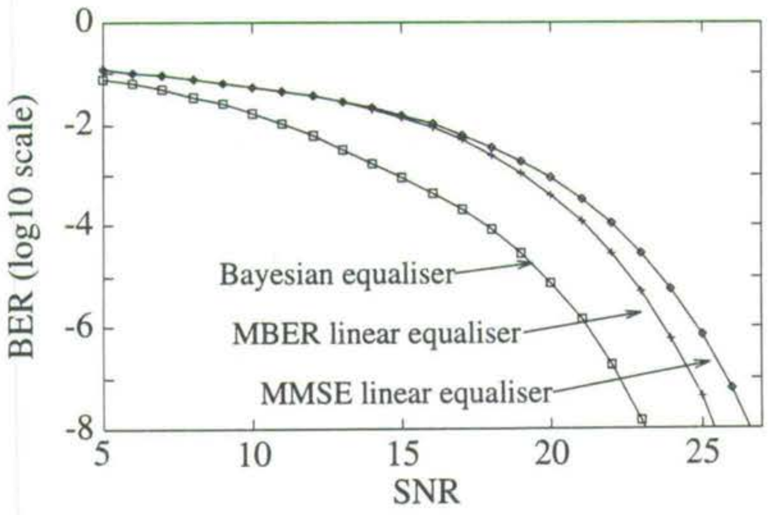

Figure 5.11: Comparing the performance of Bayesian, MBER, and MMSE feedforward equaliser for channel $H5(z) = 0.3482+0.8704z^{-1}+0.3482z^{-2}$, the parameters of the equaliser used were delay order $d = 1$ and feedforward order $m = 4$.

5.4.3 Proof that the origin is a boundary point

In the previous section, we assumed that the boundary realised by the linear equaliser should pass through the origin. We will prove in this section why this is so. The proof is presented by comparing the performance of the equalisers which have parallel decision boundaries, i.e., decision boundaries that pass through the origin, and a parallel decision boundary that does not pass through the origin. By showing that the decision boundary which passes through the origin results in the smallest misclassification probability, we conclude that the linear equaliser should have the the decision boundary passing through the origin.

The general equation of the linear equaliser's decision function with an offset term is,

$$f_l(\mathbf{r}) = \sum_{i=1}^{m} w_i r(k-i+1) + z. \tag{5.40}$$

In the case when $z = 0$, the realised decision boundary will pass through the origin and in the case when $z \neq 0$, the decision boundary will be shifted along the normal of the boundary by distance b [115], where

$$b = \frac{z}{\sqrt{\mathbf{w}^T \mathbf{w}}}. \tag{5.41}$$

The purpose of the boundary is to partition the set of channel states $C_d = \{\mathbf{c}_j\}, 1 \leq j \leq N_s$ into two regions. In the following analysis, we will assume that the delay order is chosen such that the set of channel states $C_d^{(-)}$ and $C_d^{(+)}$ are linearly separable. Given that the transmit symbol is from the set $\{\pm 1\}$ and therefore is zero mean, the set of channel states C_d will be symmetrically distributed about the origin. It is this symmetrical distribution of the channel states about the origin that is the key to proving that the offset should be 0.

Fig. 5.12 illustrates the effects of z on the linear decision boundary for the decision function $f_l(\mathbf{r}) = -2r_1 + r_2$. Note that the decision boundary of case 2 is parallel to case 1, and that the distance of the decision boundary of case 2 to case 1, along the normal of the boundary, is b.

The probability of misclassification for the equaliser with linear decision boundary is (See Eq. 5.29),

$$P_l = \frac{1}{N_s} \sum_{\mathbf{c}_j \in C_d^{(+)}} P_{lc}(\mathbf{c}_j) + \frac{1}{N_s} \sum_{\mathbf{c}_k \in C_d^{(-)}} P_{lc}(\mathbf{c}_k) \tag{5.42}$$

where N_s = number of possible channel states, $C_d^{(+)}$ and $C_d^{(-)}$ are the set of channel states associated with transmitted symbol $s(k-d) = +1$ and -1.

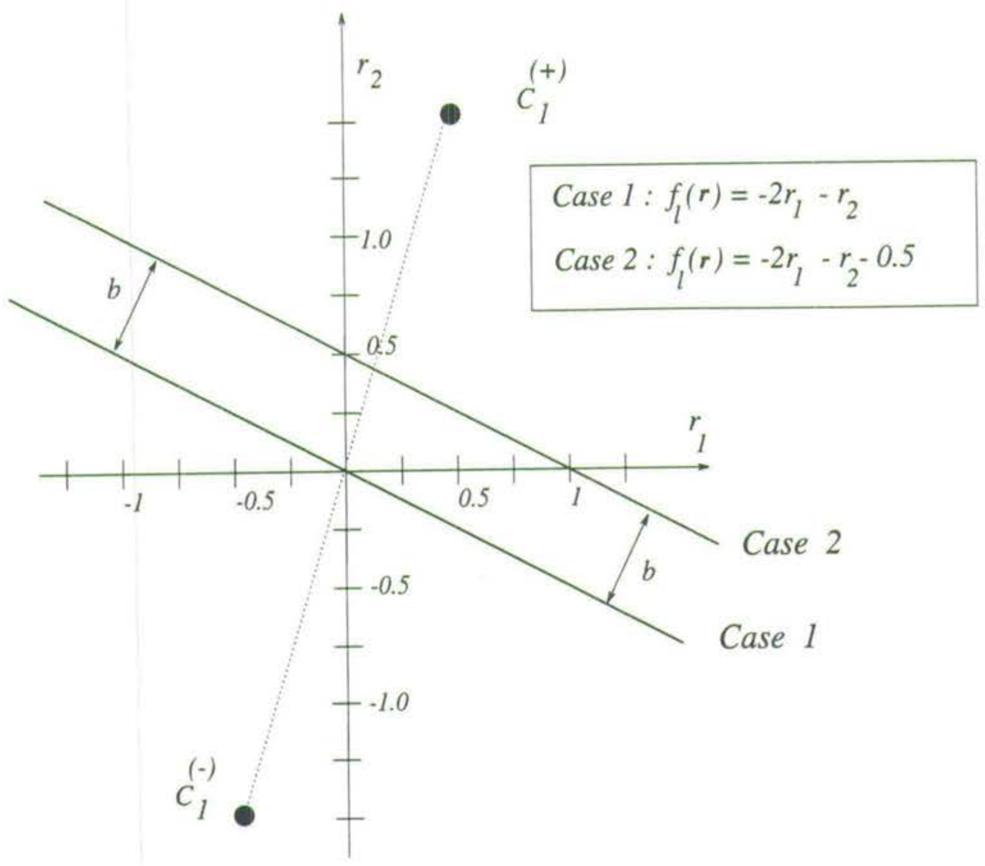
r_2
$C_1^{(+)}$
Case 1 : $f_l(\mathbf{r}) = -2r_1 - r_2$
Case 2 : $f_l(\mathbf{r}) = -2r_1 - r_2 - 0.5$
1.0
0.5
b
r_1
-1
-0.5
0.5
1
Case 2
b
-0.5
Case 1
-1.0
$C_1^{(-)}$

Figure 5.12: Parallel decision boundary with different offset values

We first show that in the case when the decision boundary passes through the origin, the sum of probability of misclassification due to channel states in the $C_d^{(+)}$ equals the sum of probability of misclassification due to the channel states in $C_d^{(-)}$. This statement can be proved by noting the symmetrical distribution of channel states about the origin, i.e., each channel state $\mathbf{c}_j^{(+)} \in C_d^{(+)}$ has a 'companion' channel state $\mathbf{c}_j^{(-)} \in C_d^{(-)}$ where the values of channel state $\mathbf{c}_j^{(-)} = -\mathbf{c}_j^{(+)}$, i.e.,

$$\mathbf{c}_j^{(+)} = F[\mathbf{s}_j(k)|s(k-d) = +1] \quad (5.43)$$

$$\mathbf{c}_j^{(-)} = F[-\mathbf{s}_j(k)|s(k-d) = -1] = -\mathbf{c}_j^{(+)} \quad (5.44)$$

where $\mathbf{s}_j(k) = [s_j(k) \ \cdots \ s_j(k-m-n_a+2)]^T$ is the j^{th} combination of $\mathbf{s}(k)$, and $F \in R^{m\times(m+n_a-1)}$ is the matrix [7] defined in Eq. 4.6.

Because of these pairings, the distance of each channel state in $C_d^{(+)}$ to the decision boundary

[7]The matrix F is

$$F = \begin{bmatrix} a(0) & a(1) & \ldots & a(n_a-1) & 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & a(0) & a(1) & \ldots & a(n_a-1) & 0 & \ldots & \ldots & \ldots & 0 \\ \vdots & \vdots & \vdots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & \ldots & \ldots & \ldots & \ldots & \ldots & a(0) & a(1) & \ldots & a(n_a-1) \end{bmatrix}.$$

is the same as the distance of the 'companion' channel state in $C_d^{(-)}$, i.e.

$$\varsigma_j^{(+)} = \left| \frac{\left(\mathbf{c}_j^{(+)}\right)^T \mathbf{w}}{||\mathbf{w}||} \right| \quad = \quad \varsigma_j^{(-)} = \left| \frac{\left(-\mathbf{c}_j^{(+)}\right)^T \mathbf{w}}{||\mathbf{w}||} \right|. \tag{5.45}$$

Therefore, the total probability of misclassification due to channel states in $C_d^{(+)}$ is equal to the probability of misclassification due to the channel states in $C_d^{(-)}$, i.e.,

$$\frac{1}{N_s} \sum_{\mathbf{c}_j \in C_d^{(+)}} P_{lc}(\mathbf{c}_j) = \frac{1}{N_s} \sum_{\mathbf{c}_k \in C_d^{(-)}} P_{lc}(\mathbf{c}_k). \tag{5.46}$$

In the case when an offset is introduced in the decision function such that the resultant decision boundary does not passes through the origin, the probability of misclassification due to channel states in $C_d^{(+)}$ will not be equal to channel states in $C_d^{(-)}$. If the decision boundary is shifted along the normal of the boundary by a distance b towards channel states $C_d^{(+)}$ (as in case 2 of Fig. 5.12), the probability of misclassifications due to channel states in $C_d^{(+)}$ increases, while the probability of misclassifications due to channel states in $C_d^{(-)}$ decreases. This is due to the fact that the distance to the boundary of channel states in $C_d^{(+)}$ decreases by b while the distance to the boundary of channel states in $C_d^{(-)}$ increases by b.

The following equations express the probability of misclassification of channel states $\mathbf{c}_j^{(+)}$ and $\mathbf{c}_j^{(-)}$ for case 1,

$$P_{lc(case1)}(\mathbf{c}_j^{(+)}) \quad = \quad Q\left(\frac{|\varsigma_j^{(+)}|}{\sigma_e}\right), \qquad P_{lc(case1)}(\mathbf{c}_j^{(-)}) = Q\left(\frac{|\varsigma_j^{(-)}|}{\sigma_e}\right), \tag{5.47}$$

and for case 2,

$$P_{lc(case2)}(\mathbf{c}_j^{(+)}) \quad = \quad Q\left(\frac{|\varsigma_j^{(+)}| - |b|}{\sigma_e}\right), \qquad P_{lc(case2)}(\mathbf{c}_j^{(-)}) = Q\left(\frac{|\varsigma_j^{(-)}| + |b|}{\sigma_e}\right). \tag{5.48}$$

Although the probability of misclassification for channel states in $C_d^{(-)}$ decreases due to the offset, the overall probability of misclassification of the equaliser increases because the probability of misclassifications due to channel states in $C_d^{(+)}$ are higher than the saving in probability of misclassification from channel states in $C_d^{(-)}$, i.e.,

$$\left| P_{lc(case2)}(\mathbf{c}_j^{(+)}) - P_{lc(case1)}(\mathbf{c}_j^{(+)}) \right| > \left| P_{lc(case2)}(\mathbf{c}_j^{(-)}) - P_{lc(case1)}(\mathbf{c}_j^{(-)}) \right|. \tag{5.49}$$

This is due to the nature of the error function $Q(.)$ which is weighted more heavily for values near zero. Applying the same observation to all the channel states, we have

$$\left|\sum_{\mathbf{c}_j^{(+)}\in C_d^{(+)}} P_{lc(case2)}(\mathbf{c}_j^{(+)}) - \sum_{\mathbf{c}_j^{(+)}\in C_d^{(+)}} P_{lc(case1)}(\mathbf{c}_j^{(+)})\right| > \left|\sum_{\mathbf{c}_j^{(-)}\in C_d^{(-)}} P_{lc(case2)}(\mathbf{c}_j^{(-)}) - \sum_{\mathbf{c}_j^{(-)}\in C_d^{(-)}} P_{lc(case1)}(\mathbf{c}_j^{(-)})\right|. \tag{5.50}$$

Therefore, any shift in the decision boundary away from the origin towards $C_d^{(+)}$ results in an increase in the probability of misclassification. By observing that the same argument can be applied if the decision boundary had been moved towards $C_d^{(-)}$, the proof is complete.

5.5 Decision Feedback equaliser

This section discusses the linear equaliser with decision feedback. The conventional DFE [3, 71] is based on a symbol-decision structure that employs a linear combination of the channel observations $\mathbf{r}(k)$ and the past decisions as illustrated in Fig. 5.13. We will call this DFE the linear-combiner DFE, in contrast to other DFE structures that use nonlinear combinations of the channel observations and past decisions [18, 73, 75, 82, 112].

The linear-combiner DFE's structure is very similar to the linear feedforward equaliser's structure. The difference lies in the DFE's decision function which now includes the application of past detected symbols [3, 71] i.e.,

$$f_l(\mathbf{r}(k), \hat{\mathbf{s}}_b(k)) = \mathbf{w}^T\mathbf{r}(k) + \mathbf{b}^T\hat{\mathbf{s}}_b(k), \tag{5.51}$$

where $\mathbf{w}$ and $\mathbf{r}(k)$ are the linear-combiner DFE's feedforward weights and feedforward input vector respectively, and $\mathbf{b}$ and $\hat{\mathbf{s}}_b(k)$ are the feedback weights and the vector of past detected symbols respectively. The vector $\hat{\mathbf{s}}_b(k)$ is

$$\hat{\mathbf{s}}_b(k) = [\hat{s}(k-d-1) \ \cdots \ \hat{s}(k-d-n)]^T \in R^n, \tag{5.52}$$

where d is the delay order parameter and n is the number of feedback terms. By comparing Eq. 5.51 to the linear equaliser's decision function (Eq. 5.4), it is obvious that the linear-combiner DFE's feedforward structure is identical to the linear feedforward equaliser and that the function of the weighted feedback term $\mathbf{b}^T\hat{\mathbf{s}}_b(k)$ is to introduce an offset in the linear boundary as discussed in the previous section (Sec. 5.4.3). Under the assumption that the equaliser's past decisions are correct, $\mathbf{b}^T\hat{\mathbf{s}}_b(k)$ can be designed to eliminate a large proportion of ISI without enhancing noise [3, 71]. The feedforward section $\mathbf{w}^T\mathbf{r}(k)$ then takes care of the remaining ISI. The weights of the DFE are usually chosen by minimising the MSE criterion. We will call such linear DFE the MMSE linear-combiner DFE.

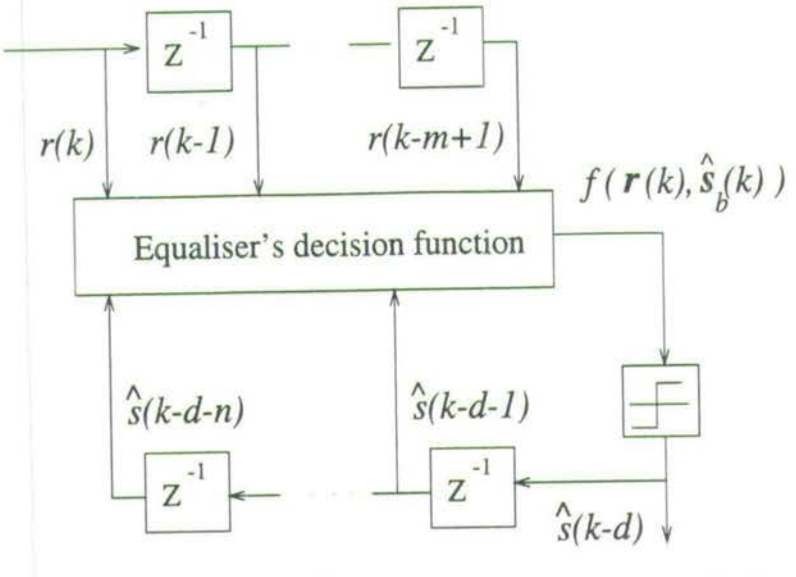

Figure 5.13: Schematic of a generic decision feedback equaliser.

5.5.1 Effects of introducing feedback

To illustrate how feedback helps to remove ISI, we apply the MMSE linear-combiner DFE to channel $H(z) = 0.5 + 1.0z^{-1}$ using delay $d = 1$, feedforward order $m = 2$, and feedback order $n = 1$ for SNR=15dB. The set of channel states for this problem is listed in Table. 5.1.

In the original equalisation problem without feedback, the linear equaliser is required to partition the full set of $N_s = 8$ channel states in C_d into two regions. By using past detected symbol information, the number of channel states that may occur is reduced from N_s to $N_s/2^n$ [18]. For example, if $s_b(k) = s(k-2) = +1$ for our equalisation problem, the set of channel states that may occur is reduced to the set $\{\mathbf{c}_1, \mathbf{c}_3, \mathbf{c}_5, \mathbf{c}_7\}$, and similarly, if the feedback $s_b(k) = s(k-2) = -1$, the set of channel states that may occur is reduced to the set $\{\mathbf{c}_2, \mathbf{c}_4, \mathbf{c}_6, \mathbf{c}_8\}$. This reduction of available combinations of channel state is illustrated in Fig. 5.14. Assuming that the feedback is correct, the reduced number of available channel states often results in decision boundaries becoming simpler and the non linearly separable equalisation problem becoming linear separable. In addition, the minimum distances of the channel states to the decision boundary normally become larger which in turn result in less probability of misclassification.

The decision boundaries of the MMSE linear-combiner DFE and linear feedforward equaliser are illustrated in Fig. 5.14. Note from the diagrams that the conditional linear-combiner DFE boundaries for feedback $\mathbf{s}_b(k) = +1$ and $\mathbf{s}_b(k) = -1$ are parallel. This can also be observed by realising that the value $\mathbf{b}^T\mathbf{s}_b(k)$ is the offset term z in the decision function as described

S/No j	Transmitted sequence $\mathbf{s}_j$ [$s(k)$	$s(k-1)$	$s(k-2)$]	Channel state $\mathbf{c}_j$ [$\hat{r}(k)$	$\hat{r}(k-1)$]
1	1	1	1	1.5	1.5
2	1	1	-1	1.5	-0.5
3	1	-1	1	-0.5	0.5
4	1	-1	-1	-0.5	-1.5
5	-1	1	1	0.5	1.5
6	-1	1	-1	0.5	-0.5
7	-1	-1	1	-1.5	0.5
8	-1	-1	-1	-1.5	-1.5

Table 5.1: Transmitted sequence and received channel states for channel $H(z) = 0.5 + 1.0z^{-1}$

in Eq. 5.40. From Fig. 5.14, we can also see that the channel states $\{\mathbf{c}_3, \mathbf{c}_5\}$ (Fig. 5.14a) and $\{\mathbf{c}_4, \mathbf{c}_6\}$ (Fig. 5.14b) are further away from the MMSE DFE boundary than the linear equaliser's decision boundary. This increase in distance away from the boundary is the reason why the MMSE linear-combiner DFE has less probability of misclassification during equaliser's operation[8]. There is however one problem when applying past detected symbols in the decision function, that of error propagation [108–111] which occurs when wrongly detected symbols are passed into the feedback vector. Error propagation however is not catastrophic and in most cases the loss due to its effect is not significant.

The results of the Monte Carlo simulation using the MMSE linear feedforward equaliser, MBER linear feedforward equaliser, and the MMSE linear-combiner DFE for the channel $H(z)$ using delay order $d = 1$, feedforward order $m = 2$, and for the DFE feedback order $n = 1$ is illustrated in Fig. 5.15. The simulation results show that the MMSE linear-combiner DFE perform better than both the equaliser without feedback and that the performance loss due to error propagation of the DFE is approximately 2db at SNR equals to 12dB, but becomes smaller at higher SNR.

[8]Although $\{\mathbf{c}_1, \mathbf{c}_7\}$ (Fig. 5.14a) and $\{\mathbf{c}_2, \mathbf{c}_8\}$ (Fig. 5.14b) becomes nearer to the MMSE linear-combiner DFE boundary, their distance to the boundary are further away than channel states $\{\mathbf{c}_3, \mathbf{c}_5\}$ and $\{\mathbf{c}_4, \mathbf{c}_6\}$ which dominates the probability of error.

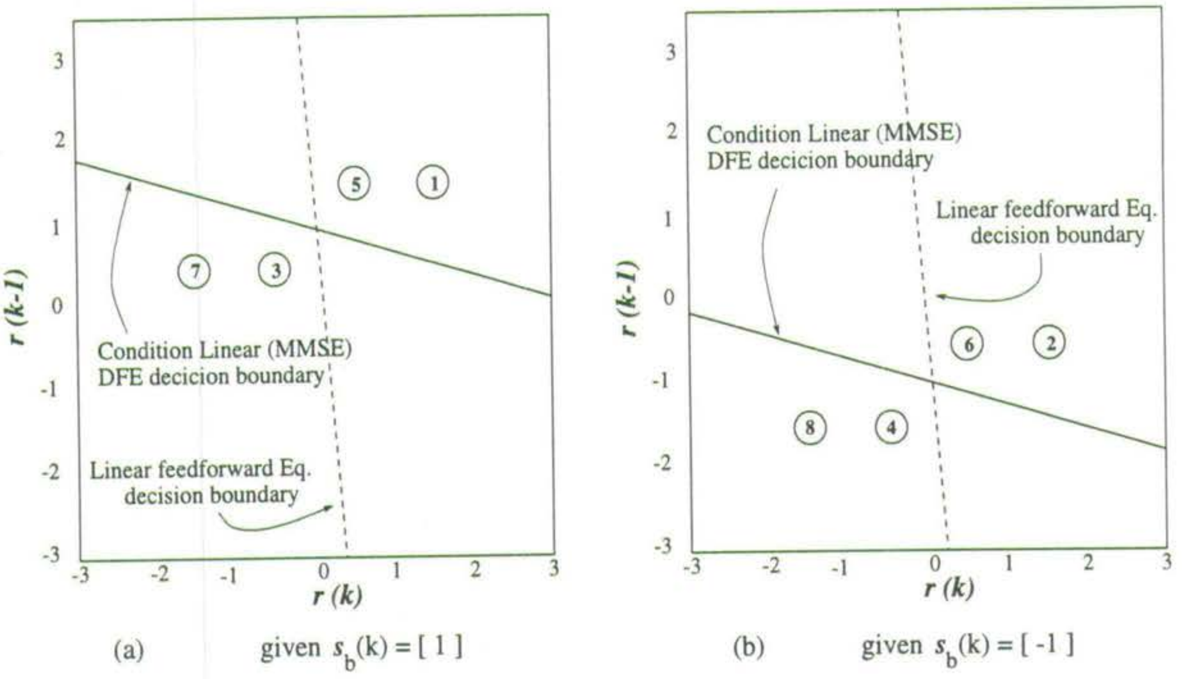

Figure 5.14: Conditional set of possible channel state locations and decision boundaries for MMSE linear-combiner DFE and MMSE linear feedforward equaliser for feedback : (a) $\mathbf{s}_b(k) = [+1]$, (b) $\mathbf{s}_b(k) = [-1]$.

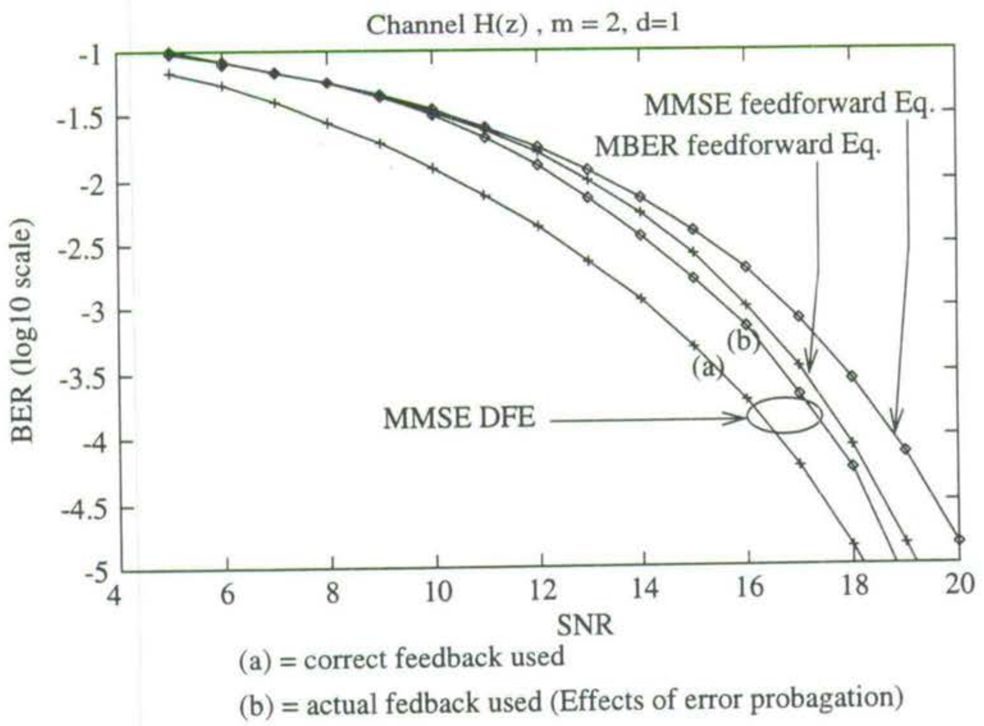

Figure 5.15: Comparison of Feedforward Eq, and DFE Equaliser for channel $H(z) = 0.5+1.0z^{-1}$ using delay $d = 1$ and $m = 2$. (DFE uses $n = 1$, i.e. feedback vector is $s(k-2)$.)

5.5.2 Alternative implementation of Linear DFE

As described in Sec. 4.5.3, it is possible to convert the DFE structure into a feedforward structure by introducing a transformation to cancel the past detected symbol's ISI from the input vector $\mathbf{r}(k)$. That is, the new transformed input vector $\mathbf{r}^{'}(k)$ to the DFE equaliser is

$$\mathbf{r}^{'}(k) = \mathbf{r}(k) - F[\mathbf{0}, \hat{\mathbf{s}}_b(k)]. \tag{5.53}$$

Fig. 5.16 illustrates the schematic layout of the DFE with the transformation process. To apply the transformation, an estimate of the channel must be available in addition to the values of the feedback vector $\hat{\mathbf{s}}_b(k)$ values. By applying the feedback information this way, the problem is reduced to a linear feedforward equaliser [9]. Therefore by reducing the DFE problem to a feedforward equalisation problem, it is obvious that the MBER optimisation procedure to optimise the weights of the feedforward weights of the DFE can be carried out.

Fig. 5.17 illustrates the translated channel states and decision boundary for channel $H(z) = 0.5 + 1.0z^{-1}$. The diagram shows the translation of subset channel states $\{\mathbf{c}_1, \mathbf{c}_3, \mathbf{c}_5, \mathbf{c}_7\}$ and $\{\mathbf{c}_2, \mathbf{c}_4, \mathbf{c}_6, \mathbf{c}_8\}$ into $\{\mathbf{c}_1^{'}, \mathbf{c}_2^{'}, \mathbf{c}_3^{'}, \mathbf{c}_4^{'}\}$. Notice that the MMSE solution in this case is far away from the best possible linear solution. In general, the minimum MSE solution is different from the minimum BER solution. Fig. 5.17 also suggests that the linear-combiner DFE can also be optimised for BER to improve on the equaliser's performance.

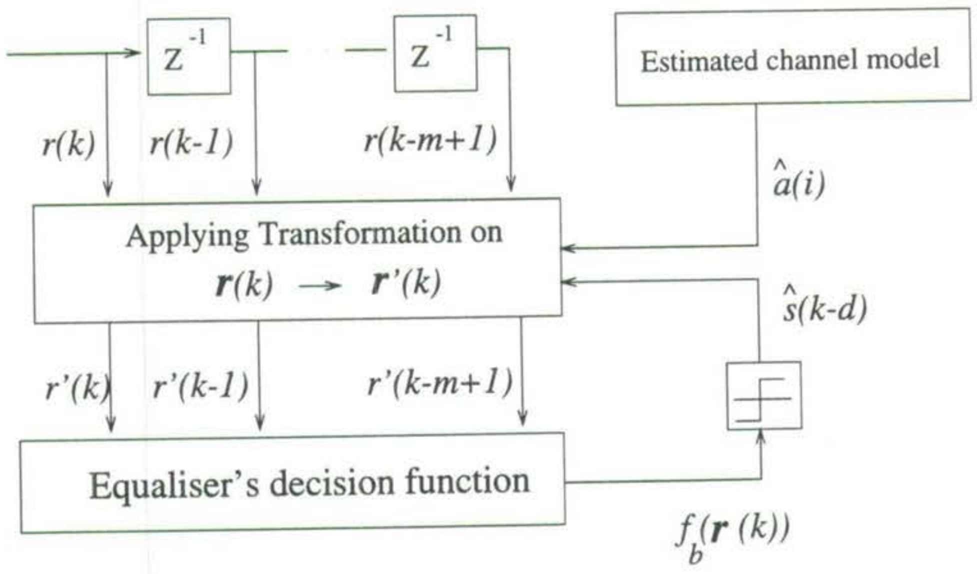

Figure 5.16: Schematic of the translation of input signal $\mathbf{r}(k) \rightarrow \mathbf{r}^{'}(k)$ for the DFE.

[9] In this study, we will assume that the statistics of the channel is completely known.

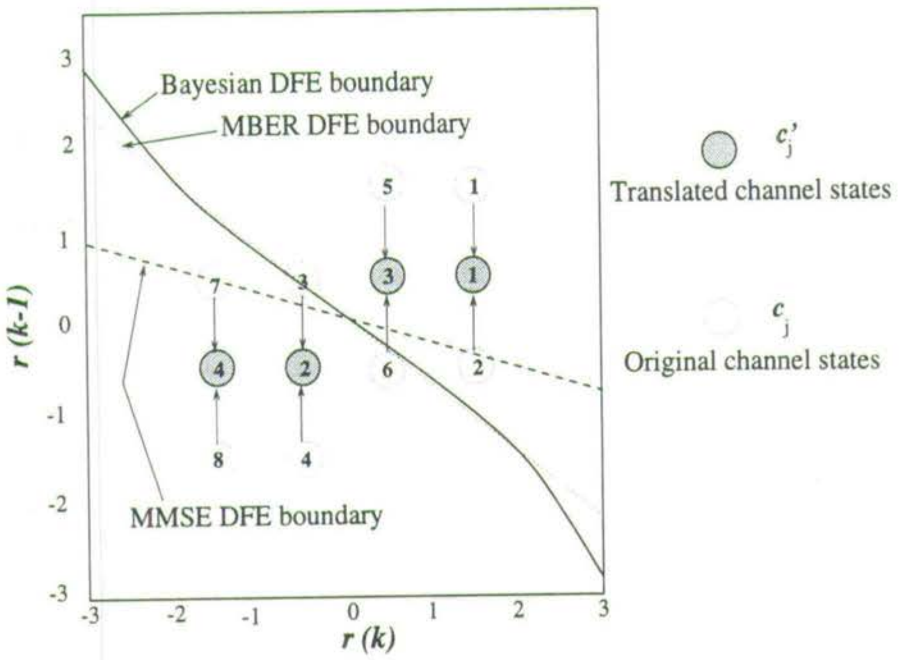

Figure 5.17: Illustration of translated channel states and the corresponding Bayesian DFE decision boundary, MBER DFE linear boundary and the MMSE DFE linear boundary for channel $H(z) = 0.5 + 1.0z^{-1}$ with $m = 2$, $d = 1$ and SNR = 10dB.

5.5.3 MBER DFE : some simulation results

Two examples are used to compare the MMSE and MBER solutions of a linear-combiner DFE. The first example uses the two-tap channel $H(z) = 0.5 + 1.0z^{-1}$ and the following equaliser's parameter $m = 2$, $d = 1$ and $n = 1$. The decision boundaries of the Bayesian DFE, the MMSE and MBER linear-combiner DFE plotted in the translated observation space $\mathbf{r}'(k)$ are shown in Fig. 5.17. Fig. 5.18 compares the BERs as a function of SNR with detected symbols being fed back for these three DFEs. Note that the MBER DFE's performance is very similar to the Bayesian DFE. This result agrees with the decision boundary plot (Fig. 5.17) which indicated that the MBER DFE's decision boundary closely approximates the Bayesian solution.

The second example uses a 5-tap channel with the following transfer function

$$H6(z) = 0.227 + 0.466z^{-1} + 0.688z^{-2} + 0.466z^{-3} + 0.227z^{-4} \tag{5.54}$$

and equaliser's parameter $m = 5$, $d = 4$, and $n = 4$. The BERs of the Bayesian DFE, the MMSE and MBER linear-combiner DFEs with detected symbols being fed back are plotted in Fig. 5.19, where it can be seen that the performance of the MBER linear-combiner DFE is better than the MMSE solution. The performance gap between the Bayesian DFE and MBER linear-combiner DFE confirms the fact that the real optimal solution for the DFE structure is generally nonlinear. The "best linear solution" is sub-optimal in nature. However, the usual MMSE solution is inferior to this "best linear solution".

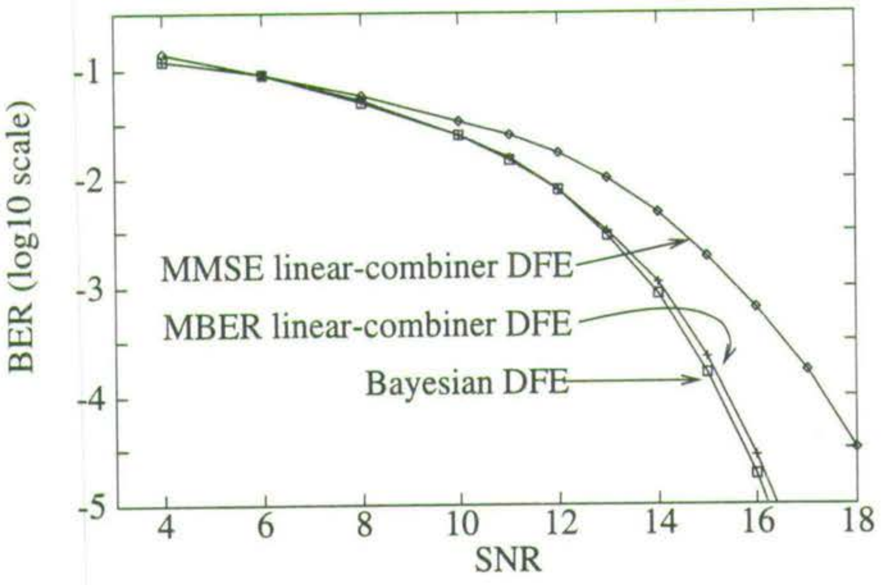

Figure 5.18: Comparing the performance of Bayesian, MBER, and MMSE DFE equalisers for channel $H(z) = 0.5{+}1.0z^{-1}$ with detected symbols being fed-back. The parameters of the equaliser used were delay $d = 1$, $m = 2$ and $n = 1$.

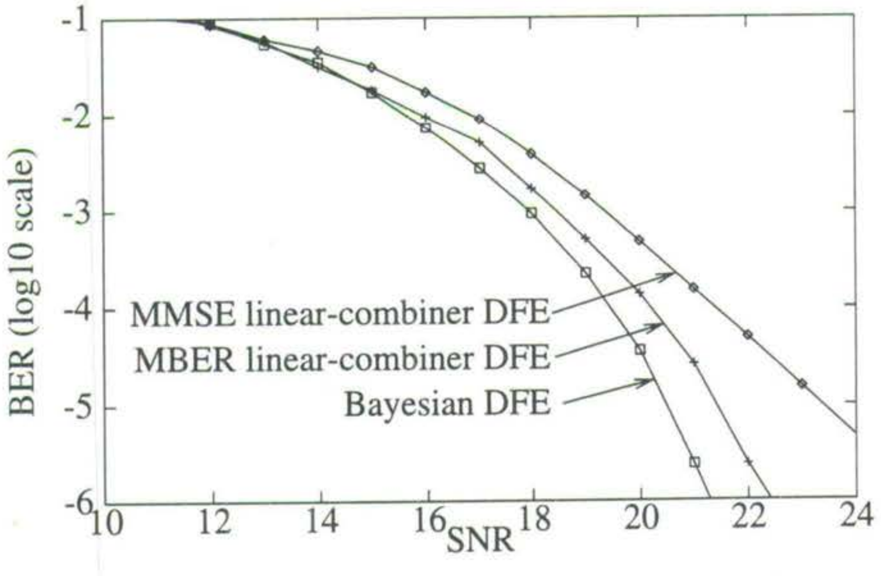

Figure 5.19: Comparing the performance of Bayesian, MBER, and MMSE DFE equalisers for channel $H6(z) = 0.227 + 0.466z^{-1} + 0.688z^{-2} + 0.466z^{-3} + 0.227z^{-4}$ with detected symbols being fed-back. The parameters of the equaliser used were delay $d = 4$, $m = 5$ and $n = 4$.

5.6 Conclusions

This chapter examined the linear equaliser's performance. In particular, we highlighted the effects of delay order on decision boundaries geometrically, and showed using computer simulation results the significant effects of delay order on BER performance. Analysis of the difference in capability of the Bayesian equaliser's decision function and the linear equaliser's decision function was also presented. To improve the MMSE linear equaliser's performance, we proposed optimising the weights using the MBER criterion. Our results indicated that in some cases, performance could be improved. This MBER optimisation was also extended to feedforward weights of the DFE.

Chapter 6

Conclusions

6.1 Introduction

The work described within this thesis is primarily concerned with the application of nonlinear models with linear-in-the-parameter structure to time series prediction and digital communication channel equalisation. In particular, the Volterra and RBF expansion techniques of introducing nonlinearity into the linear models were considered in detail. Such expansion techniques, however, can generate very large initial models which are generally un-acceptable for practical applications. To reduce the model size the OLS subset model selection technique [10, 11, 53, 54] was studied. In this study, one major objective was to compare the performance of nonlinear models to that of conventional linear models for time series prediction and equalisation problems. Our results and analysis have shown that nonlinear models have superior performance over linear models in both applications.

In the next section, the work performed is first summarised and specific achievements accomplished are highlighted (Sec. 6.2). In Sec. 6.3, the limitations of the current work are discussed and some new directions for future work are proposed.

6.2 Summary and specific achievements of work performed

The work examined in the thesis can be broken down into two major parts. In the first part, chapters 2 and 3, the problems of model selection using the OLS algorithm and the application of nonlinear predictors for time series prediction were examined. In the second part of the thesis, chapters 4 and 5, the problem of digital communication channel equalisation using linear and nonlinear filters was considered. The major outline of each chapter is given in the following paragraphs.

The objective of chapter 2 was to briefly introduce the background of work considered in the thesis. In chapter 3, three results regarding the OLS algorithm and RBF predictor were presented. The first result concerned the sub-optimum nature of the OLS algorithm selection process.

To improve the selection process, a back-tracking OLS algorithm was proposed [12]. Computer simulations were conducted to compare the performance of the subset models found using these two selection techniques. Although the results indicated that the back-tracking method can find better subset models, the improvements were also shown to be very small, and for all practical purposes, the OLS's solution would be sufficient. The second result presented in chapter 3 concerned the OLS algorithm's implementation requirement. Although the OLS algorithm is known to be very efficient, our studies revealed that it is possible to further reduce the implementation complexity by introducing some pre-processing steps [13, 16]. The pre-processing steps are introduced by performing an orthonormal transformation on the original regression problem to reduce the size of the information matrix. Two different pre-processing techniques, the reduced OLS-Gram Schmidt and reduced OLS-SVD, were considered. The computational requirement of the reduced OLS-Gram Schmidt and OLS algorithm were compared using some examples, and the results showed that the reduced OLS method can ease the implementation requirement. The third result reported in chapter 3 concerned the application of RBF predictors to time series predictions. Using computer simulations, we showed that the performance of nonlinear RBF predictors was superior to that of linear predictors for predicting nonlinear stationary time series. The performance of the RBF predictor, however, showed significant degradation in performance when the time series to be predicted exhibited homogeneous non-stationary behaviour. To improve the predictive performance, the gradient RBF network [15,16] was introduced. The GRBF predictor is based on having the nonlinearity of the model respond to the gradient of the time series. As such, the effects on the varying trends and mean would be reduced and hence the predictor's performance would be improved. This is confirmed using computer simulation results which compare the GRBF predictor and RBF predictor for such time series problems.

The second part of the thesis, chapters 4 and 5, examined the application of nonlinear filter to channel equalisation problems. Chapter 4 first introduced the channel equalisation problem and then derived the optimum solution for the symbol-by-symbol detection equalisation problem, the Bayesian equaliser. It was shown that the Bayesian equaliser's decision function is nonlinear and its structure identical to the RBF model. In addition, it was demonstrated that the RBF model can realise the Bayesian equaliser if the channel statistics are given. Three main results were reported in this chapter. The first result reported concerned the effects of delay order parameter on the equalisation problem. Our studies showed that the use of different delay order parameters not only changes the shape of the decision boundaries, it also limits the performance of the optimum equaliser operating under a fixed SNR condition [19]. That is, for the same SNR condition, the performance of the optimum Bayesian equaliser can result in significantly different levels of classification performance. To determine the optimum operating delay order, a simple method of estimating the BER performance of the Bayesian equaliser was proposed. Simulations were conducted to compare the results of the proposed estimator to those obtained from Monte Carlo simulations. The results showed that the proposed estimator is quite accurate and hence may be used to determine the effective operating delay order. The second main result concerned the selection procedure for reducing the implementation complexity of the RBF Bayesian equaliser. In the full implementation of the RBF Bayesian equaliser, a considerable amount of computational processing is required during the equaliser's operation. This makes the RBF implementation unattractive for practical applications. To reduce the implementation complexity, a model selection technique to select a reduced-size RBF equaliser was examined [19,21]. Our results showed that some centres from the RBF equaliser may be removed without introducing significant degradation to classification performance if these centres are selected carefully. The third and last result reported in this chapter concerned the application of decision feedback to the equaliser's operation. It was shown that the introduction of decision feedback results in the improvement of the equaliser's classification performance and a reduction in implementation complexity. In addition, we showed that it is possible to convert the DFE problem to the conventional feedforward equaliser's performance by introducing a transformation. Such a transformation not only allows us to view the equalisation problem in an unified manner, it also simplifies the RBF implementation of the Bayesian DFE [20].

Chapter 5 considered the application of linear techniques to channel equalisation. The purpose of this chapter was to introduce linear techniques and then compare the performance of linear equalisers to those of nonlinear equalisers. The chapter began with an introduction to the conventional linear equaliser, the MMSE linear equaliser. To highlight the difference in classification ability between the linear and nonlinear equaliser, some geometric examples which illustrate the functional mapping of the decision function were given. It was shown that the linear equaliser's decision function could only realise linear mapping and hence only linear decision boundaries. This is as compared to the nonlinear method which could generate much more complicated nonlinear mapping, and hence nonlinear decision boundaries. As the optimum decision boundary for most equalisation problem is generally nonlinear, the linear equaliser's solution is normally sub-optimum. In addition, it was also shown that the MMSE criterion is not the best criterion to use when optimising the linear equaliser's performance. To improve the linear equaliser's performance, the MBER criterion was considered. The MBER optimisation was carried out using an iterative gradient descent method. Our results showed that the MBER linear equaliser normally performs better than the conventional MMSE linear equaliser [21]. This optimisation scheme was also extended to the linear-combiner DFE by first introducing the transformation to convert the DFE to the feedforward equaliser's structure [20].

6.3 Limitations of current work and proposal for future work

This section discusses some of the limitations of work performed.

In this thesis, only the Volterra and RBF expansion techniques were considered to perform the nonlinear expansions for the linear-in-the-parameter structure. Other expansion techniques, e.g. NARMAX models [37,38], multivariate adaptive regression splines (MARS) [26,39], fuzzy basis function models [40] and so on, were not studied. Therefore, one clear direction for future work is to consider the other expansion techniques.

The other main limitation of this work was the assumption of stationarity in the problems we have considered. As most real world problems are non-stationary, adaptive implementation is an important area of future work.

Lastly, in the study of MBER optimisation for the linear equaliser, more work can be carried out to examine the nature of the linear equaliser's BER surface. In this thesis, we had assumed that the BER surface is uni-modal and had performed un-constrained optimisation when optimising for MBER performance. We have however not proved that the actual BER surface is uni-modal. Also, in this study of MBER optimisation, we had begun the optimisation process using the MMSE solution. As the computational requirement for solving for the MMSE weights is high, this may not be a practical choice. The question of having an arbitrary starting point for optimisation should be examined.

References

[1] S.HAYKIN, *Adaptive Filter Theory (2nd Edition).* Prentice-Hall, 1991.

[2] A.S.WESIGEND and N.A.GERSHENFELD, *Time series prediction : forecasting the future and understanding the past.* Addison-Wesley, 1994.

[3] S.U.H.QURESHI, "Adaptive equalization", *Proc. IEEE,* vol. 73, no. 9, pp. 1349–1387, 1985.

[4] P.ALPER, "A consideration of the discrete Volterra series", *IEEE Trans. AC,* vol. AC-10, pp. 322–327, 1965.

[5] T.KOH and E.J.POWERS, "Second-order Volterra filtering and its application to nonlinear system identification", *IEEE Trans. ASSP,* vol. 33, no. 6, pp. 1445–1455, 1985.

[6] S.Y.FAKHOURI, "Identification of the Volterra kernels of nonlinear system", *IEE Proc,* vol. 127, Part D, no. 6, pp. 296–304, 1980.

[7] M.J.D.POWELL, "Radial basis functions for multivariable interpolation: a review", *Algorithms for Approximation,* pp. 143–167, J.C.MASON and M.G.COX (Eds), Oxford, 1987.

[8] M.J.D.POWELL, "Radial basis functions approximations to polynomials", *Proc. 12th Biennial Numerical Analysis Conf. (Dundee),* pp. 223–241, 1987.

[9] D.S. BROOMHEAD and D.LOWE, "Multivariable functional interpolation and adaptive networks", *Complex Systems,* vol. 2, pp. 321–355, 1988.

[10] S.CHEN, C.F.N.COWAN, and P.M.GRANT, "Orthogonal least squares learning algorithm for radial basis function networks", *IEEE Trans. Neural Networks,* vol. 2, no. 2, pp. 302–309, 1991.

[11] S.CHEN, S.A.BILLINGS, and W.LUO, "Orthogonal least squares methods and their application to nonlinear system identification", *Int.J.Control,* vol. 50, no. 5, pp. 1873–1896, 1989.

[12] E.S.CHNG, B.MULGREW, and S.CHEN, "Backtracking Orthogonal Least Squares Algorithm for model selection", *IEE Maths Colloquium on Mathematical Aspects of Digital Signal Processing,* vol. 1994/034, pp. 10/2–10/5, Bristol, 1994.

[13] E.S.CHNG, S.CHEN, and B.MULGREW, "Efficient computational schemes for the orthogonal least squares algorithm", *IEEE Trans. Signal Processing,* vol. 43, no. 1, pp. 373-376, 1995.

[14] E.S.CHNG, S.CHEN, and B.MULGREW, "Reducing the Computational Requirement of the Orthogonal Least Squares Algorithm", *Proc. IEEE ICASSP,* vol. 3, pp. 529–532, 1994.

[15] E.S.CHNG, S.CHEN, and B.MULGREW, *Gradient radial basis function network for nonlinear and nonstationary time series prediction.* IEEE Trans. Neural Network, To appear : Nov 1995.

[16] E.S.CHNG, S.CHEN, and B.MULGREW, "Improving the radial basis function network for homogeneous nonstationary time series prediction", *Proc. of the Seventh European Signal Processing Conference*, pp. 1819–1822, 1994.

[17] S.CHEN, B.MULGREW, and P.M.GRANT, "A clustering technique for digital communications channel equalization using radial basis function networks", *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 570–579, 1993.

[18] S.CHEN, B.MULGREW, and S.McLAUGHLIN, "Adaptive Bayesian equalizer with decision feedback", *IEEE Trans. Signal Processing*, vol. 9, no. 41, pp. 2918–2927, 1993.

[19] E.S.CHNG, B.MULGREW, S.CHEN, and G.GIBSON, *Optimum lag and subset model selection for the raidial basis function equaliser.* IEEE Workshop on Neural Networks for Signal Processing (Boston 95), 1995.

[20] S.CHEN, E.S.CHNG, B.MULGREW, and G.GIBSON, On decision feedback equaliser. Submitted to *IEEE Trans. Communications*, 1995.

[21] E.S.CHNG, B.MULGREW, S.CHEN, and G.GIBSON, *Realising the Bayesian decision boundary for channel equalisation using RBF network and linear network.* Mathematics for neural networks and applications, (Jul 95), Oxford, To appear : 1995.

[22] D.A.RATKOWSKY, *Nonlinear regression modeling : A unified practical approach.* MARCEL DEKKER, INC *Statistics: Textbooks and Monographs*, 1983.

[23] D.A.RATKOWSKY, *Handbok of nonlinear regression models.* MARCEL DEKKER, INC *Statistics: Textbooks and Monographs*, 1990.

[24] R.J.BROOK and G.C.ARNOLD, *Applied regression analysis and experimental design.* MARCEL DEKKER, INC *Statistics: Textbooks and Monographs*, 1985.

[25] D.S.BOROWIAK, *Model discrimination for nonlinear regression models.* MARCEL DEKKER, INC *Statistics: Textbooks and Monographs*, 1989.

[26] J.H.FRIEDMAN, "Multivariable adaptive regression splines", *Annals Stat.*, vol. 19, pp. 1–142, 1991.

[27] S.CHEN and S.A.BILLINGS, "Representation of non-linear systems: the NARMAX model", *Int.J.Control*, vol. 49, no. 3, pp. 1013–1032, 1989.

[28] G.U.YULE, "On a method of investigating periodicities in disturbed series with special reference to Wolfer's sunspot numbers", *Philios. Transcation Royal Society*, vol. A226, pp. 267–298, 1927.

[29] G.E.P.BOX and G.M.JENKINS, *Time Series Analysis : Forecasting and Control.* Holden-Day, 1976.

[30] G.H.GOLUB and C.REINSCH, "Singular value decomposition and least squares solutions", *Numerische Math.*, vol. 14, pp. 403–420, 1970.

[31] G.H.GOLUB and C.F.VAN LOAN, *Matrix Computations (2nd Edition).* The John Hopkins University Press, Baltimore, MD, 1989.

[32] V.KLEMA and A.LAUB, "The singular value decomposition : its computation and some applications", *IEEE Trans. AC*, vol. 25, no. 2, pp. 164–176, 1980.

[33] V.VOLTERRA, *Theory of Functionals.* Blackie, 1930.

[34] P.J.W.RAYNER and M.R.LYNCH, "A new connectionist model based on a non-linear adaptive filter", *Proc. IEEE ICASSP*, vol. 2, pp. 1191–1194, 1989.

[35] J.CHEN and J.VANDEWALLE, "Study of adaptive echo canceller with Volterra expansion", *Proc. IEEE ICASSP*, vol. 2, pp. 1376–1379, 1989.

[36] S.CHEN, P.M.GRANT, and C.F.N.COWAN, "Orthogonal least-squares algorithm for training multi-output radial basis function networks", *IEE Proc*, vol. 139, Part F, no. 6, pp. 378–384, Dec 1992.

[37] I.J.LEONTARITIS and S.A.BILLINGS, "Input-output parametric modls for nonlinear-system. Part I: deterministics nonlinear systems", *Int.J.Control*, vol. 41, pp. 303–344, 1987.

[38] I.J.LEONTARITIS and S.A.BILLINGS, "Input-output parametric modls for nonlinear-system. Part II: stochastic non-linear systems", *Int.J.Control*, vol. 45, pp. 311–341, 1987.

[39] P.A.LEWIS, B.K.RAY, and J.G.STEVENS, *Modeling time series by using multivariate adaptive regression splines (MARS)*. Time series prediction : forecasting the future and understanding the past, Eds. A.S.WESIGEND and N.A.GERSHENFELD,Addison-Wesley, 1994.

[40] L.WANG and J.M.MENDEL, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning", *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 807–814, 1992.

[41] E.BIGLIERI, "Theory of Volterra processors and some applications", *Proc. IEEE ICASSP*, pp. 294–297, 1982.

[42] N.WIENER, *The extrapolation, interpolation and smoothing of stationary time series with engineering applications.* Wiley, 1949.

[43] B.PICINBONO, "Quadratic filters", *Proc. IEEE ICASSP*, pp. 298–301, 1982.

[44] T.POGGIO and F.GIROSI, "Networks for approximation and learning", *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.

[45] J.PARK and I.W.SANDBERG, "Universal approximation using radial basis function networks", *Neural Computation*, vol. 3, pp. 246–257, 1991.

[46] M.A.S.POTTS and D.S.BROOMHEAD, "Time series prediction with a radial basis function neural network", *SPIE Adaptive Signal Processing*, vol. 1565, pp. 255–266, 1991.

[47] C.A.MICCHELLI, "Interpolation of scattered data: distance matrices and conditionally positive definite functions", *Construct. Approx.*, vol. 2, pp. 11–22, 1986.

[48] S.CHEN, S.A.BILLINGS, and P.M.GRANT, "Recursive hybrid algorithm for nonlinear system identification using radial basis function networks", *Int.J.Control*, vol. 55, no. 5, pp. 1051–1070, 1992.

[49] S.CHEN and S.A.BILLINGS, "Modelling and analysis of nonlinear time series", *Int.J.Control*, vol. 50, no. 6, pp. 2151–2171, 1989.

[50] R.R.HOCKING, "The analysis and selection of variables in linear regression", *Biometrics*, vol. 32, pp. 1–49, 1976.

[51] A.J.MILLER, *Subset Selection in Regression.* Chapman and Hall, 1990.

[52] M.J.KORENBERGH, "A robust orthogonal algorithm for system identification and time series analysis", *Biological Cybernetics*, vol. 60, pp. 267–276, 1989.

[53] S.CHEN, S.A.BILLINGS, C.F.N.COWAN, and P.M.GRANT, "Nonlinear system identification using radial basis functions", *Int.J.Systems Sci.*, vol. 21, no. 12, pp. 2513–2539, 1990.

[54] S.CHEN, S.A.BILLINGS, C.F.N.COWAN, and P.M.GRANT, "Practical identification of NARMAX models using radial basis functions", *Int.J.Control*, vol. 52, no. 6, pp. 1327–1350, 1990.

[55] A.JENNINGS, *Matrix Computation for Engineers and Scientists.* John Wiley & Sons, 1977.

[56] R.A.MEESE and A.K.ROSE, "Nonlinear, nonparametric, nonessential exchange rate estimation", *American Economic Review*, vol. 80, no. 2, pp. 192–196, 1990.

[57] B.LeBARRON, *Nonlinear diagnostics and simple trading rules for high frequency foreign exchange rates.* Time series prediction : forecasting the future and understanding the past, Eds. A.S.WESIGEND and N.A.GERSHENFELD,Addison-Wesley, 1994.

[58] C.W.J.GRANGER, *Forecasting in economics.* Time series prediction : forecasting the future and understanding the past, Eds. A.S.WESIGEND and N.A.GERSHENFELD,Addison-Wesley, 1994.

[59] M.M.GABR and T.S.RAO, "The estimation and prediction of subset bilinear time series models with applications", *J.Time Series Analysis*, vol. 2, pp. 155–171, 1981.

[60] H.TONG, *Non-linear time series : A dynamical system approach.* Oxford University Press, 1990.

[61] M.CASDAGLI, "Nonlinear prediction of chaotic time-series", *Physica D*, vol. 35, pp. 335–356, 1989.

[62] N.A.GERSHENFELD and A.S.WEIGEND, *The future of time series: learning and understanding.* Time series prediction : forecasting the future and understanding the past, Eds. A.S.WESIGEND and N.A.GERSHENFELD,Addison-Wesley, 1994.

[63] C.W.J.GRANGER and A.P.ANDDERSEN, *An introduction to bilinear time series models.* GOTTINGEN: Vandenhoek and Ruprecht, 1978.

[64] T.OZAKI, "Nonlinear threshold autoregressive models for nonlinear random vibrations", *Journal of Applied Probabilities*, vol. 18, pp. 443–451, 1981.

[65] H.TONG, *Threshold models in non-linear time seres analysis.* Lecture notes in Statistics, No 21, Springer, Heidelberg, 1983.

[66] K.HORNIK, M.STINCHCOMBE, and H.WHITE, "Universal approximation of an unknown mapping and its derivative using multilayer feedforward networks", *Neural Network*, vol. 3, no. 5, pp. 551–560, 1990.

[67] N.H.PACKARD, J.P.CRUTCHFIELD, J.D.FARMER, and R.S.SHAW, "Geometry from a time series", *Phys. Rev. Lett.*, vol. 45, no. 9, pp. 712–716, 1980.

[68] F.TAKENS, "Detecting strange attractors in turbulence", *Lecture Notes in Mathematics*, pp. 366–381, Springer-Verlag, New York, 1980.

[69] T.SAUER, *Time series prediction by using delay coordinated embedding.* Time series prediction : forecasting the future and understanding the past, Eds. A.S.WESIGEND and N.A.GERSHENFELD,Addison-Wesley, 1994.

[70] M.MACKEY and L.GLASS, "Oscillation and chaos in physiological control systems", *Science*, vol. 197, pp. 287–289, 1977.

[71] J.G.PROAKIS, *Digital Communications, (2nd edition).* McGraw-Hill Book Company, 1989.

[72] A.P.CLARK, *Equalisers for digital modems.* Pentech Press, 1985.

[73] I.CHA ans S.A.KASSAM, "Channel equalisation using adaptive complex radial basis function networks", *IEEE Journal On Selected Area In Communications*, vol. 13, no. 1, pp. 122–131, 1995.

[74] S.CHEN, S.McLAUGHLIN, B.MULGREW, and P.M.GRANT, "Adaptive Bayesian decision feedback equaliser for dispersive mobile radio channels", *IEEE Trans. Communications*, vol. 43, no. 5, pp. 1937–1946, 1995.

[75] S.CHEN, S.McLAUGHLIN, and B.MULGREW, "Complex-valued radial basis function networks, Part II: Application to digital communications channel equalisayion"", *Signal Processing*, vol. 36, pp. 175–188, 1994.

[76] G.D.FORNEY, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference", *IEEE Trans. Information Theory*, vol. IT-18, no. 3, pp. 363–378, 1972.

[77] G.D.FORNEY, "The Viterbi algorithm", *Proc. IEEE*, vol. 61, pp. 268–278, 1973.

[78] M.V.EYUBOGLU and S.U.H.QURESHI, "Reduced-state sequence estimation for coded modulation on intersymbol interference channels", *IEEE Journal On Selected Area In Communications*, vol. 7, pp. 989–995, 1989.

[79] W.H.SHEEN and G.L.STUBER, "MLSE equalisation and decoding for multipath-fading channels", *IEEE Trans. Communications*, vol. 39, pp. 1455–1464, 1991.

[80] H.KUBO, K.MURAKAMI, and T.FUJINO, "Adaptive maximum-likelihood sequence estimation by means of combined equalisation and decoding in fading environment", *IEEE Journal On Selected Area In Communications*, vol. 13, no. 1, pp. 102–109, 1995.

[81] P.R.CHEVILLAT and E.ELEFTHERIOU, "Decoding of trellis-encoded signals in the presence of intersymbol interefence and noise", *IEEE Trans. Communications*, vol. 37, pp. 669–676, 1989.

[82] D.WILLIAMSON, R.A.KENNEDY, and G.W.PULFORD, "Block decision feedback equalisation", *IEEE Trans. Communications*, vol. 40, pp. 255–264, 1992.

[83] M.E.AUSTIN, *Decision-feedback equalisation for digital communication over dispersive channels.* MIT Lincoln Lab, Lexington,MA., Tech. Rep 437, Aug 1967.

[84] C.A.BELFIORE and Jr J.H.PARK, "Decision feedback equalisation", *Proc. IEEE*, vol. 67, pp. 1143–1156, 1979.

[85] J.SALZ, "Optimum mean-square decision feednack equalisation", *Bell System Technical Journal*, vol. 52, pp. 1341–1373, 1973.

[86] R.W.LUCKY, "Automatic equlisation for digital communication", *Bell System Technical Journal*, vol. 44, pp. 547–588, 1965.

[87] E.A.LEE and D.G.MESSERSCHMITT, *Digital Communication.* Kluwer Academic Publishers, 1988.

[88] B.WIDROW and Jr M.E.HOFF, *Adaptive switching circuits.* IRE WESCON Conv. Rec., pp 96-104, Pt. 4, 1970.

[89] R.D.GITLIN and S.B.WEINSTEIN, "Fractionally-spaced equalisation: An improved digital transversal equaliser", *Bell System Technical Journal*, vol. 60, pp. 275–296, 1981.

[90] G.UNGERBOECK, "Fractional tap-spacing equaliser and consequences fir clock recovery in data modems", *IEEE Trans. Communications*, vol. 24, pp. 856–864, 1976.

[91] C.R.JOHNSON Jr, "Adaptive IIE filtering: current results and open issues", *IEEE Trans. Information Theory*, vol. 30, pp. 237–250, 1984.

[92] B.MULGEW and C.F.N.COWAN, "An adaptive IIR Equaliser: A Kalman filter approach", *Proc. IEEE ICASSP*, pp. 2955–2958, 1986.

[93] A.AHO, J.E.HOPCROFT, and J.D.ULLMAN, *Data Structures and Algorithms.* Addison-Wesley, 1983.

[94] R.A.HORN and C.R.JOHNSON, *Matrix Analysis.* Cambridge University Press, 1985.

[95] P.COMON and G.H.GOLUB, "Tracking a few extreme singular values and vectors in signal processing", *Proc. IEEE*, vol. 78, pp. 1327–1343, 1990.

[96] E.LEVIN, "Hidden control neural architecture modelling of nonlinear time varying systems and its applications", *IEEE Trans. Neural Networks*, vol. 4, no. 1, pp. 109–116, 1993.

[97] S.CHEN, G.J.GIBSON, and C.F.N.COWAN, "Adaptive channel equalisation using a polynomial-perceptron structure", *IEE Proc*, vol. 137, Part I, no. 5, pp. 257–264, 1990.

[98] P.M.GRANT and B.MULGREW, "Nonlinear adaptive filters: design and applications", *To appear in : 5th IFAC symposium on adaptive systems in control and signal processing*, pp. ?, Jun, 1995.

[99] G.J.GIBSON, S.SIU, and C.F.N.COWAN, "The application of nonlinear structures to the reconstruction of binary signals", *IEEE Trans. Signal Processing*, vol. 39, no. 8, pp. 1877–1884, 1991.

[100] G.J.GIBSON, S.SIU, and C.F.N.COWAN, "Multi-layer perceptron structures applied to adaptive equalisers for data communications", *Proc. IEEE ICASSP*, pp. 1183–1186, 1989.

[101] N.W.K.LO and H.M.HAFEZ, "Neural Network channel equalization", *IJCNN 1992*, vol. 2, pp. 981–986, 1989.

[102] C.P.CALLENDER and C.F.N.COWAN, "A comparison of six different non-linear equalisation techniques for digital communication systems", *Proc. of the Seventh European Signal Processing Conference*, pp. 1524–1527, 1994.

[103] B.MULGREW and C.F.N.COWAN, "Equalisation techqniues using non-linear adaptive filters", *Adaptive Algorithms: Applications and non-classical schemes*, pp. ?, Eds. D.DOCAMPO and A.R.FIGUERAS, Publications da Universidade de Vigo,1991.

[104] R.O.DUDA and P.E.HART, *Pattern classification and scene analysis*. Wiley, 1973.

[105] K.ABEND and B.D.FRITCHMAN, "Statistical detection for communication channels with intersymbol interference", *Proc. IEEE*, vol. 58, no. 5, pp. 779–785, 1970.

[106] B.HUGHES, "On the error probability of signals in additive white Gaussian noise", *IEEE Trans. Information Theory*, vol. 37, no. 1, pp. 151–155, 1991.

[107] R.A.ILTIS, "A randomized bias technique for the importance sampling simulation of Bayesian equalisers", *IEEE Trans. Communications*, vol. 43, pp. 1107–1115, 1995.

[108] J.J.O'REILLY and A.M.de OLIVERIRA DUARTE, "Error probagation in decision feedback receivers", *IEE Proc*, vol. 132, Part F, no. 132, pp. 561–566, 1985.

[109] D.L.DUTWEILLER, J.E.MAZO, and D.G.MESSERSCHMITT, "An upper bound on the error probability in decision-feedback equaliser", *IEEE Trans. Information Theory*, vol. IT-20, pp. 490–497, 1974.

[110] R.A.KENNEDY and B.D.O.ANDERSON, "Recovery time of decision feedback equaliser on noiseless channels", *IEEE Trans. Communications*, vol. 35, pp. 1012–1021, 1987.

[111] R.A.KENNEDY, B.D.O.ANDERSON, and R.R.BITMEAD, "Tight bounds on the error probabilities of decision feedback equaliser", *IEEE Trans. Communications*, vol. 35, pp. 1022–1028, 1987.

[112] S.SIU, G.J.GIBSON, and C.F.N.COWAN, "Decision feedback equalisation using neural network structures", *Proc. 1st IEE Int. Conference on Artificial Neural Networks (London)*, pp. 125–128, 1989.

[113] D.M.HIMMELBLAU, *Applied Nonlinear Programming*. McGraw-Hill Book Company, 1972.

[114] R.E.WILLIAMSON, R.H.CROWELL, and H.F.TROTTER, *Calculus of vector functions (2nd Ed.)*. Prentice Hall, 1968.

[115] T.BANCHOFF and J.WERMER, *Linear algebra through geomery*. Springer-Verlag, 1983.

Appendix A

Original publications

The work described in this thesis has been reported in the following publications:

(i). †E. S. CHNG, S. CHEN and B. MULGREW, "Efficient computational schemes for the orthogonal least squares algorithm", in *IEEE Trans. Signal Processing*, vol 43, no 1, pg 373-376.

(ii). E. S. CHNG, S. CHEN and B. MULGREW, "Reducing the computational requirement of the orthogonal least squares algorithm", in *IEEE ICASSP 1994*, Adelaide, Australia, vol 3, pg 529-532.

(iii). †E. S. CHNG, S. CHEN and B. MULGREW, "Improving the radial basis function networks for homogeneous nonstationary time series prediction", in *European signal processing conference VII 94 (Sep 1994)*, Edinburgh, pg 1819-1822.

(iv). E. S. CHNG, B. MULGREW and S. CHEN, "Backtracking orthogonal least squares algorithm for model selection", in *IEE Colloquium on Mathematical Aspects of Digital Signal Processing*, Feb. 1994, Bristol. Digest No 1994/034, pg 10/1-10/5.

(v). S. CHEN, E. S. CHNG and K. ALKAHIMI, "Schemes for improving generalisation properties of the radial basis function network", in *14th IMACS World Congress Computational and Applied Mathematics*, Atlanta (Jul 1994), pg 86-88.

Papers to appear:

(i). E. S. CHNG, S. CHEN and B. MULGREW, "Gradient radial basis function for nonlinear and nonstationary time series prediction", to appear in Nov 1995, *IEEE Trans. Neural Networks*.

(ii). † E. S. CHNG, B. MULGREW, S. CHEN and G. GIBSON, "Optimum Lag and Subset Selection for Radial Basis Function Equaliser", to appear in *IEEE Workshop on Neural Networks for Signal Processing (NNSP'95)*, Boston.

(iii). E. S. CHNG, B. MULGREW, S. CHEN and G. GIBSON, "Realising the Bayesian decision boundary for channel equalisation using Radial Basis Function network and linear equaliser", to appear in *Mathematics of Neural Networks and Applications*, (Jul 95), Oxford.

Papers submitted:

- S. CHEN, E. S. CHNG, B. MULGREW and G. GIBSON, "On decision feedback equaliser", *IEEE Trans. Communications.*
- S. CHEN, E. S. CHNG, B. MULGREW and G. GIBSON, "Minimum-BER Linear combiner DFE", *ICC 96.*

† Reprinted in this appendix.

A.1 Publication appeared in - E. S. CHNG, S. CHEN and B. MULGREW, "Efficient computational schemes for the orthogonal least squares algorithm", in *IEEE Trans. Signal Processing*, vol 43, no 1, pg 373-376.

Efficient computational schemes for the orthogonal least squares algorithm

E.S. CHNG †, S. CHEN ‡ and B. MULGREW †

Abstract— **The orthogonal least squares (OLS) algorithm [1,2] is an efficient implementation of the forward selection [3] method for subset model selection. The ability to find good subset parameters with only a linearly increasing computational requirement makes this method attractive for practical implementations. In this correspondence, we examine the computational complexity of the algorithm and present a pre-processing method for reducing the computational requirement.**

Keywords—**OLS, nonlinear model, RBF model,volterra model.**

I. Introduction

Nonlinear predictors generated using radial basis functions (RBF) [2,4], fuzzy basis functions [5] or Volterra expansions [6] normally results in the formation of very large initial models that have the *linear-in-parameter* characteristic (figure 1). Such large initial models can normally be reduced to a much smaller parsimonious model without significant degradation in prediction performance if the subset model's parameters are chosen carefully.

To find the optimum R-parameter subset model from an original K-parameter model, it is required to calculate the performance of all the possible R-parameter subset models from the original K-parameter system and choose the best one. This requires prohibitively large amount of computation and is thus not practical.

One applicable method of subset model selection for models with the linear in parameter characteristic is the forward-selection search [3]. This method, however, has been criticised for not guaranteeing to achieve the optimum solution. Although the criticism is valid, subset models found using the forward-selection search is generally good enough for practical applications. Examples can be found in the papers describing the OLS algorithm [1,2] and Korenbergh's fast orthogonal search [7]; these two methods are derivatives of the forward-selection technique.

II. OLS Algorithm

Let us represent these nonlinear predictors that have the linear in parameter structure as a linear regression model:

$$\mathbf{y} = \mathbf{X}\mathbf{h} + \mathbf{e} \tag{1}$$

where $\mathbf{y}$ is the desired signal vector, $\mathbf{X}$ is the information matrix of size $N \times K$, $\mathbf{h}$ is the parameter vector of the model and $\mathbf{e}$ the error vector of approximating $\mathbf{y}$ by $\mathbf{Xh}$. The column vectors $\mathbf{y}$ and $\mathbf{e}$ contain N elements, that is, there are N data samples and N values of error.

The original $\mathbf{X}$ matrix have K columns. To create a parsimonious model which has R parameters, we are actually trying to pick R columns from the input matrix $\mathbf{X}$ to form a subset input matrix $\mathbf{X}s$. The OLS algorithm selects columns from the input matrix sequentially. At each selection, all the unused columns are studied to determine how each column will contribute to fit the desired vector $\mathbf{y}$ with the current subset $\mathbf{X}s$. The column that provides the best combination with $\mathbf{X}s$ to model $\mathbf{y}$ will be picked to form the new $\mathbf{X}s$. The above procedure is repeated until the number of columns in $\mathbf{X}s$ equals to R. The selection procedure is made very efficient by employing orthogonalisation schemes such as the Gram-Schmidt or the Householder transformation [8]. The details of the algorithm can be found in Chen *et al* [1,2].

III. Computation reduction of OLS method

The computational requirement in applying the OLS algorithm to find subset models from an initial information matrix $\mathbf{X}$ is proportional to the size of $\mathbf{X}$. In the situation when $N \gg K$, where N and K are the numbers of rows and columns in $\mathbf{X}$ respectively, it may be possible to reduce computation requirement of the OLS by first introducing an invariant transformation on the matrix $\mathbf{X}$ and then applying the OLS on the transformed data.

This is accomplished by pre-multiplying equation (1) by an orthonormal matrix which spans the column space of $\mathbf{X}$ [8] to transform the $N \times K$ matrix $\mathbf{X}$ and the $N \times 1$ vector $\mathbf{y}$ into a $K \times K$ matrix $\tilde{\mathbf{X}}$ and a $K \times 1$ vector $\tilde{\mathbf{y}}$. This may be thought of as a pre-processing. The OLS algorithm is then employed to select subset model based on $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{y}}$.

3.1 Reduced-OLS Gram Schmidt approach

We first examine the classical Gram-Schmidt (GS) procedure [8] for generating the orthonormal matrix used for the invariant transformation. The information matrix $\mathbf{X}$ can be decomposed into the product of an $N \times K$ matrix $\mathbf{Q}$ satisfying $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ and a $K \times K$ upper triangle matrix $\mathbf{B}$, where $\mathbf{I}$ is the identity matrix of appropriate dimension. That is

$$\mathbf{X} = \mathbf{Q}\mathbf{B} \tag{2}$$

Pre-multiplying both sides of equation (1) by $\mathbf{Q}^T$ yields

$$\mathbf{Q}^T\mathbf{y} = \mathbf{B}\mathbf{h} + \mathbf{Q}^T\mathbf{e} \tag{3}$$

† Department of Electrical Engineering, The University of Edinburgh, King's Buildings, Edinburgh EH9 3JL, Scotland.
‡ Department of Electrical and Electronics Engineering, The University of Portsmouth, Anglesea Building, Anglesea Road, Portsmouth PO1 3DJ, England.

If we introduce $\tilde{\mathbf{y}} = \mathbf{Q}^T\mathbf{y}$, $\tilde{\mathbf{X}} = \mathbf{B}$ and $\tilde{\mathbf{e}} = \mathbf{Q}^T\mathbf{y}$, we can rewrite equation (3) as

$$\tilde{\mathbf{y}} = \tilde{\mathbf{X}}\mathbf{h} + \tilde{\mathbf{e}} \tag{4}$$

$\tilde{\mathbf{y}}$ and $\tilde{\mathbf{e}}$ are $K \times 1$ vectors and $\tilde{\mathbf{X}}$ is a $K \times K$ matrix. We can then apply the OLS algorithm to perform subset selection based on $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{X}}$. We call this method the reduced-OLS GS approach.

A R-term subset model found using the reduced-OLS GS approach is identical to that of applying the OLS on the original data. This is because the transformed data $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{y}}$ are created by performing a unitary transformation [8] on $\mathbf{X}$ and $\mathbf{y}$. As such transformation preserve the length of each (column) vector and the angle between two vectors, we have not lost or created any new information when we transform equation (1) into equation (4).

The amount of computation required to apply the invariant transformation by this method requires approximately $N \times K^2$ multiplications [8]. If saving in computation by using $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{y}}$ for subset selection offsets the additional computation of pre-processing, this reduced OLS approach is justified. Computational complexity of this reduced OLS algorithm for subset model selection has been analysed, and we illustrate the results using an example showing the number of multiplications needed for subset selection working on a 500×84 information matrix (figure 2). The horizontal axis of figure 2 shows the size of the selected subset model, and the vertical axis shows the number of multiplications performed by the OLS or the reduced-OLS GS algorithm to find the required subset model.

The following equation can be used to calculate the number of multiplications performed by the OLS to select a subset model of R parameters from an information matrix $\mathbf{X}$ of size $N \times K$:

$$\text{No. multiplications (OLS)} = \sum_{i=1}^{R}(3N(K-i-1)) + \sum_{i=1}^{R-1}(2N(K-i)) \tag{5}$$

The number of multiplications required to perform the pre-processing using the GS decomposition is calculated using:

$$\text{No. multiplications (GS decomposition)} = NK^2 + NK \tag{6}$$

Therefore, the total number of multiplications required to perform a subset selection by the reduced-OLS GS approach is:

$$\text{No. multiplications (reduced-OLS GS)} = NK^2 + NK + \sum_{i=1}^{R}(3K(K-i-1)) + \sum_{i=1}^{R-1}(2K(K-i)) \tag{7}$$

3.2 Reduced-OLS SVD approach

To further reduce the computational load of the OLS, we can use an approximated matrix $\hat{\mathbf{X}}$ to represent $\mathbf{X}$. We define $\mathbf{X}$ and $\hat{\mathbf{X}}$ as

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{8}$$

$$\hat{\mathbf{X}} = \mathbf{U}_\kappa\mathbf{\Lambda}_\kappa\mathbf{V}_\kappa^T \qquad \kappa < K \tag{9}$$

where the columns of $\mathbf{U}$ are the left eigenvectors, $\mathbf{\Lambda}$ is the diagonal matrix containing the singular values and the rows of $\mathbf{V}^T$ are the right eigenvectors formed by using singular value decomposition (SVD) [8] on $\mathbf{X}$. The singular values in $\mathbf{\Lambda}$ are arranged such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_K$. The $N \times \kappa$ matrix $\mathbf{U}_\kappa$ is formed by using the first κ columns of $\mathbf{U}$, the diagonal $\kappa \times \kappa$ matrix $\mathbf{\Lambda}_\kappa$ is formed by using the first κ rows and columns of $\mathbf{\Lambda}$, and the $\kappa \times K$ matrix $\mathbf{V}_\kappa^T$ is formed by using the first κ rows of $\mathbf{V}^T$. The matrix $\hat{\mathbf{X}}$ is a rank κ approximation of the matrix $\mathbf{X}$ created by the product of $\mathbf{U}_\kappa$, $\mathbf{\Lambda}_\kappa$ and $\mathbf{V}_\kappa^T$.

If $\hat{\mathbf{X}}$ is used to approximate $\mathbf{X}$, equation (1) can be approximated by

$$\mathbf{y} \approx \hat{\mathbf{X}}\mathbf{h} + \mathbf{e} \tag{10}$$

Pre-multiplying the previous equation by $\mathbf{U}_\kappa^T$, we get

$$\mathbf{U}_\kappa^T\mathbf{y} \approx \mathbf{\Lambda}_\kappa\mathbf{V}_\kappa^T\mathbf{h} + \mathbf{U}_\kappa^T\mathbf{e} \tag{11}$$

If we introduce the $\kappa \times 1$ vectors $\tilde{\mathbf{y}}_\kappa = \mathbf{U}_\kappa^T\mathbf{y}$ and $\tilde{\mathbf{e}}_\kappa = \mathbf{U}_\kappa^T\mathbf{e}$, and the $\kappa \times K$ matrix $\tilde{\mathbf{X}}_\kappa = \mathbf{\Lambda}_\kappa\mathbf{V}_\kappa^T$, equation (11) can be written as

$$\tilde{\mathbf{y}}_\kappa \approx \tilde{\mathbf{X}}_\kappa\mathbf{h} + \tilde{\mathbf{e}}_\kappa \tag{12}$$

Since the dimensions of $\tilde{\mathbf{y}}_\kappa$ and $\tilde{\mathbf{X}}_\kappa$ are smaller than those of the vector $\tilde{\mathbf{y}}$ and matrix $\tilde{\mathbf{X}}$ in equation (4), the computation requirement is further reduced when the OLS algorithm is applied. This method is only appropriate when the approximation of $\mathbf{X}$, i.e. $\hat{\mathbf{X}}$, is created by a sufficiently large rank κ, otherwise the subset model found may not be good.

IV. Results of reduced OLS methods

Computer simulation was carried out to evaluate the quality of subset models found using the reduced-OLS methods. Subset models were selected from two different 84-tap nonlinear predictors used for predicting a chaotic Mackey-Glass time series. The first predictor was created using a degree 3 and embedding-vector-length 6 Volterra expansion. The second predictor was created by combining a 6-tap linear predictor with a 78-tap RBF predictor.

For the experiment, 500 samples from the Mackey-Glass time series were used to generate the information matrix. The information matrix $\mathbf{X}$ thus had a size of 500×84. To measure the modelling quality of the predictor, the normalised mean square error (*NMSE*) was used:

$$NMSE = 10log_{10}\left(\frac{\sum_{i=1}^{N} e_i^2}{\sum_{i=1}^{N} s_i^2}\right) \tag{13}$$

where s_i is the desired signal value at sample i, and $e_i = s_i - \hat{s}_i$. From equation (13), we can see that when we have perfect prediction, i.e. $e_i = 0$ for all i, the *NMSE* will be $-\infty$ dB. When there is no prediction, i.e. $\hat{s}_i = 0, e_i = s_i$ for all i, the *NMSE* will be 0 dB.

The subset models found using the reduced-OLS GS approach were identical to those selected by the OLS using the original data. This, however, is not true for the models

2

found using the reduced-OLS SVD approach. The reason is that the reduced-OLS SVD scheme selected subset models based on $\tilde{\mathbf{X}}_\kappa$ and $\tilde{\mathbf{y}}_\kappa$, which are approximations of the original data.

Figure 3 depicts the predictive performance of subset models selected from the Volterra predictor. The results show that when approximation rank $\kappa = 40$ is used, subset models selected with sizes less than 22 have almost equivalent performance to those selected using the full model. This suggests that information regarding the first 22 significant regressors were not lost when we approximated the rank 84 matrix $\mathbf{X}$ by the rank 40 matrix $\tilde{\mathbf{X}}_R$. When an approximation rank 60 is used, there is hardly any difference between the subset models selected by the reduced-OLS SVD algorithm and those chosen by the OLS algorithm using the original data.

Similar results were also found for subset models generated from the second nonlinear predictor (figure 4). That is, subset models found with large approximation rank κ have very similar predictive performance characteristics to those selected using the original data.

V. Conclusions

A method of reducing computational requirement of the OLS subset model selection algorithm has been presented. This reduction is significant when the number of rows in the information matrix $\mathbf{X}$ is significantly larger than the number of its columns. Two schemes of the reduced-complexity OLS method have been proposed. The first scheme is based on a Gram-Schmidt pre-processing and will provide identical results to those obtained using the original input matrix and the desired output vector. For the second scheme based on a SVD pre-processing, it has been shown that we can always trade in subset selection performance for computational complexity.

Acknowledgements

We would like to thank David Hughes for providing the Mackey-Glass data used in the simulations.

References

[1] S.CHEN, S.A.BILLINGS, and W.LUO, "Orthogonal least squares methods and their application to nonlinear system identification," *Int.J.Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[2] S.CHEN, C.F.COWAN, and P.M.GRANT, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.

[3] R.R.HOCKING, "The analysis and selection of variables in linear regression," *Biometrics*, vol. 32, pp. 1–49, 1976.

[4] M.J.D.POWELL, "Radial basis functions for multivariable interpolation: a review," *Algorithms for Approximation*, pp. 143–167, J.C.MASON and M.G.COX (Eds), Oxford, 1987.

[5] L.WANG and J.M.MENDEL, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 807–814, 1992.

[6] P.ALPER, "A consideration of the discrete Volterra series," *IEEE Trans. AC*, vol. AC-10, pp. 322–327, 1965.

[7] M.J.KORENBERGH, "A robust orthogonal algorithm for system identification and time series analysis," *Biological Cybernetics*, vol. 60, pp. 267–276, 1989.

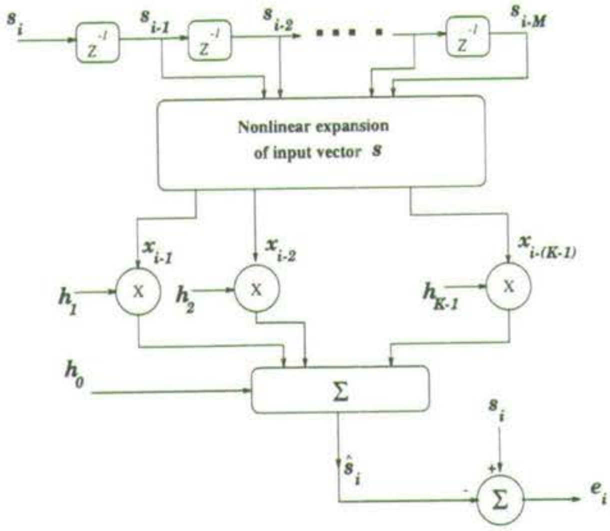

Fig. 1: Nonlinear predictor of order K

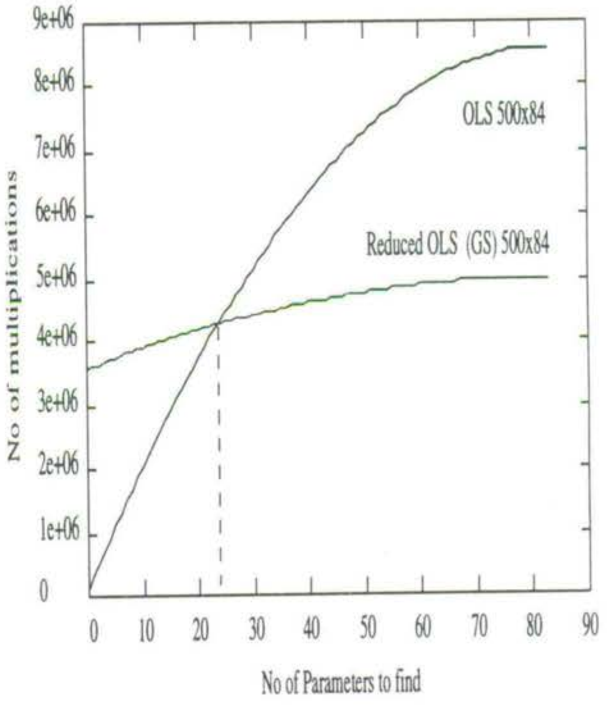

Fig. 2: Computation requirement for X matrix of size 500x84.

[8] G.H.GOLUB and C.F.VAN LOAN, *Matrix Computations (2nd Edition)*. The John Hopkins University Press, Baltimore, MD, 1989.

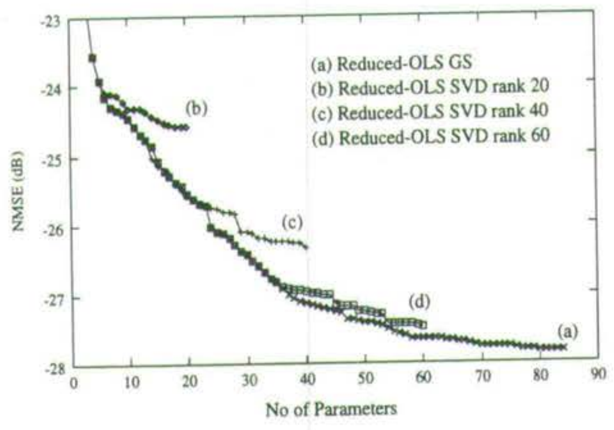

Fig. 3: Performance of subset model found using reduced-OLS on the Volterra predictor.

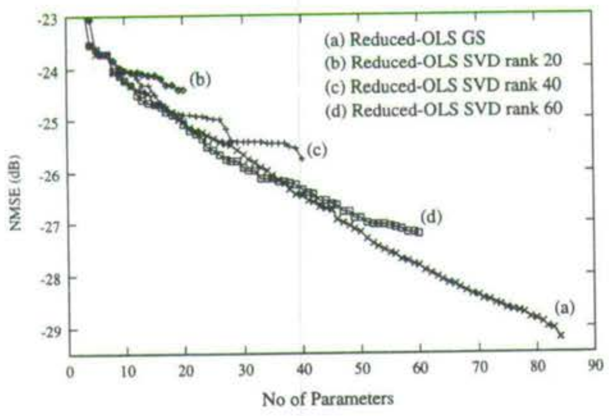

Fig. 4: Performance of subset model found using reduced-OLS on the RBF predictor.

A.2 Publication appeared in - E. S. CHNG, S. CHEN and B. MULGREW, "Improving the radial basis function networks for homogeneous nonstationary time series prediction", in European signal processing conference VII 94 (Sep 1994), Edinburgh, pg 1819-1822.

Improving the Radial Basis Function Networks for Homogeneous Nonstationary Time Series Prediction

Eng-Siong CHNG †, Sheng CHEN ‡ and Bernard MULGREW †

† *Dept. of Electrical Eng., The University of Edinburgh, Edinburgh EH9 3JL, Scotland, U.K., Tel/Fax: +44 [31] 650 5655 / 650 6554. E-Mail: echng@ee.ed.ac.uk*

‡ *Dept. of Electrical and Electronic Eng., The University of Portsmouth, Anglesea Building, Portsmouth PO1 3DJ, U.K. E-Mail: schen@ee.port.ac.uk*

Abstract. The radial basis function (RBF) network has become a popular choice of neural network to be used for nonlinear time series prediction [1–3]. Although the results have been encouraging for modelling time invariant nonlinear systems, it is difficult to achieve the same level of success for tracking nonstationary signals [4]. In this article, we present a method of modifying the classical RBF networks, which improves the predictive accuracy for nonlinear and nonstationary data.

1 Introduction

Although the RBF network has achieved considerable success in the application to stationary nonlinear time series prediction, it is unable to achieve the same level of success for tracking nonstationary series. This is because the RBF network, like many other neural network models, does not characterise temporal variability well. Since real-world signals are often not only nonlinear but also nonstationary, it is desired to develop predictors which can handle signals that exhibit both such characteristics.

To improve the predictive performance for nonstationary data, we propose a gradient RBF (GRBF) network which is a modification of the classical RBF network. In the classical RBF network, the centers of the hidden nodes can be interpreted as prototype vectors which are used to sense the presence of the input pattern. That is, if a center matches the network input vector, the corresponding hidden node will fire strongly. While in the GRBF network, a hidden node's function is to sense the presence of a prototype vector's gradient. This significantly improves the predictive capability of the network in the situation where nonstationarity of the signal is due to the variations of mean and trend.

In using this GRBF network, we are exploiting the idea that, by performing a suitable difference operation on a nonstationary signal, the resulting signal becomes stationary. This idea is used in the auto-regressive integrated moving average (ARIMA) model [5] for linear prediction of nonstationary signals. By incorporating a similar mechanism into the RBF network, we can create a network model that is capable of dealing with nonlinear and nonstationary signals.

2 The Gradient Radial Basis Function Network

The GRBF network, like the RBF network, is a single-hidden-layer feedforward neural network [3]. It consists of a input layer with M input elements, a hidden layer with K hidden nodes and, in this study, an output layer with 1 node. There are however two main differences between the RBF network and the GRBF case.

Firstly, the input vector to the RBF network contains past samples of the time series $\{y_i\}$ while the input vector to the GRBF network is generated by differencing the raw data $\{y_i\}$. The order of differencing determines the order of the GRBF network. For example, if the input vector to the RBF network at time i is given by

$$\mathbf{x}_i = [y_{i-1}, y_{i-2}, \cdots, y_{i-M}]^T \qquad (1)$$

, then the input vector of the 1st-order GRBF network at time i is

$$\begin{aligned} \mathbf{x}'_i &= \mathbf{x}_i - \mathbf{x}_{i-1} \\ &= [y_{i-1} - y_{i-2}, \cdots, y_{i-M} - y_{i-M-1}]^T \qquad (2) \end{aligned}$$

The elements of $\mathbf{x}'_i$ show the rate of change in the time-series trajectory for the past M samples.

1

Secondly, the function of the hidden node for the GRBF network is different from that of the RBF network. Figure 1 depicts the structure of the 1st-order GRBF network. Although the Gaussian function still serves as the nonlinear function which compares the similarity of the input vector to the hidden node's center, the response of the Gaussian function is now multiplied by an additional term $(y_{i-1} + \delta)$. The response of the j-th hidden node of a 1st-order GRBF network to the input vector $\mathbf{x}'_i$ is therefore given by

$$\phi'_{ij} = exp(-\alpha\|\mathbf{x}'_i - \mathbf{c}'_j\|) \times (y_{i-1} + \delta_j) \qquad (3)$$

where $\mathbf{c}'_j$ is the M-dimensional center vector of the j-th hidden node, α is a width parameter, and δ_j is a constant value associated with the center.

The term $(y_{i-1} + \delta_j)$ can be interpreted as a local single-step prediction of y_i by the j-th hidden node. From (3), if the input vector is similar to the j-th center, the value of the Gaussian function will be close to 1.0 and the predictor $(y_{i-1} + \delta_j)$ becomes fully active. As in the case of the RBF network, the output layer is a linear combiner with weights h_j, $1 \leq j \leq K$. Similar to the selection of RBF centers, $\mathbf{c}'_j$ and δ_j, $1 \leq j \leq K$, can be selected during training from the training data set $\{\mathbf{x}'_k\}_{k=1}^N$, where N is the number of training data. For each training input vector $\mathbf{x}'_k$, define $d_k = y_k - y_{k-1}$. If $\mathbf{x}'_k$ is chosen as the j-th center $\mathbf{c}'_j$, the values of δ_j is set to d_k. This ensures that the j-th hidden node is a perfect predictor of y_k.

The rationale behind the GRBF model become obvious when the network performs predictive operation. Each hidden node compares the network input vector $\mathbf{x}'_i$ with its center $\mathbf{c}'_j$. The Gaussian response of each hidden node indicates the degree of matching between $\mathbf{x}'_i$ and $\mathbf{c}'_j$. The hidden nodes thus sense the gradient of the time series rather than the series itself as in the case of the RBF model. The term $(y_{i-1} + \delta_j)$ also has a clear geometric meaning; if the j-th center $\mathbf{c}'_j$ matches the gradient $\mathbf{x}'_i$ of the series, $(y_{i-1} + \delta_j)$ is likely to be a very good prediction of y_i. Although the complexity of a GRBF hidden node is greater than that of a RBF hidden node, the GRBF has better generalisation property, particularly in predicting nonstationary time series. This often results in a smaller GRBF network. Therefore, the overall complexity of the GRBF network may not necessarily be greater than that of the RBF network in practical applications.

3 Simulation Results

We present some simulation results of time series prediction using the RBF and GRBF predictors. Initial full models were created by using all the available data in the training set as RBF and/or GRBF centers. Some linear terms were also included into the full models. Subset models were then selected from these large full models using the OLS [2] scheme, and used to evaluate single-step and multi-step prediction performance.

3.1 Results for Stationary Series

The Mackey-Glass (figure 2) chaotic time-series was used to evaluate model predictive performance. Data samples of point 100-600 were used as the training set and samples 601 to 1100 were used as the validation set. The values of M was chosen to be 6, and the width of Gaussian function was set to $\alpha = 1.0$. The following types of models were used:

i) **L-model** - The linear model of order 50.

ii) **L0-model** - A combination of the linear model and the classical RBF model.

iii) **L01-model** - A combination of the linear model, the classical RBF and 1st-order GRBF models.

The results of single-step performance for the predictors in training phase are shown in figure 3, where the vertical axis indicates the normalised mean square error (NMSE) in dB. As expected, as the size of each selected subset model increases, the accuracy of the model continued to improve. However, the rate of improvement was not the same for each model. The predictors with GRBF expansion, i.e. **L01-model**, achieved better error reduction with a smaller model size. This GRBF subset model also performed better on the validation set compared with the linear and classical RBF models, as can be seen in figure 4.

3.2 Results for Nonstationary Series

To examine how the predictors behave for nonstationary series, we used a modified Mackey-Glass time-series (figure 5). This new series was formed by adding sinusoid with amplitude 0.3 and a period of 3000 samples to the Mackey-Glass time series used in the previous example. As the training data were formed from samples 100 to 600 and the validation data consisted of samples from 601-1100, the predictors were trained without being exposed to the change in the level and trend of the test data. The results for the single-step prediction in the validation phase (figure 6) suggest that the GRBF network can perform better than the classical RBF network in a nonstationary environment.

4 Conclusions

We have presented a GRBF network for nonlinear and nonstationary time series prediction. The hidden layer of this GRBF network is designed to respond to the gradient of time-series rather than the trajectory itself. This can usually improve predictive accuracy, particularly for homogeneous nonstationary time series as are demonstrated in the simulation results. Although the discussion was based on time series prediction, this GRBF network can be applied to other signal processing applications.

References

[1] T.POGGIO and F.GIROSI, "Networks for approximation and learning", *Proc. IEEE*, vol. 78, pp. 1481–1497, 1990.

[2] S.CHEN, C.F.COWAN, and P.M.GRANT, "Orthogonal least squares learning algorithm for radial basis function networks", *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, 1991.

[3] D.S. BROOMHEAD and D.LOWE, "Multivariable functional interpolation and adaptive networks", *Complex Systems*, vol. 2, pp. 321–355, 1988.

[4] E.LEVIN, "Hidden control neural architecture modelling of nonlinear time varying systems and its applications", *IEEE Trans. Neural Networks*, vol. 4, pp. 109–116, 1993.

[5] G.E.P.BOX and G.M.JENKINS, *Time Series Analysis : Forecasting and Control*, Holden-Day, 1976.

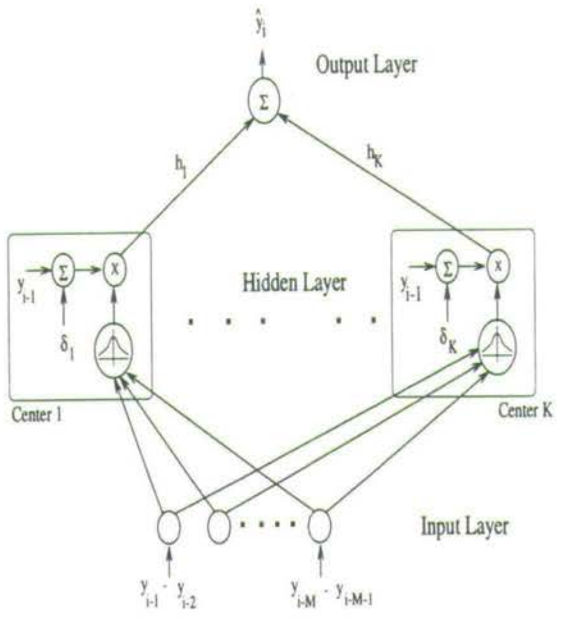

Figure 1: Topology of 1st-order GRBF network

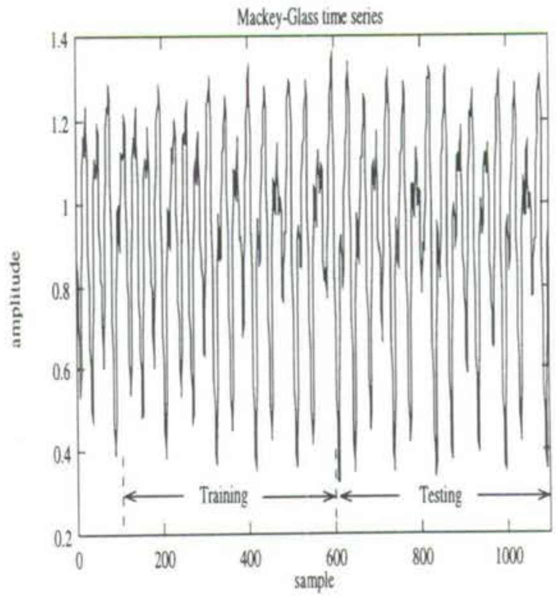

Figure 2: Mackey-Glass time series

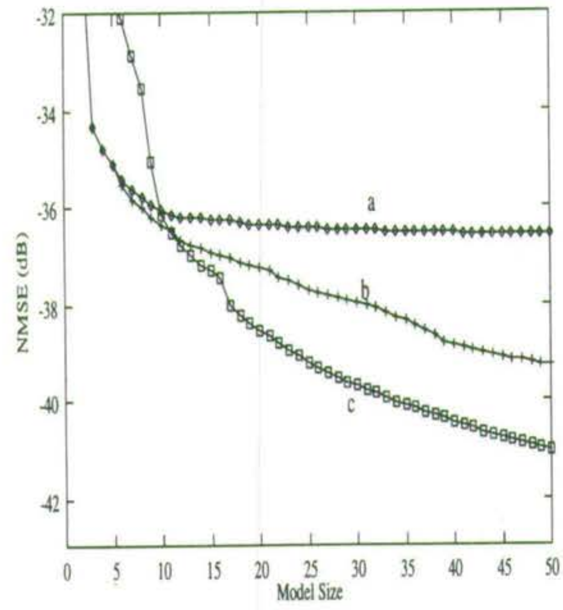

Figure 3: Performance of predictors in training phase for Mackey-Glass series
a) Linear model, b) Linear & RBF model, c) Linear, RBF & 1st order GRBF model

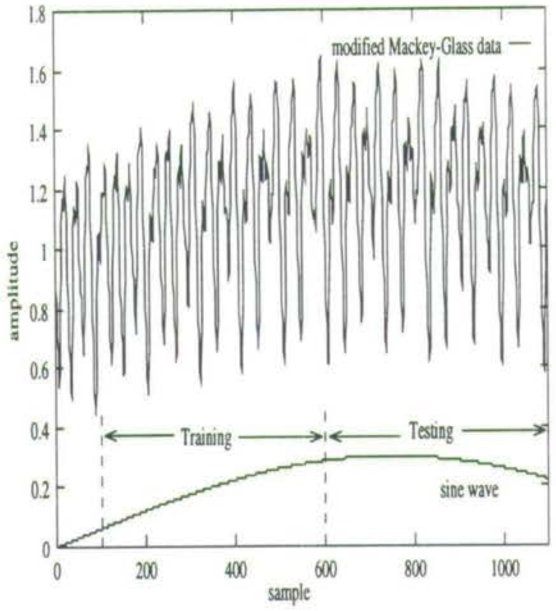

Figure 5: Modified Mackey-Glass time series

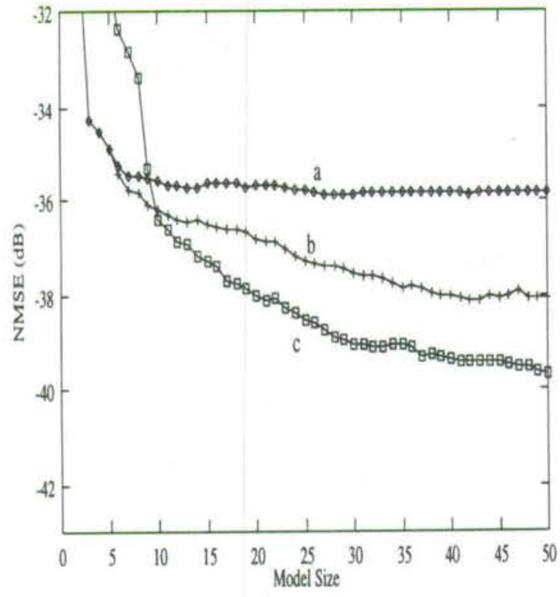

Figure 4: Performance of predictors in testing phase for Mackey-Glass series
a) Linear model, b) Linear & RBF model, c) Linear, RBF & 1st order GRBF model

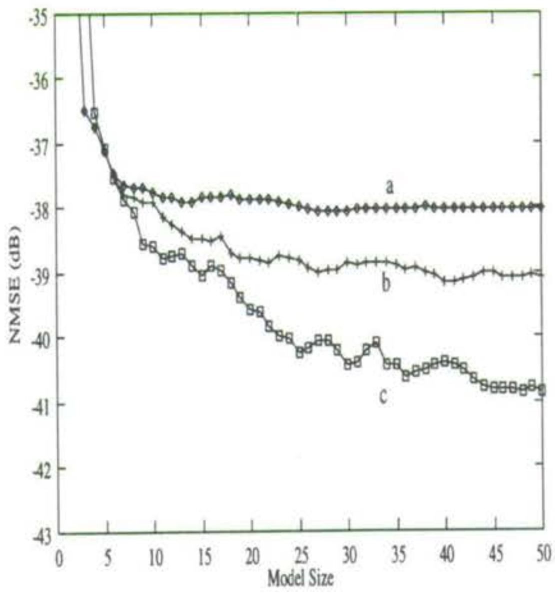

Figure 6: Performance of predictors in testing phase for modified Mackey-Glass series
a) Linear model, b) Linear & RBF model,
c) Linear, RBF & 1st order GRBF model.

A.3 Publication to appear in - E. S. CHNG, B. MULGREW, S. CHEN and G.GIBSON, "Optimum lag and subset selection for a radial basis function equaliser" in IEEE Workshop on Neural Networks for Signal Processing (NNSP'95), Boston.

OPTIMUM LAG AND SUBSET SELECTION FOR A RADIAL BASIS FUNCTION EQUALISER

E.S.CHNG †, B.MULGREW †, S.CHEN ‡ and G.GIBSON+

† *Dept. of Electrical Eng., The University of Edinburgh, U.K.*
‡ *Dept. of Electrical and Electronic Eng., The University of Portsmouth, U.K*
+ *Scottish Agricultural Statistics Service, The University of Edinburgh, U.K*

Abstract

This paper examines the application of the radial basis function (RBF) network to the modelling of the Bayesian equaliser. In particular, we study the effects of delay order d on decision boundary and attainable bit error rate (BER) performance. To determine the optimum delay parameter for minimum BER performance, a simple BER estimator is proposed.
The implementation complexity of the RBF network grows exponentially with respect to the number of input nodes. As such, the full implementation of the RBF network to realise the Bayesian solution may not be feasible. To reduce some of the implementation complexity, we propose an algorithm to perform subset model selection. Our results indicate that it is possible to reduce model size without significant degradation in BER performance.
Indexing Term: Bayesian equaliser, neural networks, RBF network, BER.

1 Introduction

It is well-known that the performance of neural network (NN) equaliser is superior to the conventional linear equaliser for the digital communication symbol-by-symbol equalisation problem [1-3]. The superiority of the NN structure is due to its ability to model the optimum Bayesian decision boundary better than the conventional linear systems. In many practical equalisation problems, the Bayesian decision boundary is often highly nonlinear, and in some cases, not linearly separable. It is thus not surprising that NN techniques, which are capable of modelling any nonlinear decision boundaries, have become very popular in equalisation problems. This paper continues this theme and investigates the application of the radial basis function (RBF) network to realise the Bayesian equaliser.

The paper is organised as follows: In Sec. 2.1, we extend the work reported in [1,2] to show the effects of delay order on the Bayesian equaliser's decision boundary and BER performance. Our analysis show that the equaliser achieves different attainable BER performance when different delay order is applied under the same signal to noise (SNR) operating condition. To determine the optimum delay order, a simple BER estimate for the equaliser is proposed in Sec. 3. The implementation complexity of the RBF equaliser is also discussed, and an algorithm to select small-sized RBF models which approximate the Bayesian solution is presented in Sec.4

2 Implementing the Bayesian equaliser

An established model of a digital communication channel subjected to inter-symbol interference (ISI) for a multi-level pulse amplitude modulation (2-ary PAM) scheme is described by the following equation [2,4]:

$$r(k) = \sum_{i=0}^{n_a-1} s(k-i)a(i) + e(k) \tag{1}$$

where $r(k)$ is the received signal at time k, $s(k)$ is an independently identically distributed (i.i.d) transmitted symbol with symbol constellation defined by the set $\{\pm 1\}$, $a(i)$ are the channel impulse response coefficients with the length of the impulse response n_a, and $e(k)$ is the additive white Gaussian noise $e(k)$ of zero mean and variance σ_e^2 [2,4]. The equaliser uses an array of received signal

$$\mathbf{r}(k) = [r(k), \cdots, r(k-m+1)]^T \tag{2}$$

to estimate the transmitted symbol $s(k-d)$, i.e. $\hat{s}(k-d)$. The integers m and d are known as the feedforward order and delay order respectively.

The transmitted symbols that affect the input vector $\mathbf{r}(k)$ is the transmit sequence $\mathbf{s}(k) = [s(k), \cdots, s(k-m-n_a+2]^T$. There are $N_s = 2^{m+n_a-1}$ possible combinations of these input sequences, i.e. $\{\mathbf{s}_j\}, 1 \le j \le N_s$ [2]. In the absence of noise, there are N_s corresponding received sequences $C_d = \{\mathbf{c}_j\}, 1 \le j \le N_s$, which are also referred to as channel states. The subscript d in C_d denotes the delay order used. The values of the channel states are defined by the following equation,

$$\mathbf{c}_j = F[\mathbf{s}_j] \quad 1 \le j \le N_s \tag{3}$$

where the matrix $F \in R^{m \times (m+n_a-1)}$ is

$$F = \begin{bmatrix} a(0) & a(1) & \dots & a(n_a-1) & 0 & \dots & \dots & 0 \\ 0 & a(0) & a(1) & \dots & a(n_a-1) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & a(0) & a(1) & \dots & a(n_a-1) \end{bmatrix} \tag{4}$$

When noise is present, the received vector $\mathbf{r}(k)$ has a Gaussian distribution with expected values corresponding to the respective $\mathbf{c}_j$.

The set of channel states $\{\mathbf{c}_j\}, 1 \le j \le N_s$ can be partitioned according to the value of $s(k-d)$, i.e., channel states associated with $s(k-d) = +1$

belong to the class $C_d^{(+)}$, and channel states associated with $s(k-d) = -1$ belong to the class $C_d^{(-)}$. The response of the Bayesian equaliser prior to the slicer is [2],

$$\begin{aligned} f(\mathbf{r}(k)) &= \sum_{\mathbf{c}_i \in C^{(+)}} p_i (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r}(k) - \mathbf{c}_i\|^2 / 2\sigma_e^2) \\ &\quad - \sum_{\mathbf{c}_j \in C^{(-)}} p_j (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r}(k) - \mathbf{c}_j\|^2 / 2\sigma_e^2) \end{aligned} \tag{5}$$

where p_i and p_j are the *a priori* probabilities of occurrence for the respective channel states. In the case of i.i.d transmitted symbols, $p_i = p_j = 1/N_s$. The output of the Bayesian equaliser $\hat{s}(k-d)$ is $\mathrm{sgn}(f(\mathbf{r}(k)))$, where sgn(.) is the signum function.

From Eq. 5, it is obvious that the structure of the RBF network is identical to the Bayesian equaliser [2], and that the RBF network realises precisely the Bayesian solution when the weights, centres and the nonlinearity of hidden units are set accordingly.

2.1 Effects of delay order on decision boundaries

The set $\{\mathbf{r}(k) | f(\mathbf{r}(k)) = 0\}$ defines the Bayesian decision boundary and is dependent on the channel state values and the delay order parameter [1, 2]. The channel states are determined by the channel impulse response and the equaliser feedforward order. The channel states however do not uniquely define the decision boundary. Given a set of channel states, the decision boundary can be changed by using different delay orders.

As an example, the Bayesian decision boundaries realised by a RBF equaliser with feedforward order $m = 2$ for channel $H(z) = 0.5 + 1.0z^{-1}$ is examined. Fig 1a lists all the 8 possible combinations of the transmitted signal sequence $\mathbf{s}(k)$ and the corresponding channel states $\mathbf{c}_i$. Fig. 1b depicts the corresponding decision boundaries for the different delay orders. Note the dramatic change in the shape of the decision boundaries for different delay orders.

The use of different delay orders also results in different limits of BER performance. To determine the optimum delay order, a computationally simple method to estimate the BER of the Bayesian equaliser is presented in Sec. 3.2.

3 Probability of mis-classification

This section presents the analysis of probability of mis-classification of the Bayesian equaliser.

S/No	Transmitted symbols			Channel State	
i	$\mathbf{s}(k)$			$\mathbf{c}_i$	
	[$s(k)$	$s(k-1)$	$s(k-2)$]	[$\hat{r}(k)$	$\hat{r}(k-1)$]
1	1	1	1	1.5	1.5
2	1	1	-1	1.5	-0.5
3	1	-1	1	-0.5	0.5
4	1	-1	-1	-0.5	-1.5
5	-1	1	1	0.5	1.5
6	-1	1	-1	0.5	-0.5
7	-1	-1	1	-1.5	0.5
8	-1	-1	-1	-1.5	-1.5

Fig (a) : State Table

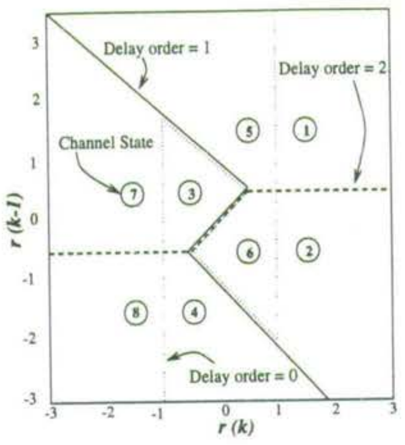

Fig (b) : Decision boundaries

Figure 1: (a) Input and desired channel states for channel $H(z)$,
(b) Bayesian decision boundaries for channel $H(z)$.

3.1 Evaluating the probability of error

We define $Z^+ \subset R^m$ to be the region of $\mathbf{r}(k)$ classified as $+1$ and $Z^- \subset R^m$ to be the region classified as -1. The probability of making a wrong decision P_e is

$$P_e = \sum_{\mathbf{c}_i \in \mathbf{C}^+} p_i \int_{\mathbf{r} \in Z^-} f_{\mathbf{r}|\mathbf{c}_i}(\mathbf{r}) d\mathbf{r} + \sum_{\mathbf{c}_j \in \mathbf{C}^-} p_j \int_{\mathbf{r} \in Z^+} f_{\mathbf{r}|\mathbf{c}_j}(\mathbf{r}) d\mathbf{r} \qquad (6)$$

where $f_{\mathbf{r}|\mathbf{c}_l}(\mathbf{r})$ is the probability density function (pdf) of the noisy received vector $\mathbf{r}$ conditioned on the received channel state being $\mathbf{c}_l$,

$$f_{\mathbf{r}|\mathbf{c}_l}(\mathbf{r}) = (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r} - \mathbf{c}_l\|^2 / (2\sigma_e^2)) \qquad (7)$$

Because the symbol constellation is symmetric, equation (6) can be reduced to

$$P_e = 2 \sum_{\mathbf{c}_i \in \mathbf{C}^+} p_i \int_{\mathbf{r} \in Z^-} f_{\mathbf{r}|\mathbf{c}_i}(\mathbf{r}) d\mathbf{r} \qquad (8)$$

3.2 Estimating the probability of error

The evaluation of BER using Eq. 8 involves evaluating m-dimensional integrals over the error region Z^-. As a closed-form solution for the expression does not exist, one must resort to numerical methods. This option however is un-attractive for large m. As our requirement to find BER performance is only one of comparing relative performance for equalisers using different delay orders, a simple approximation may be used to estimate the BER. The probability of mis-classifications, P_e, can be expressed as

$$P_e = \frac{1}{N_s} \sum_{i=1}^{N_s} P_e(\mathbf{c}_i) \tag{9}$$

where $P_e(\mathbf{c}_i)$ is the probability of mis-classification conditioned on the noise-free channel state being $\mathbf{c}_i$. It can be shown that in the case when SNR $\rightarrow \infty$, $P_e(\mathbf{c}_i)$ can be reduced to the minimum distance bound [5], i.e.,

$$P_e(\mathbf{c}_i) = Q(|\zeta_i|/\sigma_e) = \int_{|\zeta_i|}^{\infty} (2\pi\sigma_e^2)^{-1/2} \exp(-x^2/(2\sigma_e^2))dx \tag{10}$$

where $|\zeta_i|$ is the absolute minimum Euclidean distance of $\mathbf{c}_i$ to the decision boundary.

Although Eq. 10 is only valid for very high SNR, it can be applied with Eq. 9 to evaluate a rough estimate of the BER performance. Our simulation results however indicate that the proposed estimator gives good BER estimates even for low SNRs.

3.3 BER estimate : Some simulation results

Simulations were conducted to compare the BER results obtained using Eqs. 9 and 10 with those obtained by the Monte Carlo (MC) simulations. The following channels which have the same magnitude but different phase response were used,

$$H1(z) = 0.8745 + 0.4372z^{-1} - 0.2098z^{-2} \tag{11}$$
$$H2(z) = 0.2620 - 0.6647z^{-1} - 0.6995z^{-2} \tag{12}$$

For the experiment, the equaliser's feedforward order was chosen to be 4 with the transmit symbol alphabet $\{\pm 1\}$. Fig 2 compares the BER estimates of Eqs. 9 and 10 with those of MC simulations for the two channels using different delay orders. The results show that the proposed BER estimate is very accurate. To illustrate the strong dependence of the equaliser's performance with respect to the delay order, we plot the performance of the equaliser using the delay parameter as the horizontal axis in Fig 3.

4 Selecting subset RBF model

The implementation of the full RBF solution requires the use of all N_s channel states. In some cases, equivalent Bayesian solution may be realised by using a subset of the full model. For example, the decision boundaries of delay

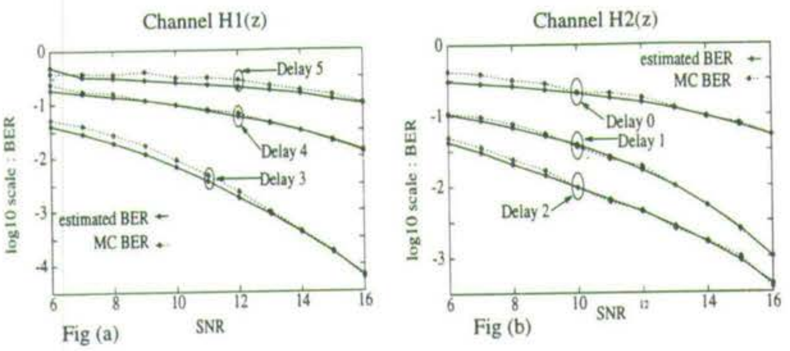

Figure 2: Estimated and MC simulations of BER vs SNR for $H1(z)$ (Fig a) and $H2(z)$ (Fig b).

order 1 and 2 (Fig 1b) can be realised by a RBF model with centres $\mathbf{c}_i$ from $\{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$ (Fig. 4a,b).

In many cases, we have observed that it is possible to find small subset RBF model to approximate the full model's solution when the decision boundary is linearly separable. The task is however much more difficult when the decision boundary is nonlinearly separable.

This section examines subset model selection algorithms to reduce implementation complexity of the RBF equaliser. The objective is to find a smaller-sized, in terms of number of centres, RBF model to realise or to approximate the same Bayesian solution as the full model. To understand how centres affect boundary, we analyse the effects of centre positions on decision boundary when $\sigma_e \to 0$. Defining the points on the boundary as $\mathbf{r}_0$, i.e. $\{\mathbf{r}_0 | f(\mathbf{r}_0) = 0\}$, Eq. 5 becomes

$$\sum_{\mathbf{c}_i \in C_d^{(+)}} p_i (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r}_0 - \mathbf{c}_i^+\|^2 / 2\sigma_e^2) = \sum_{\mathbf{c}_j \in C_d^{(-)}} p_j (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r}_0 - \mathbf{c}_j^-\|^2 / 2\sigma_e^2) \qquad (13)$$

When $\sigma_e \to 0$, the sum on the l.h.s. of Eq. 13 becomes dominated by the closest centres to $\mathbf{r}_0$, i.e.

$$U^+ = \min_{\mathbf{c}_k \in C_d^{(+)}} \{\|\mathbf{r}_0 - \mathbf{c}_k\|\} \qquad (14)$$

This is because the contribution of centres $\mathbf{c}_k \notin U^+$ converges much more quickly to zero when $\sigma_e \to 0$ than centres belonging to U^+. Similarly, the sum on the r.h.s of Eq. 13 becomes dominated by the closest centres, U^-. This

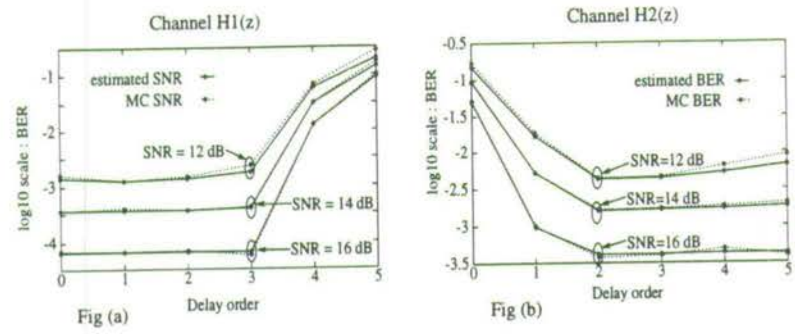

Figure 3: Estimated and MC simulations BER vs delay order for SNR 12dB, 14dB and 16dB for $H1(z)$ (Fig a) and $H2(z)$ (Fig b).

implies that the asymptotic decision boundaries are hyper-planes between pairs of U^+, U^- and that the set of all U^+, U^- defines the asymptotic decision boundaries. The following algorithm may be employed to find the set of all U^+, U^-.

Algorithm 1 : Finding U^+, U^-

```
        ⋮
For $\mathbf{c}_i \in \mathbf{c}^+$
    For $\mathbf{c}_j \in \mathbf{c}^-$
        $\mathbf{r} = \mathbf{c}_i + (\frac{\mathbf{c}_j - \mathbf{c}_i}{2})$
        if $\left[ \begin{array}{l} f(\mathbf{r}) = 0 \text{ and} \\ \mathbf{c}_i = \min_{\mathbf{c}_k \in C_d^{(+)}} \{\|\mathbf{r} - \mathbf{c}_k\|\} \end{array} \right]$        (15)
            $\mathbf{c}_i \rightarrow U^+, \mathbf{c}_j \rightarrow U^-$
    next $\mathbf{c}_j$.
next $\mathbf{c}_i$.
```

Algorithm 1 was tested to find subset models from the full RBF model (Sec. 2.1) used on channel $H(z) = 0.5 + 1.0z^{-1}$. As expected, when delay order 0 was used, all the centres, $\{\mathbf{c}_1, \ldots, \mathbf{c}_8\}$ were picked to form the subset model (Fig. 1b). For the case of using delay order 1, the selected subset model consisted of centres $\{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$ These results can be easily verified by visual inspection of the boundary formation as illustrated in Figs. 4a and 1b.

Although algorithm 1 works, the selection process is not optimum in the sense that redundant centres may be included to form the subset model. To illustrate, consider the selected subset model when delay order 2 was used.

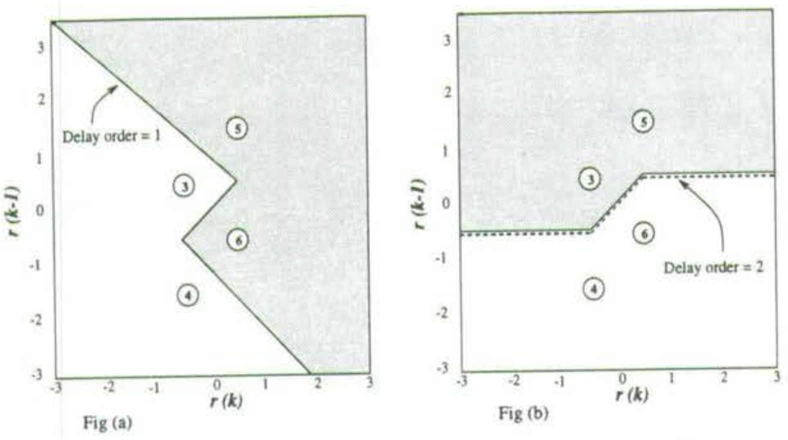

Figure 4: Realisation of decision boundary using subset RBF model.
Decision boundary for (Fig a) delay $d = 1$, (Fig b) delay $d = 2$.

By visual inspection of Figs. 4b and 1b, it is clear that the subset model with centres $\{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$ is sufficient to realise the Bayesian boundary. Algorithm 1, however, picked all the centres to form the subset model. The reason for including centres $\{\mathbf{c}_1, \mathbf{c}_2\}$ and $\{\mathbf{c}_7, \mathbf{c}_8\}$ is that these two pairs of centres satisfy Eq. 15 in algorithm 1 and thus also define the asymptotic decision boundary. They are however unnecessary because the decision boundary formed using centres $\{\mathbf{c}_3, \mathbf{c}_4\}$ and $\{\mathbf{c}_5, \mathbf{c}_6\}$ are the same.

To minimise the inclusion of redundant centres, an additional condition is introduced in Eq. 15 to verify if the new centres under consideration affect decision boundary. If the decision boundary changes with the inclusion of the new centres, they will be accepted, otherwise ignored. By adding this condition, some redundant centres will not be included in the selected subset model. The algorithm for the improved version is as follows:

```
Algorithm 2 : Finding $U^+, U^-$
        ⋮
    $\mathbf{r} = \mathbf{c}_i + (\frac{\mathbf{c}_j - \mathbf{c}_i}{2})$
    if $\left[ \begin{array}{l} f(\mathbf{r}) = 0 \text{ and} \\ \mathbf{c}_i = \min_{\mathbf{c}_k \in \mathbf{C}^+}\{\|\mathbf{r}_0 - \mathbf{c}_k\|\} \text{ and} \\ f_s(\mathbf{r}) \neq 0 \end{array} \right]$
        $\mathbf{c}_i \rightarrow U^+, \mathbf{c}_j \rightarrow U^-$
    $f_s$ = RBF model formed using $U^+, U^-$ as centres.
        ⋮
```

4.1 Subset model selection : some simulation results

Simulations were conducted to select subset models from the full model used on channels $H1(z)$ and $H2(z)$. The feed forward order used was $m = 4$, resulting in a full model with $N_s = 2^{m+n_a-1} = 64$ centres. Using SNR condition at 16dB, simulations were conducted to evaluate the performance of the subset RBF, full RBF and the linear Wiener equalisers for the two channels. The results are tabulated in Table 1a and 1b; The first column of each table indicates the delay order parameter, the second column shows the size of the subset model used while the third, fourth and fifth columns list the BER performance of the respective equalisers and the last column indicates if the decision boundary is linearly or not-linearly separable.

Our results indicate that the full RBF models' BER performance, for cases when the decision boundary is linearly separable, are normally better than those when the decision boundary is not linearly separable. This is not surprising since decision boundaries which are not linearly separable tend to be much more complicated and have more centres with different decision outputs near to each other. It was also observed that smaller-sized RBF subset models can be found for the case when the boundary is linearly separable, and their performance not significantly poorer than the full model's performance.

Delay	Subset Size	Subset BER	Full-model BER	Linear-Eq BER	Decision Boundary
0	56	-4.09	-4.09	-3.44	Linear Sep.
1	57	-4.14	-4.14	-3.07	Linear Sep.
2	32	-4.11	-4.12	-2.36	Linear Sep.
3	32	-4.11	-4.12	-1.84	Linear Sep.
4	48	-1.91	-1.91	-0.59	Not-Linear Sep.
5	64	-0.97	-0.97	-0.37	Not-Linear Sep.

Table a : Channel H1(z)

Delay	Subset Size	Subset BER	Full-model BER	Linear-Eq BER	Decision Boundary
0	56	-0.80	-1.30	-0.37	Not-Linear Sep.
1	46	-2.99	-2.99	-1.61	Linear Sep.
2	38	-3.38	-3.38	-2.67	Linear Sep.
3	56	-3.43	-3.43	-1.94	Linear Sep.
4	55	-3.32	-3.32	-1.16	Not-Linear Sep.
5	54	-3.41	-3.41	-0.80	Not-Linear Sep.

Table b : Channel H2(z)

Table 1: Comparing the performance of the full-size (64 centres) RBF equaliser, subset RBF equaliser and the Wiener equaliser for Channel $H1(z)$ (Table 1a) and Channel $H2(z)$ (Table 1b) at SNR=16db.

5 Conclusions

This paper discusses the implementation of the RBF equaliser to realise the Bayesian solution. In particular, the effects of the delay order parameter on decision boundaries and BER performance is highlighted. We have showed that the attainable BER performance depends strongly on the delay order parameter and can be significantly different for various values of the delay order. To determine the optimum operating delay order parameter, a simple BER estimator for the RBF equaliser is proposed.

The implementation complexity of the RBF equaliser to realise the Bayesian solution is also discussed. To reduce some of the implementation complexity, we have introduced an algorithm to select subset model from the full RBF implementation. Our results indicate that that good subset models with no significant degradation in BER performance may be found.

Acknowledgement

We would like to thank Achilleas G Stogioglou of Edinburgh University for his many helpful comments and ideas during the development of this work.

References

[1] G.J.GIBSON, S.SIU, and C.F.N.COWAN, "The application of nonlinear structures to the reconstruction of binary signals", *IEEE Trans. Signal Processing*, vol. 39, no. 8, pp. 1877–1884, 1991.

[2] S.CHEN, B.MULGREW, and P.M.GRANT, "A clustering technique for digital communications channel equalization using radial basis function networks", *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 570–579, 1993.

[3] C.P.CALLENDER and C.F.N.COWAN, "A comparison of six different non-linear equalisation techniques for digital communication systems", *Proc. of the Seventh European Signal Processing Conference*, pp. 1524–1527, 1994.

[4] S.U.H.QURESHI, "Adaptive equalization", *Proc. IEEE*, vol. 73, no. 9, pp. 1349–1387, 1985.

[5] B.HUGHES, "On the error probability of signals in additive white Gaussian noise", *IEEE Trans. Information Theory*, vol. 37, no. 1, pp. 151–155, 1991.

Appendix B

Maple programme to derive gradient of P_l cost function

This section discuss the derivation of $\partial P_l/\partial w_m$ where P_l is the linear equaliser's probability of mis-classification cost function, and w_m is the m^{th} weight of the linear equaliser.

To recap, the expression of the probability of mis-classification of a linear equaliser, P_l, is

$$P_l = \frac{1}{N_s} \sum_{\mathbf{c}_j \in C_d^{(+)}} P_{lc}(\mathbf{c}_j) + \frac{1}{N_s} \sum_{\mathbf{c}_k \in C_d^{(-)}} P_{lc}(\mathbf{c}_k). \tag{B.1}$$

In optimising the above function with respect to the weights $\mathbf{w}$ of the linear equaliser, the partial differentiaon of the P_l with respect to the weights needs to be calculated. The values of $\partial P_l/\partial w_1, \partial P_l/\partial w_2, \ldots, \partial P_l/\partial w_m$, where $\partial P_l/\partial w_k$, $1 \leq k \leq m$, is

$$\partial P_l/\partial w_k = \frac{1}{N_s} \sum_{j=1}^{N_s} \frac{\partial Q(|\zeta_j|/\sigma_e)}{\partial w_k}. \tag{B.2}$$

By chain rule, each $\partial Q(|\zeta_j|/\sigma_e)/\partial w_k$ can be expressed as

$$\frac{\partial Q(|\zeta_j|/\sigma_e)}{\partial w_k} = \frac{\partial Q(|\zeta_j|/\sigma_e)}{\partial \zeta_j} \frac{\partial \zeta_j}{\partial w_k}. \tag{B.3}$$

The results using the Maple symbolic partial differentiaon algorithm shows that

$$\frac{\partial Q(|\zeta_j|/\sigma_e)}{\partial w_k} = \frac{\sqrt{2}}{2} e^{-\frac{1}{2\sigma_e^2}\left(\frac{\mathbf{c}_j^T \mathbf{w}}{\mathbf{w}^T \mathbf{w}}\right)} \left(\frac{w_k(\mathbf{c}_j^T \mathbf{w}) - (sgn(\mathbf{c}_i^T \mathbf{w}) c_{jk} \sum_{i=1}^{m} w_i^2)}{\sigma_e \sqrt{\pi} (\sum_{i=1}^{m} w_i^2)^{3/2}} \right) \tag{B.4}$$

The Maple session to perform the above derivation is contained in the following pages.

```
> q`abs:=proc(val1,std`dev1)
> local a;
> #a:=(1/sqrt(2*Pi))*"int(exp(-x^2/2),x=(abs(val1))..infinity);
> a:=(1/sqrt(2*Pi))*"int(exp(-x^2/2),x= (abs(val1)/std`dev)..infinity);
> RETURN(a);
> end;
>
>
```

```
q`abs := proc(val1,std`dev1)
        local a;
            a := 1/sqrt(2*Pi)*int(exp(-1/2*x^2),x = abs(val1)/std`dev .. infinity); RETURN(a)
        end
```

```
> d1 := (c11*w1 + c12*w2 + c13*w3)/sqrt(w1*w1 + w2*w2 + w3*w3);
> d2 := (c21*w1 + c22*w2 + c23*w3)/sqrt(w1*w1 + w2*w2 + w3*w3);
> d3 := (c31*w1 + c32*w2 + c33*w3)/sqrt(w1*w1 + w2*w2 + w3*w3);
> d4 := (c41*w1 + c42*w2 + c43*w3)/sqrt(w1*w1 + w2*w2 + w3*w3);
>
>
```

$$d1 := \frac{c11\, w1 + c12\, w2 + c13\, w3}{\sqrt{w1^2 + w2^2 + w3^2}}$$

$$d2 := \frac{c21\, w1 + c22\, w2 + c23\, w3}{\sqrt{w1^2 + w2^2 + w3^2}}$$

$$d3 := \frac{c31\, w1 + c32\, w2 + c33\, w3}{\sqrt{w1^2 + w2^2 + w3^2}}$$

$$d4 := \frac{c41\, w1 + c42\, w2 + c43\, w3}{\sqrt{w1^2 + w2^2 + w3^2}}$$

```
>
> q1`abs:=q`abs(d1,std`dev);
> q2`abs:=q`abs(d2,std`dev);
> q3`abs:=q`abs(d3,std`dev);
> q4`abs:=q`abs(d4,std`dev);
>
>
```

$$q1\_abs := \frac{1}{2}\,\frac{\sqrt{2}\left(\frac{1}{2}\sqrt{2}\sqrt{\pi} - \frac{1}{2}\sqrt{2}\sqrt{\pi}\,\mathrm{erf}\left(\frac{1}{2}\,\frac{\sqrt{2}\,|c11\, w1 + c12\, w2 + c13\, w3|}{\sqrt{|w1^2 + w2^2 + w3^2|}\,std\_dev}\right)\right)}{\sqrt{\pi}}$$

$$q2\_abs := \frac{1}{2}\,\frac{\sqrt{2}\left(\frac{1}{2}\sqrt{2}\sqrt{\pi} - \frac{1}{2}\sqrt{2}\sqrt{\pi}\,\mathrm{erf}\left(\frac{1}{2}\,\frac{\sqrt{2}\,|c21\, w1 + c22\, w2 + c23\, w3|}{\sqrt{|w1^2 + w2^2 + w3^2|}\,std\_dev}\right)\right)}{\sqrt{\pi}}$$

$$q3\_abs := \frac{1}{2}\,\frac{\sqrt{2}\left(\frac{1}{2}\sqrt{2}\sqrt{\pi} - \frac{1}{2}\sqrt{2}\sqrt{\pi}\,\mathrm{erf}\left(\frac{1}{2}\,\frac{\sqrt{2}\,|c31\, w1 + c32\, w2 + c33\, w3|}{\sqrt{|w1^2 + w2^2 + w3^2|}\,std\_dev}\right)\right)}{\sqrt{\pi}}$$

$$q4\_abs := \frac{1}{2}\,\frac{\sqrt{2}\left(\frac{1}{2}\sqrt{2}\sqrt{\pi} - \frac{1}{2}\sqrt{2}\sqrt{\pi}\,\mathrm{erf}\left(\frac{1}{2}\,\frac{\sqrt{2}\,|c41\, w1 + c42\, w2 + c43\, w3|}{\sqrt{|w1^2 + w2^2 + w3^2|}\,std\_dev}\right)\right)}{\sqrt{\pi}}$$

```
> diff`q1`w1`abs:=simplify(diff(q1`abs,w1));
> diff`q2`w1`abs:=simplify(diff(q2`abs,w1));
> diff`q3`w1`abs:=simplify(diff(q3`abs,w1));
> diff`q4`w1`abs:=simplify(diff(q4`abs,w1));
>
>
>
```

$$diff\_q1\_w1\_abs := \frac{1}{2} e^{\left(-1/2 \frac{|c11\,w1 + c12\,w2 + c13\,w3|^2}{|w1^2 + w2^2 + w3^2|\, std\_dev^2}\right)} \sqrt{2} (-\mathrm{abs}(1, c11\,w1 + c12\,w2 + c13\,w3) c11\, |w1^2 + w2^2 + w3^2| + |c11\,w1 + c12\,w2 + c13\,w3| \mathrm{abs}(1, w1^2 + w2^2 + w3^2) w1) \Big/ \left(\sqrt{\pi} \, |w1^2 + w2^2 + w3^2|^{3/2} std\_dev \right)$$

$$diff\_q2\_w1\_abs := \frac{1}{2} e^{\left(-1/2 \frac{|c21\,w1 + c22\,w2 + c23\,w3|^2}{|w1^2 + w2^2 + w3^2|\, std\_dev^2}\right)} \sqrt{2} (-\mathrm{abs}(1, c21\,w1 + c22\,w2 + c23\,w3) c21\, |w1^2 + w2^2 + w3^2| + |c21\,w1 + c22\,w2 + c23\,w3| \mathrm{abs}(1, w1^2 + w2^2 + w3^2) w1) \Big/ \left(\sqrt{\pi} \, |w1^2 + w2^2 + w3^2|^{3/2} std\_dev \right)$$

$$diff\_q3\_w1\_abs := \frac{1}{2} e^{\left(-1/2 \frac{|c31\,w1 + c32\,w2 + c33\,w3|^2}{|w1^2 + w2^2 + w3^2|\, std\_dev^2}\right)} \sqrt{2} (-\mathrm{abs}(1, c31\,w1 + c32\,w2 + c33\,w3) c31\, |w1^2 + w2^2 + w3^2| + |c31\,w1 + c32\,w2 + c33\,w3| \mathrm{abs}(1, w1^2 + w2^2 + w3^2) w1) \Big/ \left(\sqrt{\pi} \, |w1^2 + w2^2 + w3^2|^{3/2} std\_dev \right)$$

$$diff\_q4\_w1\_abs := \frac{1}{2} e^{\left(-1/2 \frac{|c41\,w1 + c42\,w2 + c43\,w3|^2}{|w1^2 + w2^2 + w3^2|\, std\_dev^2}\right)} \sqrt{2} (-\mathrm{abs}(1, c41\,w1 + c42\,w2 + c43\,w3) c41\, |w1^2 + w2^2 + w3^2| + |c41\,w1 + c42\,w2 + c43\,w3| \mathrm{abs}(1, w1^2 + w2^2 + w3^2) w1) \Big/ \left(\sqrt{\pi} \, |w1^2 + w2^2 + w3^2|^{3/2} std\_dev \right)$$

```
> diff`q1`w3`abs:=simplify(diff(q1`abs,w3));
> diff`q2`w3`abs:=simplify(diff(q2`abs,w3));
> diff`q3`w3`abs:=simplify(diff(q3`abs,w3));
> diff`q4`w3`abs:=simplify(diff(q4`abs,w3));
>
>
>
```

$$diff\_q1\_w3\_abs := \frac{1}{2} e^{\left(-1/2 \frac{|c11\,w1 + c12\,w2 + c13\,w3|^2}{|w1^2 + w2^2 + w3^2|\, std\_dev^2}\right)} \sqrt{2} (-\mathrm{abs}(1, c11\,w1 + c12\,w2 + c13\,w3) c13\, |w1^2 + w2^2 + w3^2| + |c11\,w1 + c12\,w2 + c13\,w3| \mathrm{abs}(1, w1^2 + w2^2 + w3^2) w3) \Big/ \left(\sqrt{\pi} \, |w1^2 + w2^2 + w3^2|^{3/2} std\_dev \right)$$

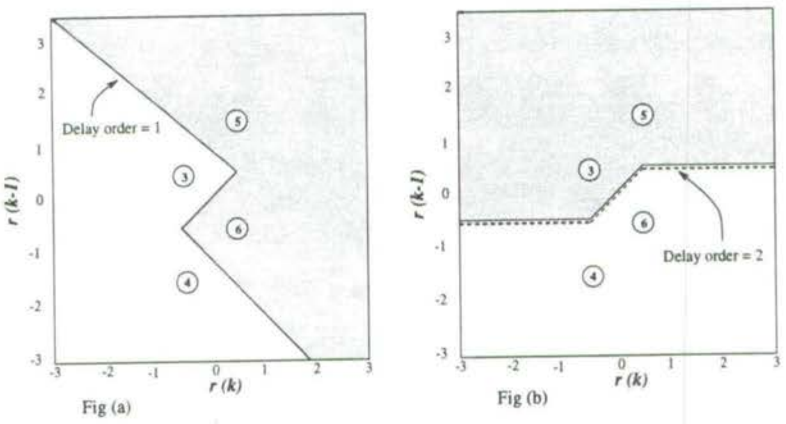

Figure 4: Realisation of decision boundary using subset RBF model. Decision boundary for (Fig a) delay $d = 1$, (Fig b) delay $d = 2$.

By visual inspection of Figs. 4b and 1b, it is clear that the subset model with centres $\{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$ is sufficient to realise the Bayesian boundary. Algorithm 1, however, picked all the centres to form the subset model. The reason for including centres $\{\mathbf{c}_1, \mathbf{c}_2\}$ and $\{\mathbf{c}_7, \mathbf{c}_8\}$ is that these two pairs of centres satisfy Eq. 15 in algorithm 1 and thus also define the asymptotic decision boundary. They are however unnecessary because the decision boundary formed using centres $\{\mathbf{c}_3, \mathbf{c}_4\}$ and $\{\mathbf{c}_5, \mathbf{c}_6\}$ are the same.

To minimise the inclusion of redundant centres, an additional condition is introduced in Eq. 15 to verify if the new centres under consideration affect decision boundary. If the decision boundary changes with the inclusion of the new centres, they will be accepted, otherwise ignored. By adding this condition, some redundant centres will not be included in the selected subset model. The algorithm for the improved version is as follows:

Algorithm 2 : Finding U^+, U^-

```
        ⋮
    $\mathbf{r} = \mathbf{c}_i + (\frac{\mathbf{c}_j - \mathbf{c}_i}{2})$
    if $\left[ \begin{array}{l} f(\mathbf{r}) = 0 \text{ and} \\ \mathbf{c}_i = \min_{\mathbf{c}_k \in \mathbf{c}^+}\{\|\mathbf{r}_0 - \mathbf{c}_k\|\} \text{ and} \\ f_s(\mathbf{r}) \neq 0 \end{array} \right]$
        $\mathbf{c}_i \to U^+,\ \mathbf{c}_j \to U^-$
    $f_s$ = RBF model formed using $U^+, U^-$ as centres.
        ⋮
```

$$diff\_q2\_w3\_abs := \frac{1}{2} e^{\left(-1/2 \frac{|c21\,w1+c22\,w2+c23\,w3|^2}{|w1^2+w2^2+w3^2|\,std\_dev^2}\right)} \sqrt{2} (-\mathrm{abs}(1, c21\,w1 + c22\,w2 + c23\,w3) c23\, |w1^2 + w2^2 + w3^2| + |c21\,w1 + c22\,w2 + c23\,w3|\, \mathrm{abs}(1, w1^2 + w2^2 + w3^2) w3) \Big/ \left(\sqrt{\pi}\, |w1^2 + w2^2 + w3^2|^{3/2}\, std\_dev\right)$$

$$diff\_q3\_w3\_abs := -\frac{1}{2} e^{\left(-1/2 \frac{|c31\,w1+c32\,w2+c33\,w3|^2}{|w1^2+w2^2+w3^2|\,std\_dev^2}\right)} \sqrt{2} (\mathrm{abs}(1, c31\,w1 + c32\,w2 + c33\,w3) c33\, |w1^2 + w2^2 + w3^2| - |c31\,w1 + c32\,w2 + c33\,w3|\, \mathrm{abs}(1, w1^2 + w2^2 + w3^2) w3) \Big/ \left(\sqrt{\pi}\, |w1^2 + w2^2 + w3^2|^{3/2}\, std\_dev\right)$$

$$diff\_q4\_w3\_abs := -\frac{1}{2} e^{\left(-1/2 \frac{|c41\,w1+c42\,w2+c43\,w3|^2}{|w1^2+w2^2+w3^2|\,std\_dev^2}\right)} \sqrt{2} (\mathrm{abs}(1, c41\,w1 + c42\,w2 + c43\,w3) c43\, |w1^2 + w2^2 + w3^2| - |c41\,w1 + c42\,w2 + c43\,w3|\, \mathrm{abs}(1, w1^2 + w2^2 + w3^2) w3) \Big/ \left(\sqrt{\pi}\, |w1^2 + w2^2 + w3^2|^{3/2}\, std\_dev\right)$$