IDENTIFICATION & CONTROL OF

NONLINEAR SYSTEMS

A thesis presented for the degree of Doctor of Philosophy in the Department of Engineering, University of Warwick, UK, by

Q. M. Zhu, B. Sc., M. Sc.

August 1, 1989

Table of Contents

Acknowledgements i

Summary ii

List of published work based on the thesis iii

Preface iv

SECTION 1 Incomplete time series analysis and modelling 1

S1.1 Survey 1

S1.2 List of notations 4

S1.3 List of figures and tables 4

Chapter 1 Fourier amplitude and spectrum estimation 6

1.1 Introduction 6

1.2 Estimator for Fourier transform 9

1.2.1 Unbiased estimate 9

1.2.2 Minimum mean square error estimate 9

1.3 Estimator for PSD 11

1.3.1 Unbiased estimate 11

1.3.2 Minimum mean square error estimate 11

1.4 Properties of estimators 12

1.4.1 Compensating factors in limiting cases 12

1.4.2 Spurious periodicity 14

1.4.3 Direct estimators 17

1.5 Experiment results 18

1.6 Discussion 27

1.7 Appendix 29

Chapter 2 Frequency response estimation 34

2.1 Introduction 34

2.2 Estimator for frequency response 37

2.2.1 Unbiased estimate 37

2.2.2 Minimum mean square estimate 38

2.3 Properties of estimators 40

2.3.1 Closed-loop system 40

2.3.2 Spurious periodicity 41

2.3.3 Direct estimator 42

2.4 Experiment results 42

2.5 Appendix 50

SECTION 2 System identification and parameter estimation 53

S2.1 Survey 53

S2.2 List of notations 54

S2.3 List of figures and tables 54

Chapter 3 Structure detection and parameter estimation 58

3.1 Introduction 58

3.2 Structure detection and parameter estimation 61

3.2.1 Structure detection 61

3.2.2 Parameter estimation 69

3.3 Simulation experiment 71

3.4 Conclusion 82

Chapter 4 A variable weighted least squares algorithm 83

4.1 Introduction 83

4.2 Parameter estimation 86

4.2.1 Off-line algorithm 86

4.2.2 Properties of algorithm 87

4.2.3 Modified (on-line) algorithm 89

4.3 Experimental results 90

4.4 Application -- Analysis of a saturating second order system 95

4.4.1 Jump resonances 95

4.4.2 Jump effect prediction 97

4.4.3 Simulation experiment 97

4.4.4 Step response experiment 103

SECTION 3 Self-tuning controller design 107

S3.1 Survey 107

S3.2 List of notations 111

S3.3 List of figures 111

Chapter 5 NLGPC design 112

5.1 Introduction 112

5.2 Plant model and output prediction 114

5.3 Nonlinear general predictive controller 117

5.3.1 Controller design 117

5.3.2 Fast recursive root-solving 118

5.3.3 Self-tuning implementation 122

5.4 Simulation experiment 122

5.5 Conclusion 130

5.6 Appendix 131

Chapter 6 NLDBC design 134

6.1 Introduction 134

6.2 HCARIMA model and HCARMA model 135

6.3 General nonlinear feedback controller 138

6.3.1 Nonlinear feedback controller 138

6.3.2 Nonlinear deadbeat controller 140

6.3.3 NLDBC STC implementation 141

6.4 Simulation experiment 142

6.5 Conclusion 147

6.6 Appendix 147

SECTION 4 Conclusions and bibliography 149

S4.1 Overall conclusions 149

S4.2 Bibliography 151

SECTION 5 Appendix 158

S5.1 Papers published 158

S5.2 Computer programs 181

Acknowledgement

I am indebted to my supervisor, Professor J. L. Douce, for his encouragement, technical guidance and inspiration; to Professor K. Warwick for his encouragement and guidance in the self-tuning control study; to Mr. A. Hulme for his help and advice with use of the Prime computer, Sun station, and design of programs; to Mr. P. M. Obene for his fruitful discussion of many problems; to colleagues in Control and Instrument Systems Centre, University of Warwick, for their collaboration; to Mr. R. Kent and Mrs. V. Kent for their help in reading the draft thesis.

The project is funded by ORSA(Overseas Research Students Awards) in part, and I am grateful for the associated financial support.

Summary

This thesis investigates some problems on nonlinear system identification, parameter estimation, and signal processing.

Random signal spectral analysis and system frequency response estimation are studied from incomplete time series. Both recursive and direct estimators are presented based on either an unbiased or minimum mean square error criterion.

Nonlinear system identification and parameter estimation are studied. A quantisation technique is developed to give a clear geometrical interpretation for structure detection and parameter estimation. A new concept, state amplitude distance between current and previous operating states, is introduced, and results in a Variable Weighted Least Squares (VWLS) algorithm. A modified version makes on-line application possible. Jump resonance is predicted by the VWLS algorithm as one of the applications.

Self-tuning controllers, including a nonlinear general predictive controller and a nonlinear deadbeat controller, are designed. A vector backward shift operator is defined to simplify the expression of the Hammerstein model, and is introduced to analyse the general feedback controller design problem for nonlinear plant described by the Hammerstein model. A fast root-solver developed facilitates nonlinear model treatment in on-line applications.

Theoretical results are confirmed by simulation studies.

List of publications based on the thesis

1 Douce, J.L., Zhu, Q.M., Spectral analysis of records with missing data, IFAC Symposium on identification & system parameter estimation, 1988, Beijing, China.

2 Warwick, K., Zhu, Q.M., Douce, J.L., An adaptive controller for nonlinear systems, IFAC Symposium on adaptive control & signal processing, 1989, Glasgow, UK.

3 Douce, J.L., Zhu, Q.M., Spectrum estimation and system identification from incomplete data, International conference on system science, 1989, Wroclaw, Poland. 1989.

List of intended publications based on the thesis

1 A general predictive controller for nonlinear systems (with Prof. K. Warwick and Prof. J.L. Douce), Being reviewed by IEE Journal Part D.

2 A nonlinear deadbeat controller (with Prof. K. Warwick and Prof. J.L. Douce), In preparation.

3 Modelling the behaviour of non-linear systems (with Prof. J.L. Douce), In preparation.

Seminar presented

1 Incomplete time series analysis and modelling, Control and Instrument Systems Centre, University of Warwick, UK, Feb. 1989.

Preface

The motivation of the thesis stems from the importance of modelling, identification, parameter estimation, and controller design for a wide range of nonlinear systems whatever they are physical or social. The relevant problems in nonlinear systems have been actively studied by many workers over a long period, however present knowledge is incomplete due to the difficulties not encountered in linear systems. For nonlinear systems, a complete and unified theory is probably unattainable. Attention has historically therefore been focussed on the analysis of special cases such as Van der Pol oscillators, relay systems or systems consisting of a small number of linear dynamic elements with connected static nonlinearities. In the thesis attention is paid to a wide range of nonlinear systems.

To the end of identifying and controlling nonlinear systems as simply as possible, some new methods are developed in the thesis. Both linearization and nonlinear techniques are considered in the belief that the combination of the two kinds of techniques will bring satisfactory results.

The thesis consists of five sections.

The **first section** concentrates on the problems of incomplete time series analysis and modelling. It is not uncommon that a time series, such as an operational record of an industrial installation or market fluctuation statistics, has some missing observations due to many reasons. Bard (1974) has classified the necessary analysis as a kind of special nonlinear parameter estimation. However study shows that it is appropriate to apply linear approximating techniques to the estimation of Fourier amplitude, power spectral density, and frequency response functions. Both recursive and direct algorithms are developed using either an unbiased or minimum mean square error criterion. The properties of the resulting estimators are discussed in detail. Chapter one studies the estimation of Fourier amplitude and power spectral density of random processes. Chapter two considers estimation of the system frequency response function.

The **second section** studies several problems related to nonlinear system; structure detection, parameter estimation, and jump effect prediction. A concept, state distance or amplitude effect, results in some novel understanding and approaches to the nonlinear problems even though the basic idea, used seldom in nonlinear system identification, has been known for long time. In chapter three an amplitude quantisation technique illustrates the relationship between system structure detection and its parameter estimation in a set of three dimensional spaces. There has not been found any publication for this concept and geometric interpretation. Another interesting topic studied in chapter four is a weighted least squares algorithm based on the nearness between the current state and previous states, i.e. the amplitude distance between current state and previous ones, allowing the approximation of the behaviour of a wide range of nonlinear systems. The consideration of on-line applications leads to a modified parameter estimator. One of the applications of the algorithm, the jump effect prediction for nonlinear systems, is investigated in this chapter.

The **third section** designs two types of self-tuning controllers, the general predictive controller and the dead-beat controller, for nonlinear systems described by a combined Hammerstein + ARMA model. In chapter five a nonlinear general predictive controller is designed in the form of a self-tuning algorithm, and a fast root-solving routine is developed to find the inverse Hammerstein characteristic parameters. In chapter six a nonlinear dead-beat self-tuning controller is designed by a direct method which generalises a HARMA model with a new operator. This simplifies the nonlinear dead-beat controller design to be the same as linear dead-beat controller design. A general nonlinear feedback controller is also presented based on the HARMA model.

The **fourth section** presents the conclusions of the thesis and contains the bibliography.

The **fifth section** gives computer programs and the publications based on the thesis.

The thesis emphasizes the development of new concepts and the exploration of potential applications. The validity and effectiveness of the theoretical results are demonstrated with computer simulations. Hopefully experiments based on laboratory tests and further work in real environments will be carried on in my future work.

Section 1 Incomplete time series analysis and modelling

S1.1 Survey

In time series modelling and parameter estimation, a common situation is met in which the measured sequence is not a complete set of the observations, but the measurements corresponding to some time instants are missing, not known or unreliable.

Missing data can arise from a number of causes, such as failure of recording equipment, clerical errors, rejection of outliers, or because of an inability to observe the phenomenon at certain times. The pattern of the missing data, i.e. the distribution of the missing data position, may be in one of two categories, one deterministic or periodic, as in the case, for example, of a single sensor which is time shared to measure and record different processes, or a random or aperiodic phenomenon as in the case of an unreliable sensor which fails intermittently.

One solution to the problem of uncertain observation is interpolation, which estimates uncertain values using the known values and then reconstructs the time series. The technique is used quite often in the statistical field, a number of authors (Bard 1974; John and Prescott 1975; Jarrett 1978; Smith 1981) having studied the problem and obtained some fruitful results. One common feature of the previous authors work is the reliance on parametric models. However the interpolating method has some disadvantages (Robinson 1983; Harris 1987). Satisfactory results will be probably obtained by simple or complicated methods of interpolating the missing values, followed by application of some standard techniques to calculate such as parameters of ARMA model or power spectrum density etc., only when discrete data are sampled at a very fast speed or when the missing data points are infrequent. Furthermore data interpolation and fitting procedures cannot be easily incorporated into a computer program, particularly when quite a few data values are missing, and to ensure good interpolation, interaction between the program and the analysis is required which would often be not practicable for many industrial systems. Douce and Zhu (1989) reported that the interpolating method is not always effective in frequency domain analysis. The

simulation experiment in the section will demonstrate the point of view.

Another more general solution is straightforward and derives the estimated statistics directly by means of some criterion such as UNBiase (UNB) or minimum Mean Square Error (MSE) without reconstruction of the time series. The first of the advantages of the method is that less a priori knowledge of the properties of the time series is needed than in the interpolating method. The second advantage is its computational efficiency which is a key factor in the field of real time analysis and processing. The third factor is that it has better accuracy than some simple interpolating techniques in certain environments. The author preference (Douce and Zhu 1988, 1989) is based on these three considerations, which will be applied for analysis and estimation of Fourier transform, spectrum, and frequency response in the section. It is noted that since 1962 (Jones 1962; Parzen 1963), especially 1969 (Nahi) some useful results (Robert and Gaster 1980; McGiffin and Murthy 1980, 1981; Harris 1987) have been published in signal modelling and parameter estimation from time series with uncertain observations, by other sub-optimal methods. A good survey of some representative techniques can be found in McGiffin and Murthy (1980).

The new method developed in the section for the estimation of Fourier transform, power spectrum density and frequency response has general applications in many fields, including analysis of operational records of industrial processes, market fluctuation data, or population statistics. The present study concentrates on stationary time series.

In summary this section considers the spectrum analysis of finite duration data where the available data contains one or more uncertain observations or missing points. Both recursive and direct algorithms are developed using either an unbiased or minimum mean square error criterion to obtain estimates of the Fourier transform and the power spectrum density of the complete data record. The analysis is extended to include frequency response identification of a wide range of systems with missing data at the output.

It is shown that it is important to consider the position within the record as well as the total number of the missing points. One result of a periodic

distribution of the missing points is the production of a spurious periodicity in the measured power spectrum. It is believed that this phenomenon has not been reported previously.

A comparison of the new method with traditional techniques including simple interpolation method is presented, and simulation results demonstrate the resulting improved performance.

A combination of the relevant techniques is suggested for the estimation of power spectrum density and the frequency response function.

S1.2 List of notations

u	system input, no missing observations.
x	system output, no missing observations.
y	system output, with missing observations.
X(Ω), $S_{xx}(\Omega)$, etc.	amplitude and power spectra of x.
$\hat{X}$	estimate of X(Ω) based on Y(Ω).
M	number of missing points.
σ_x^2	variance of specified signal.
$\bar{S}_{yy}(\Omega)$	smoothed value of estimate of $S_{yy}(\Omega)$.
$G(\Omega)$	system frequency response.
E[.]	expectation.
X^*	complex conjugate of X.
CM	Covariance Method.
IM	Interpolation Method.
PM	Periodogram Method.
UNB	UNBiased or UNB criterion.
MSE	Mean Square Error or MSE criterion.
Ω	harmonic frequency.

S1.3 List of figures and tables

Fig. 1.2.1	Recursive estimators
Fig. 1.4.1	Spurious periodicity
Fig. 1.5.1	Spectrum estimates with 10% missing points (SYS 1)
Fig. 1.5.2	Spectrum estimates with 20% missing points (SYS 1)
Fig. 1.5.3	Spectrum estimates with 10% missing points (SYS 2)

Fig. 1.5.4	Spectrum estimates with 20% missing points (SYS 2)
Fig. 1.6.1	Spectrum estimation procedure
Fig. 2.1.1	Closed-loop system
Fig. 2.1.2	System with missing points at output
Fig. 2.2.1	Recursive estimators
Fig. 2.4.1	Frequency response estimates with 10% missing points (SYS 1)
Fig. 2.4.2	Frequency response estimates with 20% missing points (SYS 1)
Fig. 2.4.3	Frequency response estimates with 10% missing points (SYS 2)
Fig. 2.4.4	Frequency response estimates with 20% missing points (SYS 2)
Table 1.5.1	Errors in spectrum estimation
Table 1.5.2	Fit in spectrum estimation
Table 2.4.1	Errors in frequency response estimation
Table 2.4.2	Fit in frequency response estimation

Chapter 1 Fourier amplitude and spectrum estimates

1.1 Introduction

The Power Spectrum Density (PSD) is an important parameter in the description of random process. There are two general methods currently available for PSD estimation with uncertain observations. We call one the Covariance Method (CM), the alternative technique the Periodogram Method (PM). A brief introduction to CM and PM will be presented after a definition of the problem.

The time series, assumed zero mean and normally distributed, with no missing observations is written

$$x = x_1, \cdots, x_i, \cdots, x_N \quad , \quad i = 1, N$$

With missing observations, the series is written

$$y_i = x_i \cdot g_i \qquad (1.1.1)$$

where $g_i = 0$ for a missing point and unity otherwise. This is a reasonable choice when the time series has zero mean value. The missing data consists of M missing points, a member of this set being x_m. The incomplete time series y may be recognised as the product of the original time series x amplitude modulated by the function g_i.

The discrete Fourier Transform (FT) of x_i is

$$X(\Omega) = \sum_{i=1}^{N} x_i (cos\ i\Omega - j \sin i\Omega) = X_R + jX_I \qquad (1.1.2)$$

where $\Omega = \frac{2\pi k}{N}$ with k integer .

The power spectral density is

$$S_{xx}(\Omega) = \frac{X(\Omega)X^*(\Omega)}{N} \qquad (1.1.3)$$

Alternatively, the power spectral density can be expressed in term of the measured autocovariance function according to

$$S_{xx}(\Omega) = R_{xx}(0) + 2\sum_{\tau=1}^{N} R_{xx}(\tau)cos(\Omega\tau) \tag{1.1.4}$$

where

$$R_{xx}(\tau) = \frac{1}{N}\sum_{i=1}^{N-\tau} x_i x_{i+\tau} \qquad \text{for } \tau \geq 0 \tag{1.1.5}$$

The covariance method for spectral analysis with missing points (Jones 1962; Parzen 1963; McGiffin and Murthy 1980) uses eqn (1.1.1) to give

$$R_{yy}(\tau) = R_{gg}(\tau).R_{xx}(\tau) \tag{1.1.6}$$

Knowing the positions of the missing points, $R_{gg}(\tau)$ can be calculated, to give

$$R_{yy}(\tau) = \sum_{i=1}^{N-\tau} y_i y_{i+\tau}$$

$$R_{gg}(\tau) = \sum_{i=1}^{N-\tau} g_i g_{i+\tau} \tag{1.1.7}$$

$$\hat{R}_{xx}(\tau) = \frac{R_{yy}(\tau)}{R_{gg}(\tau)} \tag{1.1.8}$$

By eqn(1.1.8) computing N+1 covariances, consistent estimates can be obtained as $\hat{R}_{xx}(\tau)$ converges to the true value as $N \rightarrow \infty$ (McGiffin and Murthy 1980), then using eqn (1.1.4) the spectral density of x(t) can be estimated.

The main drawback of this method is that different covariances are computed with different accuracy since the variance of the estimate $\hat{R}_{xx}(\tau)$ increases with increasing τ, and for a given τ it decreases with increasing value of the denominator of eqn(1.8) for $\hat{R}_{xx}(\tau)$, thus for a given sequence with a particular pattern of missing observations, determining the optimal τ's to be used in computing $\hat{R}_{xx}(\tau)$ is a difficult problem and has not yet been studied (McGiffin and Murthy 1980). Another disadvantage is that the CM is not statistically efficient (McGiffin and Murthy 1980). The method can not be used if in case of $R_{gg}(\tau) = 0$. Jones (1962) and Parzen (1963) proved the restriction for the case of regularly missing observations.

An alternative approach termed the periodogram method (Harris 1987) is based directly on the measured power spectrum $S_{yy}(\Omega)$. This is defined by

$$\hat{S}_{xx}(\Omega) = \frac{N}{N-M} S_{yy}(\Omega) \tag{1.1.9}$$

where (N - M) is the number of non-zero terms in y(t). A obvious drawback is that no consideration is given to the positions of the missing points.

In general, both the above estimators are sub-optimal, as shown later in this chapter.

Furthermore a simple Interpolation Method (IM) may be used to obtain PSD by the reconstruction of time series. The rule estimating missing value is given by

$$y_i = \frac{x_{i-1} + x_{i+1}}{2} \tag{1.1.10}$$

It can be expected that this IM gives good results for narrow-band signals, on the other hand, it is unsatisfactory for wide-band signals. Therefore IM has less generality and needs some advance knowledge of the time series.

The new estimator of the property Θ_x (for example the power spectral density) of the process giving the sequence x, is derived from the estimate of Θ_y from the sequence y according to

$$\hat{\Theta}_x = k\ \Theta_y \tag{1.1.11}$$

where k is a frequency-dependent factor depending also on the position of the missing points chosen to satisfy a criterion such as zero bias or minimum mean square error. This real gain factor k can be expressed in terms of the true value of Θ_x, and hence both a recursive method and a direct method are introduced to determine its value. Simulation studies show that the recursion introduced converges rapidly for the wide range of examples considered whilst the direct method gives the same results as the recursive one with a significant reduction in computing time.

1.2 Estimator for Fourier transform

1.2.1 Unbiased estimate

The requirement for an unbiased estimator is given by

$$E[X-\hat{X}] = E[(X_R - k_1 Y_R) + j(X_I - k_2 Y_I)] = 0 \qquad (1.2.1)$$

that is

$$E[X_R - k_1 Y_R] = 0 \text{ and } E[X_I - k_2 Y_I] = 0 \qquad (1.2.2)$$

In this special case k_1 and k_2 are unity since $E[X_R] = E[Y_R] = E[X_I] = E[Y_I] = 0$.

1.2.2 Minimum mean square estimate

The MSE criterion is written in the form

$$E[(X-\hat{X})(X-\hat{X})^*] = E[(X_R - k_1 Y_R)^2 + (X_I - k_2 Y_I)^2] \qquad (1.2.3)$$

Differentiating with respect to k_1, and setting the result to zero and noting that Y is known,

$$E[X_R Y_R] = k_1 Y_R^2 \qquad (1.2.4)$$

that is

$$k_1 = \frac{E[X_R Y_R]}{Y_R^2} \qquad (1.2.5)$$

and similarly

$$k_2 = \frac{E[X_I Y_I]}{Y_I^2} \qquad (1.2.6)$$

Appendix A1.1 derives expressions for k_1 and k_2 on the two assumptions:

(a) The missing points are separated such that the cross-correlation between values at missing positions may be neglected, and

(b) The missing points are not too near the end of the record.

The results are

$$k_1 = \frac{f_1(\frac{N}{2} - \sum \cos^2 m\Omega) + f_2\gamma_1}{Y_R^2}$$

$$k_2 = \frac{f_1(\frac{N}{2} - \sum \sin^2 m\Omega) - f_2\gamma_1}{Y_I^2} \qquad (1.2.7)$$

where

$$f_1 = \frac{E[X_R^2 + X_I^2]}{N} \approx \frac{k_3[Y_R^2 + Y_I^2]}{N} \qquad (1.2.8)$$

where k_3 will be determined in the calculation of the gain for PSD estimation. f_2 and γ_1 can be estimated directly, and Y_R^2 and Y_I^2 can be also expressed as the functions of f_1 and f_2, see appendix A1.1 for the details.

Eqn(1.2.7) and eqn(1.2.8) give a recursive form of estimator in which f_1 is updated from the current estimates of k_1, k_2, and k_3 (with initial values $k_1=k_2=1$, $k_3=\frac{N}{(N-M)}$ assumed) according to eqn(1.2.8) and then k_1 and k_2 are calculated, with k_3 obtained from the PSD estimator. Therefore the estimators for FT and PSD are connected recursively, as demonstrated in the block diagram as shown in Fig. 1.2.1.

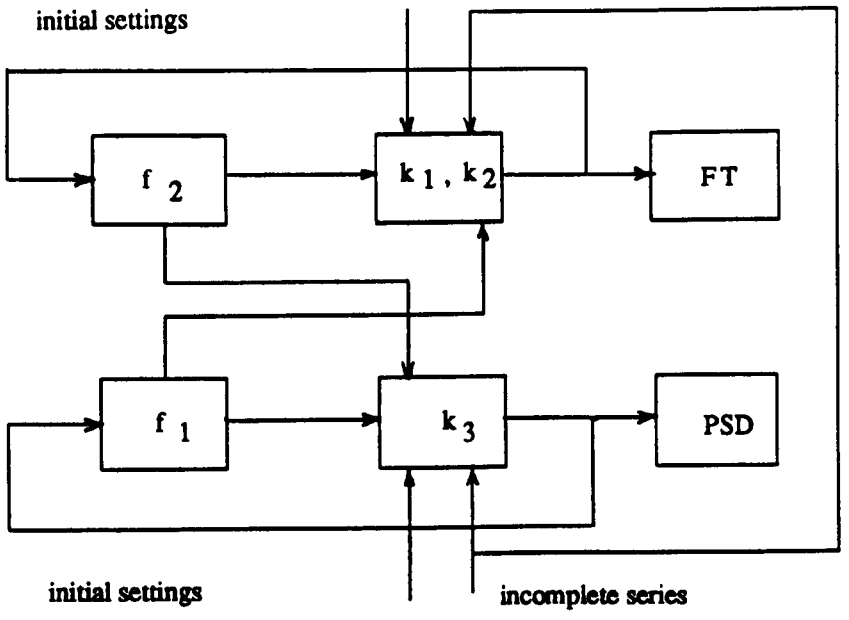


Figure 1.2.1 Recursive estimators

1.3 Estimator for PSD

The reason for two gains being introduced adopted in the Fourier transform estimator is that there are two independent variables, the real and imaginary parts, in that criterion. However the PSD estimate is a real quantity, hence just one gain k_3 is set when compensates the measured PSD directly.

1.3.1 Unbiased estimate

Firstly UNB estimator is developed in term of

$$E\,[(X_R^2+X_I^2)-k_3(Y_R^2+Y_I^2)] = 0 \tag{1.3.1}$$

the gain k_3 is

$$k_3 = \frac{E\,[X_R^2+X_I^2]}{Y_R^2+Y_I^2} = \frac{N\,f_1}{f_1(N-2M)+\sigma^2 M} \tag{1.3.2}$$

the derivation is given in appendix A1.1.

1.3.2 Minimum mean square error estimate

Secondly MSE estimator is developed in term of minimisation of

$$E\,\{[(X_R^2+X_I^2) - k_3(Y_R^2+Y_I^2)]^2\} \tag{1.3.3}$$

Differentiating with respect to k_3, and setting the result to zero and noting y is known,

$$k_3 = \frac{E\,[(X_R^2+X_I^2)(Y_R^2+Y_I^2)]}{[(Y_R^2+Y_I^2)]^2}$$

$$= \frac{E\,[X_R^2Y_R^2+X_I^2Y_R^2+X_R^2Y_I^2+X_I^2Y_I^2]}{[(Y_R^2+Y_I^2)]^2}$$

$$= \frac{[E\,[X_R^2]+E\,[X_I^2]][Y_R^2+Y_I^2]+2[[E\,[X_R\,Y_R\,]]^2+[E\,[X_I\,Y_I\,]]^2+[E\,[X_R\,Y_I\,]]^2+[E\,[X_I\,Y_R\,]]^2]}{[(Y_R^2+Y_I^2)]^2} \tag{1.3.4}$$

where, for Gaussian signal (Godfrey and Jones 1986), it has

$$E\,[X_R^2Y_R^2] = E\,[X_R^2]Y_R^2+2[E\,[X_R\,Y_R\,]]^2$$

$$E[X_I^2 Y_R^2] = E[X_I^2]Y_R^2 + 2[E[X_I Y_R]]^2$$

$$E[X_R^2 Y_I^2] = E[X_R^2]Y_I^2 + 2[E[X_R Y_I]]^2$$

$$E[X_I^2 Y_I^2] = E[X_I^2]Y_I^2 + 2[E[X_I Y_I]]^2 \qquad (1.3.5)$$

These above terms can be evaluated in terms of f_1, f_2, β_1, γ_1 and γ_2. Appendix A1.1 derives the detailed expressions for eqns (1.3.4) and (1.3.5).

1.4 Properties of the new estimators

1.4.1 Compensating factors in limiting cases

In this part, two special signals are selected for analysis, to indicate the dependence of the estimators on the signal characteristics. $E[Y_R^2 + Y_I^2]$ is used instead of $Y_R^2 + Y_I^2$ for the following theoretical analysises.

The signals selected are white noise and a narrow-band signal.

1. White noise signal ($E(x) = 0, \sigma^2 = 1$)

From eqn(A1.2), eqn(A1.3), and eqn(A1.12),

$$f_1 = 1 \ , \ f_2 = 0 \ , \ \beta_1 = \gamma_2 \approx 0 \qquad (1.4.1)$$

First consider MSE FT estimator, from eqn(1.2.3)

$$k_1 = \frac{E[X_R Y_R]}{E[Y_R^2]}$$

$$= \frac{f_1(\frac{N}{2} - \sum\cos^2 m\Omega) + f_2\gamma_1}{f_1(\frac{N}{2} - 2\sum\cos^2 m\Omega) + \sigma^2\sum\cos^2 m\Omega + 2f_2\gamma_1}$$

$$k_2 = \frac{E[X_I Y_I]}{E[Y_I^2]}$$

$$= \frac{f_1(\frac{N}{2} - \sum\sin^2 m\Omega) - f_2\gamma_1}{f_1(\frac{N}{2} - 2\sum\sin^2 m\Omega) + \sigma^2\sum\sin^2 m\Omega - 2f_2\gamma_1} \qquad (1.4.2)$$

Substitute eqn(1.4.1) into eqn(1.4.2), gives

$$k_1 = 1 \quad , \quad k_2 = 1 \qquad (1.4.3)$$

As mentioned before, the gains for UNB estimator are undefined. For the UNB and MSE estimators $k_1 = k_2 = 1$. The position and number of the missing data have no effect on the estimator in the case.

Second consider UNB PSD estimator, substitute eqn(1.4.1) into eqn(1.3.2), gives

$$k_3 = \frac{N}{N-M} \tag{1.4.4}$$

which is the same as the traditional PM.

Now consider MSE PSD estimator, from eqn(1.3.3)

$$k_3 = \frac{E[(X_R^2+X_I^2)(Y_R^2+Y_I^2)]}{E[(Y_R^2+Y_I^2]^2}$$

$$= \frac{E[X_R^2Y_R^2+X_I^2Y_R^2+X_R^2Y_I^2+X_I^2Y_I^2]}{E[Y_R^4+2Y_R^2Y_I^2+Y_I^4]} \tag{1.4.5}$$

Substitute eqn(1.4.1) into eqn(1.4.5) with lengthy operation, gives

$$k_3 = \frac{N(N-M)+2[(\frac{N}{2}-\sum\cos^2 m\Omega)^2+(\frac{N}{2}-\sum\sin^2 m\Omega)^2+2(\sum\cos m\Omega\ \sin m\Omega)^2]}{3(N-M)^2-4(\frac{N}{2}-\sum\cos^2 m\Omega)(\frac{N}{2}-\sum\sin^2 m\Omega)+4(\sum\cos m\Omega\ \sin m\Omega)^2}$$

$$= \frac{N(N-M)+(N^2-2NM+2M^2)}{3(N-M)^2-N^2+2NM}$$

$$= \frac{2N^2-3NM+2M^2}{2N^2-4NM+3M^2}$$

$$= \frac{2-3\,M/N+2\,(N/M)^2}{2-4\,M/N+3\,(N/M)^2} \tag{1.4.6}$$

From the analysis, UNB and MSE estimators depend only on the number of missing points for a white noise signal, and the MSE PSD estimator is biased. However there is no big difference when the number of missing data is small, for instance M/N = 0.1, k_3 = 1.1 in UNB, k_3 = 1.06 in MSE. In particular they are the same when M = 0.

2. Narrow-band signal

In this case

$$f_1 \to \infty \quad , \quad |f_2|, |\beta_1|, |\gamma_1|, |\gamma_2| < R, \ R \text{ is positive and finite} \tag{1.4.7}$$

two limitating cases of the arguments, (1) $\sum\cos^2 m\Omega=0$ and $\sum\sin^2 m\Omega=M$ and (2) $\sum\cos^2 m\Omega=M$ and $\sum\sin^2 m\Omega=0$, these both lead to $\sum\cos m\Omega \sin m\Omega=0$.

First consider MSE FT estimator, substitute eqn(1.4.7) into eqn(1.4.2), to give, in case (1)

$$k_1 = 1 \quad , \quad k_2 = \frac{N-2M}{N-4M} = \frac{1-2M/N}{1-4M/N} \tag{1.4.8}$$

in case (2),

$$k_1 = \frac{N-2M}{N-4M} = \frac{1-2M/N}{1-4M/N} \quad , \quad k_2 = 1 \tag{1.4.9}$$

Now consider UNB PSD estimator, substitute eqn(1.4.7) into eqn(1.3.1), giving, in both cases,

$$k_3 = \frac{N}{N-2M} = \frac{1}{1-2M/N} \tag{1.4.10}$$

Now consider MSE PSD estimator, substitute eqn(1.4.7) into eqn(1.4.5) with lengthy operation, to give, in both cases,

$$k_3 = \frac{N(N-2M)+2[(\frac{N}{2}-\sum\cos^2 m\Omega)^2+(\frac{N}{2}-\sum\sin^2 m\Omega)^2]}{3(N-2M)^2-4(\frac{N}{2}-2\sum\cos^2 m\Omega)(\frac{N}{2}-2\sum\sin^2 m\Omega)}$$

$$= \frac{N^2-2NM+M^2}{N^2-4NM+6M^2}$$

$$= \frac{1-2M/N+(M/N)^2}{1-4M/N+6(M/N)^2} \tag{1.4.11}$$

1.4.2 Spurious periodicity

An interesting property is the spurious periodic distribution of PSD over a range of unimportant harmonic frequencies due to a periodic distribution of missing points. Consider a PSD estimator by PM with reference to eqn(1.1.9),

$$\hat{S}_{xx} = \frac{N}{N-M}\bar{S}_{yy}$$

$$= \frac{1}{N-M} E[X-X_m][X-X_m]^*$$

$$= \frac{1}{N-M}[E(XX^*)-2E(X_R X_{mR}+X_I X_{mI})+E(X_m X_m^*)]$$

$$\approx \frac{1}{N-M}(f_1(N-2M)+E(X_m X_m^*))$$

$$\approx \frac{1}{N-M}(N(N-2M)S_{xx}+M\sigma_x^2+2R_m(\alpha)cos(\alpha\Omega)) \qquad (1.4.12)$$

where

$$R_m(\alpha) = E[X_{mi}X_{mj}] \quad , \quad \alpha = |m_i - m_j| \quad , \quad j = i-1 \qquad (1.4.13)$$

if S_{xx} has low-pass filter property , then

$$\lim_{\Omega \to \pi} S_{xx} \to 0$$

or

$$S_{xx} \approx 0 \quad , \quad \text{for } h < \Omega \le \pi \qquad (1.4.14)$$

where h is a measure of the cut-off frequency of the signal.

Eqn(1.4.13) becomes

$$\hat{S_{xx}} \approx \frac{1}{N-M}[\sigma_x^2 M+2R_m(\alpha)\cos\alpha\Omega) \qquad h < \Omega \le \pi \qquad (1.4.15)$$

Clearly there exists a periodic spectrum at high harmonics segment, which consists of a constant plus a cosine function. The period of the spectrum is

$$T_m = \frac{N}{\alpha} = \frac{N}{|m_i - m_j|} \quad , \quad j = i-1 \qquad (1.4.16)$$

This follows since

$$cos(\alpha\Omega) = cos(\frac{\alpha 2\pi k}{N}) \quad , \quad k = 0, \ldots, N-1 \qquad (1.4.17)$$

The frequency is

$$f_m = \frac{(2\pi\alpha)}{2\pi N} = \frac{\alpha}{N} \qquad (1.4.18)$$

The average value is

$$\frac{\sigma_x^2 M}{N-M} = \frac{\sigma_x^2 M/N}{1-M/N} \qquad (1.4.19)$$

From eqn(1.4.19), increasing the number of missing point increases the level of the constant spurious estimated spectrum at high frequencies. A graphic presents the analysis as shown in Fig. 1.4.1.

Another qualitative explanation of the phenomenon considers the periodic missing points (of period P points) as a sampled version of original sequence x(i). This sampled sequence is subtracted from x(i) to produce the observed sequence y(i). The sampled sequence, sampled at the low frequency (1/P), introduces aliasing of the original signal, so that the original term at zero frequency produces alias terms at the frequency 1/P, 2/P, etc. This is a periodic phenomenon, giving a repetitive power spectrum, repeating over (N/P) harmonic frequencies.

It can be concluded that a spurious periodic spectrum appears at high harmonics which is much smaller than main part of spectrum, its period and average value being determined by the number and position of missing data. The phenomenon is caused by periodically distributed missing points, and occurs for any spectrum with band limited property, no matter whether low-pass or high-pass.

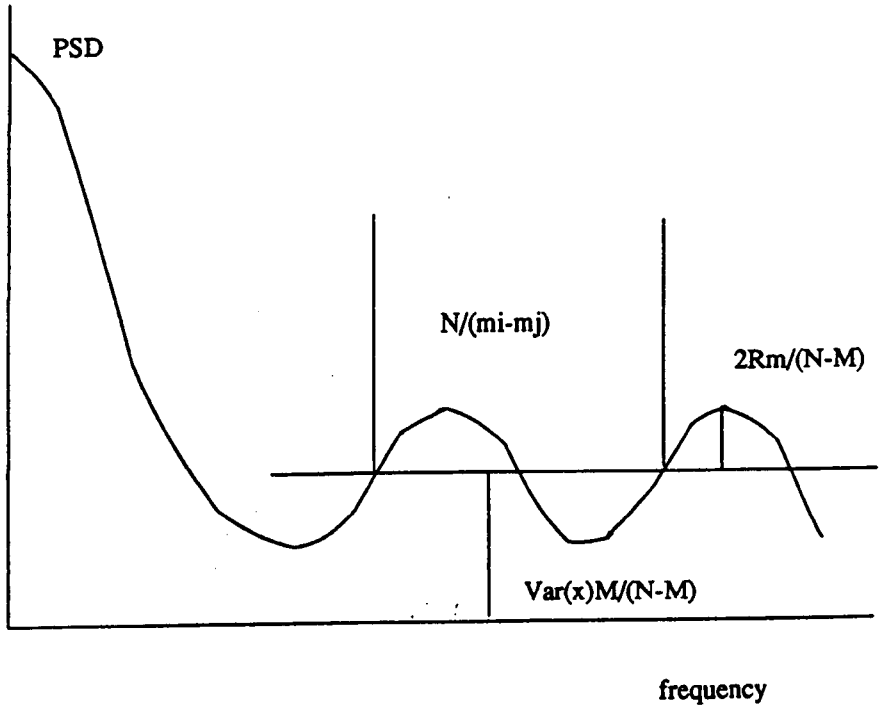


Figure 1.4.1 Spurious periodicity

1.4.3 Direct estimators

A further development is that the recursive estimators may be modified to be direct estimators by the substitutions

$$X_R = Y_R + X_{mR}$$

$$X_I = Y_I + X_{mI} \tag{1.4.20}$$

Appendix A1.2 presents the detailed derivations for the resulting estimators. It is noticed that the recursive estimators use the variable substitutions like

$$Y_R = X_R - X_{mR}$$

$$Y_I = X_I - X_{mI} \tag{1.4.21}$$

where X_R and X_I are real values which are estimated recursively.

Summarying this section:

1. UNB estimators depend only on the signal characteristics and the number of missing data. However this accords with the assumptions given previously.

2. MSE estimators depend on the signal characteristics and the number, but also the position of missing data. Since MSE is a sort of quadratic criterion, the terms for the gains include quadratic form such as $(\frac{N}{2}-\sum\cos^2 m\Omega)^2$, $(\sum\cos m\Omega \sin m\Omega)^2$ for PSD and $\sum\cos^2 m\Omega$, $\sum\sin^2 m\Omega$ for Fourier transform.

3. The ratio M/N is an important parameter in the estimators. N/M is a representative of Signal to Noise Ratio (SNR), since missing data introduce noise into the analysis. Some of the estimators will fail when $M/N \geq 1/2$, consistent with the results of Jones (1962) and Parzen (1963).

4. A spurious periodic spectrum gives information about the distribution of missing points.

5. The direct estimators reduce significantly computing time comparing with recursive estimators.

1.5 Experiment results

Three systems excited by Gaussian white noise have been studied to compare the traditional and new methods for spectral analysis. The experiments have been completed to examine the effect of smoothing over several blocks and over adjacent frequencies on the resulting estimates.

Three quantitative measures of performance have been used for the comparison. The first one is the sum over all frequencies of the error squared between the true spectrum with no missing points and the estimated spectrum with missing points. This is a measure of bias, and is given by

$$e1 = \frac{\sum_{\Omega=0}^{\pi} (E(A) - E(\hat{A}))^2}{N} \tag{1.5.1}$$

where A is the PSD calculated from time series without missing data and $\hat{A}$ is the PSD estimated by either the UNB estimator or the MSE estimator from time series with missing data. N is the time series length without missing data. The second one is the sum over all frequencies of mean square error, that is

$$e2 = \frac{\sum_{\Omega=0}^{\pi} E(A - \hat{A})^2}{N} \tag{1.5.2}$$

The third one is a linear regression of the true spectrum on the estimated spectrum again over all frequencies, given by

$$fit = \frac{\sum_{\Omega=0}^{\pi} E(A) E(\hat{A})}{\sum_{\Omega=0}^{\pi} (E(\hat{A}))^2} \tag{1.5.3}$$

In the all experiments, a block length of 128 points has been used, and the whole data length equals the length of one block multiplied by the number of blocks, which is the data length used by CM. The missing points have been introduced at positions 10, 20, ..., 120 firstly, that is ten percent of missing points in the series, and then at 5, 10, ..., 120, which corresponds to twenty percent of missing points in the series. In those experiments each involving one block, smoothing over two adjacent frequencies for system 1, and four adjacent frequencies for

system 2 has been used to estimate the PSD for the calculation of k_1, k_2, k_3. For the experiments involving more blocks, the spectrum is smoothed over the blocks for this calculation. A lag window (Bartlett) is used to smooth the covariance function when the correlation method is used. In the experiment, expectation operation E[.] concerned in theoretical derivation is replaced by smoothing operation $\sum$ [.], i.e. arithmetical averaging. Three particular systems have been considered.

1. First order system (SYS 1, narrow-band)

The system considered is

$$x_k = 0.9\, x_{k-1} + u_k$$

in which u_k is a white noise normally distributed signal of unity variance and zero mean value. Fig. 1.5.1 compares the errors in an experiment consisting of 1000 blocks with 10% missing points using traditional and the new methods, and shows the true PSD and the estimated PSD with semilogarithmic axis to display clearly the periodic phenomenon at high harmonics. Fig. 1.5.2 shows results with 20% missing points. The maximum errors in the traditional methods occur at zero frequency, corresponding to the frequency of the peak value of the true spectrum.

2. Fourth order system (SYS 2)

The signal is produced by passing the previously defined white noise through the process described by Harris (1987)

$$y_{k-3} = 1.7143y_{k-4} - 0.9048y_{k-5} + u_k$$

$$y_k = 1.0732y_{k-1} - 0.9512y_{k-2} + y_{k-3}$$

This produces a power spectrum with two pronounced resonances.

The periodogram method is again inferior to the correlation method, both being worse than the new methods. Fig. 1.5.3 shows, with 10% missing points, how the errors in spectral estimation vary with frequency, the frequencies of the maxima corresponding to the resonant frequencies of the system. The maximum of the error is approximately 10% of the true value. It demonstrates again the periodic phenomenon predicted by theory. Fig. 1.5.4 shows the results with 20%

missing points.

Ensembles of experiments, each of duration equal to one, five, ten and one hundred blocks have been completed to investigate the new estimators performances with comparison to the traditional estimators, Table 1.5.1 summaries the results and shows the significant reduction in errors resulting from the new estimators, UNB has the minimum bias and the subminimum mean square error, MSE has the minimum mean square error and the subminimum bias, compared with the other methods except IM for SYS1 and SYS2. Table 1.5.2 lists the ensemble average of the linear regression of the true on the estimated spectrum. The new estimators substantially reduce the deviation of this factor from unity.

For above two systems, IM gives the best results for the considered methods due to the narrow-band characteristics. The following SYS 3 is mainly selected to check the IM generality.

3. First order system (SYS 3, wide-band)

The system considered is

$$x_k = 0.1\, x_{k-1} + u_k$$

The experiment shows that IM is worse than UNB and MSE. The comparisons have been recorded in Table 1.5.1 and Table 1.5.2.

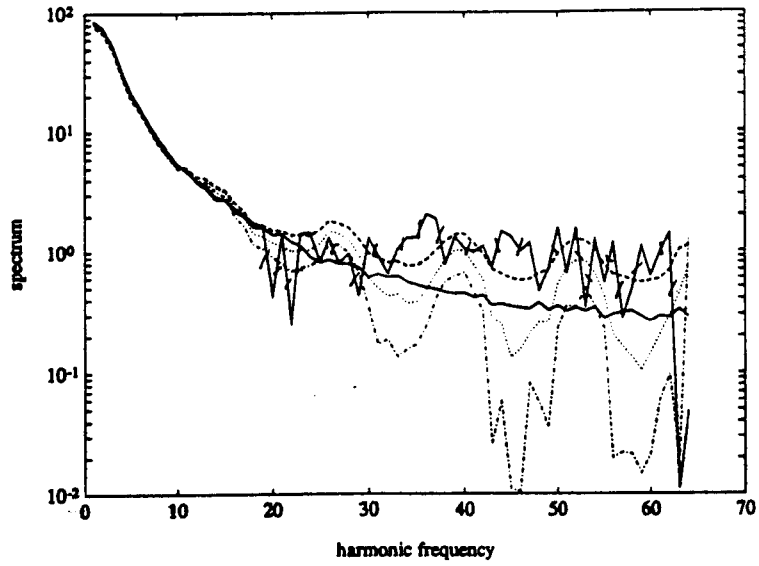


Figure 1.5.1a. Spectrum estimates (sys1, mis 10%).
——————— with no missing points
- - - - - - - - - - periodogram method
-/-/-/-/-/-/-/-/- covariance method
..... minimum bias
— · — · — minimum mean square error

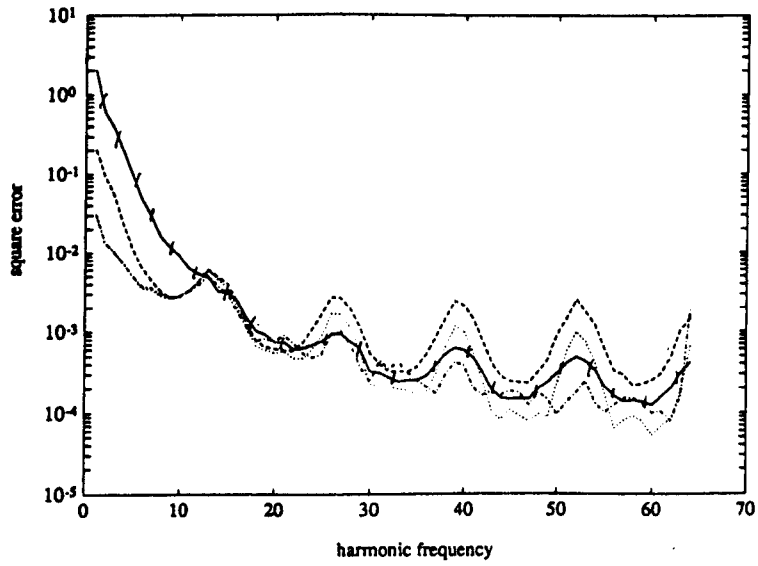


Figure 1.5.1b. Error squared as a function of harmonic.
- - - - - - - - - - covariance method
-/-/-/-/-/-/-/-/- periodogram method
..... minimum bias
— · — · — minimum mean square error

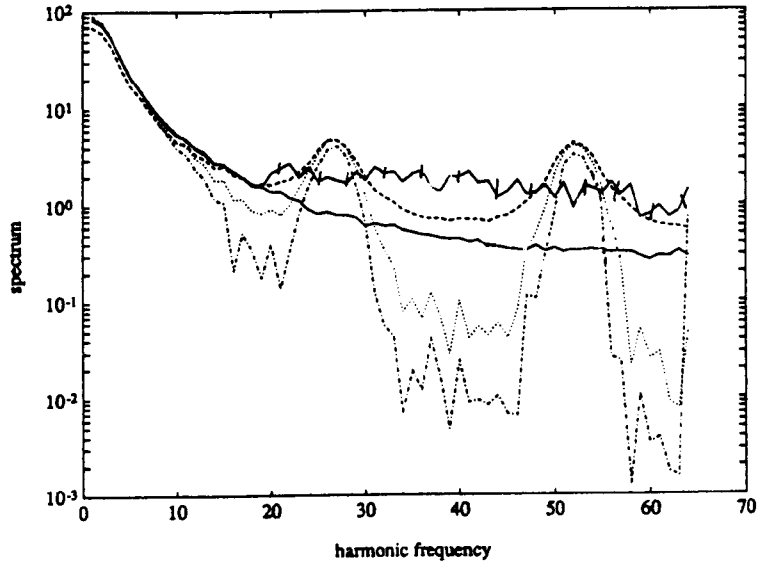


Figure 1.5.2a. Spectrum estimates (sys1, mis 20%).
——— with no missing points
- - - - - - - - - - periodogram method
-/-/-/-/-/-/-/-/- covariance method
..... minimum bias
— . — . — minimum mean square error

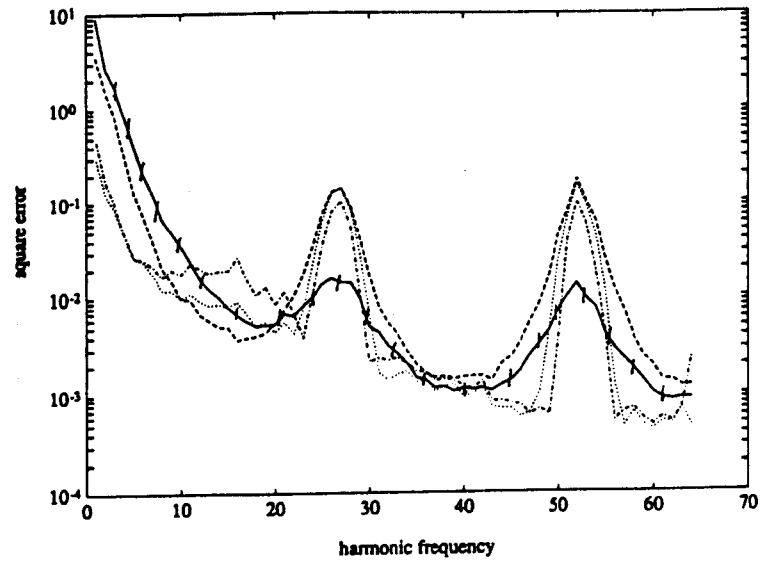


Figure 1.5.2b. Error squared as a function of harmonic.
- - - - - - - - - - covariance method
-/-/-/-/-/-/-/-/- periodogram method
..... minimum bias
— . — . — minimum mean square error

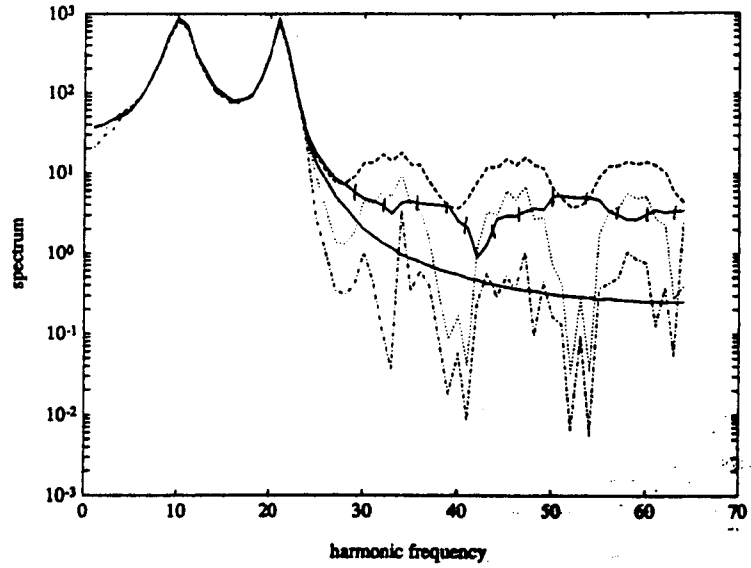


Figure 1.5.3a. Spectrum estimates (sys2, mis 10%).

———————— with no missing points

- - - - - - - - - - periodogram method

-/-/-/-/-/-/-/-/- covariance method

..... minimum bias

— · — · — minimum mean square error

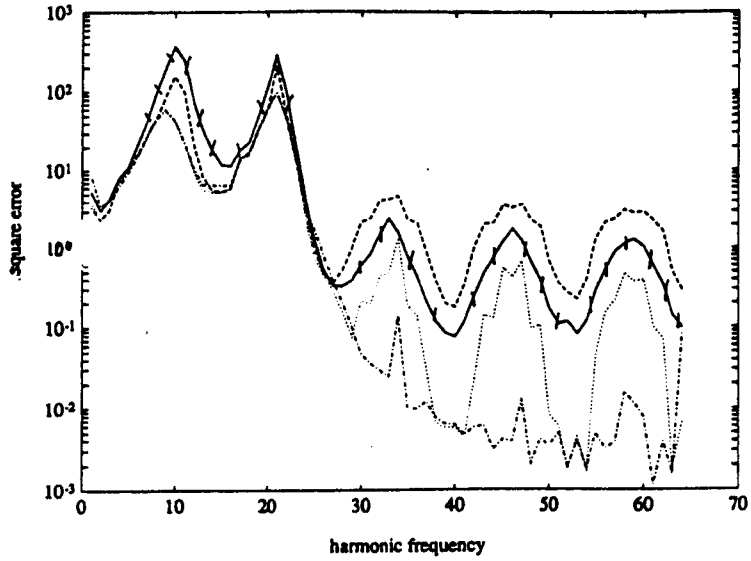


Figure 1.5.3b. Error squared as a function of harmonic.

- - - - - - - - - - covariance method

-/-/-/-/-/-/-/-/- periodogram method

..... minimum bias

— · — · — minimum mean square error

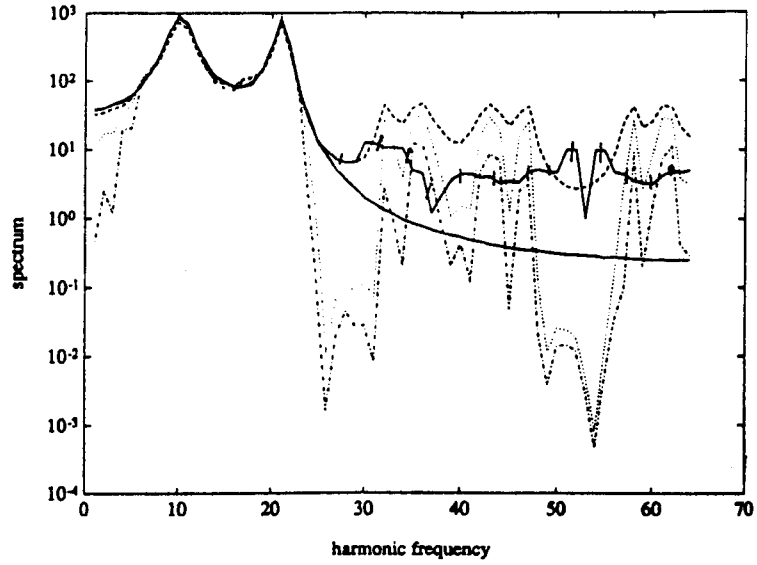


Figure 1.5.4a. Spectrum estimates (sys2, mis 20%).

———— with no missing points

- - - - - - - - - - periodogram method

-/-/-/-/-/-/-/-/- covariance method

..... minimum bias

— · — · — minimum mean square error

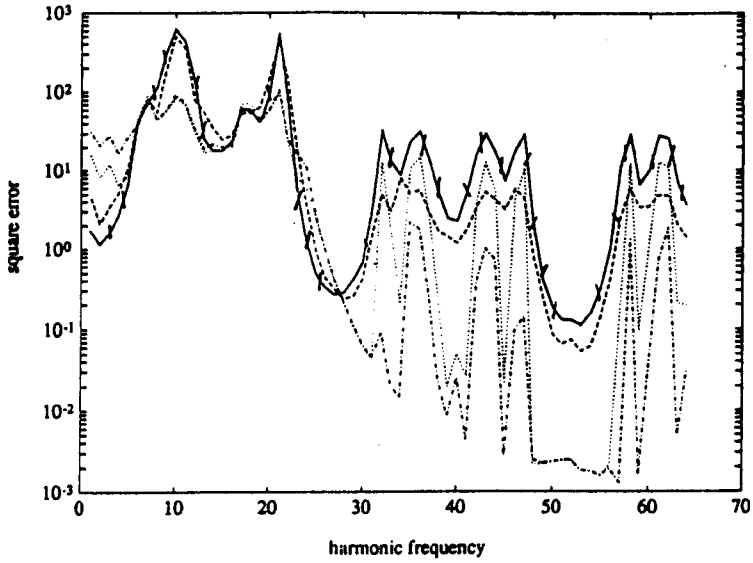


Figure 1.5.4b. Error squared as a function of harmonic.

- - - - - - - - - - **covariance method**

-/-/-/-/-/-/-/-/- **periodogram method**

..... minimum bias

— · — · — minimum mean square error

Errors in spectrum estimation

data length		PM		CM		IM		UNB		MSE	
(10% missing)		e1	e2	e1	e2	e1	e2	e1	e2	e1	e2
sys1	128	50.39	105.9	46.88	83.01	0.338	0.677	12.01	22.98	13.17	21.12
sys2	128	2600	5942	2507	5020	680.8	1025	1871	4191	2009	4001
sys3	128	0.171	0.317	0.176	0.312	0.187	0.973	0.130	0.282	0.133	0.257
sys1	128*5	5.031	10.36	4.912	9.957	0.133	0.633	1.934	8.248	2.852	8.201
sys2	128*5	553.0	1261	370.1	1220	205.2	365.0	184.3	1032	317.9	1022
sys3	128*5	0.047	0.278	0.042	0.266	0.063	0.713	0.034	0.249	0.041	0.238
sys1	1280	3.679	7.847	3.552	6.594	0.066	0.385	0.993	3.760	1.687	3.530
sys2	1280	284.2	1233	190.2	1187	137.1	310.6	167.4	945.1	170.2	900.8
sys3	1280	0.043	0.274	0.039	0.268	0.059	0.732	0.032	0.248	0.034	0.253
sys1	12800	2.7982	5.982	0.941	4.102	0.014	0.573	0.137	2.085	0.155	1.751
sys2	12800	238.8	1197	166.7	900.9	96.52	258.5	21.40	709.7	34.01	668.1
sys3	12800	0.009	0.267	0.007	0.266	0.017	0.298	0.002	0.216	0.009	0.201
data length		PM		CM		IM		UNB		MSE	
(20% missing)		e1	e2	e1	e2	e1	e2	e1	e2	e1	e2
sys1	128	261.3	350.3	253.1	340.8	0.746	1.028	66.81	88.81	78.95	84.95
sys2	128	10042	21042	9253	18000	818.2	1305	6345	9947	7069	9905
sys3	128	0.626	1.355	0.611	1.298	0.812	2.342	0.574	1.027	0.581	1.019
sys1	128*5	9.520	48.01	8.922	40.55	0.296	1.072	5.373	19.41	6.704	19.00
sys2	128*5	2000	4479	2111	3899	679.5	1248	559	2079	897.9	2088
sys3	128*5	0.152	0.877	0.150	0.876	0.167	0.963	0.128	0.703	0.129	0.700
sys1	1280	8.352	33.44	5.962	26.53	0.141	1.039	1.404	6.596	2.127	6.453
sys2	1280	1903	4305	1923	3801	193.6	385.1	507.8	2057	847.2	2009
sys3	1280	0.114	0.853	0.110	0.791	0.137	0.921	0.066	0.623	0.071	0.597
sys1	12800	6.198	21.28	3.862	16.24	0.026	1.157	1.106	3.783	1.651	2.959
sys2	12800	1508	4000	347.9	3790	153.8	367.7	207.4	1800	231.7	1778
sys3	12800	0.049	0.451	0.044	0.538	0.073	0.552	0.031	0.403	0.062	0.393

Table 1.5.1

Fit in spectrum estimation						
data length (10% missing)		PM fit	CM fit	IM fit	UNB fit	MSE fit
sys1	128	1.113	1.093	1.029	1.069	1.077
sys2	128	1.266	1.095	1.038	1.041	1.079
sys3	128	0.913	0.925	1.213	0.974	0.968
sys1	128*5	1.108	1.088	1.011	1.064	1.084
sys2	128*5	1.102	1.062	1.030	1.039	1.045
sys3	128*5	1.021	0.980	1.187	1.013	1.014
sys1	1280	1.097	1.070	0.995	1.010	1.014
sys2	1280	1.064	1.040	1.019	0.980	0.971
sys3	1280	1.013	1.012	1.070	1.008	1.009
sys1	12800	1.065	1.025	1.001	1.007	1.012
sys2	12800	1.038	1.031	1.013	1.021	1.027
sys3	12800	1.012	1.010	1.068	1.003	1.005
data length (20% missing)		PM fit	CM fit	IM fit	UNB fit	MSE fit
sys1	128	1.239	1.095	1.041	1.079	1.107
sys2	128	1.270	1.259	1.041	1.163	1.215
sys3	128	0.823	0.829	1.247	0.912	0.913
sys1	128*5	1.223	1.089	1.008	1.067	1.087
sys2	128*5	1.217	1.184	1.039	1.155	1.100
sys3	128*5	1.165	1.154	1.225	1.083	1.091
sys1	1280	1.216	1.075	1.004	0.988	0.980
sys2	1280	1.193	1.068	1.032	0.970	0.941
sys3	1280	1.162	1.153	1.218	1.032	1.037
sys1	12800	1.202	1.031	1.002	0.991	1.015
sys2	12800	1.183	1.045	1.028	0.998	0.979
sys3	12800	1.015	1.011	1.098	1.009	1.011

Table 1.5.2

1.6 Discussion

From the simulation experiment, several common points may be noted

1. The shape and peak positions of estimated PSD by traditional PM or CM are correct. However the errors vary at different harmonics, the largest errors occur at peak positions. This arises since traditional methods use average gain compensation, on the other hand, the new methods make use of varying gain compensation, the more error the more compensation is given, therefore the better results obtained.

2. The location of missing points has substantial influence on the errors introduced by these missing points. Even though UNB estimator does not depend on the position, it relies on the assumption that these missing points are adequately separated.

3. An interesting and new spurious periodic spectral phenomenon has been predicted and observed, caused by a periodic distribution of missing points. It is noted that CM does not produce this phenomenon.

4. The effect of missing data is to introduce uncorrelated white noise into the spectral analysis. Quantitative results are presented for the spectral density of this noise background.

5. The new method presented with MSE has a further improvement compared with a sort of subminimum MSE method developed by the authors in previous work (Douce and Zhu 1988). Specifically the new method has a revised definition of the mean square error.

6. The same results have been obtained from both recursive and direct operations by UNB and MSE estimators.

7. A combination of the techniques studied in the chapter is suggested for the estimation of Fourier transformation and PSD as shown in Fig. 1.6.1.

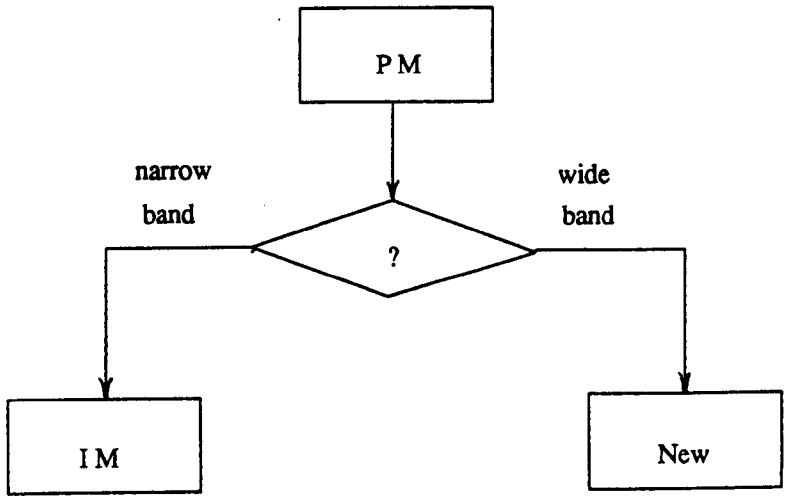


Figure 1.6.1 PSD estimation procedure

1.7 Appendix 1

A1.1 Recursive algorithm

For the estimation of Fourier transform and PSD, the compensating gains depend on five factors f_1 and f_2, γ_1, γ_2, and β_1. This section presents the detailed definitions and derivations. Consider

$$E[X_R^2] = E[\sum\sum x_i x_j \cos i\Omega \cos j\Omega]$$

$$= \sum\cos^2 j\Omega \sum E[x_i x_j]\cos(i-j)\Omega - (\sum \sin 2j\Omega \sum E[x_i x_j]\sin(i-j)\Omega)/2$$

and

$$E[X_I^2] = E[\sum\sum x_i x_j \sin i\Omega \sin j\Omega]$$

$$= \sum\sin^2 j\Omega \sum E[x_i x_j]\cos(i-j)\Omega + (\sum \sin 2j\Omega \sum E[x_i x_j]\sin(i-j)\Omega)/2$$

(A1.1.1)

define

$$E[X_R^2] + E[X_I^2] = N\sum E[x_i x_j]\cos(i-j)\Omega = N f_1 \qquad \text{(A1.1.2)}$$

$$(\sum \sin 2j\Omega \sum E[x_i x_j]\sin(i-j)\Omega)/2 = (\sum_{\tau}^{N-1} \hat{R}(\tau)\sin(\tau\Omega)\sin 2\Omega)/2 = f_2 \sin 2\Omega$$

(A1.1.3)

where

$$\hat{R}(\tau) = Ave\ (y_i y_{i+\tau}) = estimate\ of\ E(x_i x_{i+\tau}) \qquad \text{(A1.1.4)}$$

similarly the remaining terms are derived, which are

$$E[X_R Y_R] = E[X_R^2 - X_R X_{mR}]$$

$$= f_1(\frac{N}{2} - \sum\cos^2 m\Omega) + f_2 \gamma_1 \qquad \text{(A1.1.5)}$$

where

$$E[X_R X_{mR}] = \sum\cos^2 m\Omega \sum E[x_i x_m]\cos(i-m)\Omega - (\sum \sin 2jm\Omega \sum E[x_i x_m]\sin(i-j)\Omega)/2$$

$$= f_1 \sum\cos^2 m\Omega - f_2 \gamma_1$$

$$\gamma_1 = \begin{cases} \sum_{m}^{\tau} \sin 2m\Omega - \sum_{m \geq N-\tau+1}^{N} \sin 2m\Omega, & \tau \leq N/2 \\ \sum_{m}^{N-\tau} \sin 2m\Omega - \sum_{m \geq \tau+1}^{N} \sin 2m\Omega, & \tau > N/2 \end{cases} \tag{A1.1.6}$$

$$E[X_I Y_I] = E[X_I^2 - X_I X_{mI}]$$

$$= f_1\left(\frac{N}{2} - \sum \sin^2 m\Omega\right) - f_2 \gamma_1 \tag{A1.1.7}$$

$$E[Y_R^2] = E[X_R^2 - 2X_R X_{mR} + X_{mR}^2]$$

$$= f_1\left(\frac{N}{2} - 2\sum \cos^2 m\Omega\right) + 2f_2 \gamma_1 + \sigma^2 \sum \cos^2 m\Omega \tag{A1.1.8}$$

where

$$E[X_{mR}^2] \approx \sigma^2 \sum \cos^2 m\Omega \tag{A1.1.9}$$

$$E[Y_I^2] = E[X_I^2 - 2X_I X_{mI} + X_{mI}^2]$$

$$= f_1\left(\frac{N}{2} - 2\sum \sin^2 m\Omega\right) - 2f_2 \gamma_1 + \sigma^2 \sum \sin^2 m\Omega \tag{A1.1.10}$$

where

$$E[X_{mI}^2] \approx \sigma^2 \sum \sin^2 m\Omega \tag{A1.1.11}$$

Define cross part

$$E[X_I X_R] = E[\sum\sum x_i x_j \sin i\Omega \cos j\Omega]$$

$$= \sum \cos^2 j\Omega \sum E[x_i x_j] \sin (i-j)\Omega + (\sum \sin 2j\Omega \sum E[x_i x_j] \cos (i-j)\Omega)/2$$

$$= \sum_{\tau=1}^{N-1} \hat{R}(\tau) \sin^3 \tau\Omega = \beta_1 \tag{A1.1.12}$$

noticing that

$$\cos^2 j\Omega = \cos^2 (N-j)\Omega \quad , \quad \cos^2 j\Omega - \cos^2 (N\Omega) = \sin^2 j\Omega \tag{A1.1.13}$$

$$E[X_I Y_R] = E[X_I X_R - X_I X_{mR}]$$

$$= \beta_1 - \gamma_2 - f_1 \sum \cos m\Omega \sin m\Omega \qquad \text{(A1.1.14)}$$

where

$$E[X_I X_{mR}] = \sum \cos^2 m\Omega \sum E[x_i x_m] \sin(i-m)\Omega + (\sum \sin 2m\Omega \sum E[x_i x_m] \cos(i-m)\Omega)/2$$

$$= \gamma_2 + f_1 \sum \cos m\Omega \sin m\Omega$$

$$\gamma_2 = \begin{cases} \sum_{\tau=1}^{N-1} \hat{R}(\tau) \sin(\tau\Omega)\left(\sum_{m}^{\tau} \cos^2 m\Omega - \sum_{m \geq N-\tau+1}^{N} \cos^2 m\Omega\right), & \tau \leq N/2 \\ \sum_{\tau=1}^{N-1} \hat{R}(\tau) \sin(\tau\Omega)\left(\sum_{m}^{N-\tau} \cos^2 m\Omega - \sum_{m \geq \tau+1}^{N} \cos^2 m\Omega\right), & \tau > N/2 \end{cases} \qquad \text{(A1.1.15)}$$

$$E[X_R Y_I] = E[X_R X_I - X_R X_{mI}] = E[X_I Y_R] \qquad \text{(A1.1.16)}$$

$$E[Y_R Y_I] = E[X_R X_I - X_R X_{mI} - X_I X_{mR} + X_{mR} X_{mI}]$$

$$= \beta_1 - 2\gamma_2 + \sigma^2 \sum \cos m\Omega \sin m\Omega \qquad \text{(A1.1.17)}$$

where

$$E[X_{mR} X_{mI}] \approx \sigma^2 \sum \cos m\Omega \sin m\Omega \qquad \text{(A1.1.18)}$$

It should be noticed that the parameters f_2, β_1, γ_1, and γ_2 are all much smaller than f_1, therefore these have no important effect on the Fourier transform and PSD estimates. Simulation results have confirmed this in detail.

A1.2 Direct algorithm

The five factors f_1 and f_2, γ_1, γ_2, and β_1 have been defined in the part A2.1. In this section, they will be used to derive a direct algorithm for the various estimates. Consider two variable substitutions

$$X_R = Y_R + X_{mR}$$

$$X_I = Y_I + X_{mI} \tag{A1.2.1}$$

The remaining terms may be derived with the substitutions, which are

$$E[X_R Y_R] = E[Y_R^2 + X_R X_{mR} - X_{mR}^2]$$

$$= E[Y_R^2] + (f_1 - \sigma^2)\sum \cos^2 m\Omega - f_2\gamma_1 \tag{A1.2.2}$$

where

$$E[X_R X_{mR}] = \sum \cos^2 m\Omega \sum E[x_i x_m] cos\,(i-m)\Omega - (\sum sin\,2jm\,\Omega \sum E[x_i x_m] sin\,(i-j)\Omega)/2$$

$$= f_1 \sum \cos^2 m\Omega - f_2\,\gamma_1$$

$$E[X_{mR}^2] \approx \sigma^2 \sum \cos^2 m\Omega \tag{A1.2.3}$$

Similarly

$$E[X_I Y_I] = E[Y_I^2 + X_I X_{mI} - X_{mI}^2]$$

$$= E[Y_I^2] + (f_1 - \sigma^2)\sum \sin^2 m\Omega + f_2\gamma_1 \tag{A1.2.4}$$

where

$$E[X_{mI}^2] \approx \sigma^2 \sum \sin^2 m\Omega \tag{A1.2.5}$$

$$E[X_R Y_I] = E[X_I Y_R] = E[Y_R Y_I + X_I X_{mR} - X_{mR} X_{mI}]$$

$$= E[Y_R Y_I] + \gamma_2 + (f_1 - \sigma^2)\sum \cos m\Omega \sin m\Omega \tag{A1.2.6}$$

where

$$E[X_I X_{mR}] = \sum \cos^2 m\Omega \sum E[x_i x_m] \sin(i-m)\Omega + (\sum \sin 2m\Omega \sum E[x_i x_m] \cos(i-m)\Omega)/2$$

$$= \gamma_2 + f_1 \sum \cos m\Omega \sin m\Omega$$

$$E[X_{mR} X_{mI}] \approx \sigma^2 \sum \cos m\Omega \sin m\Omega \qquad \text{(A1.2.7)}$$

As mentioned in appendix A1.1, f_2, β_1, γ_1 and γ_2 may be neglected in the calculations.

Chapter 2 Frequency response estimation

2.1 Introduction

The importance of frequency response characteristics is well known for system identification and controller design. The essence of frequency response estimation is the estimation of cross-spectrum between input and output and input auto-spectrum of a system. This chapter studies the estimation of the frequency response function from input and output data of a system in which there exist missing points at the output. Formally speaking, as far as author know, there had not been any published result until the authors' publication (Douce and Zhu 1988) even though one can naturally borrow idea and techniques from power spectral density estimation with missing data.

The definition of the frequency response estimation is given by, without missing data,

$$G = \bar{S}_{vx} / \bar{S}_{vu} \tag{2.1.1}$$

With reference to Fig. 2.1.1, S_{vx} is a cross-spectrum between input and output of a system, S_{vu} is a cross-spectrum between input to the system and input to the plant, which equals to S_{uu}, the auto-spectrum of the system input, for open loop systems.

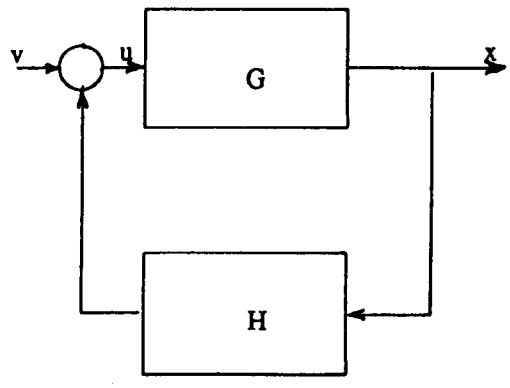
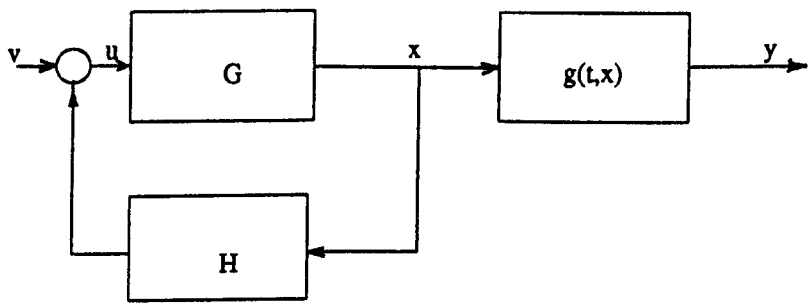


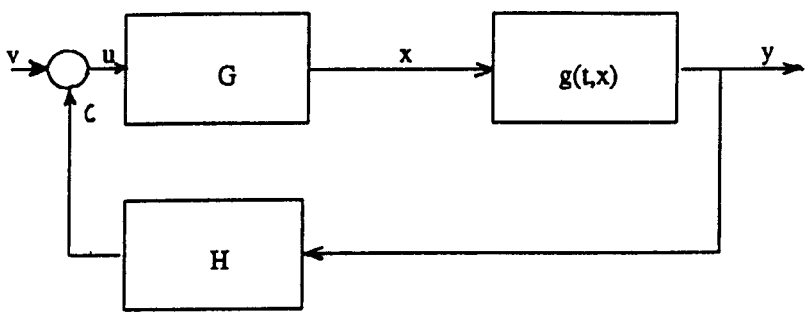
Figure 2.1.1 Closed-loop system

The problems studied are shown in Fig. 2.1.2. The modulating function g(t,x) has been defined previously, with g(t,x) = 0 for a missing point and unity otherwise.

(a)



(b)



(c)

Figure 2.2.1 Systems with missing points at output

Problem 1: open loop system with missing data at output, shown in Fig. 2.1.2(a). This is the simplest case. Douce and Zhu (1988) defined the frequency response estimate to be, with missing data at output,

$$\hat{G} = \bar{S}_{v\hat{x}} / \bar{S}_{vv} \tag{2.1.2}$$

where $S_{v\hat{x}}$ is an estimate of S_{vx}.

Problem 2: closed loop system with missing data at output, seen in Fig. 2.1.2(b). This presents for example the case in which the recording instrument fails, but the sensor which generates the feedback signal is working normally. The estimation procedure is the same as eqn(2.1.2).

Problem 3: A closed loop system with missing data at output, effecting the feedback path, shown in Fig. 2.1.2(c). This is a case in which the recording instrument and sensor fail simultaneously. A definition of the frequency response estimation is given by

$$\hat{G} = \bar{S}_{v\hat{x}} / \bar{S}_{v\hat{u}} \tag{2.1.3}$$

At present, we can not give a general rule to deal with this problem, the difficulty being the estimation of the cross-spectrum $S_{v\hat{c}}$ in which the missing data appears at the plant input. However it can be solved under certain conditions such as unit feedback (H=1), corresponding to

$$\hat{G} = \bar{S}_{v\hat{x}} / \bar{S}_{vu} \tag{2.1.4}$$

As shown later, this is the same as eqn(2.1.2). The simplified situation, that is unit feedback (H = 1), presents a heuristic solution to the problem in the chapter.

As mentioned above, the frequency response estimation mainly depends on the cross-spectrum $S_{v\hat{x}}$ no matter whether open loop or closed loop is considered. The idea and techniques developed in PSD estimation with missing data are developed further to solve the problems. Both recursive and direct estimators will be considered with UNB or MSE performance criteria.

2.2 Estimator for frequency response

2.2.1 Unbiased estimate

Firstly the UNB estimator is presented in term of real and imaginary parts respectively, that is k_4 and k_5 are chosen to satisfy

$$E[Re(U^*X) - k_4 Re(U^*\hat{X})] = E[(U_R X_R + U_I X_I) - k_4(U_R Y_R + U_I Y_I)] = 0$$

$$E[Im(U^*X) - k_5 Im(U^*\hat{X})] = E[(U_R X_I - U_I X_R) - k_5(U_R Y_I - U_I Y_R)] = 0 \tag{2.2.1}$$

where Re and Im denote the real and imaginary parts respectively. Solving the linear equation set, gives

$$k_4 = \frac{A_0}{A_1 + A_2} = \frac{N}{N-M}$$

$$k_5 = \frac{B_0}{B_1 + B_2} = \frac{N}{N-M} \tag{2.2.2}$$

where

$$A_0 = E[U_R X_R + U_I X_I]$$

$$A_1 = E[U_R Y_R]$$

$$A_2 = E[U_I Y_I]$$

$$B_0 = E[U_R X_I - U_I X_R]$$

$$B_1 = E[U_I Y_R]$$

$$B_2 = E[U_R Y_I] \tag{2.2.3}$$

This important result demonstrates that the PM, derived on an ad-hoc basis for spectral analysis, is, when the stated assumptions are satisfied, an unbiased estimate of the cross-spectrum. It therefore leads to unbiased estimates of the system frequency response.

Appendices A2.1 and A2.2 give the detailed derivations of these expressions.

2.2.2 Minimum mean square estimate

Secondly the MSE estimator is presented in term of minimisation of

$$E[(U^*X - U^*\hat{X})(U^*X - U^*\hat{X})^*] \tag{2.2.4}$$

Two kinds of gains can be obtained from this criterion. One directly compensates the output Fourier transformation Y, and another one indirectly compensates the cross-spectrum.

For the first case, let

$$\hat{X} = k_4 Y_R + k_5 Y_I \tag{2.2.5}$$

differentiating eqn(2.2.5) with respect to k_4, and setting the result to zero and noting that u and y are known,

$$k_4 = \frac{E[S_{uu} X_R Y_R]}{S_{uu} Y_R^2} = \frac{E[U_R^2 X_R Y_R + U_I^2 X_R Y_R]}{U_R^2 Y_R^2 + U_I^2 Y_R^2} \tag{2.2.6}$$

and similarly

$$k_5 = \frac{E[S_{uu} X_I Y_I]}{S_{uu} Y_I^2} = \frac{E[U_R^2 X_I Y_I + U_I^2 X_I Y_I]}{U_R^2 Y_I^2 + U_I^2 Y_I^2} \tag{2.2.7}$$

where

$$\begin{aligned}
E[U_R^2 X_R Y_R] &= U_R^2 E[X_R Y_R] + 2E[U_R X_R] U_R Y_R \\
E[U_I^2 X_R Y_R] &= U_I^2 E[X_R Y_R] + 2E[U_I X_R] U_I Y_R \\
E[U_R^2 X_I Y_I] &= U_R^2 E[X_I Y_I] + 2E[U_R X_I] U_R Y_I \\
E[U_I^2 X_I Y_I] &= U_I^2 E[X_I Y_I] + 2E[U_I X_I] U_I Y_I
\end{aligned} \tag{2.2.8}$$

For the second case, let

$$U^*\hat{X} = k_4 Re(U^*Y) + jk_5 Im(U^*Y) \tag{2.2.9}$$

differentiating eqn(2.2.10) with respect to k_4, and setting the result to zero gives

$$\begin{aligned}
k_4 &= \frac{E[Re(U^*X) Re(U^*Y)]}{(Re(U^*Y))^2} \\
&= \frac{E[(U_R X_R + U_I X_I)(U_R Y_R + U_I Y_I)]}{(U_R Y_R + U_I Y_I)^2}
\end{aligned} \tag{2.2.10}$$

and similarly

$$k_5 = \frac{E[Im(U^*X)Im(U^*Y)]}{(Im(U^*Y))^2}$$

$$= \frac{E[(U_R X_I - U_I X_R)(U_R Y_I - U_I Y_R)]}{(U_R Y_I - U_I Y_R)^2} \qquad (2.2.11)$$

It should be noted that although the two derived gains have different values, they will produce the frequency response estimate with minimum MSE, i.e. the resultant estimate is the same since the same performance criterion is chosen.

Fig. 2.2.1 shows the recursive relationship of the estimators in block diagram form.

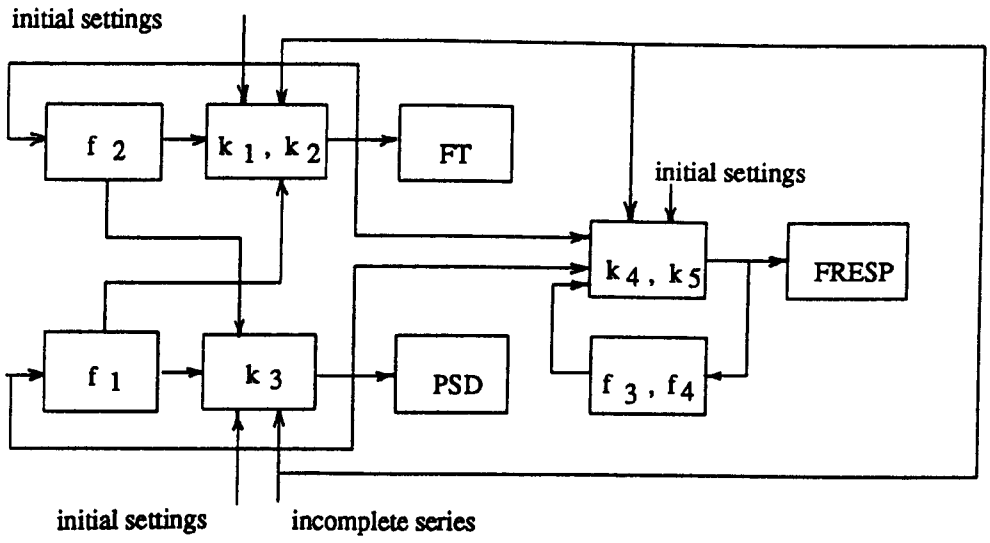


Figure 2.1.2 Recursive estimators

2.3 Properties of the new estimators

2.3.1 Closed-loop system

The first property concerns the equivalence between problem 2 and problem 3 when the system is unity feedback, (H=1). With reference to Fig. 2.1.2(c)

For no missing points,

$$u(t) = v(t) - x(t) \tag{2.3.1}$$

With missing points, it has

$$\begin{aligned} u_m(t) &= v(t) - x_m(t) = v(t) - x(t) + d(t) \\ &= u(t) + d(t) \end{aligned} \tag{2.3.2}$$

where

$$d(t) = (1 - g(t,x))x(t) \tag{2.3.3}$$

may be recognised as a disturbance with values equal to the values at missing points and zero otherwise, and is white noise due to the condition given in PSD estimation that missing points are separated such that cross-correlation between values at missing points may be neglected.

Let D be the Fourier transform of the disturbance d(t), from eqn(2.1.3), this leads to

$$\begin{aligned} \hat{G} &= \bar{S}_{v\hat{x}} / \bar{S}_{v\hat{u}} \\ &= E[V^* \hat{X}] / E[V^* \hat{U}] \\ &= E[V^* \hat{X}] / E[V^* U] \\ &= \bar{S}_{v\hat{x}} / \bar{S}_{vu} \end{aligned} \tag{2.3.4}$$

This the same as eqn(2.1.2), where

$$\begin{aligned} & E[V^* \hat{U}] \\ &= E[V^* U + V^* D] \\ &= E[V^* U] + E[V^* D] \end{aligned}$$

$$= E[V^* U] + E[V^*]E[D]$$

$$= E[V^* U] \tag{2.3.5}$$

This follows since E[D] = 0.

2.3.2 Spurious periodicity

The spurious periodic phenomenon predicted and observed in PSD estimation is not presented in the frequency response estimate. With reference to appendices A2.1 and A2.2, this can be proved by considering

$$|E[U^* Y]|$$

$$= |E[(U_R + jU_I)((X_R - X_{mR}) - j(X_I - X_{mI}))]|$$

$$= |E[(U_R + jU_I)((X_R - jX_I) - (X_{mR} - jX_{mI}))]|$$

$$= |G\bar{S}_{vu}| - |E[(U_R + jU_I)(X_{mR} - jX_{mI})]|$$

$$= |G\bar{S}_{vu}| - c\,|f_3 + jf_4|$$

$$= |G\bar{S}_{vu}| - c\,|G| \tag{2.3.6}$$

where c is a constant. If frequency response G has low-pass filter property, then

$$\lim_{\Omega \to \pi} G \to 0 \tag{2.3.7}$$

or

$$G \approx 0 \quad , \quad h < \Omega < \pi \tag{2.3.8}$$

where h is the cut-off frequency. Hence there is no spurious periodic phenomenon at high harmonics in frequency response estimation. However it may be proved that this phenomenon is present in measured squared cross spectrum. Consider

$$|E[(U^* Y)(U^* Y)^*]| = |E[(U^* U Y^* Y)]|$$

$$= |E[(U_R^2 + U_I^2)(Y_R^2 + Y_I^2)]|$$

$$= |E[U_R^2]E[Y_R^2] + 2E[U_R Y_R]$$

$$+ E[U_I^2]E[Y_I^2] + 2E[U_I Y_I]$$

$$+ E[U_R^2]E[Y_I^2] + 2E[U_R Y_I]$$

$$+ E[U_I^2]E[Y_R^2] + 2E[U_I Y_R]\,|$$

$$= |E[(U_R^2 + U_I^2)]E[(Y_R^2 + Y_I^2)] + 2[E[U_R Y_R] + E[U_I Y_I] + E[U_R Y_I] + E[U_I Y_R]]\,| \tag{2.3.9}$$

Chapter 1 has demonstrated the spurious periodicity in $E[(Y_R^2 + Y_I^2)]$ from both theory and experiment. $E[(U_R^2 + U_I^2)]$ is a constant for uncorrelated Gaussian input signals. The terms $E[U_R Y_R] = E[U_I Y_I] = E[U_R Y_I] = E[U_I Y_R] \to 0$ since $E[U^* Y] \to 0$ when $\Omega \to \pi$ for low-pass filter systems.

2.3.3 Direct estimators

The third feature is that the recursive estimators may be modified to be direct estimators by the same variable substitutions as eqn(1.4.19) in Chapter1. This gives a significant saving in computing time. A detailed derivation is given in appendix A2.2.

2.4 Experiment results

The three problems mentioned in the introduction have been studied with the new methods, UNB and MSE, and traditional methods CM and IM as comparisons. The same performance criteria as used previously have been evaluated in each case.

The error performances and linear fit defined in chapter 1 are modified to be the measurements of magnitude instead of amplitude, to indicate both amplitude and phase characteristics.

For SYS 1, Fig. 2.4.1 and Fig. 2.4.2 show the results in an experiment consisting of 1000 blocks with 10% missing points and 20% missing points respectively, and compare the new methods, UNB and MSE, with the traditional method CM to demonstrate the improved performances predicted in theory. Periodic errors are found in the mean squared error due to periodic missing points.

For SYS 2, Fig. 2.4.3 and Fig. 2.4.4 again show significant improvements.

Table 2.4.1 summarizes the results, from experiments with one, five, ten, and one hundred blocks, to indicate the significant improvements in the reduction of the errors. Table 2.4.2 shows the ensemble average of linear regression of the true on the estimated frequency response function, the new estimators substantially reduce the deviation of this factor from unity. The experiment again confirms the restriction of interpolation method.

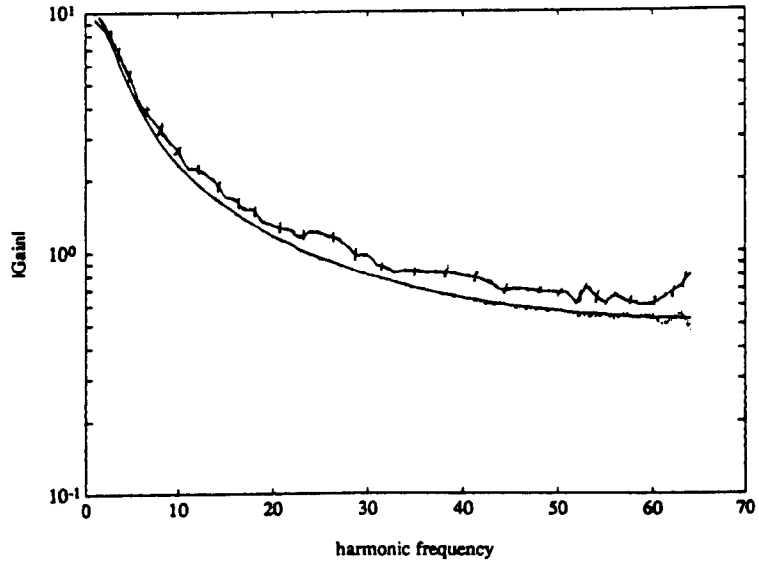


Figure 2.4.1a. Frequency response estimates (sys1, mis 10%).
———— with no missing points
-/-/-/-/-/-/-/- covariance method
..... minimum bias
— · — · — minimum mean square error

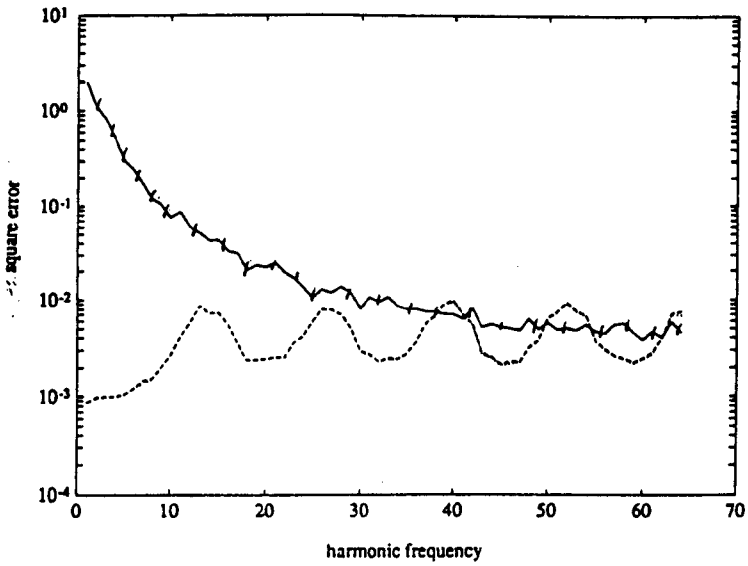


Figure 2.4.1b. Error squared as a function of harmonic.
-/-/-/-/-/-/-/- covariance method
..... minimum bias
— · — · — minimum mean square error

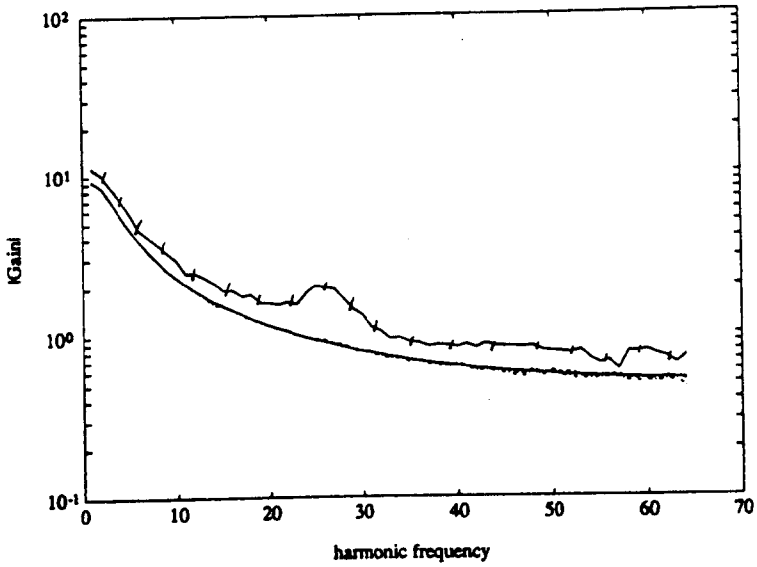


Figure 2.4.2a. Frequency response estimates (sys1, mis 20%).

———— with no missing points

-/-/-/-/-/-/-/-/- covariance method

..... minimum bias

— · — · — minimum mean square error

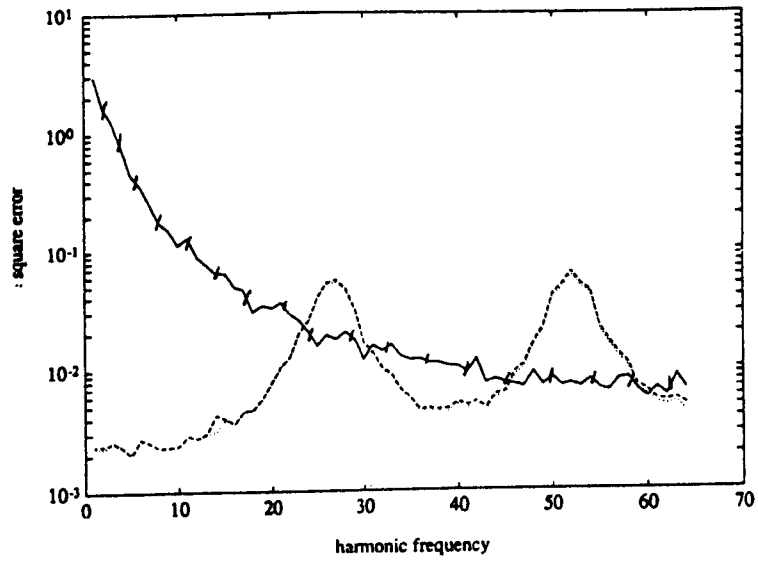


Figure 2.4.2b. Error squared as a function of harmonic.

-/-/-/-/-/-/-/-/- covariance method

..... minimum bias

— · — · — minimum mean square error

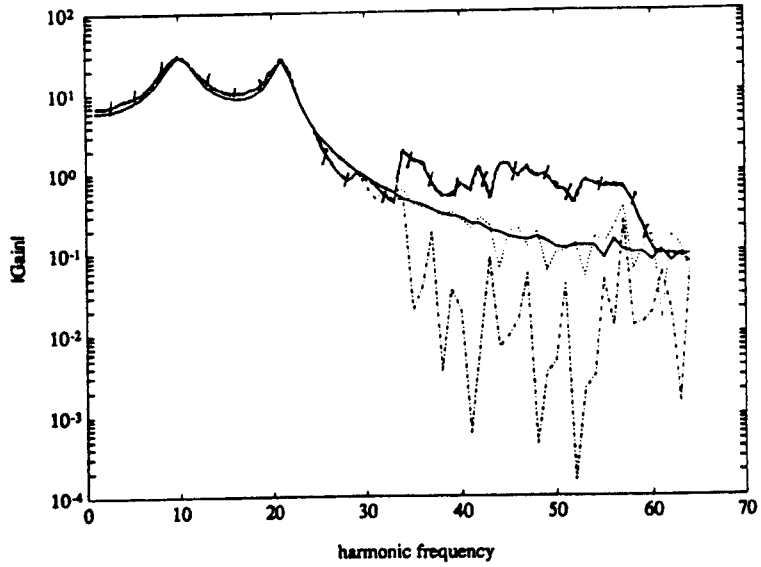


Figure 2.4.3a. Frequency response estimates (sys2, mis 10%).
———— with no missing points
-/-/-/-/-/-/-/- covariance method
..... minimum bias
— · — · — minimum mean square error

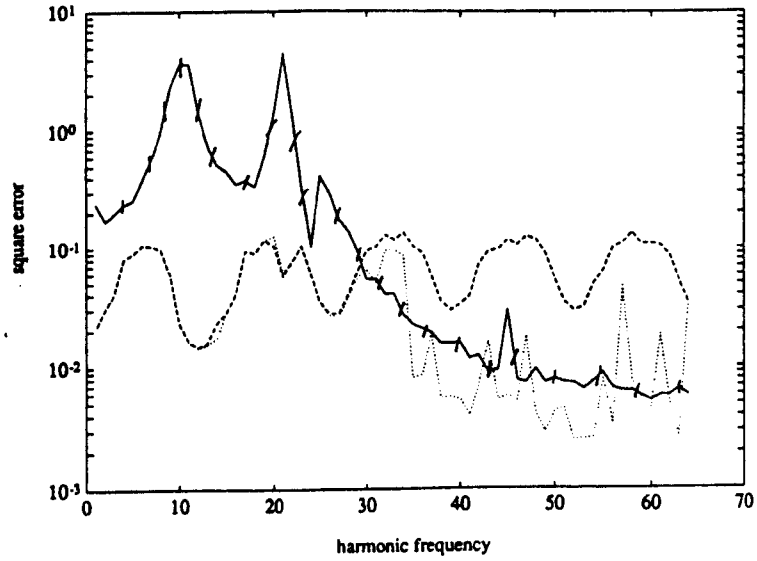


Figure 2.4.3b. Error squared as a function of harmonic.
-/-/-/-/-/-/-/- covariance method
..... minimum bias
— · — · — minimum mean square error

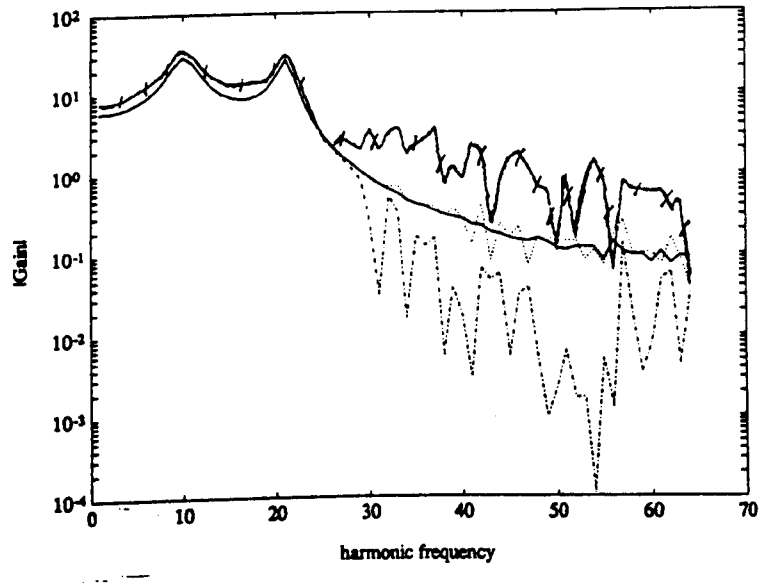


Figure 2.4.4a. Frequency response estimates (sys2, mis 20%).

———— with no missing points

-/-/-/-/-/-/-/-/- covariance method

..... minimum bias

— . — . — minimum mean square error

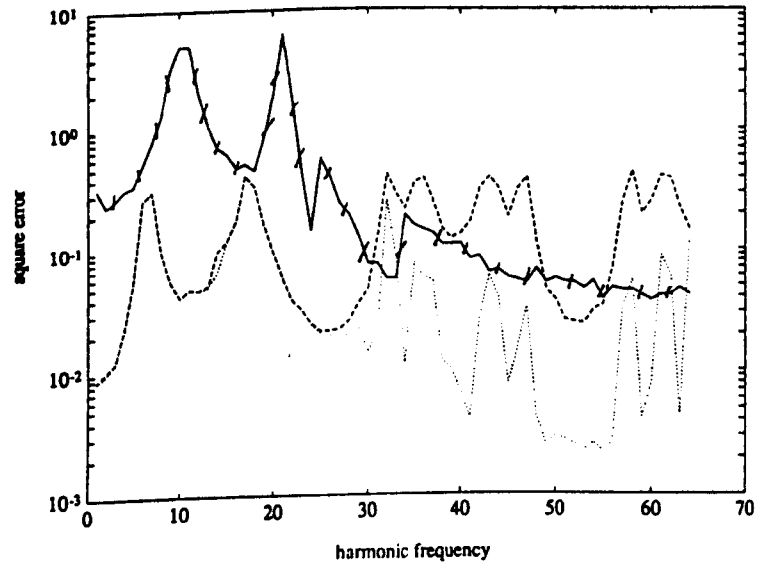


Figure 2.4.4b. Error squared as a function of harmonic.

-/-/-/-/-/-/-/-/- covariance method

..... minimum bias

— . — . — minimum mean square error

Errors in frequency response estimation

data length (10% missing)		CM		IM		UNB		MSE	
		e1	e2	e1	e2	e1	e2	e1	e2
sys1	128	16.15	30.98	0.059	0.129	1.078	2.523	1.081	2.229
sys2	128	64.65	85.19	0.926	2.435	5.732	9.418	5.331	9.001
sys3	128	0.047	0.215	0.054	0.223	0.025	0.157	0.028	0.149
sys1	128*5	2.631	8.001	0.018	0.067	0.097	0.438	0.098	0.433
sys2	128*5	14.22	17.37	0.322	1.273	2.768	4.853	2.903	4.370
sys3	128*5	0.035	0.174	0.037	0.180	0.018	0.134	0.019	0.129
sys1	1280	2.047	6.999	0.010	0.058	0.057	0.583	0.061	0.575
sys2	1280	10.94	12.74	0.135	0.818	0.597	3.106	0.721	2.909
sys3	1280	0.018	0.139	0.021	0.145	0.012	0.089	0.017	0.086
sys1	12800	1.839	5.566	0.005	0.057	0.009	0.536	0.010	0.528
sys2	12800	8.661	11.11	0.058	0.678	0.025	3.009	0.029	2.898
sys3	12800	0.012	0.135	0.013	0.139	0.001	0.102	0.002	0.101
data length (20% missing)		CM		IM		UNB		MSE	
		e1	e2	e1	e2	e1	e2	e1	e2
sys1	128	18.17	52.19	0.085	0.210	2.872	5.137	2.929	4.997
sys2	128	183.4	372.4	1.134	2.857	14.43	19.28	15.84	18.55
sys3	128	0.165	0.377	0.177	0.392	0.048	0.307	0.050	0.301
sys1	128*5	7.694	20.66	0.042	0.182	0.275	1.417	0.278	1.175
sys2	128*5	88.67	143.2	0.551	1.558	4.306	19.47	4.542	18.15
sys3	128*5	0.091	0.314	0.093	0.355	0.043	0.253	0.044	0.249
sys1	1280	5.404	12.93	0.034	0.146	0.120	1.322	0.131	1.311
sys2	1280	18.30	41.45	0.249	1.077	1.417	8.267	1.646	5.616
sys3	1280	0.072	0.298	0.075	0.306	0.022	0.242	0.025	0.237
sys1	12800	3.611	8.317	0.020	0.124	0.026	1.177	0.028	1.128
sys2	12800	14.29	25.71	0.201	1.401	0.038	7.625	0.057	5.007
sys3	12800	0.045	0.237	0.048	0.298	0.002	0.233	0.003	0.199

Table 2.4.1

Fit in frequency response estimation					
data length (10% missing)		CM fit	IM fit	UNB fit	MSE fit
sys1	128	0.78	1.092	1.023	1.024
sys2	128	0.707	1.068	0.784	0.789
sys3	128	0.837	0.824	0.891	0.890
sys1	128*5	0.874	1.009	0.981	0.980
sys2	128*5	0.811	1.064	0.973	0.969
sys3	128*5	0.910	1.099	1.045	1.052
sys1	1280	0.877	1.005	1.007	1.008
sys2	1280	0.813	1.016	0.987	0.983
sys3	1280	1.033	1.093	0.989	0.988
sys1	12800	0.899	1.001	1.002	1.003
sys2	12800	0.828	1.011	0.989	0.985
sys3	12800	1.044	1.082	0.994	0.992
data length (20% missing)		CM fit	IM fit	UNB fit	MSE fit
sys1	128	0.323	1.099	0.460	0.456
sys2	128	0.242	1.092	0.604	0.584
sys3	128	0.792	0.789	0.858	0.855
sys1	128*5	0.707	1.013	0.946	0.945
sys2	128*5	0.647	1.082	0.969	0.963
sys3	128*5	0.810	1.137	0.924	0.922
sys1	1280	0.796	1.009	0.979	0.976
sys2	1280	0.723	1.036	0.974	0.971
sys3	1280	1.059	1.134	0.972	0.970
sys1	12800	0.797	1.002	1.002	1.003
sys2	12800	0.741	1.002	1.003	1.004
sys3	12800	1.053	1.114	0.990	0.989

Table 2.4.2

2.5 Appendix 2

A2.1 Recursive algorithm

This part will define f_3, f_4, and derive some relevant terms in eqn(2.2.6) and eqn(2.2.7). Consider

$$E[U_R X_R] = E[\sum\sum u_i x_j \cos i\Omega \cos j\Omega]$$

$$= \sum \cos^2 j\Omega \sum E[u_i x_j] \cos(i-j)\Omega - (\sum \sin 2j\Omega \sum E[u_i x_j] \sin(i-j)\Omega)/2 \quad \text{(A2.1.1)}$$

similarly

$$E[U_I X_I] = \sum \sin^2 j\Omega \sum E[u_i x_j] \cos(i-j)\Omega + (\sum \sin 2j\Omega \sum E[u_i x_j] \sin(i-j)\Omega)/2 \quad \text{(A2.1.2)}$$

define

$$E[U_R X_R] + E[U_I X_I] = N \sum E[u_i x_j] \cos(i-j)\Omega = N f_3 \quad \text{(A2.1.3)}$$

f_3 represents the real part of the cross-spectrum. Consider

$$E[U_I X_R] = E[\sum\sum u_i x_j \sin i\Omega \cos j\Omega]$$

$$= \sum \cos^2 j\Omega \sum E[u_i x_j] \sin(i-j)\Omega + (\sum \sin 2j\Omega \sum E[u_i x_j] \cos(i-j)\Omega)/2 \quad \text{(A2.1.4)}$$

similarly

$$E[U_R X_I] = -\sum \sin^2 j\Omega \sum E[u_i x_j] \sin(i-j)\Omega + (\sum \sin 2j\Omega \sum E[u_i x_j] \cos(i-j)\Omega)/2 \quad \text{(A2.1.5)}$$

define

$$E[U_I X_R] - E[U_R X_I] = N \sum E[u_i x_j] \sin(i-j)\Omega = N f_4 \quad \text{(A2.1.6)}$$

f_4 represents the imaginary part of the cross-spectrum.

The rest of the terms can be derived according to the definitions, which are

$$U_R Y_R = E[U_R X_R - U_R X_{mR}]$$

$$= f_3(\frac{N}{2} - \sum \cos^2 m\Omega) + f_4 \sum \cos m\Omega \sin m\Omega \qquad \text{(A2.1.7)}$$

where

$$E[U_R X_{mR}] = f_3 \sum \cos^2 m\Omega - f_4 \sum \cos m\Omega \sin m\Omega$$

$$U_I Y_I = E[U_I X_I - U_I X_{mI}]$$

$$= f_3(\frac{N}{2} - \sum \sin^2 m\Omega) - f_4 \sum \cos m\Omega \sin m\Omega \qquad \text{(A2.1.8)}$$

where

$$E[U_I X_{mI}] = f_3 \sum \sin^2 m\Omega + f_4 \sum \cos m\Omega \sin m\Omega$$

$$U_I Y_R = E[U_I X_R - U_I X_{mR}]$$

$$= f_4(\frac{N}{2} - \sum \cos^2 m\Omega) - f_3 \sum \cos m\Omega \sin m\Omega \qquad \text{(A2.1.9)}$$

where

$$E[U_I X_{mR}] = f_4 \sum \cos^2 m\Omega + f_3 \sum \cos m\Omega \sin m\Omega$$

$$U_R Y_I = E[U_R X_I - U_R X_{mI}]$$

$$= -f_4 \frac{N}{2} - \sum \sin^2 m\Omega - f_3 \sum \cos m\Omega \sin m\Omega \qquad \text{(A2.1.10)}$$

where

$$E[U_R X_{mI}] = -f_4 \sum \sin^2 m\Omega + f_3 \sum \cos m\Omega \sin m\Omega$$

A2.2 Direct algorithm

This part will present a direct algorithm by the following substitutions

$$X_R = Y_R + X_{mR}$$

$$X_I = Y_I + X_{mI} \qquad \text{(A2.2.1)}$$

The remaining terms may be derived similarly, to give

$$E[U_R X_R] = E[U_R Y_R + U_R X_{mR}]$$

$$= E[U_R Y_R] + f_3 \sum \cos^2 m\Omega + f_4 \sum \cos m\Omega \sin m\Omega \qquad \text{(A2.2.2)}$$

where

$$E[U_R X_{mR}] = f_3 \sum \cos^2 m\Omega - f_4 \sum \cos m\Omega \sin m\Omega$$

$$E[U_I X_I] = E[U_I Y_I + U_I X_{mI}]$$

$$= E[U_I Y_I] + f_3 \sum \sin^2 m\Omega + f_4 \sum \cos m\Omega \sin m\Omega \qquad \text{(A2.2.3)}$$

where

$$E[U_I X_{mI}] = f_3 \sum \sin^2 m\Omega + f_4 \sum \cos m\Omega \sin m\Omega$$

$$E[U_I X_R] = E[U_I Y_R + U_I X_{mR}]$$

$$= E[U_I Y_R] + f_4 \sum \cos^2 m\Omega + f_3 \sum \cos m\Omega \sin m\Omega \qquad \text{(A2.2.4)}$$

where

$$E[U_I X_{mR}] = f_4 \sum \cos^2 m\Omega + f_3 \sum \cos m\Omega \sin m\Omega$$

$$E[U_R X_I] = E[U_R Y_I + U_R X_{mI}]$$

$$= E[U_R Y_I] - f_4 \sum \sin^2 m\Omega + f_3 \sum \cos m\Omega \sin m\Omega \qquad \text{(A2.2.5)}$$

where

$$E[U_R X_{mI}] = -f_4 \sum \sin^2 m\Omega + f_3 \sum \cos m\Omega \sin m\Omega$$

Section 2 System identification and parameter estimation

S2.1 Survey

The need for a system model in controller design for both linear and nonlinear system is well known. Critical points in system identification are the detection of system structure, the determination of the order of the system, finally the estimation of the parameters of the system when a parametric model is suitable for the description of the system. For linear system identification, many methods have been developed successfully. The tutorial text book in theory and application by Ljung (1987) gives a good survey and unifies many fundamental methods as a set of tool boxes.

For nonlinear system identification there are two distinct methods, producing either a linear approximating model or alternatively a nonlinear model. In the first case structure detection is reduced to the order and time delay determination (Billings and Voon 1983). Many algorithms may be then used to estimate the parameters of the system models, and a linear covariance analysis of the residuals can be applied to test the adequacy of the fitted model. The most popular linear model is the Auto-Regressive Moving Average (ARMA) model which may be used to approximate some system with mild nonlinearities around operating points. However linear approximating model is inappropriate under some circumstances.

The second method has better accuracy but tends to be more complicated and problem specific. Some representative parametric models have been studied such as Hammerstein model (Narendra and Gallman 1966), Nonlinear Auto-Regressive Moving Average model with eXogenous inputs (Leontaritis and Billings 1985), Bilinear model (Svoronos, Stephanopoulos, and Aris 1981, Fnaiech and Ljung 1987), and nonlinear output affine model (Chen and Billings 1988), in which attention was concentrated on parameter estimation with the assumption that the system structure is known. Furthermore, traditional models like Volterra (1930) or Wiener series (Wiener 1958, Billings 1980) have been studied for long time, but the implications for controller design makes them inferior to those

models mentioned above.

The parameter estimation of nonlinear models, particularly nonlinear models which are linear in the parameters, have been studied in detail. However so far nonlinear system structure detection has been only investigated by a few authors (e.g. West 1965, Douce 1976, Billings and Voon 1983) due to the associated difficulties.

In the section, attention is concentrated on parametric models. A concept considered seldom in system identification and parameter estimation, amplitude effect to nonlinear system, is introduced, in other applications even though it has been known for a long time. Almost all authors paid attention to time functions, that is the evolution of input and output series along time process. In the case of system identification and parameter estimation, this is appropriate for linear systems. However it is well understood that the characteristics of nonlinear systems depend on signal amplitude. Whereas nonlinear system identification and parameter estimation should consider both time axis and amplitude axis, unfortunately the latter is often ignored without any explanation in many publications.

In summary, chapter three presents a new method, regarding to amplitude sampling, to detect the structure of a class of nonlinear systems and to estimate the systems parameters. With the aid of multi-dimensional graphics a clear geometrical interpretation is obtained. The nonlinear characteristic of the systems, dynamic or static, can be directly identified. Following the idea, a recursive parameter estimation procedure is developed in which the parameters to be estimated are the slopes on the corresponding projecting planes. The principle and implementation of the method are explained in detail and a simulation experiment confirms the validity and efficiency of the method.

Chapter four presents a variable weighted least squares algorithm according to the amplitude distance between current state and previous states, which is demonstrated to be appropriate for a wide range of nonlinear systems and the same as ordinary least squares algorithm in linear cases. The idea leads to a variable weight algorithm with for on-line parameter estimation. Jumping effect prediction is investigated in order to check the off-line algorithm application. Several

systems, typical and special, are chosen in the simulation experiment to confirm the algorithms performance.

S2.2 List of notations

E[.]	expectation
Var[.]	variance
Cov[.]	covariance
Integ[.]	interger operation
Ave[.]	arithmetic average operation
ARMA	Auto-Regressive Moving Average
NARMAX	Nonlinear Auto-Regressive Moving Average with eXogenous inputs
OLS	Original Least Squares
VWLS	Variable Weighted Least Squares
PID	Proportional, Integral, and Derivative
Matlab	Matrix laboratory package

S2.3 List of figures and tables

Fig. 3.1.1	Nonlinear systems
Fig. 3.2.1	3D quantisation space
Fig. 3.2.2	First order systems
Fig. 3.2.3(a1)	System (sys1) output $\bar{y}_i$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-1}$
Fig. 3.2.3(a2)	Projecting plane of $\bar{y}_i$ and $\hat{x}_i$
Fig. 3.2.3(a3)	Projecting plane of $\bar{y}_i$ and $\hat{y}_{i-1}$
Fig. 3.2.3(b1)	System (sys2) output $\bar{y}_i$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-1}$

Fig. 3.2.3(b2)	Projecting plane of $\bar{y}_i$ and $\hat{x}_i$
Fig. 3.2.3(b3)	Projecting plane of $\bar{y}_i$ and $\hat{y}_{i-1}$
Fig. 3.3.1(a)	System (sys1) output $\bar{y}_i$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-1}$
Fig. 3.3.1(b)	System (sys2) output $\bar{y}_i$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-1}$
Fig. 3.3.2(a)	Auxiliary variable (sys1) $\bar{f}_{a1}$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-1}$
Fig. 3.3.2(b)	Projecting plane of $\bar{f}_{a1}$ and $\hat{x}_i$
Fig. 3.3.2(c)	Projecting plane of $\bar{f}_{a1}$ and $\hat{y}_{i-1}$
Fig. 3.3.2(d)	Auxiliary variable (sys1) $\bar{f}_{a2}$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-2}$
Fig. 3.3.2(e)	Projecting plane of $\bar{f}_{a2}$ and $\hat{y}_{i-2}$
Fig. 3.3.2(f)	Auxiliary variable (sys1) $\bar{f}_{a1}$ versus input $\hat{x}_i$ and previous input $\hat{x}_{i-1}$
Fig. 3.3.2(g)	Projecting plane of $\bar{f}_{b1}$ and $\hat{x}_{i-1}$
Fig. 3.3.3(a)	Projecting plane of $\bar{y}_i$ and $\hat{x}_i$
Fig. 3.3.3(b)	Projecting plane of $\bar{y}_i$ and $\hat{y}_{i-1}$
Fig. 3.3.3(c)	Projecting plane of $\bar{y}_i$ and $\hat{y}_{i-2}$
Fig. 3.3.3(d)	Projecting plane of $\bar{y}_i$ and $\hat{x}_{i-1}$
Fig. 4.2.1	Weighting geometric interpretation
Fig. 4.3.1	Experimental systems
Fig. 4.4.1	Frequency response curve showing jump resonance
Fig. 4.4.2	Analysis of saturating second order system
Fig. 4.4.3(a)	Sinusoidal response with frequency increase
Fig. 4.4.3(b)	Sinusoidal response with frequency decrease

Fig. 4.4.4(a)	Model frequency response characteristics
Fig. 4.4.4(b)	System frequency response characteristics
Fig. 4.4.5	Phase-plane response of the saturating system
Fig. 4.4.6(a)	Step response in one step prediction
Fig. 4.4.6(b)	Step response in multi step prediction
Table 3.3.1	Recursive parameter estimation
Table 4.3.1	Errors in one step prediction and model response

Chapter 3 Structure detection and parameter estimation

3.1 Introduction

Structure detection and model validation tests are fundamental parts of most identification procedures. It is often necessary to determine the structural form or type of model representation which approximates, in the sense of some specified criterion, to the process as the first step in system identification and finally tests the chosen model fitness against the available input and output data. Whereas structure detection involves the determination of the model form which will most appropriately fit the data, model validity checks are designed to indicate the adequacy of the fitted model. Most studies relating to these procedures assume that the system under investigation is linear. Structure detection then reduces to the problem of determining the model order and time delay of the system (Goring and Unbehauen 1973).

This becomes rather complicated in the case of nonlinear systems, and few authors have studied the problem. West (1965) considered nonlinear distortion correlation by studying static nonlinear characteristics. By splitting the output from the nonlinear element into two portions, one proportional to the input signal and the other a distortion noise, West showed that there is no correlation between the input and distortion signal whenever the input belongs to the separable class of random process. Douce (1976) proved that the same property occurs for a specific class of nonlinear dynamic systems. The nonlinear distortion can however, be detected by cross-correlating the residual with a test signal obtained by passing the system input through a specified nonlinearity and Douce developed an identification procedure based on this result. Billings and Voon (1983) introduced higher-order correlation functions as a simple method of computing measures of nonlinearities, which were shown to avoid complicated computation. Billings (1980) gives a good survey on nonlinear system identification.

Billings and Voon (1983) defined structure detection as a method of detecting nonlinearity and of distinguishing this from linear effects and additive noise. Further, they defined model validity as testing terms in residuals which if ignored will cause bias in the parameter estimates. There is no need in this latter case to distinguish between linear, nonlinear or correlated noise effects since any one of these can introduce bias into the estimates. The structure detection and model validation are usually distinct and iteration is needed to modify the model and its parameters to produce a proper approximation of the system concerned.

One common feature of the representative techniques is that they mainly consider time correlation characteristics of input and output signals. A potential development is to study signal amplitude effects as an aid to nonlinear system identification. This chapter aims, by a signal amplitude selection technique, to detect the structure of the whole system concerned, including determination of the position and characteristics of any nonlinearity, then to estimate the parameters of the system.

Consider a general form of nonlinear system given by

$$y_i = f(y_{i-1}, \cdots, y_{i-na}, x_i, \cdots, x_{i-nb}) + \varepsilon_i \qquad (3.1.1)$$

where f is a single valued nonlinearity such as a polynomial (Hammerstein model, Wiener model), saturation or relay, and x_i and y_i are the current input and output respectively. ε_i is an uncorrelated disturbance with zero mean value and variance σ_ε^2.

Two classes of the nonlinear systems have been studied as shown in Fig. 3.1.1.

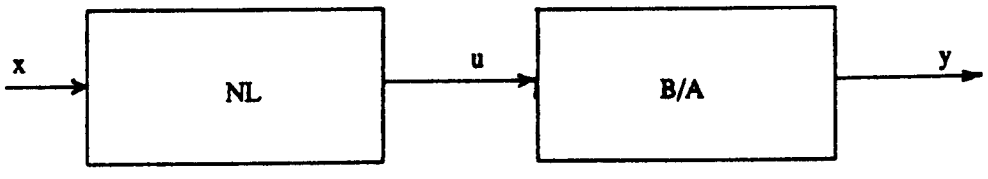
For the first system, defined as sys1 shown in Fig. 3.1.1(a), eqn(3.1.1) gives

$$y_i = b_0 f(x_i) + \cdots + b_{nb} f(x_{i-nb}) - a_1 y_{i-1} - \cdots - a_{na} y_{i-na} + \varepsilon_i \qquad (3.1.2)$$

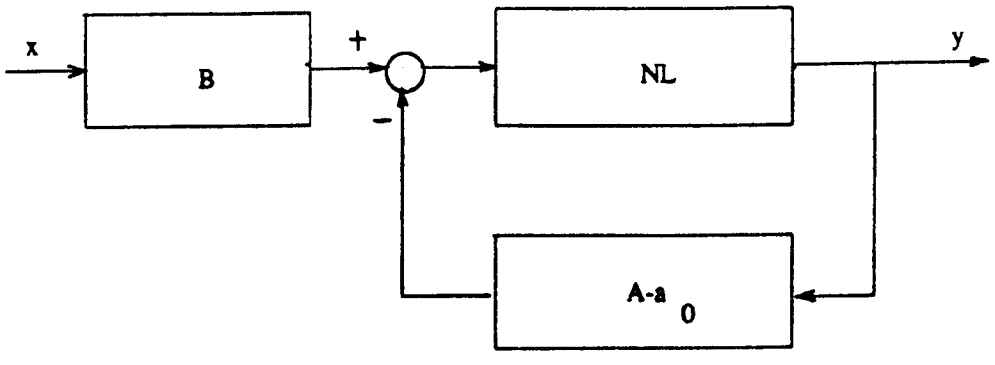
For the second system, defined as sys2 shown in Fig. 3.1.1(b), eqn(3.1.1) gives

$$y_i = f(b_0 x_i + \ldots + b_{nb} x_{i-nb} - a_1 y_{i-1} - \cdots - a_{na} y_{i-na}) + \varepsilon_i \qquad (3.1.3)$$

The work described in this chapter represents the results of preliminary studies, in that it is restricted mainly to first order systems. However it is suggested that further study may demonstrate that the ideas have wider applicability.



(a)



(b)

$$A = a_0 + a_1 z^{-1} + \cdots + a_{na} z^{-na}$$

$$B = b_0 + b_1 z^{-1} + \cdots + b_{nb} z^{-nb}$$

Figure 3.1.1 Nonlinear systems

3.2 System identification

3.2.1 Structure detection

Consider a simple form of eqn(3.1.1)

$$y_i = f(y_{i-1}, x_i) \tag{3.2.1}$$

where the analytical form is unknown and the input and output signals are available.

A quantisation operation, including input and past output signal amplitude quantisations and current output signal amplitude averaging, will be developed to find the analytical expression to describe the system.

Let $\hat{x_i}$ and $\hat{y}_{i-1}$ denote quantisation variable of x_i and y_{i-1} respectively, that is

x_i , y_{i-1} *is quantized following time axis with data length n.*

$\hat{x_i}$, $\hat{y}_{i-1}$ *is quantized following amplitude axis with data length nx, ny.*

Usually $nx, ny \ll n$, and

$$\hat{x_i} = Integ\left[\frac{nx\ (x_i - min(x_i))}{max(x_i) - min(x_i)} \right] + 1$$

$$\hat{y}_{i-1} = Integ\left[\frac{ny\ (y_i - min(y_i))}{max(y_i) - min(y_i)} \right] + 1$$

$$\bar{y}_i = Ave[\, y_i \,], \quad \text{for } \hat{x_i}, \hat{y}_{i-1} \tag{3.2.2}$$

where Integ[.] denotes taking integer operation, Ave[.] denotes arithmetic average operation, and max(.) and min(.) take the maximum and minimum of the series respectively.

In fact the quantisation operation smoothes the corresponding signals within a small area specified by $\hat{x_i}$ and $\hat{y}_{i-1}$. This method considers both time and amplitude correlation characteristics of signals.

From $\bar{y}_i$, $\hat{x_i}$ and $\hat{y}_{i-1}$, a 3D space is correspondingly built up, as shown in Fig. 3.2.1. The structure detection and the parameter estimation will be carried out based on this 3D space and the corresponding projecting planes.

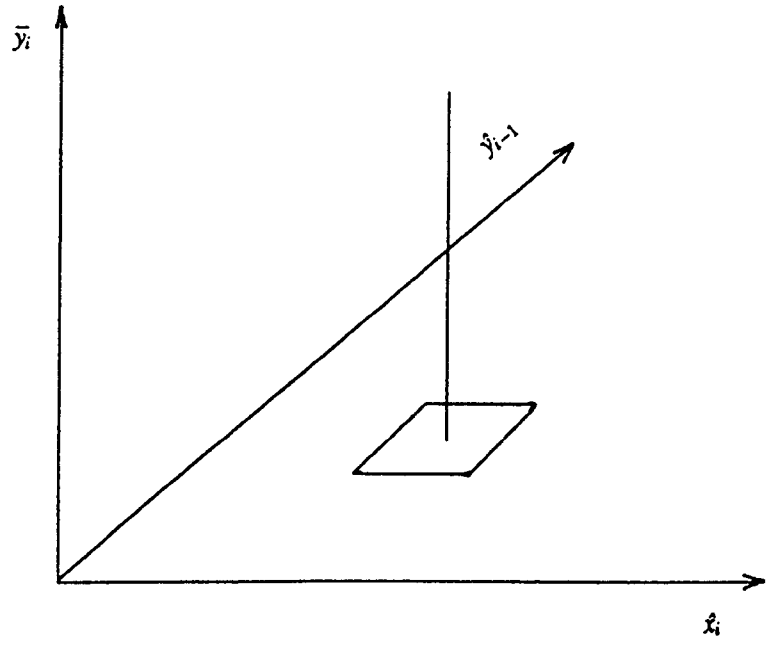


Figure 3.2.1 3D quantisation space

For the linear system

$$y_i = a_1 y_{i-1} + b_0 x_i \tag{3.2.3}$$

points on the 3D plot defined above all lie on the plane

$$y_i - a_1 y_{i-1} - b_0 x_i = 0 \tag{3.2.4}$$

For nonlinear systems, points lie on a surface. Inspection of this surface can reveal information about the location and characteristic of a nonlinear element and the linear dynamic parameters within the system.

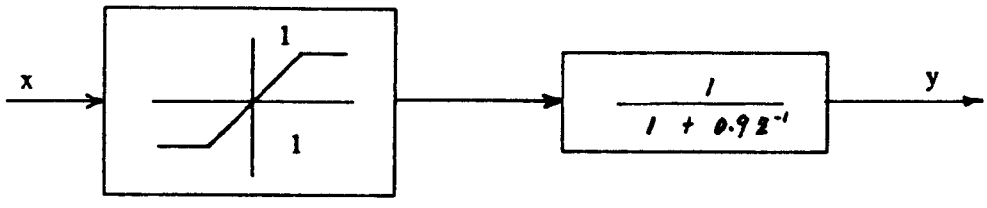
Two examples, both first order systems, as shown in Fig. 3.2.2, are selected. The nonlinearity is saturation and the linear dynamics are

$$A = 1 + 0.9z^{-1}$$

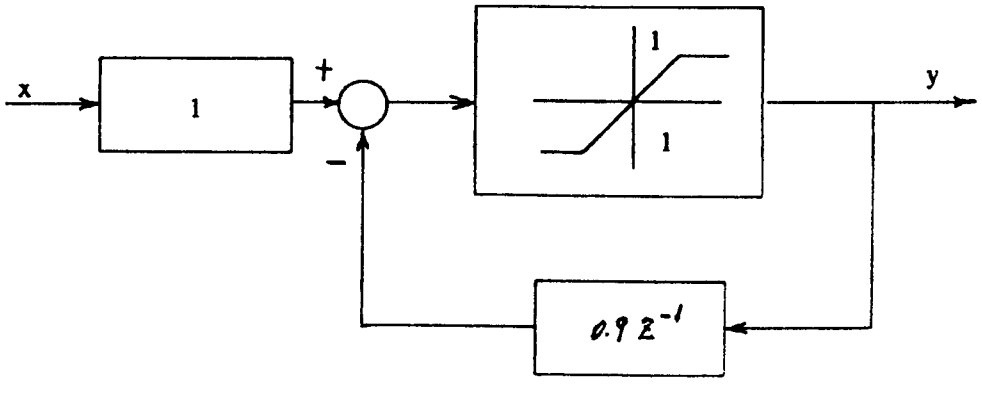
$$B = 1 \tag{3.2.5}$$

The input signal is a Gaussian white random signal with zero mean value and variance 4. The data length of the input and output is set to 4000. The

quantised data lengths nx and ny are set to 20 equally, hence the quantised area consists of 20*20=400 small areas.



(a)



(b)

Figure 3.2.2 First oder systems

Two projecting planes $\bar{y}_i, \hat{x}_i$ and $\bar{y}_i, \hat{y}_{i-1}$ are defined as one of the planes perpendicular to $\hat{y}_{i-1}$ and $\hat{x}_i$ respectively in the 3D space. They will be used for the linear dynamic parameter estimation of the systems.

The examples will be used to show directly several features on the 3D graphic surface, including

1 System structure

2 Nonlinearity and its position

Consider the plot shown in Fig. 3.2.3(a1). This shows the output (vertical) as a function of input and previous output of a system excited by white normal distributed noise. Quantisation has been introduced as described above, primarily to render the plotting process simple within Matlab (a proprietary software package), with the additional advantage, not persued here, that disturbance effects are reduced due to averaging when sufficient data points are available.

The horizontal plane, corresponding to $\bar{y}_i = 0$, is used to indicate absence of data pairs in the corresponding $\hat{x}_i$, $\hat{y}_{i-1}$ space.

Visual inspection of this plot indicates that $\bar{y}_i$ is a linear function of $\hat{y}_{i-1}$ for the range of signal amplitude available. Further, $\bar{y}_i$ is obviously a nonlinear function of $\hat{x}_i$, with some evidence of the presence of a saturation characteristic for large values of $\hat{x}_i$.

This dependence of the output on a signal variable (which may be a combination of system variables) is demonstrated by viewing plots such as Fig. 3.2.2(a1) from a selected direction. This direction is defined by a vector perpendicular to the dependant variable and to the selected independent variable.

In this example, the viewing direction is perpendicular to the $\hat{x}_i$ axis. Projecting the surface onto this plane gives the nonlinear relationship between $\bar{y}_i$ and $\hat{x}_i$, as shown in Fig. 3.2.3(a2).

Further, the angle or the slope of the projection relative to the $\hat{y}_{i-1}$ axis gives the linear coefficient relating $\bar{y}_i$ to $\hat{y}_{i-1}$, as shown in Fig. 3.2.3(a3). In summary, Fig. 3.2.3(a1) enables the system to be identified as

$$y_i = a_1 y_{i-1} + b_0 f(x_i) \tag{3.2.6}$$

That is, the structure is as in Fig. 3.2.2(a).

Similarly consider the plot shown in Fig. 3.2.3(b1). Visual inspection of this plot indicates that $\bar{y}_i$ is a saturating nonlinear function of both $\hat{y}_{i-1}$ and $\hat{x}_i$ for the range of signal amplitude available.

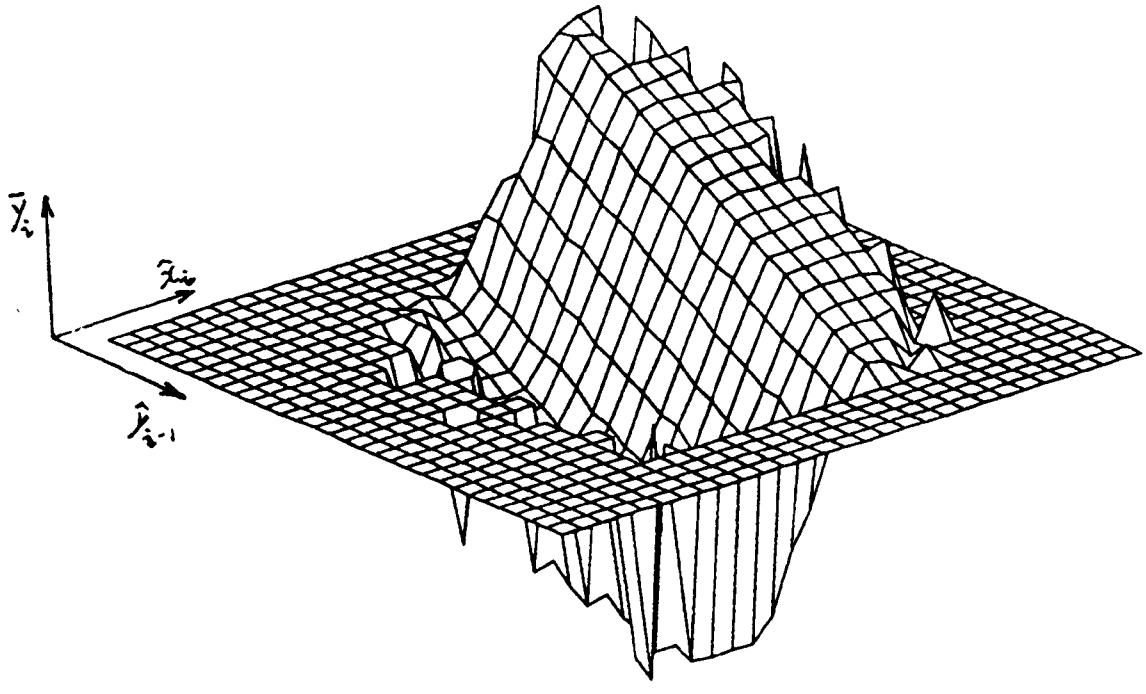


Figure 3.2.3(a1) System (sys1) output $\bar{y}_i$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-1}$

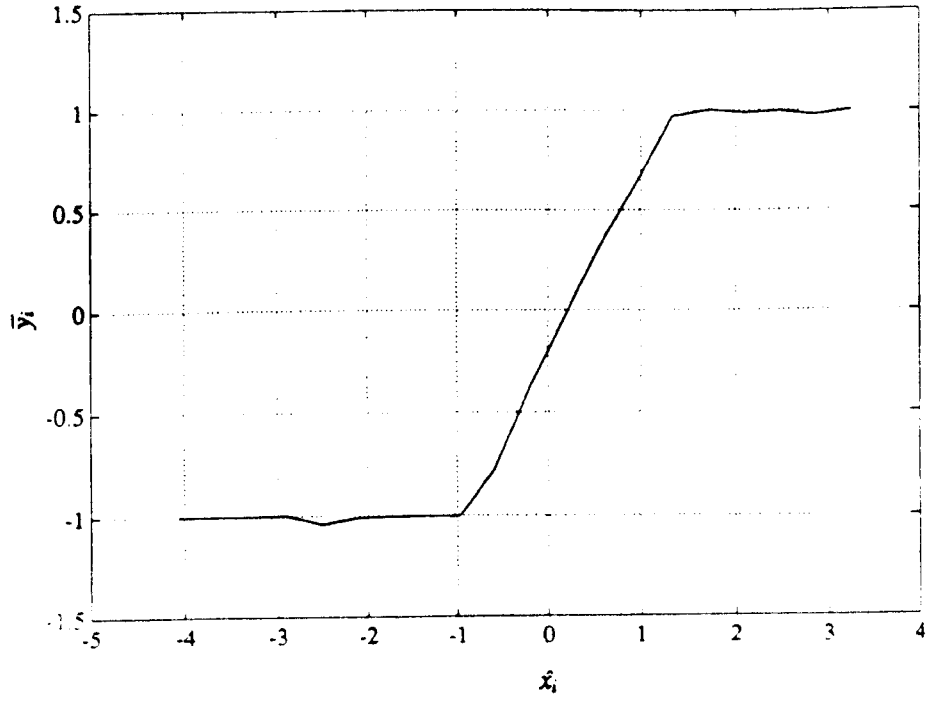


Figure 3.2.3(a2) Projecting plane of $\bar{y}_i$ and $\hat{x}_i$

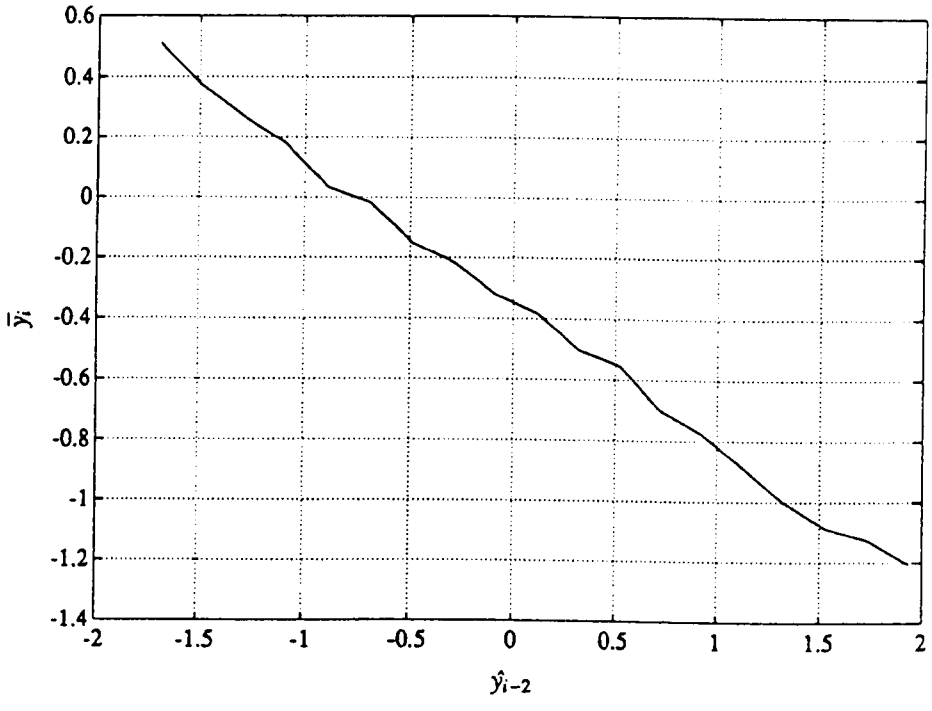


Figure 3.2.3(a3) Projecting plane of $\bar{y}_i$ and $\hat{y}_{i-1}$

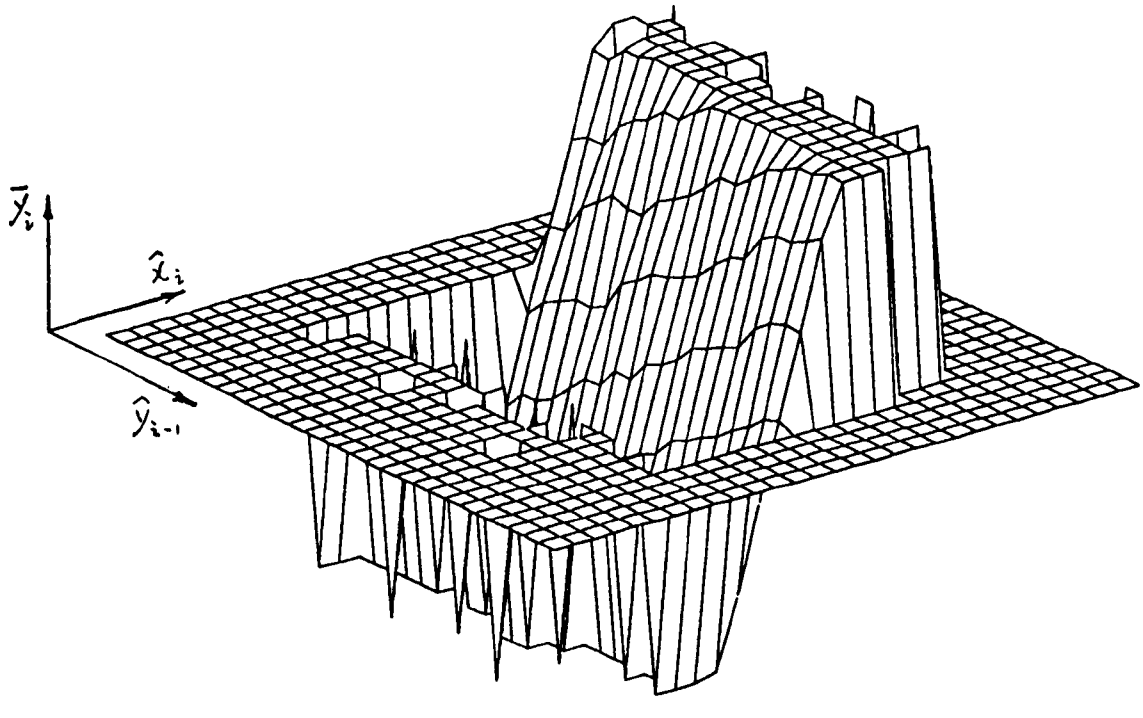


Figure 3.2.3(b1) System (sys2) output $\bar{y}_i$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-1}$

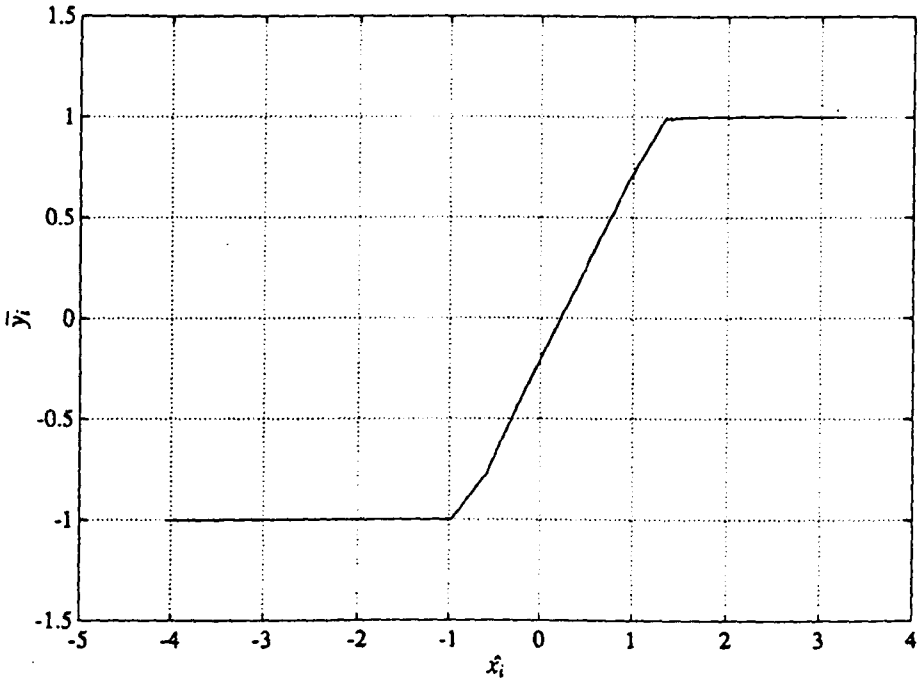


Figure 3.2.3(b2) Projecting plane of $\bar{y}_i$ and $\hat{x}_i$

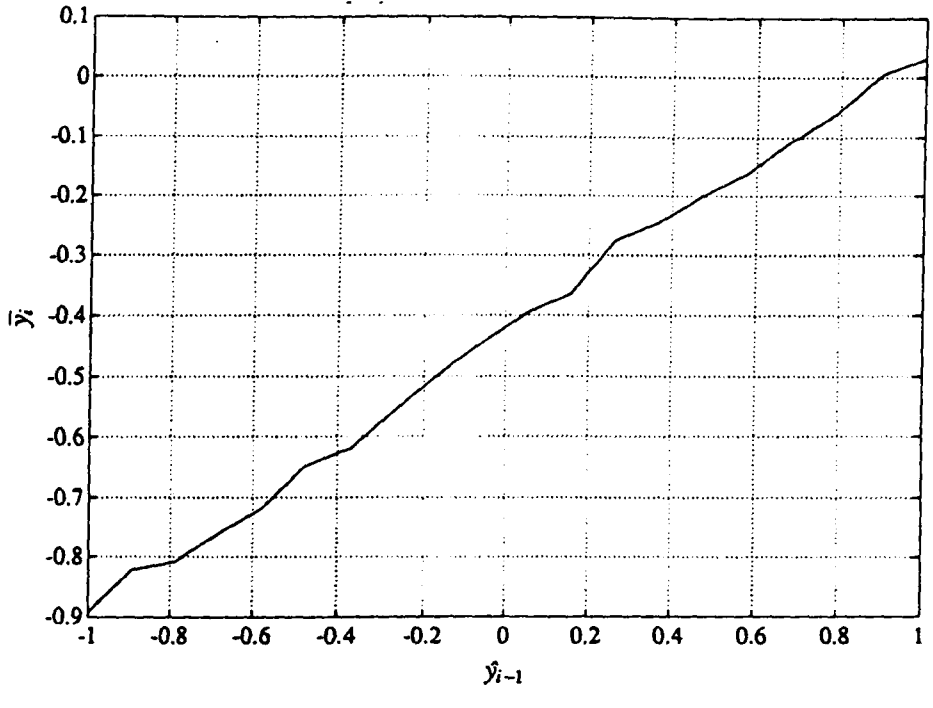


Figure 3.2.3(b3) Projecting plane of $\bar{y}_i$ and $\hat{y}_{i-1}$

The relationship between $\bar{y}_i$ and $\hat{x}_i$ shown in Fig. 3.2.3(b2) is the same as the analysis for the first system, however the visual inspection has indicated, in Fig. 3.2.3(b1), the nonlinearity depending on the combination of $\hat{y}_{i-1}$ and $\hat{x}_i$, saturation appears when $| a_1 y_{i-1} + b_0 x_i | > 1$. Fig. 3.2.3(b3) is the projecting plane $\bar{y}_i$, $\hat{y}_{i-1}$ in case of $| a_1 y_{i-1} + b_0 x_i | < 1$, this is convenient for parameter estimation in linear area.

In summary, from Fig. 3.2.3(b1) the system structure is identified as

$$y_i = f(a_1 y_{i-1} + b_0 x_i) \tag{3.2.7}$$

That is, the structure is as in Fig. 3.2.2(b).

It is learnt from the system structure given in Fig. 3.2.2 that the relationship between output and input only depends on $\bar{y}_i$ and $\hat{x}_i$, similarly the relationship between output and past output only depends on $\bar{y}_i$ and $\hat{y}_{i-1}$. Therefore the quantisation algorithm may be used to distinguish the two types of nonlinear systems in the 3D space. For system one, it is clear that the nonlinearity can only be visualized from $\bar{y}_i$, $\hat{x}_i$ projecting direction. For system two, it is found that the nonlinearity appears from both projecting directions $\bar{y}_i$, $\hat{x}_i$ and $\bar{y}_i$, $\hat{y}_{i-1}$. For linear system, there are linear relationships viewed from the two projecting directions, this is confirmed within the linear area in the 3D space.

The structure detection method developed from the simple cases is also available for some more general cases. This is because the relationship between input $\hat{x}_i$ and output $\bar{y}_i$ indicates the static nonlinear characteristic, leading to detection of the static nonlinearity. The second reason is that the displayed relationship between past output $\hat{y}_{i-1}$ and output $\bar{y}_i$ indicates the presence or absence of a nonlinear relationship, even though neglecting other terms leads to a poor indication of the linear coefficient a_{i-1}. Therefore the first step is used to detect the linearity or nonlinearity of the systems, and if nonlinearity is detected then further inspection is used to distinguish the two types of systems.

3.2.2 Parameter estimation

For simple cases, system structure detection and parameter estimation are simultaneously achieved. The parameters may be obtained, as mentioned above, from the slope on the corresponding projecting planes, as shown in Fig. 3.2.3(a2, a3, b2, b3). However the quantisation technique for the general case may not be directly applied to obtain system parameters from the slopes on the projecting planes as for first order systems. This is because for the general case (high order systems) the variables not involved in building the 3D space affect the variables in the 3D space, the slopes on the projecting planes in the 3D space are not true representatives for the parameters to be estimated. It is argued that the technique is still applicable, with some modifications.

Alternatively we divide the work, as mentioned in the last section, into two parts, one for structure detection having been solved, another for parameter estimation. The modified algorithm, by introducing auxiliary variables, generates a set of equivalent first order systems, or a set of 3D spaces with the same properties as the first order systems. We have demonstrated that parameter estimates can be directly obtained in the 3D space representing a first order system, hence the problem for parameter estimation in general case may be solved. The implementation of the parameter estimation algorithm is based on the following argument, which presents, by introducing auxiliary variables, a recursive procedure to estimate parameters.

Argument

The quantisation algorithm developed from the first order system is also available for high order system when a set of auxiliary variables are defined as

for sys1

$$f_{ak} = y_i - \sum_{j=1, j\neq k}^{na} a_j y_{i-j} - \sum_{j=1}^{nb} b_j f(x_{i-j}), \qquad k=1, \cdots, na$$

$$f_{bk} = y_i - \sum_{j=1}^{na} a_j y_{i-j} - \sum_{j=1, j\neq k}^{nb} b_j f(x_{i-j}), \qquad k=1, \cdots, nb$$

for sys2

$$f_{ak} = \sum_{j=1,j\neq k}^{na} a_j y_{i-j} + \sum_{j=0}^{nb} b_j x_{i-j}, \qquad k=1,\cdots,na$$

$$f_{bk} = \sum_{j=1}^{na} a_j y_{i-j} + \sum_{j=0,j\neq k}^{nb} b_j x_{i-j}, \qquad k=1,\cdots,nb \tag{3.2.8}$$

Proof: according to eqn(3.1.2), for sys1

$$f_{ak} = a_k y_{i-k} + b_0 f(x_i)$$

$$f_{bk} = b_k f(x_{i-k}) + b_0 f(x_i)$$

According to eqn(3.1.3), for sys2

$$y_i = f(a_k y_{i-k} + f_{ak})$$

$$y_i = f(b_k x_{i-k} + f_{bk}) \tag{3.2.9}$$

These have the same form as the first order systems given in Fig. 3.2.2, therefore a set of 3D spaces may be built up from the set of first order systems and the parameters of the systems may be obtained directly by the quantisation algorithm developed from the 3D spaces.

It should be noticed that f_{ak} and f_{bk} include the parameters to be estimated. We use the most recent parameter estimates instead of the true parameters to calculate f_{ak} and f_{bk}, then update the parameter estimates. Obviously this is a recursive process, with the algorithm given by, for sys1

$$f_{ak}(r+1) = y_i - \sum_{j=1,j\neq k}^{na} a_j(r) y_{i-j} - \sum_{j=1}^{nb} b_j(r) f(x_{i-j}), \qquad k=1,\cdots,na$$

$$f_{bk}(r+1) = y_i - \sum_{j=1}^{na} a_j(r) y_{i-j} - \sum_{j=1,j\neq k}^{nb} b_j(r) f(x_{i-j}), \qquad k=1,\cdots,nb$$

$$f_{ak}(r+1) = a_k(r+1) y_{i-k} + b_0 f(x_i)$$

$$f_{bk}(r+1) = b_k(r+1) f(x_{i-k}) + b_0 f(x_i)$$

for sys2

$$f_{ak}(r+1) = \sum_{j=1,j\neq k}^{na} a_j(r) y_{i-j} + \sum_{j=0}^{nb} b_j(r) x_{i-j}, \qquad k=1,\cdots,na$$

$$f_{bk}(r+1) = \sum_{j=1}^{na} a_j(r) y_{i-j} + \sum_{j=0,j\neq k}^{nb} b_j(r) x_{i-j}, \qquad k=1,\cdots,nb$$

$$y_i = f\,(a_k(r+1)y_{i-k} + f_{ak}(r+1))$$

$$y_i = f\,(b_k(r+1)x_{i-k} + f_{bk}(r+1)) \qquad (3.2.10)$$

where r is a recursive time. The extensive simulation experiment has confirmed the algorithm to produce correct estimates.

3.3 Simulation experiment

Two noise free second order nonlinear systems are selected with the different structures as shown in Fig. 3.1.1. The nonlinearity is saturation, and the system linear dynamics are set to

$$A = 1 - z^{-1} + 0.3z^{-2}$$

$$B = 1 - 0.5z^{-1} \qquad (3.3.1)$$

The data length of the input and output is 4000. The quantised data lengths nx and ny are set to 20, hence the quantised area consists of 20*20=400 small areas. A Gaussian random signal input with zero mean value and variance 4 is chosen as input.

The horizontal plane, corresponding to $\bar{y}_i = 0$, is used to indicate absence of data pairs in the corresponding $\hat{x}_i$, $\hat{y}_{i-1}$ space.

The first step is to detect the nonlinear system structure. Three variables, the output $\bar{y}_i$, past output $\hat{y}_{i-1}$ and input $\hat{x}_i$ are selected to build up a 3D space by the quantisation operation given in eqn(3.2.2). The results are illustrated for the open-loop system (sys1) in Fig. 3.3.1(a) and the closed-loop system (sys2) in Fig. 3.3.1(b).

First consider Fig. 3.3.1(a), $\bar{y}_i$ is a nonlinear function, with evidence of saturation, of $\hat{x}_i$ and a linear function of $\hat{y}_{i-1}$ by visual inspection of this plot. It is also found that $\bar{y}_i$ is a nonlinear function, with evidence of saturation, of $\hat{x}_{i-1}$ and a linear function of $\hat{y}_{i-2}$ by visual inspection of the 3D space ($\bar{y}_i$, $\hat{y}_{i-2}$, $\hat{x}_{i-1}$) not shown in the chapter. According to the discussion in section 3.2 the system is identified as

$$y_i = a_1 y_{i-1} + a_2 y_{i-2} + b_0 f\,(x_i) + b_1 f\,(x_{i-1}) \qquad (3.3.2)$$

where function $f(.)$ denotes saturation for the specific system and the system order is assumed known by someway.

Similarly the visual inspection of Fig. 3.3.1(b) indicates that $\bar{y}_i$ is a saturating nonlinear function both of $\hat{x}_i$ and $\hat{y}_{i-1}$. The same structure is also found in the 3D space $(\bar{y}_i, \hat{y}_{i-2}, \hat{x}_{i-1})$ not shown in the chapter. Hence the system is identified as

$$y_i = f(a_1 y_{i-1} - a_2 y_{i-2} + b_0 x_i + b_1 x_{i-1}) \tag{3.3.3}$$

The rule to determine the system structure characteristics in the 3D space is

sys1 Nonlinearity (viewing perpendicular to plane $\bar{y}_i, \hat{x}_i$), linearity (viewing perpendicular to plane $\bar{y}_i, \hat{y}_{i-1}$).

sys2 Nonlinearity (viewing perpendicular to plane $\bar{y}_i, \hat{x}_i$), nonlinearity (viewing perpendicular to plane $\bar{y}_i, \hat{y}_{i-1}$).

Linear system

Linearity (viewing perpendicular to plane $\bar{y}_i, \hat{x}_i$) linearity (viewing perpendicular to plane $\bar{y}_i, \hat{y}_{i-1}$), which is confirmed by visual inspection in linear area of the 3D space.

It has been noted that the slopes or angles relative to the elemental plane $\hat{x}_i$, $\hat{y}_{i-1}$ are not true representatives for the linear dynamic parameters of the systems in the structure detection 3D space, however they may be used as the initial values for recursive parameter estimation confirmed by the following experiment.

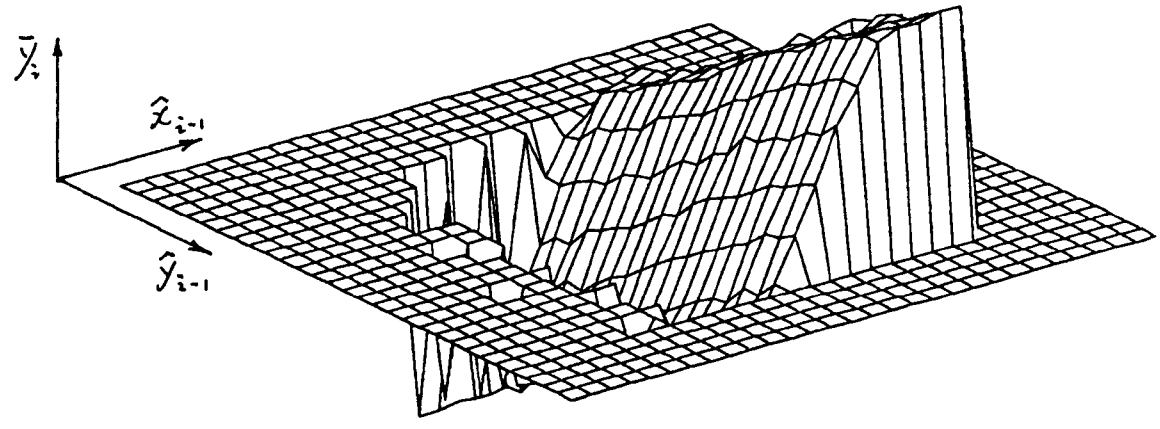


Figure 3.3.1(a) System (sys1) output $\bar{y}_i$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-1}$

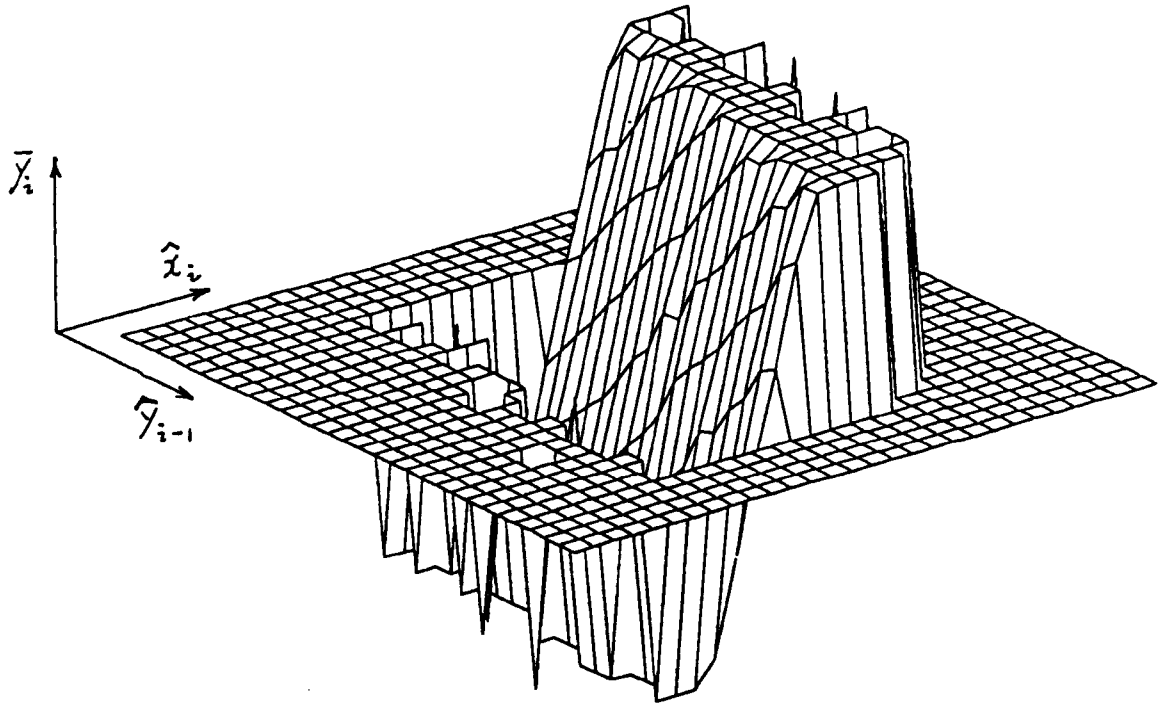


Figure 3.3.1(b) System (sys2) output $\bar{y}_i$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-1}$

The second step is to estimate the linear dynamic parameters of the systems identified in the last experiment. According to the algorithm presented in section 3.2.2, a set of 3D spaces, by introducing auxiliary variables, are chosen as

for sys1

space $(f_{a1}\, y_{i-1}\, x_i)$ for estimation of a_1 and b_0

space $(f_{a2}\, y_{i-2}\, x_i)$ for estimation of a_2

space $(f_{b1}\, x_{i-1}\, x_i)$ for estimation of b_1

where

$$f_{a1} = y_i - a_2 y_{i-2} - b_1 f(x_{i-1})$$

$$f_{a2} = y_i - a_1 y_{i-1} - b_1 f(x_{i-1})$$

$$f_{b1} = y_i - a_1 y_{i-1} - a_2 y_{i-2} \qquad (3.3.4)$$

for sys2

space $(y_i\, y_{i-1}\, f_{a1})$ for estimation of a_1

space $(y_i\, y_{i-2}\, f_{a2})$ for estimation of a_2

space $(y_i\, x_i\, f_{b0})$ for estimation of b_0

space $(y_i\, x_{i-1}\, f_{b1})$ for estimation of b_1

where

$$f_{a1} = a_2 y_{i-2} + b_0 x_i + b_1 x_{i-1}$$

$$f_{a2} = a_1 y_{i-1} + b_0 x_i + b_1 x_{i-1}$$

$$f_{b0} = a_1 y_{i-1} - a_2 y_{i-2} + b_1 x_{i-1}$$

$$f_{b1} = a_1 y_{i-1} - a_2 y_{i-2} + b_0 x_i \qquad (3.3.5)$$

The initial values for the iteration are set to $(a_1\, a_2\, b_0\, b_1)$=(-0.4 0 1 0) for the sys1 and $(a_1\, a_2\, b_0\, b_1)$=(-0.5 0 1 0) for the sys2. Parameters a_1 and b_0 are chosen initially from the 3D space $(y_i\, y_{i-1}\, x_i)$ used for the system structure detection in the first experiment. They are not true parameters but suitable initial settings. Since no initial information is available for a_2 and b_1 in the 3D space, their

initial values are set to zero.

After five iterations using the last parameters estimated and the available input and output data for new parameter estimation, the surfaces in the set of 3D spaces arrive at steady state. Consider sys1, the resultant 3D spaces and projecting planes are shown in Fig. 3.3.2. Spaces $(f_{a1}\, y_{i-1}\, x_i)$ shown in Fig. 3.3.2(a) and $(f_{a2}\, y_{i-2}\, x_i)$ shown in Fig. 3.3.2(d) have the same shape, that is, $\bar{f}_{a1}$ is a saturating nonlinear function of $\hat{x}_i$ shown in Fig. 3.3.2(b) and a linear function of $\hat{y}_{i-1}$ shown in Fig. 3.3.2(c), $\bar{f}_{a2}$ is a saturating nonlinear function of $\hat{x}_i$ and a linear function of $\hat{y}_{i-2}$ shown in Fig. 3.3.2(e). The different angle of the projection relative to the $\hat{y}_{i-1}$ axis in planes $\bar{f}_{a1}\, y_{i-1}$ and $\bar{f}_{a2}\, y_{i-2}$ indicates the different parameters estimated. In space $(f_{b1}\, x_{i-1}\, x_i)$, as shown in Fig. 3.3.2(f), the visual inspection indicates that $\bar{f}_{b1}$ is a saturating nonlinear function both of $\hat{x}_{i-1}$ and $\hat{x}_i$, this is coincident with the system structure identified given in eqn(3.3.2) and Fig. 3.3.2(g) shows the resultant slope for the parameter b_1 estimate.

Similarly for sys2, four projecting planes, as shown in Fig. 3.3.3(a, b, c, d) are obtained from their 3D spaces. The projecting planes $\bar{y}_i$, $\hat{f}_{a1}$ and $\bar{y}_i$, $\hat{f}_{a2}$ are chosen in case of $|a_1 y_{i-1} - a_2 y_{i-2} + b_0 x_i + b_1 x_{i-1}| < 1$, hence a linear characteristic is displayed.

Table 3.3.1 summarises the recursive parameter estimating results for both sys1 and sys2.

Recursive parameter estimation Sys1: open-loop						
Est. para. \ recursion	1	2	3	4	5	Real para.
$\hat{a}_1$	-0.4	-0.6	-0.8	-0.9	-0.99	$a_1 = -1$
$\hat{a}_2$	0	0.1	0.2	0.25	0.3	$a_2 = 0.3$
$\hat{b}_0$	1	1	1	1	1	$b_0 = 1$
$\hat{b}_1$	0	-0.22	-0.3	-0.4	-0.5	$b_1 = -0.5$
Recursive parameter estimation Sys2: closed-loop						
Est. para. \ recursion	1	2	3	4	5	Real para.
$\hat{a}_1$	-0.5	-0.72	-0.93	-0.96	-0.98	$a_1 = -1$
$\hat{a}_2$	0	0.23	0.3	0.3	0.3	$a_2 = 0.3$
$\hat{b}_0$	1	1	1	1	1	$b_0 = 1$
$\hat{b}_1$	0	-0.3	-0.43	-0.48	-0.49	$b_1 = -0.5$

Table 3.3.1

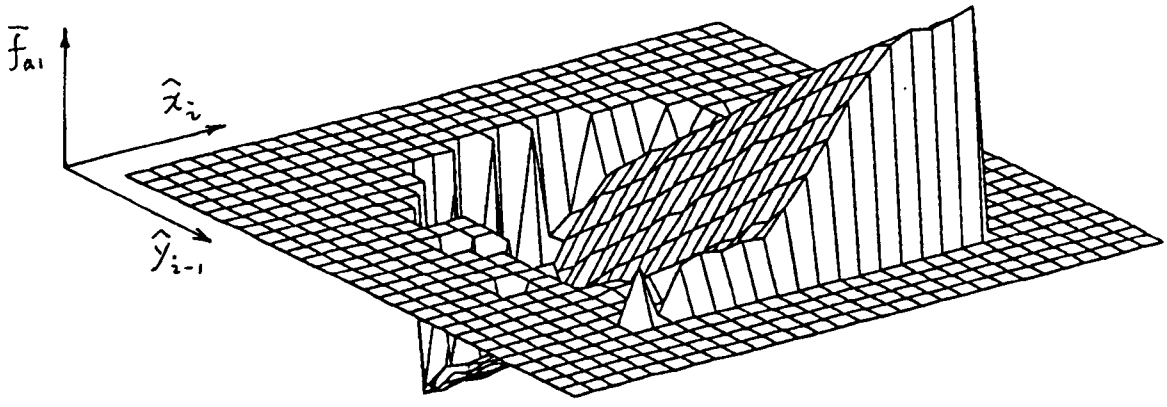


Figure 3.3.2(a) Auxiliary variable (sys1) $\bar{f}_{a1}$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-1}$

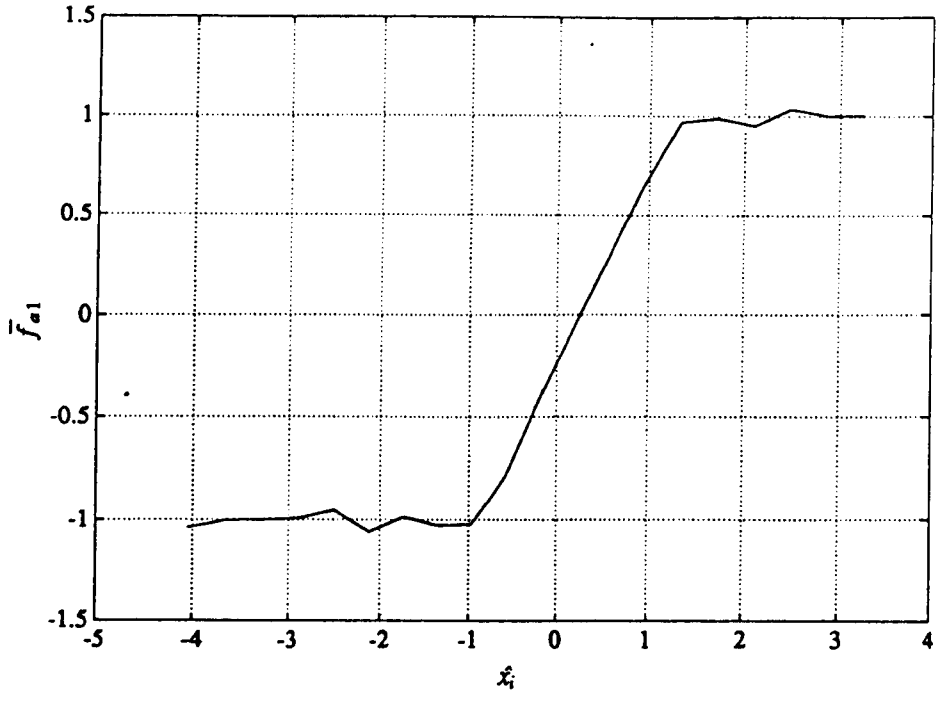


Figure 3.3.2(b) Projecting plane of $\bar{f}_{a1}$ and $\hat{x}_i$

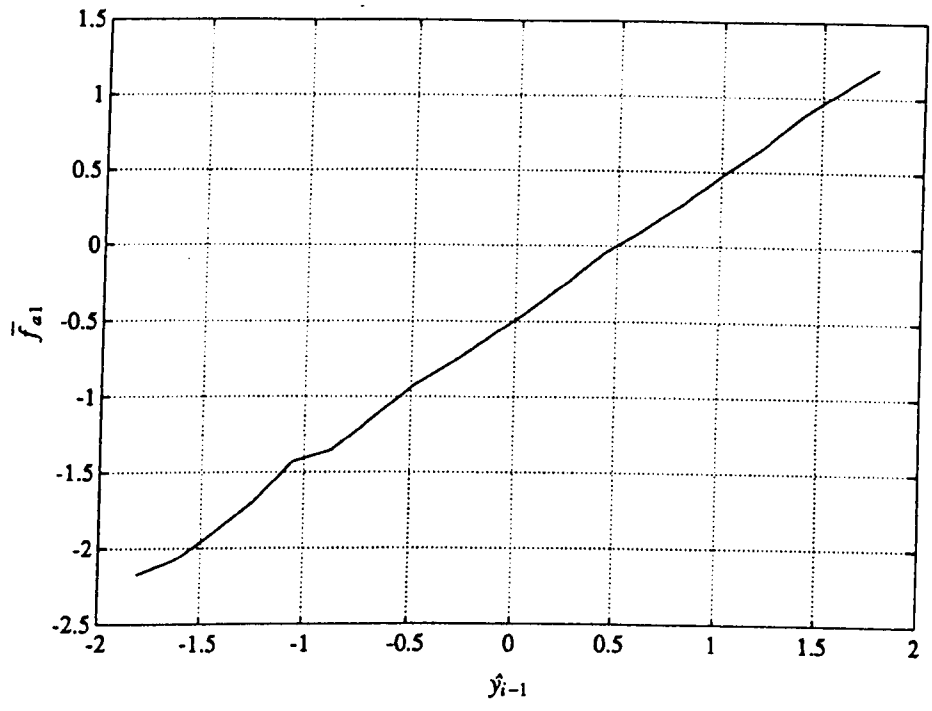


Figure 3.3.2(c) Projecting plane of $\bar{f}_{a1}$ and $\hat{y}_{i-1}$

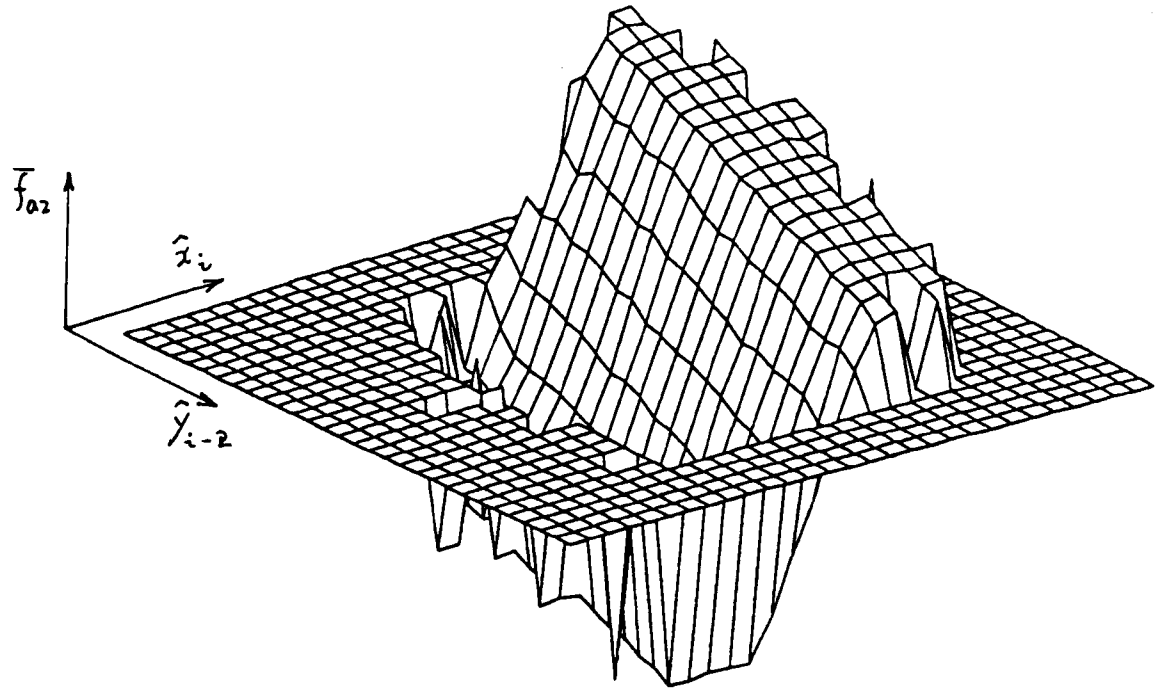


Figure 3.3.2(d) Auxiliary variable (sys1) $\bar{f}_{a2}$ versus input $\hat{x}_i$ and previous output $\hat{y}_{i-2}$

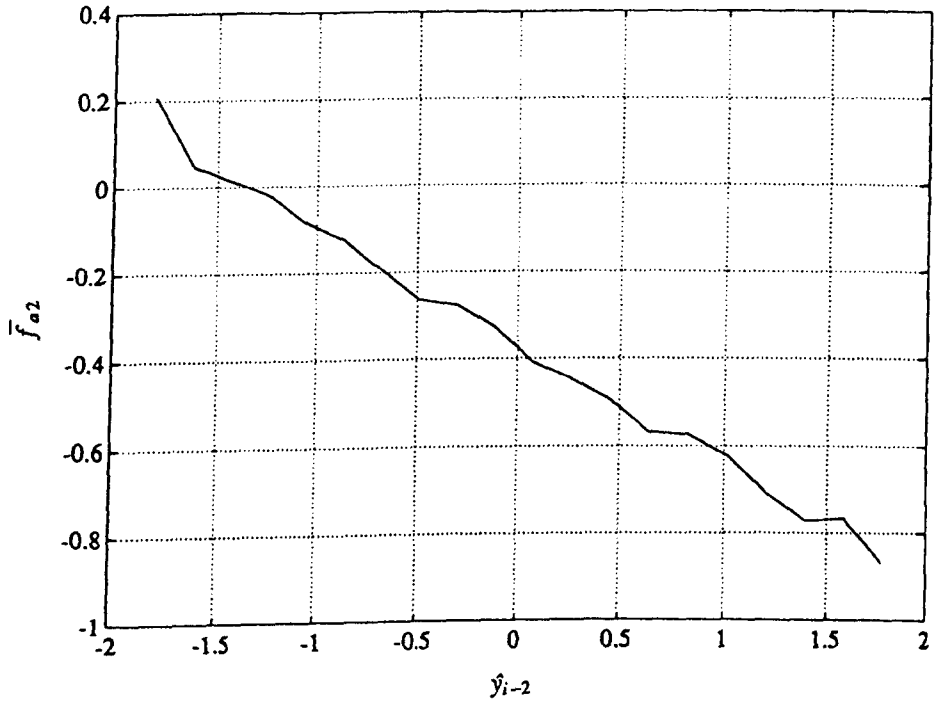


Figure 3.3.2(e) Projecting plane of $\bar{f}_{a2}$ and $\hat{y}_{i-2}$

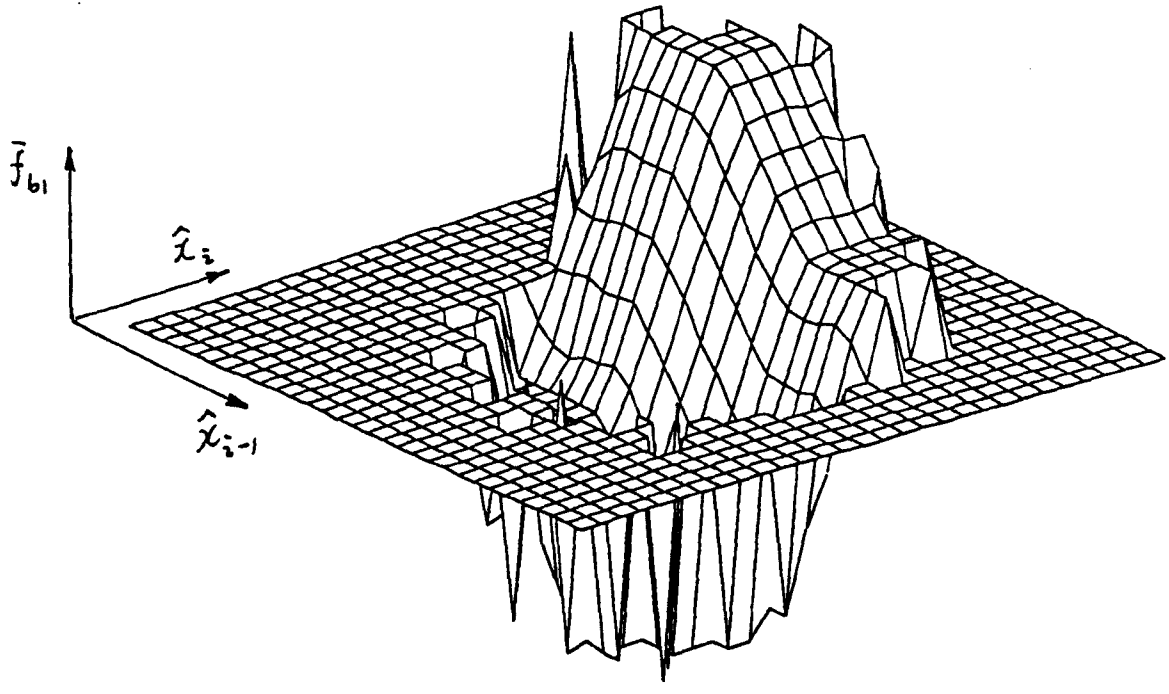


Figure 3.3.2(f) Auxiliary variable (sys1) $\bar{f}_{b1}$ versus input $\hat{x}_i$ and previous input $\hat{x}_{i-1}$

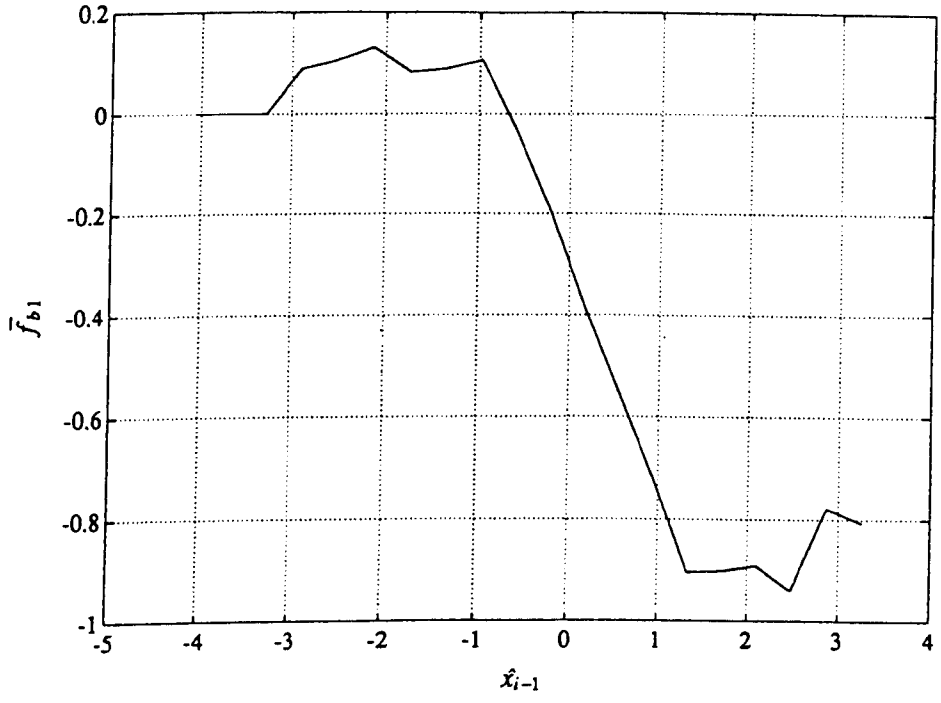


Figure 3.3.2(g) Projecting plane of $\bar{f}_{b1}$ and $\hat{x}_{i-1}$

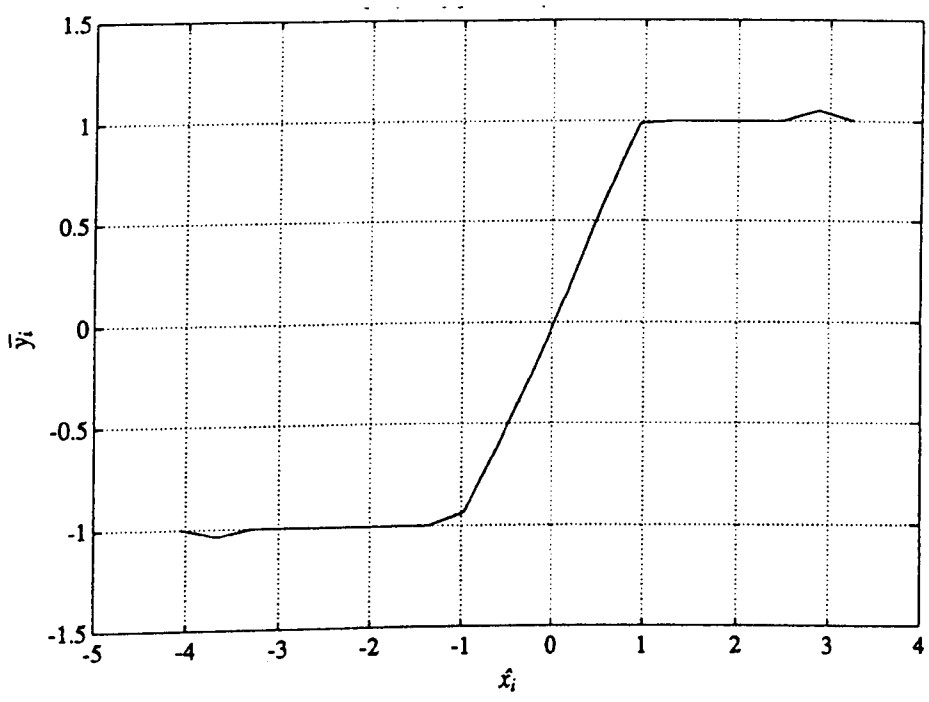


Figure 3.3.3(a) Projecting plane of $\bar{y}_i$ and $\hat{x}_i$

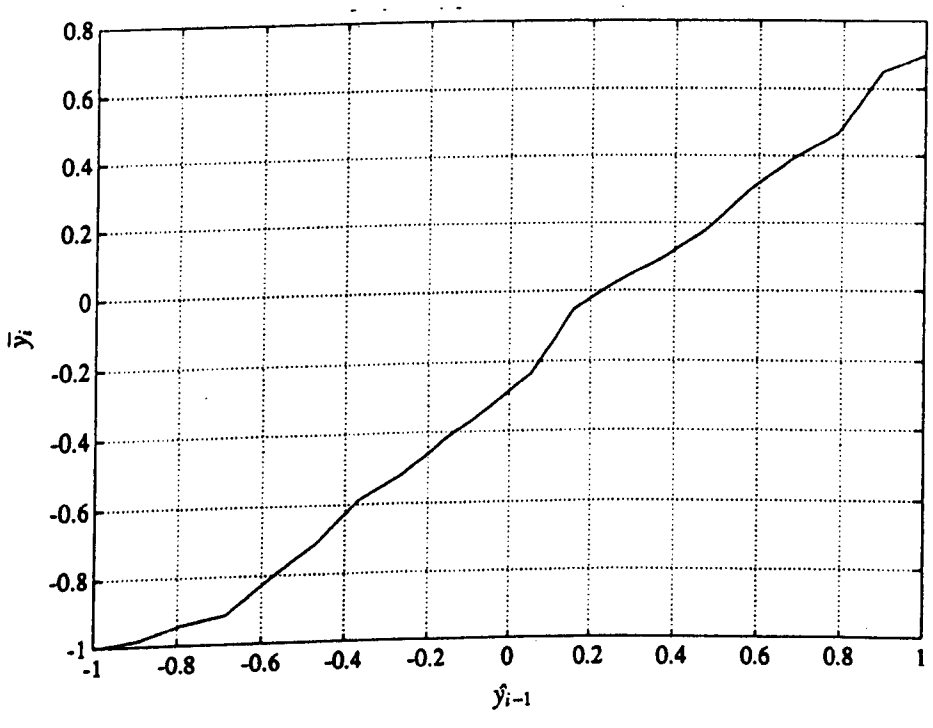


Figure 3.3.3(b) Projecting plane of $\bar{y}_i$ and $\hat{y}_{i-1}$

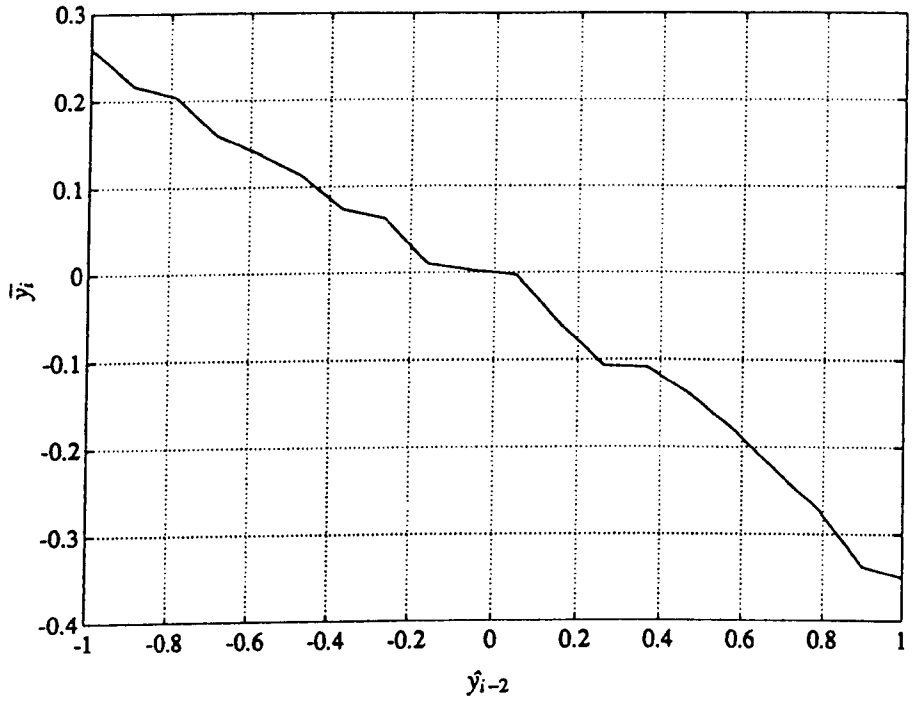


Figure 3.3.3(c) Projecting plane of $\bar{y}_i$ and $\hat{y}_{i-2}$

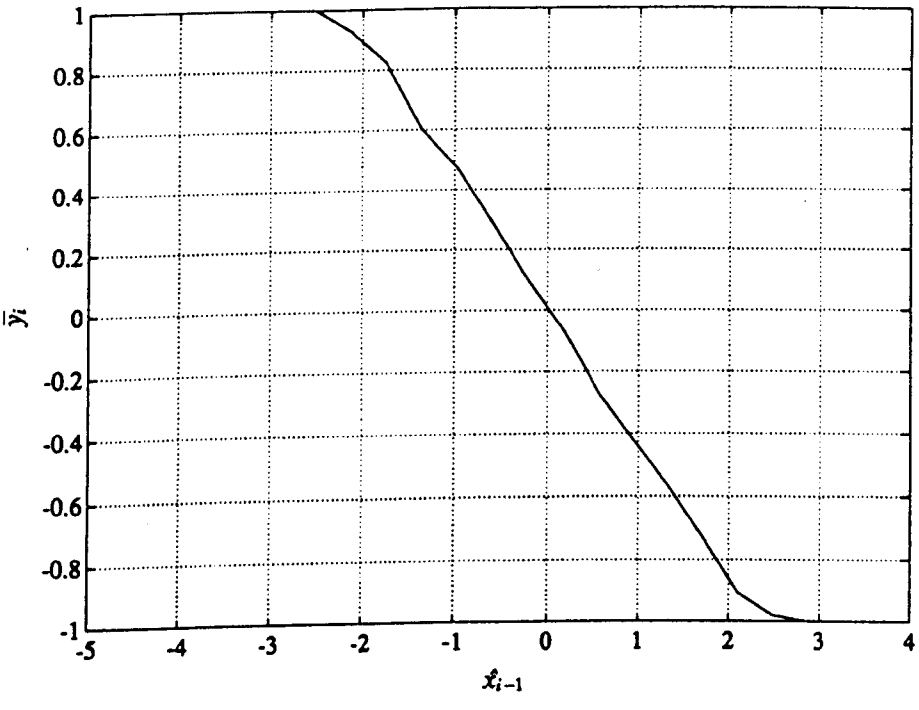


Figure 3.3.3(d) Projecting plane of $\bar{y}_i$ and $\hat{x}_{i-1}$

3.4 Conclusion

This chapter has introduced a potentially valuable method for nonlinear system identification. When a system contains a nonlinear characteristic which is piece-wise linear, then the display of output against input and past outputs is a series of planes or hyper planes. For each linear region, a linear model can be constructed. In simple cases, it is possible to determine the variables which affect the nonlinear output signal, and the characteristics of the nonlinearity. Thus the structure and parameters of the system can be found.

The idea introduced with reference to first order nonlinear systems have been demonstrated with two second order examples.

This chapter is submitted as a novel idea prompting substantial further work. Areas for further research include the automatic determination of the amplitude range of validity of a linear model. For each range, the parameters of the linearised model should be used, in terms of the angles of the elemental planes, to determine the function affecting the nonlinear behaviour of the system. A projection of the surface onto the appropriate plane may enable the nonlinear characteristic to be found.

Discussion has been limited to noise free systems. It is not clear how an independent noise source will interfere with the analysis procedures proposed. It is expected, however, that the effect of low variance disturbances can be minimised by averaging and by careful choice of the quantisation amplitudes.

For the consideration of experiment three recommendations for input signal x_i choice are

1 Input signal amplitude should include all nonlinearities in the range of operation in order to identify system fully.

2 The input signal amplitude should not be too big in order to guarantee the algorithm precision.

3 Properly set quantised data length nx in order to keep a trade off between algorithm precision and computing time.

Chapter 4 A variable weighted least squares algorithm

4.1 Introduction

Many papers have paid attention to nonlinear system identification, especially to parameter estimation (Narendra and Gallmann 1966, Bard 1974, Douce 1976 Svoronos, Stephanopoulos and Aris 1981, Leontaritis and Billings 1985, Chen and Billings 1988) under the assumption that the system structure is known or that the system can be approximated by a chosen model, from input and output data. One significant reason parametric models are so popular is that the Proportional Integral and Derivative (PID) regulator and most of the modern controllers, such as the self-tuning controller, adaptive controller, optimal controller, etc., use a parametric model. Thus, for a wide range of practical applications parametric models are used in controller design.

A wide range of nonlinear systems can be represented by Nonlinear AutoRegressive Moving Average models with eXogenous inputs (NARMAX) (Leontaritis and Billings 1985 a, b), and these have been successfully applied for self tuning controller design. The model mainly involves the determination of the order of the system, the variables to be employed and the estimation of the parameters. In fact this is a type of generalised nonlinear regressive analysis model in linear parameters.

It is well known that most control systems encounted in practice are nonlinear to some extent. However it may be possible to represent systems which are perturbed over a restricted operating range by a linear model. In these cases an AutoRegressive Moving Average (ARMA) model is commonly used with a combination of least squares algorithms to estimate the parameters. This is generally called a piecewise linearisation technique.

Least squares type algorithms, such as ordinary least squares and weighted least squares, are popular for use in both off line and on line parameter estimation. The ordinary least squares algorithm is known to have optimal properties

when the parameters are time invariant, however it is unsuitable for either tracking time-varying parameters or approximating nonlinear systems, and can give biased parameter values. Thus considerable research effort has been directed towards the development of modified versions of the algorithm. Research has led to study of special weighting functions, where the weighting function is time varying. The best known of these modified ordinary least squares algorithm is exponential data weighting (Goodwin and Payne 1977) and the modified version (Salgado, Goodwin, and Middleton 1988) with exponential resetting and forgetting when the excitation is poor. Another interesting idea in this regard is the variable forgetting factor algorithm of Fortescue, Kershenbaum, and Ydstie (1981), in which at each step a weighting factor is chosen to maintain constant a scalar measure of the information content of the estimator. It has been shown that, for nearly deterministic systems, such an approach enables the parameter estimates to follow both slow and sudden changes in the plant dynamics. Approximations allowing modelling of nonlinear systems has also been studied.

The algorithms mentioned above are particularly useful for dynamic performance assessment in the presence of poor excitation.

In this chapter, a second order ARMA model is adopted to approximate a wide range of nonlinear systems, and a Variable Weighted Least Squares (VWLS) algorithm is developed with a suitable weighting choice to deal with nonlinear characteristics. The weighting choice concerns the state amplitude distance between the current state and past states. This novel weighting is introduced to handle the amplitude effects in nonlinear systems. This algorithm removes some of the difficulties in nonlinear system structure detection.

First the background relevant to the new algorithm is presented.

Consider the modelling of a single-input-output nonlinear system by the general expression

$$y(n) = f\,[\,y(n-1), y(n-2),\ldots u(n), u(n-1),\ldots] \qquad (4.1.1)$$

where u and y are respectively the input and output of the system, and f is an unknown general nonlinear function.

Given a record of input-output signal u_R, y_R the response to a specific input sequence u_T is to be determined.

The basic technique is to linearise the system about the current, instantaneous operating point, to predict the output $\hat{y}_T(n)$. For the known inputs, $u_T(n)$, $u_T(n-1)$... and past outputs (known or predicted) $y_T(n-1)$, $y_T(n-2)$... the amplitude distance in state space between this point and every point on the given record of u_R and y_R is computed.

A weighted least squares technique is then used to calculate a linearised model, with each set of data in the records u_R, y_R being weighted according to the inverse square of this distance.

Using this linearised model, the predicted output $\hat{y}_T(n)$ is obtained. This weighting is repeated for each output point to be estimated.

The behaviour of a system described by a linear ARMA model is defined by

$$y(t) = \phi^T(t)\Theta + \varepsilon(t) \tag{4.1.2}$$

where

$$\phi^T(t) = [y(t-1), \ldots, y(t-na), u(t), \ldots, u(t-nb)]$$

$$\Theta^T = [a_1, \ldots, a_{na}, b_0, \ldots, b_{nb}] \tag{4.1.3}$$

u(t) and y(t) are the system input and output respectively, ε is a disturbance with zero mean value and limited variance σ^2.

The standard Weighted Least Squares (WLS) algorithm is given by

$$\hat{\Theta}(t) = [\,R(t)\,]^{-1} f(t) \tag{4.1.4}$$

where

$$R(t) = \frac{1}{t} \sum_{k=1}^{t} \beta(t,k)\, \phi(k)\, \phi^T(k)$$

$$f(t) = \frac{1}{t} \sum_{k=1}^{t} \beta(t,k)\, \phi(k)\, y(k) \tag{4.1.5}$$

$\beta(t,k)$ is a time varying weighting function. The corresponding recursive form is

$$\hat{\Theta}(t) = \hat{\Theta}(t-1) + R^{-1}(t)\, \phi(t)\, [\, y(t) - \phi^T(t)\, \hat{\Theta}(t-1)\,]$$

$$R(t) = \lambda(t) R(t-1) + \phi(t) \phi^T(t) \tag{4.1.6}$$

where the weighting sequence $\lambda(t)$ has the following property,

$$\beta(t,k) = \lambda(t)\beta(t-1,k) \quad 1 \le k \le t-1$$

$$\beta(t,t) = 1 \quad 0 < \lambda \le 1 \tag{4.1.7}$$

this means

$$\beta(t,k) = \prod_{j=k+1}^{t} \lambda(j) \tag{4.1.8}$$

The weights take account of the time correlation of current data with past data, that is the farther from current time the less weight, hence $\lambda(t)$ is usually called forgetting factor. By applying the matrix inversion lemma, eqn(4.1.6) may be simplified to a standard form given in many textbooks.

4.2 Parameter estimation

This part will presents both off-line and on-line VWLS algorithms and analyse the properties of the off-line algorithm.

4.2.1 Off-line algorithm

A second order ARMA model is given by

$$y(t) = a_1 y(t-1) + a_2 y(t-2) + b_0 u(t) + b_1 u(t-1) + c_0 + \varepsilon(t) \tag{4.2.1}$$

It may be written in vector form:

$$y(t) = \phi^T(t)\Theta + \varepsilon(t) \tag{4.2.2}$$

where

$$\phi^T(t) = [y(t-1), y(t-2), u(t), u(t-1), 1]$$

$$\Theta^T = [a_1, a_2, b_0, b_1, 1] \tag{4.2.3}$$

The parameter vector is estimated by a VWLS algorithm

$$\hat{\Theta} = [R(t)]^{-1} f(t) \tag{4.2.4}$$

where

$$R(t) = \frac{1}{t} \sum_{k=1}^{t} w(t,k)\phi(k)\phi^T(k)$$

$$f(t) = \frac{1}{t} \sum_{k=1}^{t} w(t,k)\phi(k)y(k) \qquad (4.2.5)$$

Equations (4.2.1) to (4.2.5), are the particular cases of the expressions given in eqn(4.1.2) to eqn(4.1.5) in the introduction.

The critical novel point is the choice of the weight w(t,k). In principle, it can be any non-negative function which decreases with increasing distance. In practice, it is convenient to use a square law, with the maximum value limited for sufficiently small distance, this is set to be

$$\frac{1}{w(t,k)} = (u(t) - u(t-k))^2 + (u(t-1) - u(t-k-1))^2$$
$$+ (y(t-1) - y(t-k-1))^2 + (y(t-2) - y(t-k-2))^2 \qquad (4.2.6)$$

which is determined from the nearness of current state to the previously measured states. That is, the more the weight is given, the smaller the amplitude distance between current state and previous known states.

4.2.2 Properties of algorithm

Two properties of the VWLS algorithm are of fundamental importance. First is noticed the similarity between the VWLS and Ordinary Least Squares (OLS) techniques for linear systems, and secondly the geometrical interpretation of the weight choice. These are now demonstrated.

1. For a time-invariant linear system, the VWLS algorithm is equivalent in expected value to the OLS algorithm when the uncorrelated disturbance has zero mean value.

Proof

According to the conditions given, it follows, from eqn(4.2.5),

$$E[R(t)] = E[R(t+1)]$$

$$E[f(t)] = E[f(t+1)] \qquad (4.2.7)$$

Substituting eqn(4.2.7) into eqn(4.2.4), produces

$$E[\hat{\Theta}_v(t)] = E[\hat{\Theta}_v(t+1)] \tag{4.2.8}$$

where v denotes the parameter vector estimated by VWLS algorithm.

For OLS algorithm, it follows, from eqn(4.1.2) and eqn(4.1.3) ($\beta(t,k)=1$)

$$E[\hat{\Theta}_o(t)] = E[\hat{\Theta}_o(t+1)] \tag{4.2.9}$$

where o denotes the parameter vector estimated by OLS algorithm. It may be proved, Norton (1987), that both $E[\hat{\Theta}_v(t)]$ and $E[\hat{\Theta}_o(t)]$ are unbiased estimates for uncorrelated disturbance with zero mean value, hence

$$E[\hat{\Theta}_v(t)] = E[\hat{\Theta}_o(t)] = \Theta \tag{4.2.10}$$

For nonlinear systems eqn(4.2.7) does not hold.

It should be noticed that the OLS algorithm has the minimum covariance property for linear time invariant systems when error is uncorrelated (Norton 1987). Therefore for the linear systems

$$Cov[\hat{\Theta}_v] \geq Cov[\hat{\Theta}_o] \tag{4.2.11}$$

For nonlinear systems eqn(4.2.11) does not hold.

2. The weight chosen for the state amplitude distance is the inverse of the squared radius of a hypersphere with center at the current state (u(t), u(t-1), y(t-1), y(t-2)).

This may be understood by letting

$$\begin{aligned} R^2(t,k) &= (u(t)-u(t-k))^2 + (u(t-1)-u(t-k-1))^2 \\ &\quad + (y(t-1)-y(t-k-1))^2 + (y(t-2)-y(t-k-2))^2 \end{aligned} \tag{4.2.12}$$

this is a hypersphere with radius R(t,k) and centre at (u(t), u(t-1), y(t-1), y(t-2)). The radius varies with differing previous states, but the centre is fixed. Therefore the integration of the weights is determined by a set of radii of a concentric hypersphere, that is

$$w(t,k) = \frac{1}{R^2(t,k)} \tag{4.2.13}$$

An example is shown in Fig. 4.2.1 for the simplest case of $R^2(t,k) = (u(t)-u(t-k))^2 + (y(t-1)-y(t-k-1))^2$.

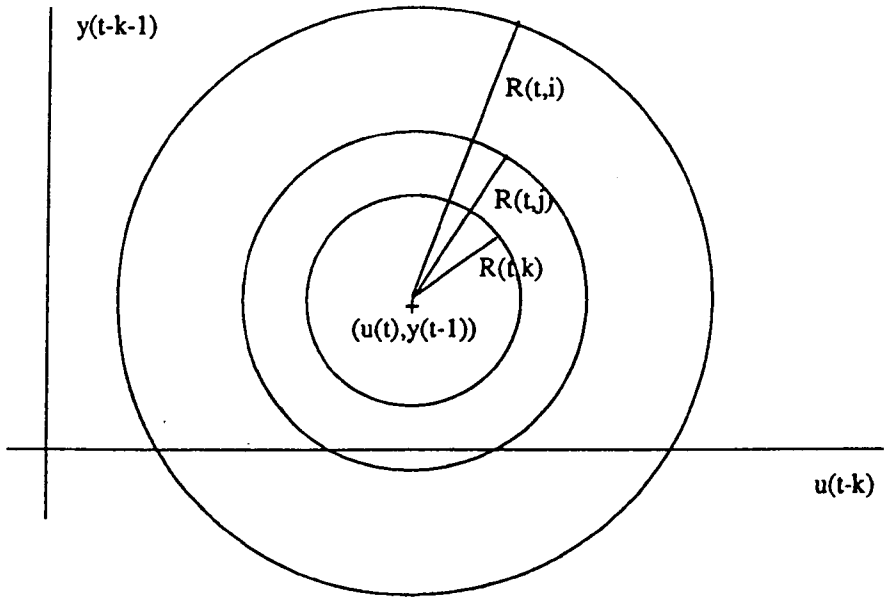


Figure 4.2.1 Weighting geometric interpretation

4.2.3 Modified (on-line) algorithm

It has been shown that for the off-line algorithm, for every current state all weights must be calculated, and the memory length increases with the growth of data series. Hence the algorithm can not be directly written in recursive form. However it is possible to apply the technique in a modified algorithm with the aim of on-line implementation.

With reference to eqn(4.2.4) and eqn(4.2.5), the algorithm is given by:

$$\hat{\Theta} = [R(t)]^{-1} f(t) \tag{4.2.14}$$

where

$$R(t) = \frac{1}{t-i} \sum_{k=t-i}^{t} w(t,k)\phi(k)\phi^T(k)$$

$$f(t) = \frac{1}{t-i} \sum_{k=t-i}^{t} w(t,k)\phi(k)y(k) \tag{4.2.15}$$

This uses only a limited amount of past data, extending over i points, reducing the computation required at the trade off of a reduced data set leading to

reduced accuracy. An alternative possibility, for time invariant systems, is to smooth the gathered data, so that a new input and output measurement is combined with other measured data which has similar values of all the states u(t), u(t-1), y(t-1), y(t-2). The algorithm is similar to that in eqn(4.2.15), the differences are

$$\phi^T(k) = [\,\bar{y}(k-1), \bar{y}(k-2), \bar{u}(k), \bar{u}(k-1), 1\,] \qquad (4.2.16)$$

where – denotes the smoothed data.

4.3 Experiment results

Four representative systems as shown in Fig. 4.3.1 are selected for the simulation study. These are

1 Second order linear system defined by

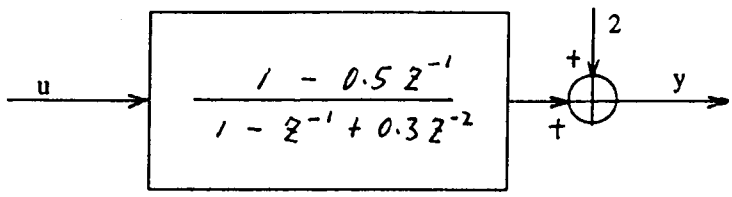
$$y(t) = a_1 y(t-1) + a_2 y(t-2) + b_0 u(t) + b_1 u(t-1) + c_0 \qquad (4.3.1)$$

where the parameters are set to $(a_1, a_2, b_0, b_1, c_0)$=(1, -0.3, 1, -0.5, 2). This example will be used to check the equivalence for the given performance criterion between VWLS and OLS in linear case. Since OLS is optimal algorithm, VWLS is also expected to have the same property when the variance of the disturbance is small.

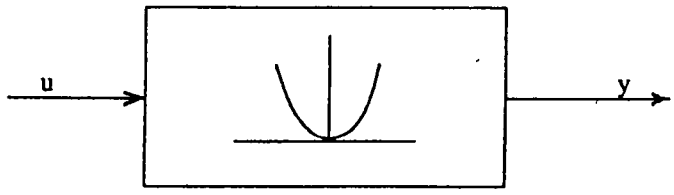
2 Static square defined by

$$y(t) = u^2(t) \qquad (4.3.2)$$

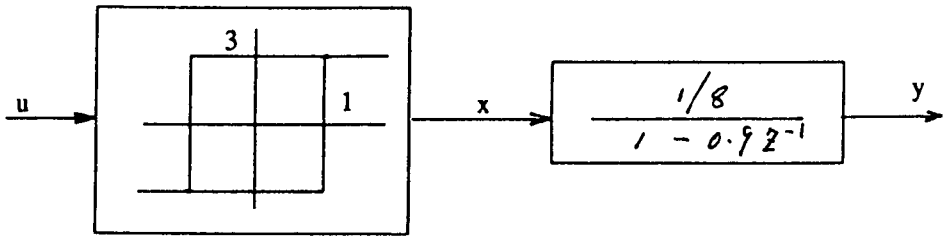
which will be used to check the algorithm for a static nonlinearity; a very common situation in industrial areas.



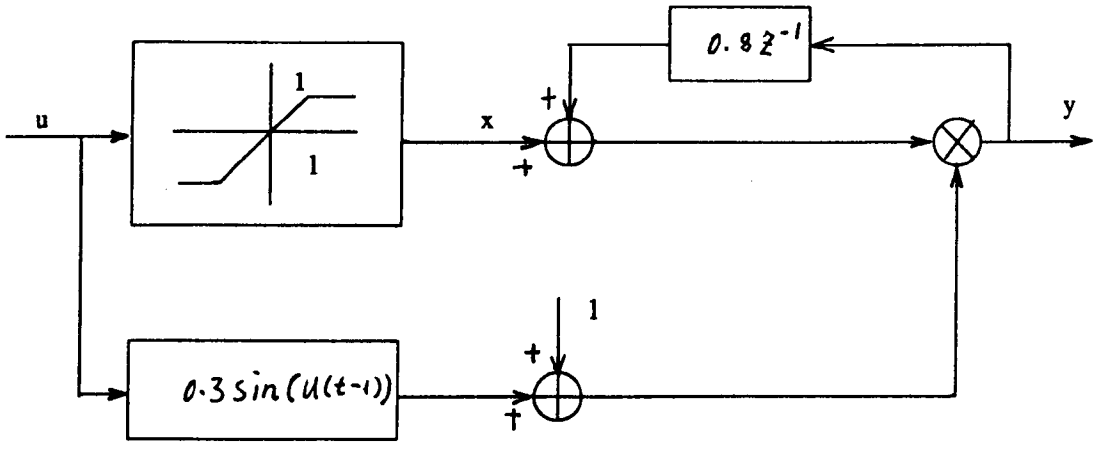
(a)



(b)



(c)



(d)

Figure 4.3.1 Experimental systems

3 First order lag after hysteresis defined by

$$y(t) = 0.9y(t-1) + x(t)/8$$

$$x(t) = \begin{cases} 3sign(u(t)) & |u(t)| \geq 1 \\ x(t-1) & otherwise \end{cases}$$

$$sign(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \qquad (4.3.3)$$

which will be used to check the technique for a class of nonlinear systems encountered in many engineering fields. It represents the cascade of a multi-valued nonlinearity with a linear dynamic system.

4 Complex nonlinear defined by

$$y(t) = (0.8y(t-1) + x(t))^* (1 + 0.3sin(u(t-1))$$

$$x(t) = \begin{cases} sign(u(t)) & |u(t)| \geq 1 \\ u(t) & otherwise \end{cases} \qquad (4.3.4)$$

which will be used to check the approximation for a complex nonlinear system.

Four input signals, sinusoid, Gaussian, step, and random amplitude steps are available to demonstrate the model behaviour for different input signals

A noise signal is superposed on system input, that is

$$z(t) = u(t) + \varepsilon(t) \qquad (4.3.5)$$

where u(t) is the input signal without contamination. This is a much worse case than when an uncorrelated noise is superimposed on the system output. The variance of the noise $\varepsilon(t)$ is selected for each experiment.

Two working modes are (a) the one step predictor and (b) the model response or so called multi-step predictor. The first one is given by

$$\hat{y}(t) = a_1 y(t-1) + a_2 y(t-2) + b_0 u(t) + b_1 u(t-1) + c_0 \qquad (4.3.6)$$

where $\hat{}$ denotes predicted value. The second one is given by

$$\hat{y}(t) = a_1\hat{y}(t-1) + a_2\hat{y}(t-2) + b_0u(t) + b_1u(t-1) + c_0 \qquad (4.3.7)$$

The criterion to evaluate the algorithm performance is a normalised output error squared defined by

$$e = \frac{\sum(y(t) - \hat{y}(t))^2}{\sum y^2(t)} \qquad (4.3.8)$$

An interactive simulation procedure is implemented in the matrix laboratory package Matlab. This includes a learning phase at the beginning of the program execution, in which the measured output is recorded as a function of past output and current and past input values. The length of input and output data is usually set to 800, the first 500 data points are dedicated to the learning phase.

In order to prevent weighting factors from being very large, a limiter is used, i.e.

$$w(t) = \begin{cases} w & \frac{1}{w} \geq \alpha \\ \alpha & \frac{1}{w} \leq \alpha \end{cases} \qquad (4.3.9)$$

where α is a preset positive constant.

A comparison has been made with results using the OLS algorithm. Table 4.3.1 shows the resultant errors defined in eqn(4.3.8), in which e1 and e2 denote one step prediction error and model response error respectively. This table demonstrates the VWLS has better properties in both one step prediction and model response for the concerned nonlinear systems, and is the same as OLS for linear systems. For the linear system it should be noted that the OLS parameter estimates are expected to have lower variances for big disturbances due to its minimum variance property, and hence lower error measures even though the same error measures are obtained by the two algorithms with small disturbance in the experiment.

Predictor errors
Input: Gaussian

system	OLS		VWLS	
	e1	e2	e1	e2
1	0.000	0.001	0.000	0.001
2	0.521	0.522	0.108	0.111
3	0.023	0.225	0.017	0.191
4	0.047	0.238	0.024	0.111

Predictor errors
Input: sinusoid

system	OLS		VWLS	
	e1	e2	e1	e2
1	0.000	0.003	0.000	0.003
2	0.312	0.414	0.184	0.187
3	0.039	0.031	0.000	0.003
4	0.005	0.009	0.000	0.004

Predictor errors
Input: random amplitude step

system	OLS		VWLS	
	e1	e2	e1	e2
1	0.000	0.003	0.000	0.003
2	0.341	0.691	0.123	0.181
3	0.004	0.533	0.002	0.383
4	0.017	0.690	0.011	0.538

Predictor errors
Input: step

system	OLS		VWLS	
	e1	e2	e1	e2
1	0.000	0.002	0.000	0.002
2	0.552	0.566	0.482	0.486
3	0.000	0.012	0.000	0.011
4	0.000	0.033	0.000	0.023

Table 4.3.1

4.4 Application--Analysis of a saturating second order system

This part will analyse the behaviour, including frequency response and step response characteristics, of a saturating second order system presented by Douce (1963), as one of the applications of the VWLS algorithm.

4.4.1 Jump resonances

Jump resonance is one of the phenomena exhibited in nonlinear systems. Some basic concepts are introduced before a new technique for the prediction of jump resonance is introduced. Ogata (1970) summarised the phenomenon.

In carrying out experiments on the forced oscillations of a system with differential equation,

$$m\frac{d^2x}{dt^2} + f\frac{dx}{dt} + kx + k'x^3 = p\ \cos\Omega t \tag{4.4.1}$$

where

$$p\ \cos\Omega t = forcing\ function \tag{4.4.2}$$

one may observe a number of phenomena, such as multivalued responses, and a variety of periodic motions (such as subharmonic oscillations and superharmonic oscillations). These phenomena do not occur in the response of linear systems.

In carrying out experiments in which the amplitude p of the forcing function is held constant, while its frequency is varied slowly and the amplitude x of the response is observed, one may obtain a frequency response curve similar to that shown in Fig. 4.4.1. Suppose that $k' < 0$ and that the forcing frequency Ω is low at the start at point 1 on the curve of Fig. 4.4.1. As the frequency Ω is increased, the amplitude x increases smoothly and continuously until point 2 is reached. A further increase in the frequency Ω will cause a jump from point 2 to point 3, with accompanying changes in amplitude and phase. This phenomenon is called a jump resonance. As the frequency Ω is increased further, the amplitude x follows the curve from point 3 toward point 4. In performing the experiment in the other direction, i.e., starting from a high frequency, one observes that as Ω is decreased, the amplitude x slowly increases through point 3, until point 5 is reached. A

further decrease in Ω will cause another jump from point 5 to point 6, accompanied again by changes in amplitude and phase. After this jump, the amplitude x decreases with Ω and follows the curve from point 6 toward point 1. Thus, the response curve is actually discontinuous, and a representative point on the response curve follows different paths for increasing and decreasing frequencies. The response corresponding to the curve between point 2 and point 5 correspond to unstable oscillations, and they can not be observed experimentally. One thus sees that for a given amplitude p of the forcing function there is a range of frequencies over which either of the two stable responses can occur. It is noted that for jump resonance to take place, it is necessary that the damping term should be small and that the amplitude of the forcing function should be large enough to drive the system into a region of appreciable nonlinear operation.

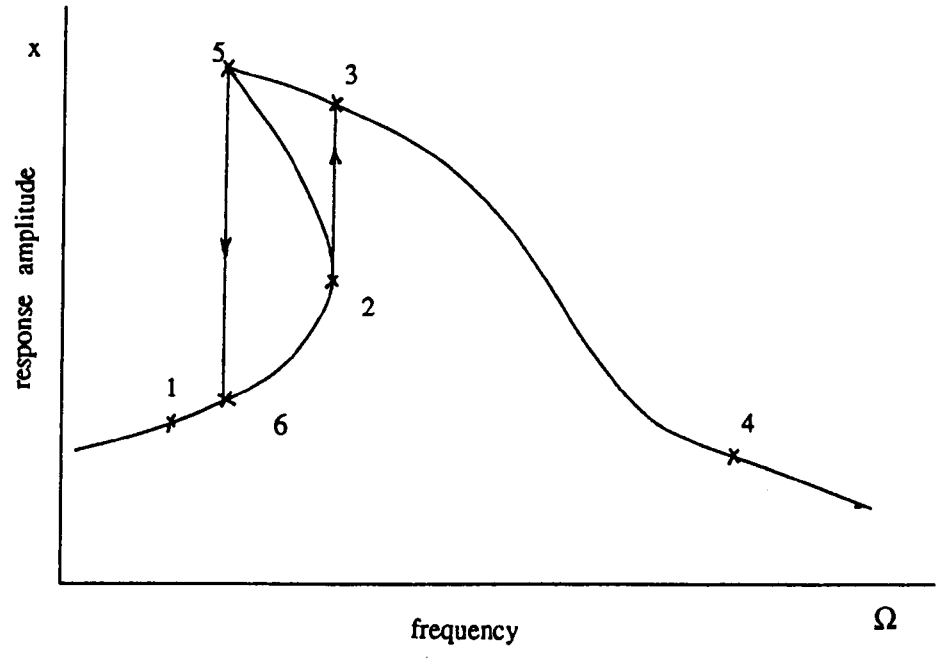


Figure 4.4.1 frequency response curve showing jump resonance

4.4.2 Jump effect prediction

The new modelling technique has been evaluated by applying it to the prediction of jump resonance, and comparing these predictions with experiments on the system selected. The learning phase involves recording the system response to a wide band zero-mean Gaussian signal so that a record is built up of system output as a function of past values of the output and present and past values of the system input.

A multi-step predictor as given in eqn(4.3.7) is used to obtain system output response excited by a set of sinusoidal inputs with different frequency, the varying parameters in the model being estimated by the VWLS algorithm. A critical point is that a learning phase proceeds from the real system before the multi-step predictor is involved. When the multi-step predictor starts working, that is as the computer simulation is initiated, the knowledge of the system kept in the learning data file is available to guide the model parameter estimation and output prediction.

4.4.3 Simulation experiment

In order to study the problem, a model is set up to describe a typical amplifier-motor combination as shown in Fig. 4.4.2, reproduced from Douce 1963. The motor torque is assumed proportional to the voltage fed to the amplifier. This linear relationship is obeyed only if the input voltage does not exceed a limited value, due to overloading or saturation of some elements of the system. This causes the system to operate in a nonlinear manner.

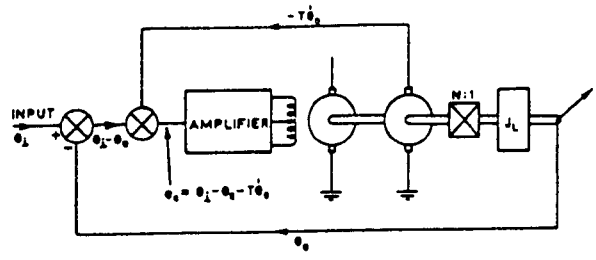
For a large input voltage, the output current from the amplifier remains approximately constant, and the magnetic flux in the motor, limited by saturation of the iron, attains its maximum value. Thus the motor torque, and the acceleration of the motor, is independent of amplifier input voltage if this voltage is greater than a particular magnitude. In this simple angular position-control system, the linear equations are applicable only for small control signals.

The schematic of the second-order position-control system is shown in Fig. 4.4.2(a). The motor produces a torque proportional to the amplifier input signal, unless a torque is demanded greater than the maximum available. Let the magnitude of the control signal which is just sufficient to apply maximum motor torque be h.

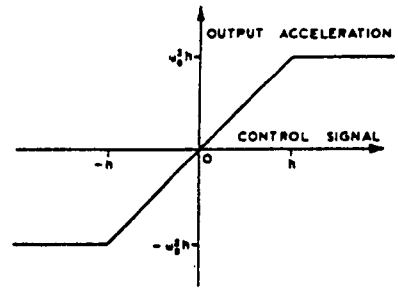
For $|e_c| < h$ linear analysis is applicable, giving, with the usual notation,

$$\frac{d^2\theta_o}{dt^2} = w_o^2(e_c) = w_o^2(\theta_i - \theta_o - T\frac{d\theta_o}{dt}) \qquad (4.4.6)$$

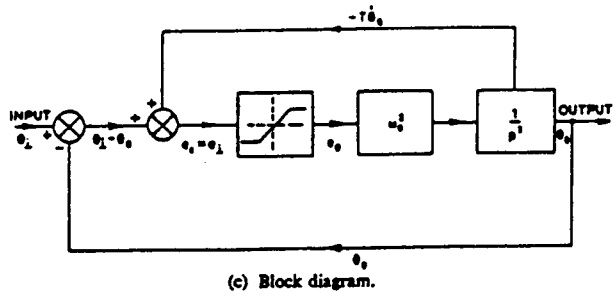
When the signal is equal to or exceeds h, the acceleration of the output has its maximum value $w_o^2 h$. This gives the relationship between output acceleration and control signal magnitude shown in Fig. 4.4.2(b).



(a) The basic system.



(b) Characteristic of the saturating element.



(c) Block diagram.

Figure 4.4.2 Analysis of saturating second order system

The block diagram of the system can now be constructed in Fig. 4.4.2(c). This shows the nonlinear characteristic as a separate element, an essential step in analysing the behaviour of any nonlinear system for any input.

Noting that the relationship between the output signal of the nonlinearity, e_o, and the input signal e_i can be written as follows,

$$e_o = \begin{cases} e_i & |e_i| \le h \\ h & e_i > h \\ -h & e_i < -h \end{cases}$$

or

$$e_o = \frac{1}{2}(|e_i + h| - |e_i - h|) \tag{4.4.7}$$

When the effect of saturation is considered, direct solution of the differential equation is not possible in general and the superposition theorem is inapplicable as well known. Therefore the response to a particular input can not be determined from the known response to a different input signal. The method of analysis to be adapted depends on the form of the input signal applied. In this case, the signal to be considered is sinusoid. The discrete form of the experimental system is given by

$$e_1 = \theta_i(t) - \theta_o(t-1)$$

$$e_i = e_1 - T\dot{\theta}_o(t-1)$$

$$e_o = \begin{cases} e_i & |e_i| \le h \\ h & e_i > h \\ -h & e_i < -h \end{cases}$$

or

$$e_o = \frac{1}{2}(|e_i + h| - |e_i - h|) \tag{4.4.8}$$

$$\dot{\theta}_o(t) = \Delta e_o + \dot{\theta}_o(t-1)$$

$$\theta_o(t) = \Delta \dot{\theta}_o(t) + \theta_o(t-1) \tag{4.4.9}$$

which is a numerical algorithm to be implemented in computer program, where Δ is the sampling interval.

In this first experiment, the length of data in the learning phase is set to 3000 points. Then Gaussian input is changed into a set of sinusoidal inputs and the model output is predicted to obtain the corresponding frequency response characteristics. The relevant experimental parameters are set as follows

Input signal	$\theta_i = 1.5\sin\Omega t$
Frequency range of input	$\Omega = 0.01*(2\pi)$——$0.2*(2\pi)$
Frequency change step	$0.01*(2\pi)$
Data length per harmonic	100
System damping	T = 0.2
Break point of saturation	b = 1
Height of saturation	h = 1

Fig. 4.4.3(a, b) shows the model response against time. The first 3000 data points are generated from the system with a Gaussian input, for the learning phase. The last 4000 data points are obtained by the VWLS algorithm for the model response to a set of sinusoidal inputs with 200 data points per harmonic. The gain of the frequency response characteristics is calculated by smoothing the peak values in the range of last 100 data points for the each frequency. Fig. 4.4.3(a) shows the model response for increasing input frequency and Fig. 4.4.3(b) shows the results with decreasing frequency.

Fig. 4.4.4(a) shows the model frequency response characteristic. The obvious jump resonance may be observed by comparing the characteristics for frequency increasing and decreasing. Fig. 4.4.4(b) shows the system frequency response characteristic. Comparison of these two figures shows good qualitative agreement, particularly at high and low frequencies. The magnitude of the jump effect is, however, appreciably smaller for the model than for the true system. Two reasons for this are proposed. First, the learning phase is unlikely to include all amplitudes of states encountered in the sinusoidal input experiment, so linear extrapolation is invoked fairly extensively. Second, the method uses local linearisation. The jump effect is believed to be highly sensitive to the form of the nonlinear characteristic, and this smoothing is expected to reduce the magnitude of

this effect.

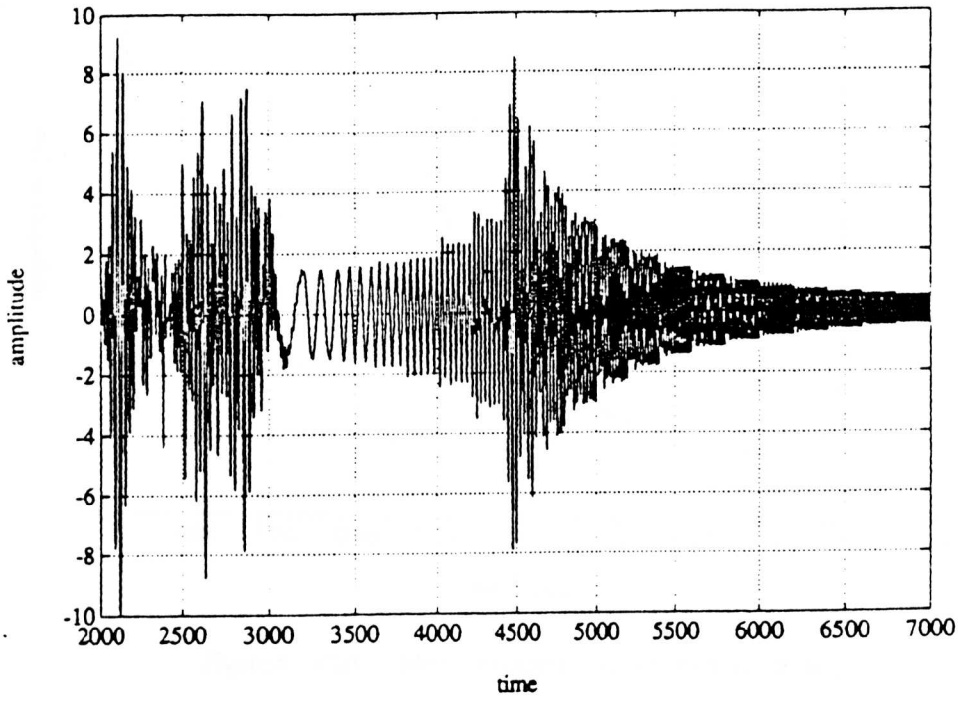


Figure 4.4.3(a) Sinusoidal response with frequency increase

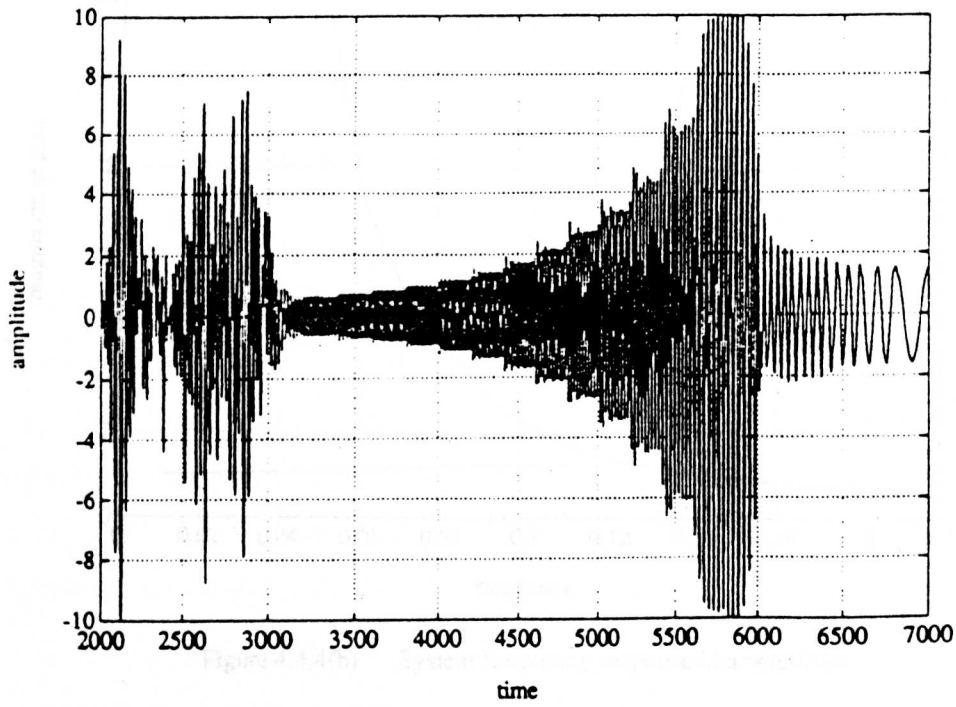


Figure 4.4.3(b) Sinusoidal response with frequency decrease

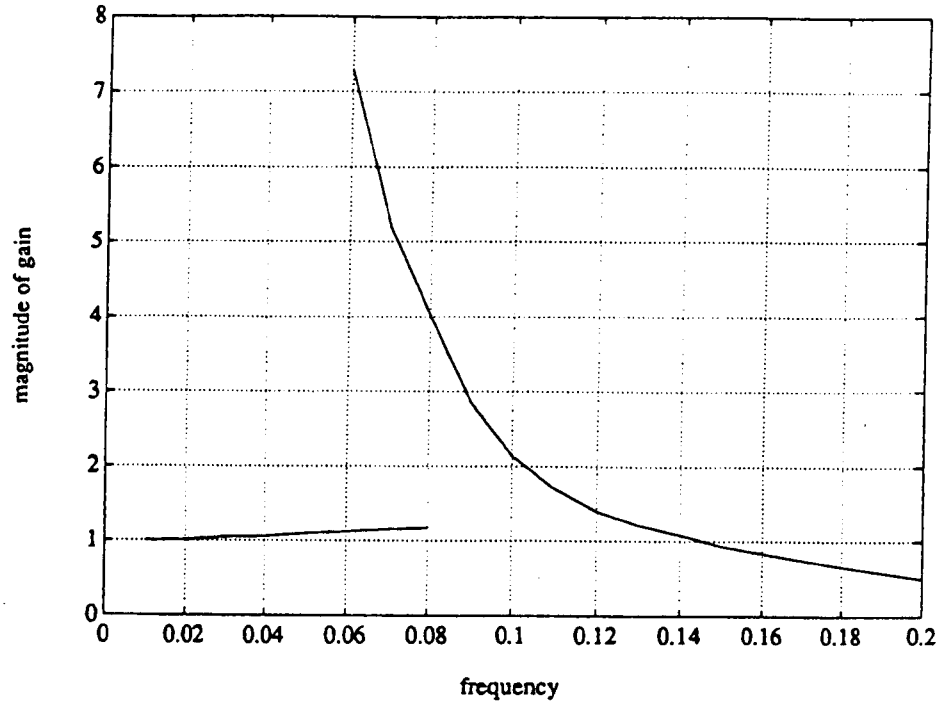


Figure 4.4.4(a) Model frequency response characteristics

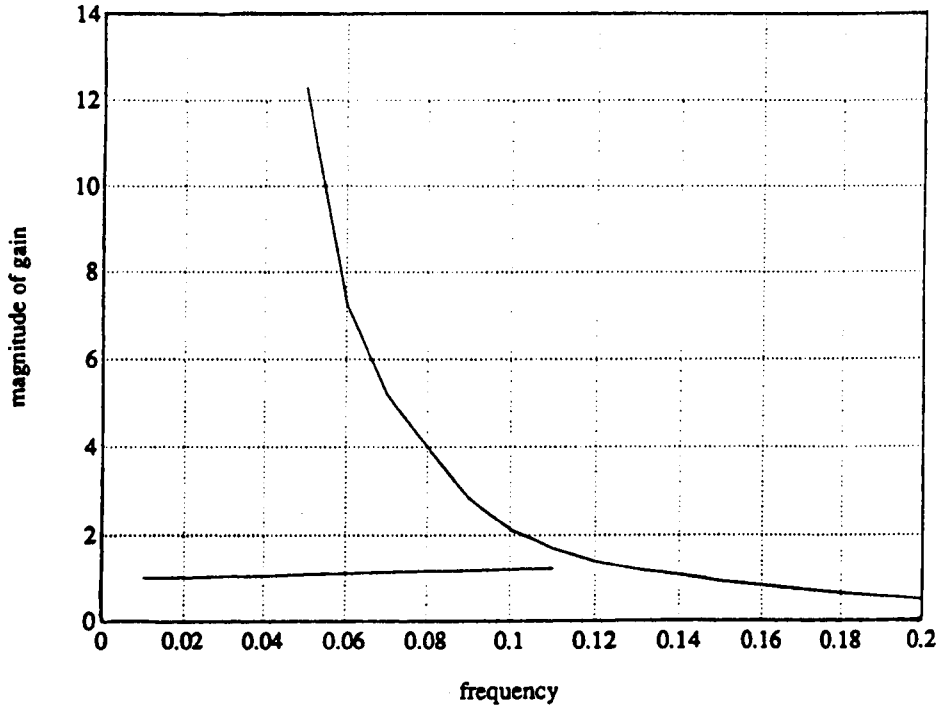


Figure 4.4.4(b) System frequency response characteristics

4.4.4 Step response experiment

The second experiment is carried out to check the validity of the VWLS algorithm to predict the response of the saturating second order system, shown in Fig. 4.4.1, to a step input.

Douce (1963) gave a detailed theoretical study of the response of the saturating second order system to a step input, which is summarised in Fig. 4.4.5 reproduced from Douce (1963).

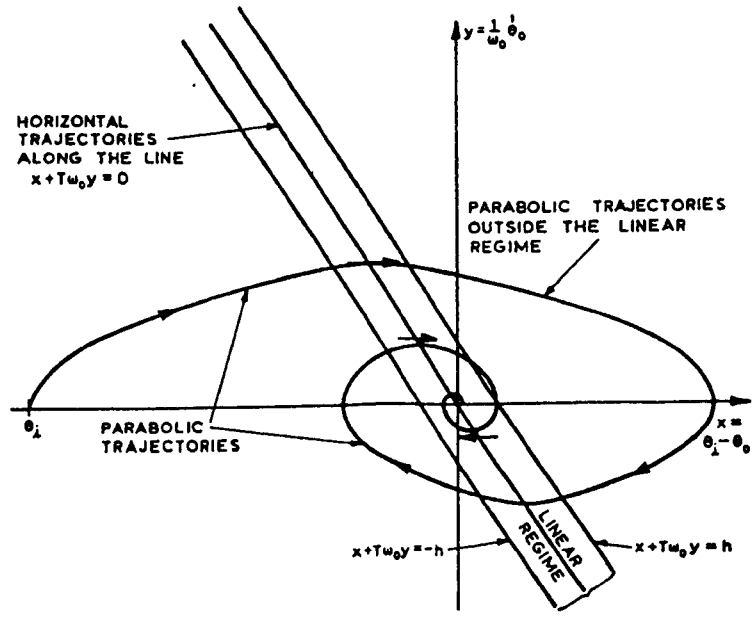


Figure 4.4.5 Phase-plane response of the saturating system

With reference to Fig. 4.4.2, when a step of magnitude greater than h is applied, maximum acceleration is applied initially, producing a parabolic trajectory on the phase plane, until the system enters the linear regime. Form this time the acceleration decreases, becoming zero when the trajectory cuts the line $x + Tw_0 y = 0$. Deceleration ensues, and if the trajectory leaves the linear regime it follows a further parabolic path.

A critical damped ($Tw_0 = 2$) system for small signal operation is selected in the experiment. For linear system there is no overshoot at the output for a step input with any amplitude. For the nonlinear system, Douce (1963) predicted that

there will be overshoot at the output for a step input with sufficiently large amplitude.

First 1000 data points as a learning phase are generated from random Gaussian input with zero mean value and variance 25, in which the measured output is recorded as a function of past output and current and past input values. Then inputing a step signal with amplitude 50 to the system, the output response is obtained in one step prediction mode given in eqn(4.3.6) and model response or multi-step prediction mode given in eqn(4.3.7) by the VWLS algorithm. The one step prediction shown in Fig. 4.4.6(a) indicates fairly the output response properties both in transient and steady states. The model response shown in Fig. 4.4.6(b) indicates errors in transient state and zero error in steady state. This is the worst test condition to the VWLS algorithm due to using all predicted past outputs instead of real past outputs for the model response calculation.

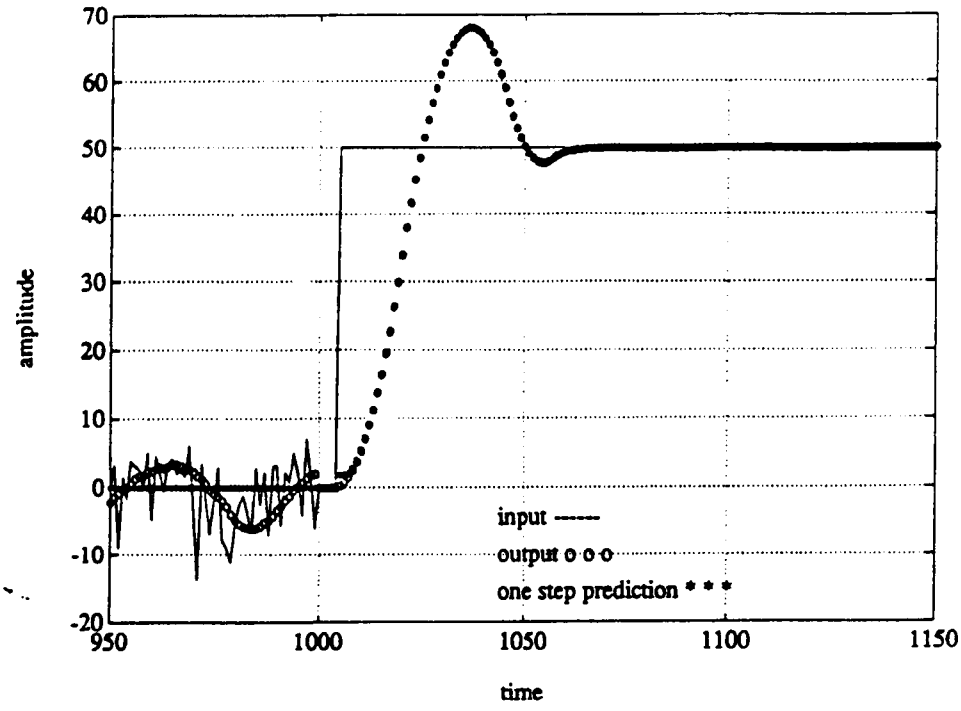


Figure 4.4.6(a) Step response of one step prediction

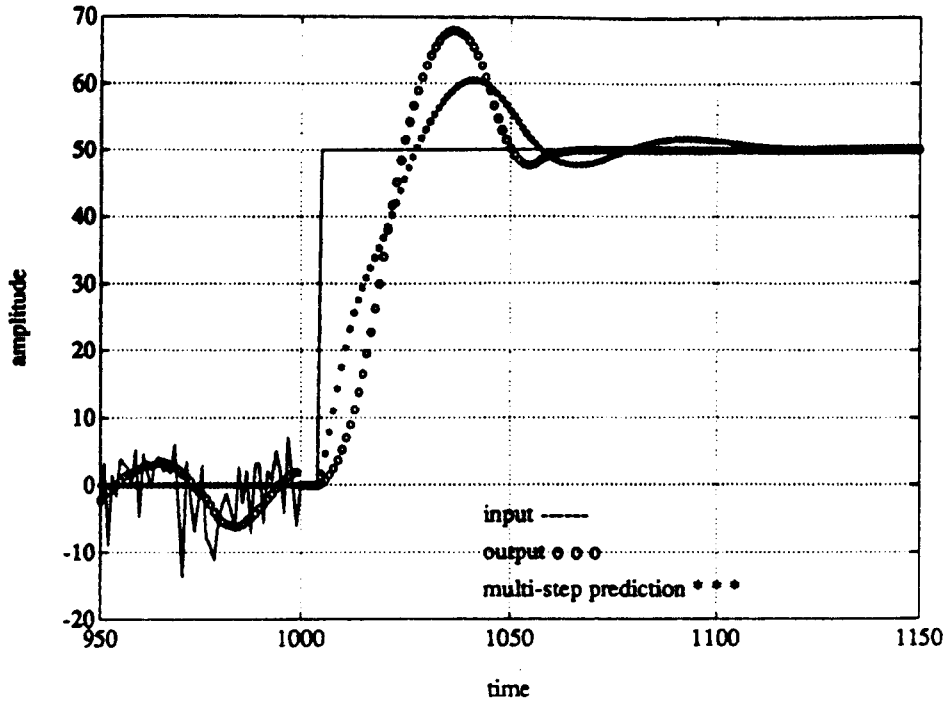


Figure 4.4.6(b) Step response of multi-step prediction

4.5 Conclusion

This chapter has introduced a novel method to model nonlinear system behaviour by the ARMA model with a weighted least squares algorithm.

The method involves a learning phase, during which input-output data is recorded. Using a priori knowledge, or tests with this data set, the order of the locally linearised model is determined.

To implement the model, for each time step the learning data is scanned, and this data is inserted into the model for linearisation according to the distance between the current state and each recorded state.

An interesting feature of the method is that the input data for the learning phase can differ substantially from that used in the modelling phase. The same modelling technique predicts two nonlinear phenomena which, to date, have been examined using totally different techniques.

The computer simulation studies have demonstrated the validity and efficiency of the method.

The areas for further research include experiments in laboratory or real environments and extension of the idea from the weighting choice to other typical parameter estimation algorithms such as maximum likelihood estimation and prediction error estimation.

Section 3 Self-tuning controller design

S3.1 Survey

A general problem in control theory and application is to design control laws or controllers which achieve better performance for any member of a specified class of plants or processes. This naturally involves system identification and parameter estimation which are often separate steps in the classical approach. With the increase of control quality demand and complicated plant control, adaptive control strategy, a suitable combination of above steps, has been suggested, studied, and tested in laboratories and real environments, and applied to solve real problems for a few decades. In particular, a representative starting milestone that may be recognised publicly was in year of 1958 when Kalman postulated the design of a machine which adjusts itself automatically to control an arbitrary dynamic process.

This section does not intend to give an exhaustive survey of the topic all-round. A brief introduction to nearly a decade of development of relevant branches, immediately highlight the subject---Self-Tuning Control (STC) for non-linear systems.

In the 1960s, the appearance of modern control theory, such as state space technique, Kalman filter theory and stability theory, was important for the development of adaptive control. Some fruitful results in stochastic control, such as dynamic programming (Bellman 1957, 1961), system identification and parameter estimation (Astrom and Eykoff 1971), and some fundamental contributions by Tsypkin (1971, 1973), increased the understanding of adaptive control development.

In the 1970s, there was a rapid and vigorous expansion both in theory and applications. Various novel and applicable controllers, such as minimum variance controller (Astrom 1970), general minimal variance controller (Clarke and Gawthrop 1975), the pole-assignment controller (Wellstead, Prager and Zanker 1979), and pole-zero placement controller (Astrom and Wittenmark 1980), were presented. Then many papers on the understanding, experimental studies and

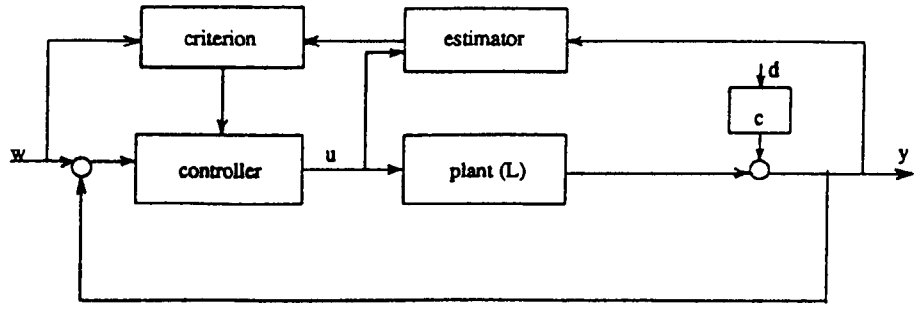
applications, and modified controllers based on the controllers were published. Meanwhile people began to consider and test the application to various plants and processes with the controllers. It is worthwhile to mention that Astrom and co-workers made great contributions to this field. The period was a revolutionary decade.

More recently, adaptive controllers entered a period of maturity. The advantages of different adaptive controllers based on linear models have been combined to build up some complex controllers such as the general predictive controller (Peterka 1984, Clarke, Mohtadi and Tuffs 1987), general pole placement controller (Lelic and Zarrop 1987, Lelic and Wellstead 1987). Astrom in 1983 made a summary of the current development of adaptive control. Although important theoretical results on stability and structure had been established, much theoretical work still remained to be done. The advent of microprocessors and industrial feasibility studies have contributed to a better understanding of the practical aspects of adaptive control, and a number of adaptive regulators had appeared on the market.

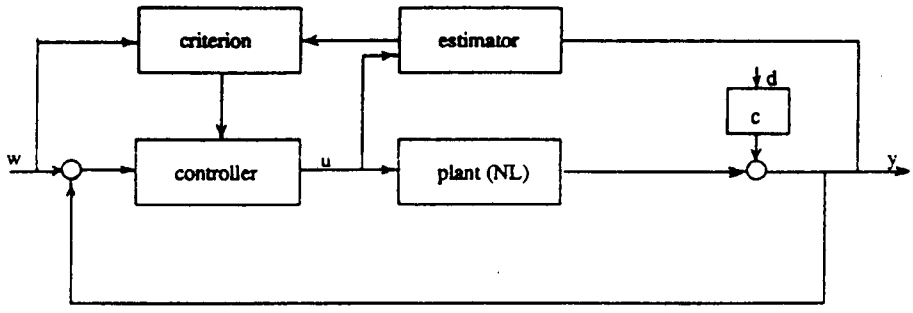
Astrom and Eykoff (1971), Clarke and Gawthrop (1979), Isermann (1982), Astrom (1983), Seborg, Shah, and Edgar (1986), Gupta (1986), Midelleton, Goodwin, Hill, and Mayne (1988), have also provided important reviews.

STC for nonlinear systems, one of the active branches of adaptive control, has been given attention since 1980s. The critical point is adaptive controller design based on a nonlinear plant model. Astrom and Wittenmark (1973) Clarke and Gawthrop (1979), and Isermann (1982) all consider the problem of how to run a self-tuner for long periods of time on a strongly nonlinear process. It is possible to construct algorithms similar to those developed from linear models as long as the nonlinear model structure is linear in the parameters.

Fig. 3.1 shows the general STC system framework. The difference between linear system STC shown in Fig. 3.1(a) and the nonlinear system STC shown in Fig. 3.1(b) is the plant structure. Much more attention should be paid to the model treatment instead of developing new control scheme for nonlinear systems.



(a) STC scheme for linear plants



(b) STC scheme for nonlinear plants

Figure 3.1 STC schemes

Although Astrom and people mentioned above forecast the problem and possible solution, almost no results were reported untile Anbumani, Patnaik and Sarma (1981), and Lachmann (1982) publications appeared, which improved considerably the control performance with NLSTC for nonlinear plants.

Controller design, whatever the plant is linear or nonlinear, and controller is traditional PID or modern STC, generally requires the following three procedures:

1 A proper model to approximate the plant.

2 A criterion for controller design and calculation of controller output based on the criterion chosen.

3 System test, such as stability analysis, output response to typical inputs, ability to suppress the system response to disturbance or noise.

The adaptive controllers studied in the section are of specified structure with the parameter values to be selected and all variables are of sampled form for implementation on digital computers.

In chapter five, a general predictive controller is designed based on a combined Hammerstien + ARMA system model, which is termed NonLinear General Controller (NLGPC). In chapter six, A NonLinear DeadBeat Controller is designed, based on the same model as in chapter five, by a direct method in which a new operator is introduced to simplify the design procedure to be the same as for linear deadbeat controller design.

S3.2 List of notations

STC	Self-Tuning Control
GPC	General Predictive Control
DBC	DeadBeat Control
NLSTC	NonLinear Self-Tuning Control
NLGPC	NonLinear General Predictive Control
NLDBC	NonLinear DeadBeat Control
HCARMA	Hammerstein Controlled Auto-Regressive Moving Average
HCARIMA	Hammerstein Controlled Auto-Regressive Integrated Moving Average
ERLS	Enhanced Recursive Least Squares

S3.3 List of figures

Fig. 3.1	Self-tuning control framework
Fig. 5.1.1	Model of plant characteristics
Fig. 5.3.1	Root-solving computational flow diagram
Fig. 5.4.1	Static nonlinearities
Fig. 5.4.2	NLDBC operating on the plant L1+NL1
Fig. 5.4.3	NLGPC operating on the plant L1+NL1
Fig. 5.4.4	NLGPC operating on the plant L2+NL1
Fig. 5.4.5	NLGPC operating on the plant L2+NL2
Fig. 6.4.1	NLDBC operating on the plant L1+NL1
Fig. 6.4.2	NLDBC operating on the plant L2+NL2
Fig. 6.4.3	NLDBC operating on the plant L3+NL1

Chapter 5 Nonlinear general predictive controller design

5.1 Introduction

Self-tuning control techniques are, in the main, based on the philosophy that the system to be controlled can be regarded as being linear. To this end, the topic is well developed both in term of theory and applications (Harris and Billings (eds) 1985, Warwick (ed) 1988). For a large number of systems such a linearizing approach is acceptable, any relatively small nonlinearities being effectively linearized by the controller. The resulting performance is deemed to be satisfactory but is nevertheless below that which would be expected, had the plant been perfectly linear. In order to obtain improved control over systems with small nonlinearities or to deal with systems with stronger nonlinearities, whilst obtaining the benefits of a self-tuning control method, it is necessary to take account of the nonlinearities in an appropriate way.

Because of the large number of different types of nonlinearity which can occur in practice (Atherton 1975, Cook 1986), extending a basic linear control scheme to account for all possibilities is not a realistic proposition and would necessarily result in a toolbox approach which requires a certain amount of operator selection and therefore plant knowledge. A much more sensible way of tackling the general problem is to employ a framework with which a large number of nonlinear plants can be adequately modelled. The most appropriate solution as far as self-tuning control is concerned (Anbumani, Patnaik, and Sarma 1981, Lachmann 1982, Agarwal and Seborg 1987), is to assume a parametric plant description via a Hammerstein model which basically constitutes a linear ARMA system model coupled with a polynomial of powers of the control input. By this means the plant to be controlled is thought of as consisting of a linear part cascaded with a nonlinear part. The ARMA model is then used to represent the linear part, whereas the extra power polynomial approximates the nonlinear part, as shown in Fig. 5.1.1. In fact such an underlying philosophy has been shown as sensible within an adaptive control scheme.

The feasibility of using a Hammerstein model based in adaptive controller has been studied (Grimble 1985). This has highlighted a major problem in the use of such models. This is the necessity for the on-line computation of a root solving algorithm for every recursion of the controller updating procedure. Not only is this an extremely time consuming process, which can be a particular problem when the sampling period is small, but also the accuracy of the root solving routine, which is needed to obtain a unique solution to the nonlinear plant model polynomial, can be poor, and this can produce stability problems and usually requires an odd number of polynomial roots in order to guarantee at least one real root.

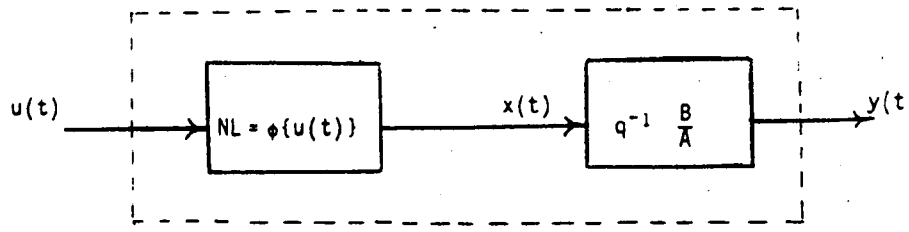


Figure 5.1.1 Model of plant characteristics

In this chapter an adaptive nonlinear controller is designed, based on the Hammerstein model, in term of NLGPC. The controller has the flexibility of a linear system form of General Predictive Control (GPC) (Clark, Mohtadi and Tuffs 1987), but can also deal effectively with a range of nonlinear systems. The design procedure entails two distinct parts, that is linear GPC design and polynomial root-solving. The method is therefore indirect. The overall technique can best be understood by reference to Fig. 5.1.1. Under the assumption that the model approximates the plant concerned very well, then a linear controller is designed to provide self-tuning GPC for the linear dynamic $q^{-1}B/A$ of the plant, which relates x(t), output of the plant nonlinear static part, to y(t), the plant output. The second step invokes a root-solving procedure, to calculate the inverse

function of the nonlinear part NL, to find a controller output u(t). The intermediate variable x(t), which is the input to the linear part of the system, is not a measurable quantity. However it may be estimated within the plant model due to the separability of the process (Billings and Fakhouri 1978). Note if $x(t)=\psi(u)=ku(t)$, where k is a scalar gain, the overall control problem reduces to a linear GPC requirement.

The novel approach taken in the chapter is to employ a one or two step Newton-Raphson iteration, for every recursion of the overall algorithm, in place of a complex root solving routine. The technique makes use of the signal value from the previous recursion of the algorithm, such that on-line computation is significantly reduced to a few arithmetic operations. The second benefit in the algorithm is that the variation of the controller output signal to be reduced from one sample period to the next. This follows since each input signal is based on its previous value so that the transmitted signal is filtered automatically. A positive feature of this controller is therefore the reduction of control input swings.

5.2 Plant model and output prediction

Consider a class of plants described by Hammerstein model

$$Ay(t) = Bx(t-1) + C\varepsilon(t) \tag{5.2.1}$$

where the polynomials A, B, and C are defined as,

$$A = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na}$$

$$B = b_0 + b_1 q^{-1} + \cdots + b_{nb} q^{-nb}$$

$$C = c_0 + c_1 q^{-1} + \cdots + c_{nc} q^{-nc}$$

$$x(t) = r_0 + r_1 u(t) + \cdots + r_{nr} u^{-nr}(t) = \sum_{i=0}^{nr} r_i u^i(t) \tag{5.2.2}$$

where q^{-1} is the backward shift operator such that $q^{-i}y(t) = y(t-i)$, u(t) is the plant input or controller output, y(t) is the measured variable or system output. $\varepsilon(t)$ is a disturbance affecting the system such that $\delta\varepsilon(t)=\xi(t)$, $\delta=1-q^{-1}$, which allows for nonzero offset on a zero mean, white noise signal $\xi(t)$ (Tuffs and

Clarke 1985). This representation of the model disturbance comes from the consideration in practice that two principal disturbances are encountered (Clarke, Mohtadi and Tuffs 1987), random steps occuring at random times such as changes in material quality and Brownian motion found in plants relying on energy balance. x(t) is an intermediate variable which is the nonlinear static element output or linear dynamic element input in the plant model.

The parameters a_i, b_i, and r_i cab be estimated by an Enhanced Recursive Least Squares (ERLS) approach (Kortamann and Unbehauen 1987), when the plant model is continuously updated within a self-tuning controller. It can be assumed initially that these are known or identified values, and the requirement for them to be estimated can be reintroduced later specifically for adaptive controller design purposes. Also, let C=1 for simplicity of explanation.

As a fundamental aspect of a predictive control-algorithm, a prediction of the plant output signal is required, which is based on information available at a particular time instant. By considering merely the linear part of the plant represented in eqn(5.2.1), a prediction of the output at time instant t+k is obtained directly, based on information available at time instant t. For this purpose a Diophantine identity is introduced (Clarke, Mohtadi, and Tuffs 1987, Owens and Warwick 1988)

$$1 = E_k A \delta + q^{-k} F_k \tag{5.2.3}$$

in which $k \geq 1$ and E_k, F_k are unique polynomials for any given integer k and polynomial A. Also E_k is monic and of order $ne = k-1$ such that

$$E_j = 1 + e_1 q^{-1} + \cdots + e_{ne} q^{-ne}$$

and

$$F_k = f_0 + f_1 q^{-1} + \cdots + f_{nf} q^{-nf} \tag{5.2.4}$$

where nf=na.

If now eqn(5.2.1) is multiplied throughout by $q^k E_k \delta$ and the Diophantine identity given in eqn(5.2.3) is made use of, it follows that

$$y(t+k) = B E_k \delta x(t+k-1) + F_k y(t) + E_k \varepsilon(t+k) \tag{5.2.5}$$

in which the disturbance terms present are all future values, this is due to the fact that ne=k-1.

The optimal output predictor for the output signal at time instant t+k, made at time instant t, is therefore

$$\hat{y}(t+k/t) = BE_k \delta x(t+k-1) + F_k y(t) \tag{5.2.6}$$

It is thus an obvious point that from eqn(5.2.5) and eqn(5.2.6), the actual system output can be regarded as

$$y(t+k) = \hat{y}(t+k/t) + E_k \varepsilon(t+k) \tag{5.2.7}$$

However, at an instant t, it is possible to select a range of values k for which an output prediction can be made.

Assume that a set of output predictions is made from k=1 to N, where $N \geq 1$, when the output eqn(5.2.5) may be rewritten in vector form

$$Y^T = GX + f + \varepsilon \tag{5.2.8}$$

where

$$\begin{aligned} Y^T &= [y(t+1) \cdots y(t+N)] \\ X^T &= [\delta x(t) \cdots \delta x(t+N-1)] \\ f^T &= [f(t+1) \cdots f(t+N)] \\ \varepsilon^T &= [E_1\varepsilon(t+1) \cdots E_N\varepsilon(t+N)] \end{aligned} \tag{5.2.9}$$

Further, the matrix G is of dimension N*N and has elements such that

$$G = \begin{matrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ \cdot & g_1 & \cdots & 0 \\ \cdot & \cdot & \cdots & 0 \\ \cdot & \cdot & \cdots & 0 \\ g_{N-1} & g_{N-2} & \cdots & g_0 \end{matrix} \tag{5.2.10}$$

where g_j is the coefficient in the integrated plant step-response, i.e. $g_j = h_j$ from

$$H = (\delta A)^{-1}B = h_0 + h_1 q^{-1} + h_2 q^{-2} \cdots \tag{5.2.11}$$

The vector GX in eqn(5.2.8) therefore represents a set of unknown values, due to the vector X, at time instant t. Note that the signal x(t),and therefore $\delta x(t)$, is considered to be unknown until it has been calculated and applied, thus at the instant that the output predictions are made, it is an unknown signal.

The vector f in eqn(5.2.8) represents a set of known values at time instant t, such that each term in f is given by

$$f(t+k) = F_k y(t) + h_k \qquad (5.2.12)$$

in which

$$h_k = \{BE_k - [\, h_0 + h_1 q^{-1} \cdots + h_{k-1} q^{-(k-1)}]\}\delta x(t+k-1) \qquad (5.2.13)$$

In the appendix 5.6.2, a detailed recursive algorithm is presented to implement the operations, including initial value settings.

5.3 Nonlinear general predictive controller

5.3.1 Controller design

The design of the overall control input to be applied to the plant contains two parts, that relating to the linear dynamic part of the plant and that relating to the nonlinear static part of the plant. The linear part is considered firstly, and for predictive control a cost function given by

$$J = \sum_{k=1}^{N} [y(t+k) - w(t-k)]^2 + \sum_{k=1}^{N} \lambda(k)[\delta x(t+k-1)]^2 \qquad (5.3.1)$$

is employed in which N is the maximum prediction horizon, $\lambda(k)$ is a weighting applied to the control inputs and w(t) is a reference input signal, applied at time instant t. Note that the control horizon and the output prediction horizon have been selected as the same value, N. This is not a necessary requirement, in fact other control horizon selections have been investigated elsewhere (Clarke, Mohtadi, and Tuffs 1987, Warwick and Clarke 1988).

The control objective is to obtain a vector of future control inputs X which will minimize the cost function in eqn(5.3.1). On taking the expected value of the cost function J,

$$E(J) = E[(GX + f + \varepsilon - W)^T (GX + f + \varepsilon - W) + X^T \lambda X] \qquad (5.3.2)$$

where λ is a N*N diagonal matrix whose diagonal elements are

$$\lambda(1), \cdots, \lambda(N) \qquad (5.3.3)$$

and

$$W^T = [w(t+1) \cdots w(t+N)] \qquad (5.3.4)$$

By differentiating eqn(5.3.2) with respect to X and setting the result to zero, a cost function minimum is given by the minimum mean square control

$$X = (G^T G + \lambda)^{-1} G^T (W - f) \qquad (5.3.5)$$

such that

$$x(t) = x(t-1) + g^T (W - f) \qquad (5.3.6)$$

where g^T is the first row of $(G^T G + \lambda)^{-1} G^T$.

So the control signal x(t) is the signal required to be applied to the linear part of the system, and which is based on a set of known future reference signals W along with the known vector f.

5.3.2 Fast recursive root-solving

The signal x(t) obtained in eqn(5.3.6) is an intermediate variable acting as a solution to the linear predictive controller problem. It remains for the nonlinear part of the controller problem to be solved, and this is done directly, remembering that x(t) is related to the control input to be applied to the plant, u(t), by the definition given in eqn(5.2.2).

The nonlinear problem can be stated as, with reference to eqn(5.2.2), given any signal x(t), at time instant t, and the appropriate coefficients (r_i, i=0, 1,...,nr). find the control input signal u(t).

With reference to Fig. 5.1.1 and eqn(5.2.2), the relationship between x(t) and u(t) may be written as

$$x(t) = \phi[u(t)] \qquad (5.3.7)$$

where $\phi(.)$ denotes a functional operator whatever it is linear or nonlinear, In this case it expresses a polynomial for the Hammerstein model. The requirement is to calculate the function inversion, i.e.

$$u(t) = \phi^{-1}[x(t)] \tag{5.3.8}$$

which means that one of the Hammerstein polynomial roots must be found in order to produce a possible control input signal u(t).

One suggested solution is provided by the Newton-Raphson recursive method (Gerald 1987), whereby

$$u_{n+1} = u_n - \frac{[\phi(u_n(t)) - x(t)]}{\phi'(u_n(t))}$$

$$\phi'(u_n(t)) = \frac{d\phi(u_n(t))}{du_n} \tag{5.3.9}$$

where the subscript n denotes the order of iteration, such that the (n+1)th iteration is obtained from the nth iteration, $n \geq 0$.

Two problems occur when appling eqn(5.3.9) directly, The first is that $\phi'(u_n(t)) \simeq 0$ in the neighbourhood of a solution. This is a critical point because in practice it cannot be guaranteed that the derivative of the function will not approximately equate with zero after any particular recursion due to model variation, the estimated error, and even an unsuitable initial value. The second problem is the possibility of no real root of the polynomial existing, thus causing a breakdown of the algorithm.

In order to overcome these drawbacks, whilst retaining simplicity, an improved root-solving approach based on eqn(5.3.9) has been developed. which is explained as follows.

When $\phi'(u_n(t)) \simeq 0$, there exist two possibilities. Either $u_n(t)$ is a root which satisfies the polynomial equation, or it is not. this can easily be checked by taking account of $\phi(u_n(t)) - x(t)$, which will be within a preset small value, i.e. approximately zero, if $u_n(t)$ is one of the roots. On the other hand, if $u_n(t)$ is not a root then a new alternative initial value is set and the recursion process is repeated. If no real root exists or if several searches have been carried out with alternative

initial values, a monitoring loop instructs the root-solving algorithm to stop and a default value is taken such that u(t)=x(t)/α, where α is a positive constant.

It is possible (Anbumani, Patnaik, and Sarma 1981) to restrict the polynomial order to be odd, thereby ensuring that there is at least one real root. Unfortunately this may introduce modelling error and restricts the types of the nonlinearities concerned. No such restriction is placed in the method described.

The solution to the second problem as mentioned above is to set the initial value to be $u_0(t) = u(t-1)$, i.e. the initial value is equal to the previously applied control input. This is a particularly suitable choice when the signal to noise ratio of the plant is high and /or when the reference input changes are either small or infrequent. In general it is found that with this initialisation procedure, the root-solving procedure used here involves, in most cases, a single iteration of eqn(5.3.9), each time a new solution is required. If the resultant solution does not fit well, a further preset finite number of recursions depending on the sampling period are carried out. If this is still not satisfactory, an alternative initial value is tried, as described above.

Note that a standard, more complex, root-solving algorithm can, if desired, be employed in order to obtain a solution. Care must be taken, however, where on-line, real-time applications are involved, due to the computing time requirements. A flow diagram for the nonlinear plant part solution is shown in Fig. 5.3.1, where it takes r_i, u(t-1), and x(t) as input values, and u(t) as output value, at any time instant. Four monitoring factors c1, c2, c3, and c4 are preset. c1 is used to control the iterating time of the algorithm, which is often chosen from a trade off between solution accuracy and solution obtainability, c2 is a error measurement that will determine the accuracy of the solution, c3 is a default value in case no suitable root is found. c4 is used to check whether the derivative $\phi'(u(t))$ enters a small region around zero, which is an indiction of either a possible solution or an inflecting point of the polynomial curve.

Another suggested solution is to generate a look-up table from the relationship $x(t) = \phi[u(t)]$, and then the controller output u(t) may be found for the given x(t).

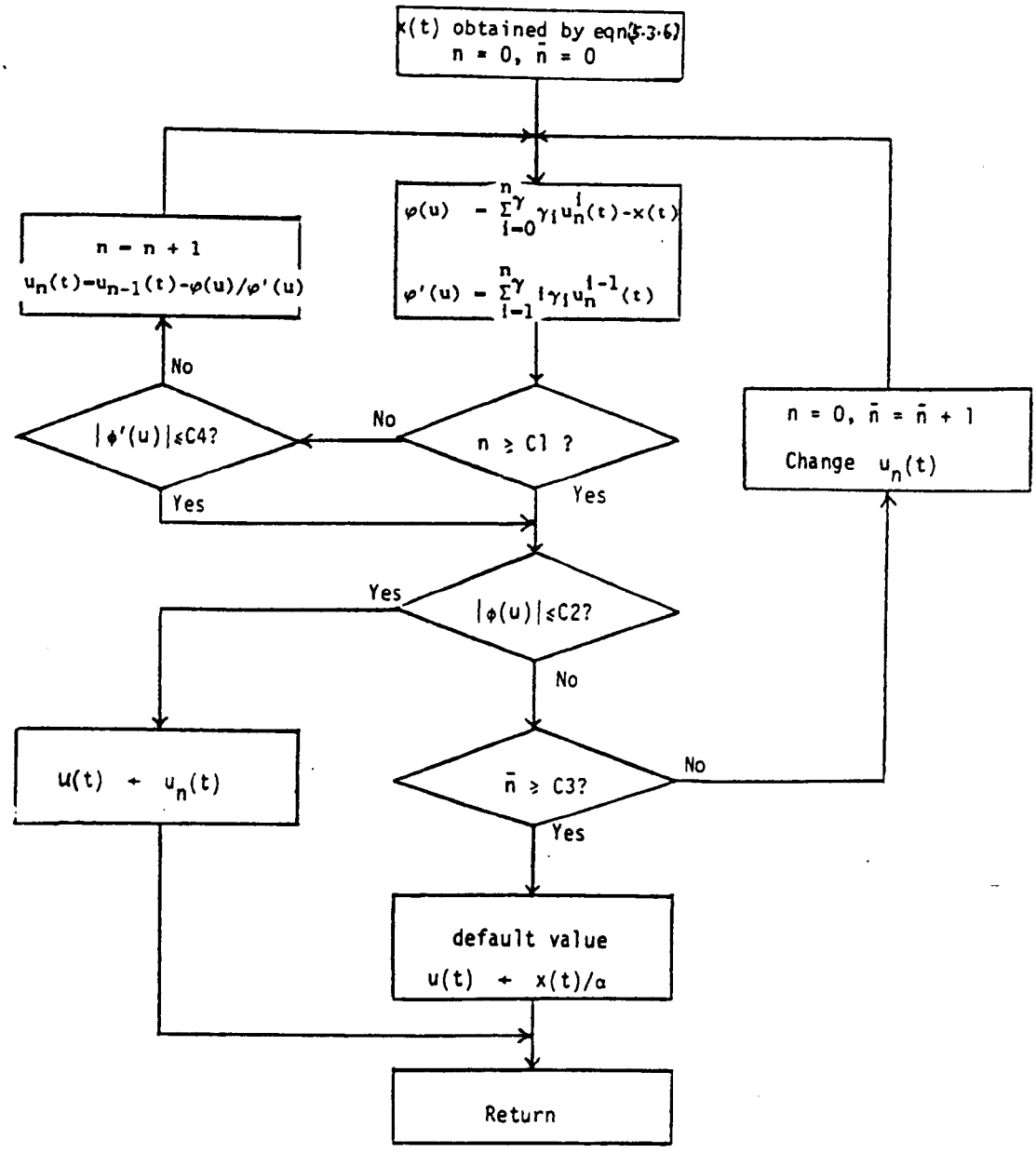


Figure 5.3.1 Root-solving computational flow diagram

5.3.3 Self-tuning implementation

The steps required to employ a self-tuning NLGPC on a plant described by a Hammerstein model are as folloes during every sampling interval:

Step 1	Sample the system output, (time instant t).
Step 2	Update the plant model parameter estimates a_i, b_i, r_i by ERLS.
Step 3	Using the estimated model coefficients a_i, b_i calculate linear intermediate signal x(t). This is a linear GPC design routine.
Step 4	Calculate the controller output u(t) from x(t) with aid of a root-solving routine.
Step 5	Apply the controller output u(t) to the plant input.
Step 6	Update the input and output vector of the plant and store other necessary data.
Step 7	Wait for the next sampling instant before returning to step 1.

5.4 Simulation experiment

In order to investigate the usefullness of the NLGPC scheme, described above, simulations have been carried out for particular systems. A comparison with NonLinear DeadBeat Controller (NLDBC) on the same plant models is included.

Two different nonlinearities are selected (Anbumani, Patnaik, and Sarma 1981), as described in eqn(5.2.2)

$$NL1:\ r_0 = 1, r_1 = 1, r_2 = -1, r_3 = 0.2 \tag{5.4.1}$$

$$NL2:\ r_0 = 0, r_1 = 1, r_2 = 0, r_3 = -1 \tag{5.4.2}$$

The nonlinear characteristics are shown in Fig. 5.4.1. For the linear dynamic part, again two different models are selected with reference to eqn(5.2.2)

$$L1:\ a_1 = -0.9, b_0 = 1, b_1 = 2 \tag{5.4.3}$$

$$L2:\ a_1 = -2.87,\ a_2 = 2.74,\ a_3 = -0.87$$

$$b_0 = 0.04,\ b_1 = 0.002,\ b_2 = -0.037 \qquad (5.4.4)$$

This means that L1 is open-loop stable and non-minimum phase, whereas L2 is open-loop unstable.

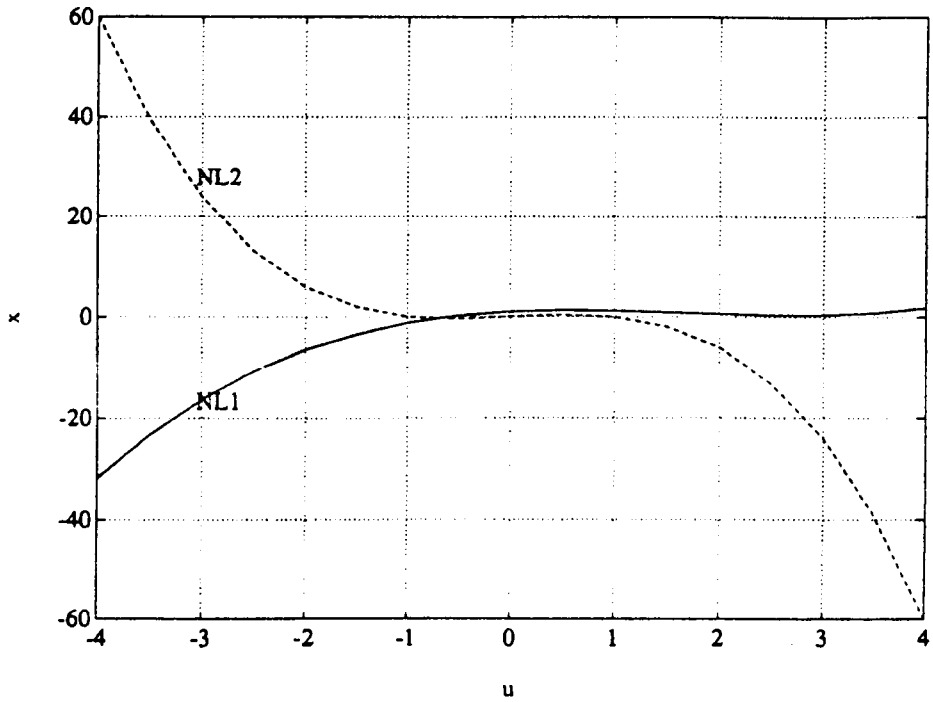


Figure 5.4.1 Static nonlinearities

An ERLS estimator is used with fixed forgetting factor 0.95, and no noise is introduced into the system, $\varepsilon(t)=0$. In order to overcome the large initial input signal deviations which occur during tuning, a relay providing unity magnitude bang-bang control is employed for the first 10 samples. It is, further, assumed within the estimator that the model structure is known.

In order to consider transient behaviour, a sequence of set-point value is assigned as follows,

Samples	set point values
1-10	20
11-30	20
31-50	60
51-70	20
71-90	0

The cycle from 11-90 samples is then repeated periodically. In each of the Figs. 5.4.2 to 5.4.5, the plots in 'a' show the control input signal u(t) as a continuous line with the intermediate variable x(t) shown by a broken line. The plots given in 'b' show the reference set-point signal as a continuous line with the actual system output signal y(t) shown by a broken line.

Fig. 5.4.2 shows the behaviour of a nonlinear deadbeat controller when operating on a system whose linear part is described by L1 and whose nonlinear part is described by NL1. The dead time (transport delay) of the linear part in each case is unity, as is defined in the original model eqn(5.2.1), further the order of the B polynomial is also unity. It can be seen from Fig. 5.4.2b that following a change in reference signal, the system output reaches the new set point value after only 2 sample periods and no overshoot occurs. These results are those which would be expected from a deadbeat controller operating on a purely linear plant (Warwick 1986) whose characteristics are represented in L1, the 2 sample periods corresponding to the sum of the dead time are the order of the B polynomial, both of which are unity.

It can also be seen that once initial tuning in of the parameter estimator has occurred, after approximately 70 sample periods, operation of the overall controller, including the root solver is satisfactory. It is assumed however that the structure of both the linear and nonlinear system part is known exactly, hence for Fig. 5.4.2, 1 a_i parameter and 2 b_i parameters are selected, along with 3 r_i parameters and r_0 being set to unity.

There is no reason to suggest why the indirect design method, described in the chapter, would not be suitable for use with controller in which the control

objective is of a type other than deadbeat, e.g. pole placement or minimum variance control. Self-tuning control is therefore applicable to a wide variety of nonlinear systems, which can be regarded as separable, in the sense of the need for an intermediate variable x(t) to be estimated. It must be pointed out though, that where noise affects the system, due to the cancelling effects of the deadbeat action, the noise will be filtered by the open-loop denominator A. this means that the system under control is required to be open-loop stable, otherwise noise enhancement may result.

Fig. 5.4.3 shows the behaviour of a nonlinear general predictive controller when operating on a stable but nonminimum phase system, whose linear part is described by L1 and whose nonlinear part is described by NL1. The predictive nature of the controller can be clearly witnessed in the plots, where advance knowledge of a reference value change has allowed the actual output signal to commence its distinct set value variation before the alteration in reference value has occurred.

Fig. 5.4.4 and Fig. 5.4.5 show the behaviour of a NLGPC when operating on an unstable plant whose linear part is described by L2 and whose nonlinear part is described by NL1 in the case of Fig. 5.4.4, and NL2 in the case of Fig. 5.4.5. Note that 400 data points are plotted in these figures, rather than 200 as is the case for Fig. 5.4.2 and Fig. 5.4.3. Again, exact structural knowledge of the plant, i.e. correct number of a_i, b_i, and r_i parameters, is assumed for parameter estimation purposes, and the control weightings $\lambda(k)$ are taken to be equal to zero.

The plots in Fig. 5.4.4b and Fig. 5.4.5b show that with the unstable open-loop plants in question, NLGPC is still able to produce a stable, satisfactory output response, although the initial tuning period is a lot longer than previously. Also, as can be seen from Fig. 5.4.4a and Fig. 5.4.5a, in order to provide a suitable controlling action, the control input signal has a high variance and rapid variations occur. A conclusion can be made that NLGPC is potentially suitable for the control of nonlinear plants. However, controller quality is much more sensitive to the choice of control weightings $\lambda(k)$. In fact, improper choice of $\lambda(k)$ may be such that a stable closed-loop system response is not achievable, even when the

controller parameters are finely tuned.

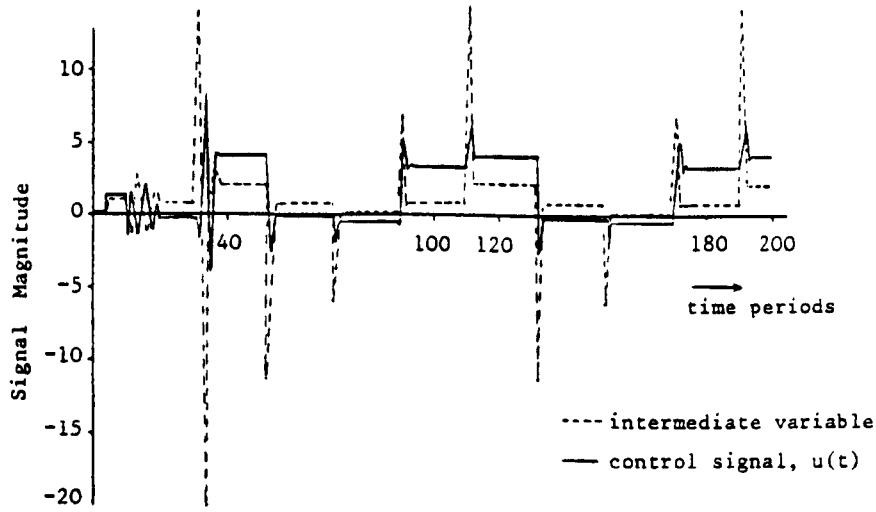


Figure 5.4.2a NLDBC operating on the plant L1+NL1

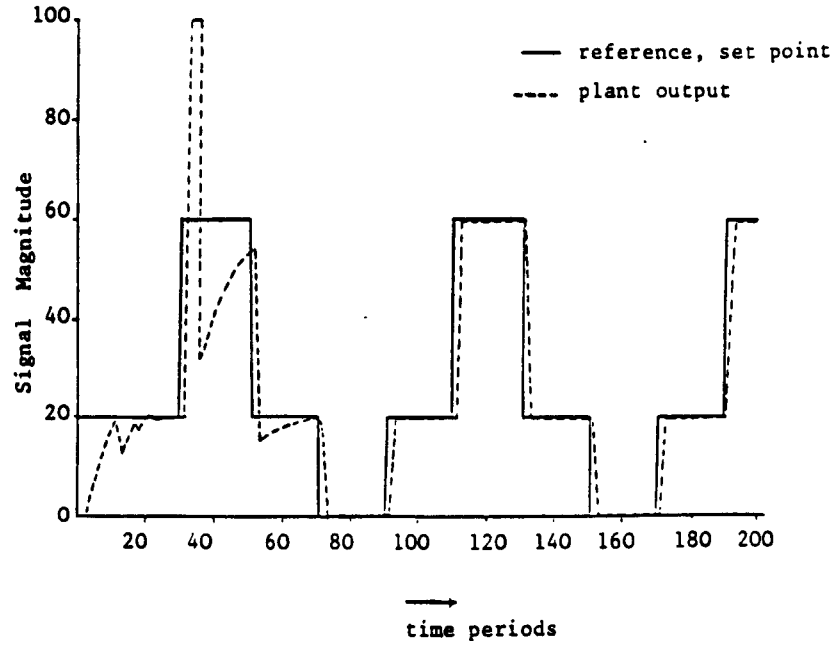


Figure 5.4.2b NLDBC operating on the plant L1+NL1

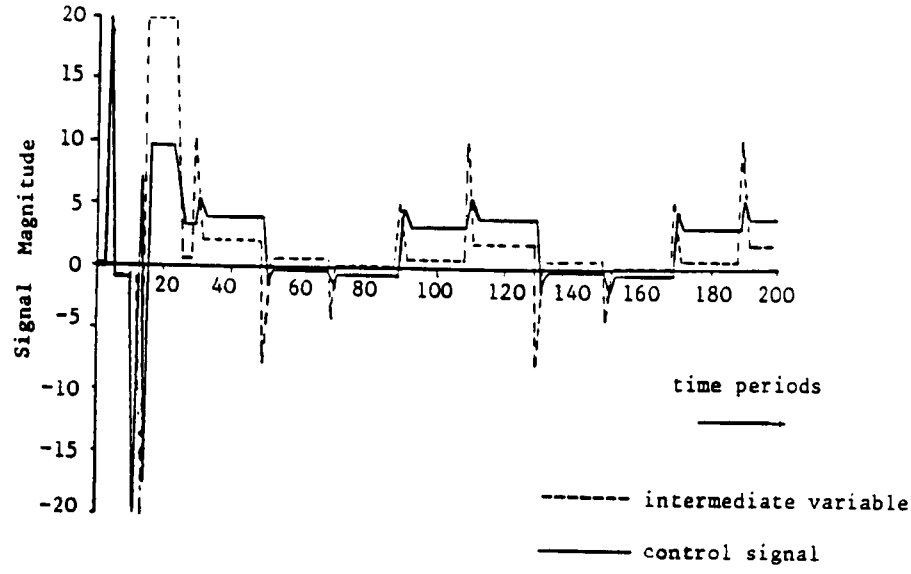


Figure 5.4.3a NLGPC operating on the plant L1+NL1

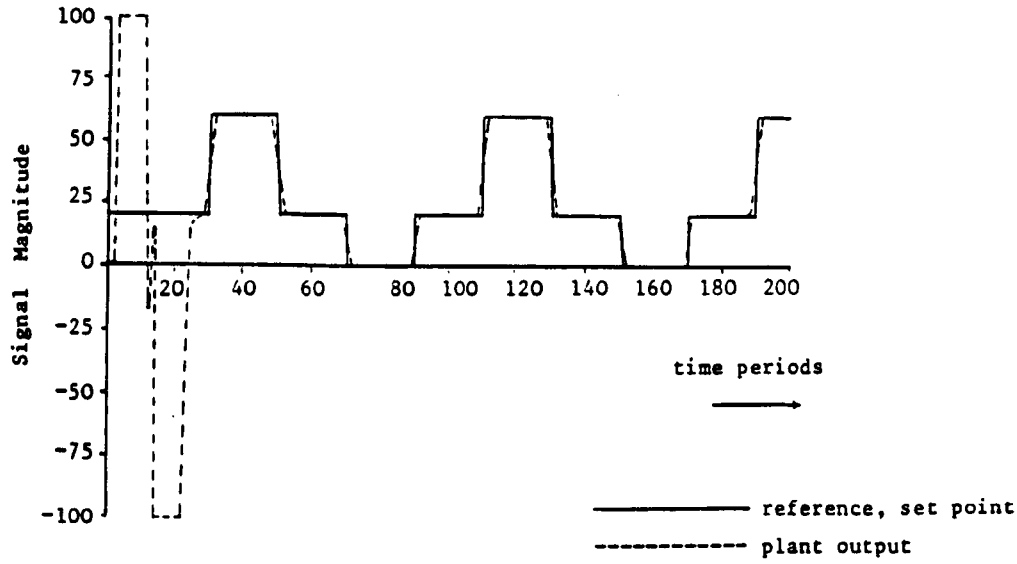


Figure 5.4.3b NLGPC operating on the plant L1+NL1

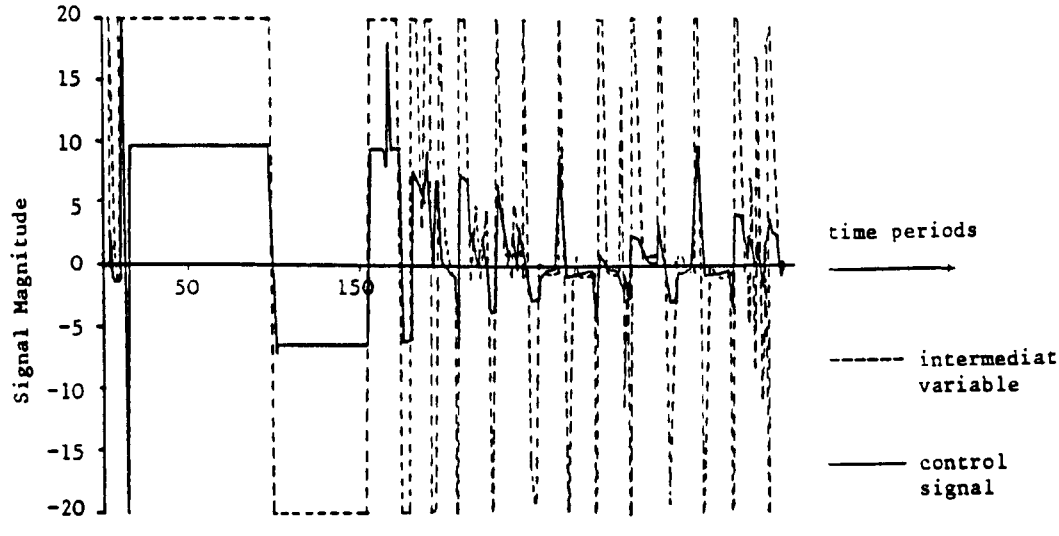


Figure 5.4.4a NLGPC operating on the plant L1+NL2

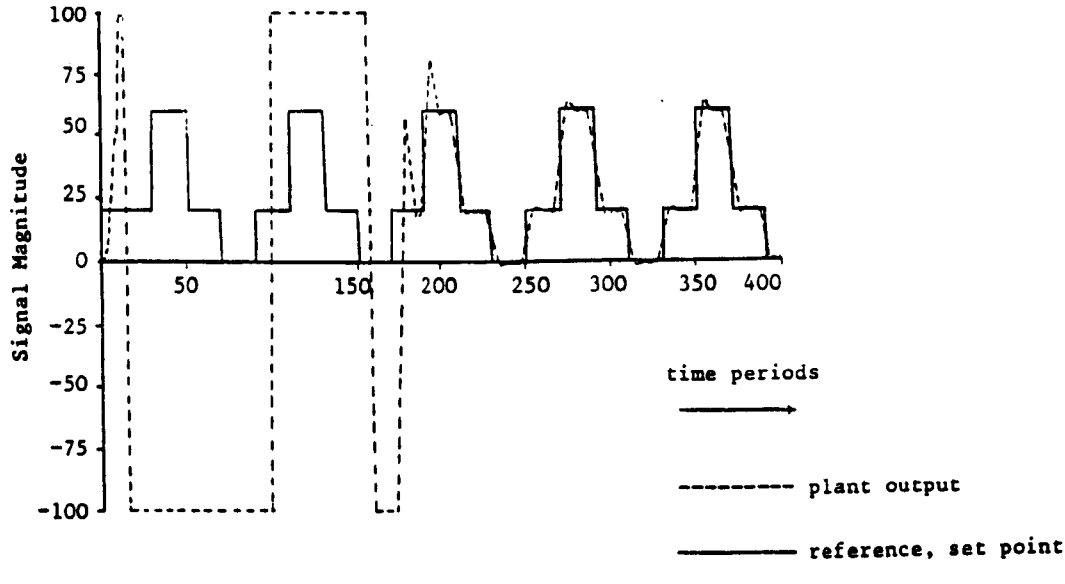


Figure 5.4.4b NLGPC operating on the plant L1+NL2

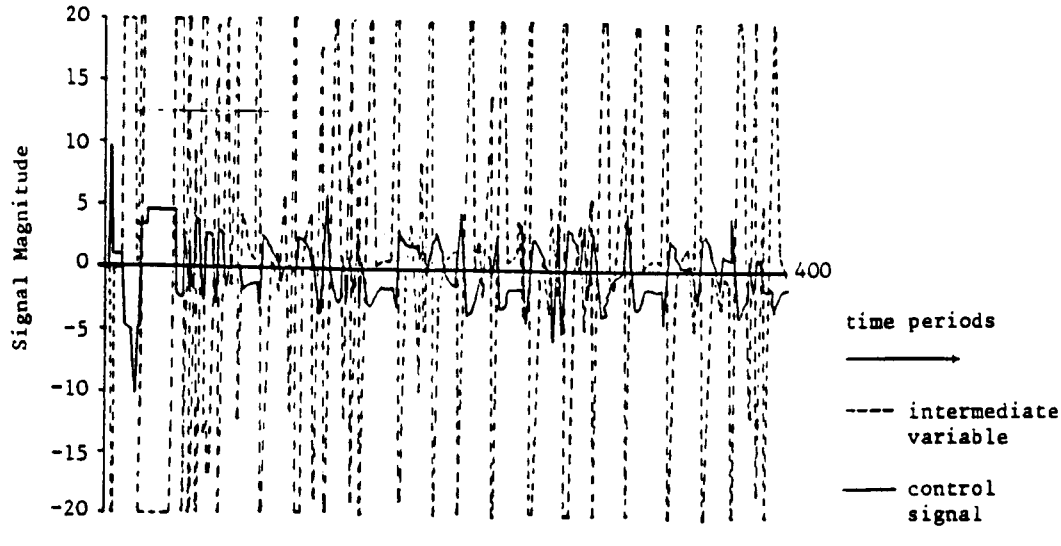


Figure 5.4.5a NLGPC operating on the plant L2+NL2

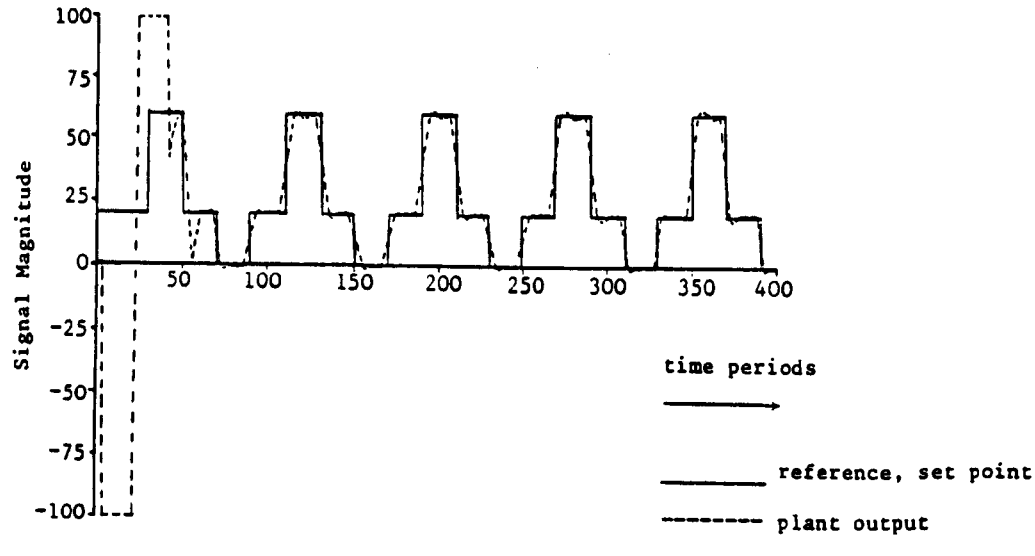


Figure 5.4.5b NLGPC operating on the plant L2+NL2

5.5 Conclusion

In the chapter an indirect self-tuning controller has been presented for the control, by means of either General Predictive Control or Deadbeat Control, of a class of nonlinear systems which can be adequately modelled by Hammerstein model. The simple Newton-Raphson root solver is in fact also applicable with other control objectives such as pole placement or minimum variance output. In fact the characteristics of the particular linear control objective, such as GPC, are retained in the nonlinear controller and hence much more attention should be paid to the nonlinear model treating technique.

The self-tuning controller described in the chapter will operate well whether the system under control is linear or nonlinear, although in common with other self-tuning control algorithms, a detailed theoretical analysis, in particular a determination of the transient behaviour, is of little value except in some very simple or specialised cases. The feasibility of NLGPC is therefore considered and compared with NLDBC, by means of several simulation studies. These indicate that not only can NLGPC be successfully applied for nonlinear system control, but also that it can be applied in a relatively simple fashion.

5.6 Appendix

5.6.1 Enhanced Recursive Least Squares estimator

An ERLS algorithm is used to estimate the parameters of the Hammerstein model given in eqn(5.2.1) and eqn(5.2.2).

$$\hat{\Theta}(t+1) = \hat{\Theta}(t) + L(t)[y(t) - \phi^T(t)\hat{\Theta}(t)]$$

$$L(t) = \frac{P(t-1)\phi(t)}{1 + \phi^T(t)P(t-1)\phi(t)}$$

$$P(t) = [P(t-1) - \frac{P(t-1)\phi(t)\phi^T(t)P(t-1)}{1 + \phi^T(t)P(t-1)\phi(t)}]/\lambda$$

where

$$\hat{\Theta} = [\beta_{00}\ \beta_{01}\ \cdots\ \beta_{nb1}\ \beta_{02}\ \cdots\ \beta_{nb2}\ \cdots\ \beta_{0nr}\ \cdots\ \beta_{nbnr}\ a_1\ a_2\ \cdots\ a_{na}]$$

$$\phi^T(t) = [1\ u(t-1)\ \cdots\ u(t-nb-1)\ u(t-1)^2\ \cdots\ u(t-nb-1)^2\ \cdots$$

$$u(t-1)^{nr}\ \cdots\ u(t-nb-1)^{nr}\ -y(t-1)\ \cdots\ -y(t-na)]$$

$$\beta_{00} = (b_0 + b_1 + \cdots + b_{nb})r_0,\quad \beta_{ij} = b_i r_j$$

Let r_1=1, it follows that

$$b_i = \beta_{i1}\quad r_j = \frac{\sum_{i=0}^{nb} b_i \beta_{ij}}{\sum_{i=0}^{nb} b_i^2}$$

λ is a forgetting factor to effect the convergence of the parameter estimates and to cope with slowing time-varying plant. Sodestrom, Ljung and Gustavsoon (1976) presented a way to choose this constant, for plants or processes with constant parameters

$$\lambda(t) = 1$$

or

$$\lambda(t+1) = \lambda_0 \lambda(t) + (1 - \lambda_0)$$

$$0.95 \le \lambda_0 = \lambda(0) \le 1$$

For plant with slowly time varying parameters

$$\lambda(t) = \lambda_0,\ 0.85 \le \lambda_0 \le 1$$

The most suitable forgetting factor depends on the plant model and the kind of disturbances. For the case of low order model and no stochastic disturbance, a smaller value λ_0 (e.g. λ_0=0.85) can be used to speed up the convergence of the parameter estimates, otherwise λ_0 must be set to 1 to guarantee the accuracy of the parameter estimates, and results in slow convergence of the estimates. Therefore the choice of the forgetting factor is a trade off between convergence speed and accuracy of the estimator.

The ERLS estimator is an unbiased for an uncorrelated disturbance sequences. For correlated disturbance sequence, an Enhanced Recursive Maximum Likelihood (ERML) estimator is available which may be derived from linear RML estimator.

5.6.2 Recursive computation of Diophantine equation (E, F) and polynomial $G_j(q^{-1})$

1 Initialisation (j=1)

$$E_1(q^{-1}) = 1$$

$$F_1(q^{-1}) = q(1 - \tilde{A})$$

$$G_1(q^{-1}) = B$$

where

$$\tilde{A} = \delta A = (1 - q^{-1})A = \alpha_0 + \alpha_1 q^{-1} + \cdots + \alpha_{n\alpha} q^{-n\alpha}$$

$$\alpha_0 = 1,\ \alpha_{n\alpha} = -a_{na},\ \alpha_i = a_i - a_{i-1},\ n\alpha = na + 1$$

2 For j=1...N-1 (N maximum prediction horizon)

$$E_{j+1}(q^{-1}) = E_j(q^{-1}) + f_{j0}q^{-j}$$

$$F_{j+1}(q^{-1}) = f_{(j+1)0} + f_{(j+1)1}q^{-1} + \cdots + f_{(j+1)i}q^{-i},\ i \in [0, na]$$

$$f_{(j+1)i} = f_{(j)i+1} - \alpha_{i+1}f_{j0}$$

$$f_{(j+1)na} = -\alpha_{n\alpha}f_{j0}$$

$$G_{j+1}(q^{-1}) = E_{j+1}(q^{-1})B = g_{(j+1)0} + g_{(j+1)1}q^{-1} + \cdots + g_{(j+1)i}q^{-i}$$

$$g_{(j+1)i} = g_{ji} + f_{j0}b_{i-j},\ i \in [0, nb+j]$$

with

$$b_{i-j} = 0,\ i < j$$

Chapter 6 Nonlinear deadbeat controller design

6.1 Introduction

Most of the NLSTC design methods, using parametric models, are indirect ones. The NLSTC design routine is, as shown in Fig. 5.1.1, to first construct a linear STC from the relationship between x and y and then to calculate the inverse function of the nonlinear part to obtain the controller output u, usually involving a root-solving routine. The internal or intermediate variable x can be estimated due to the assumption of the nonlinear plant model being separable, so that the intermediate variable is a bridge connecting STC and NLSTC. If x=f(u)=ku, k is a constant, that the STC design is derived for a linear system. By this method, Anbumani, Patnaik and Sarma (1981), Lachmann (1982), and Grimble (1985) presented several schemes for NLSTC design. On the other hand, a direct design method results from an integration of the STC design idea and different modelling technique to produce the NLSTC output u directly without the requirement for estimating the intermediate variable x. A few papers were published recently using a direct design method in some special cases, and these obtained straightforwardly the NLSTC output u instead of estimating the intermediate variable x, see Lachmann (1982), Agarwal and Seborg (1987), and Zhang and Lang (1988) for details.

A large number of control systems are designed with the objective that the response of the system should attain its desired value as quickly as possible. Such a control system is called a minimum-time control system or a time-optimal control system (Kuo 1980). One of the typical forms in digital control system is so called deadbeat controller, which combines fastest response to reference signals with easy implementation in self-tuning algorithms.

Previous work concerned with deadbeat controller design mainly concentrates on the following aspects. Deadbeat regulators, either single variable (Tou 1964, Jordan and Korn 1980, Iserman 1981) or multivariable (Kaczorek 1982, Chen, Chiang and Hsiao 1984, Beelen and Dooren 1988), are basic designs, which emphasise the ability to follow step reference signal. The deadbeat

tracking design deals with the problem of following irregular reference signals (Bradshow and Pooter 1976, Kucera 1980, Ichikawa 1989). Robust deadbeat controllers (Zhao and Kimura 1986, Warwick 1986) describe many schemes to overcome the drawbacks such as the excessive control signals to achieve the fastest regulation speed and less robustness with respect to plant variations, allowing a trade off between speed of response and robustness.

It should be noticed that all proposals mentioned above use a linear plant model. However in a real environments linear model is not always proper due to the existence of severe nonlinearities. Kaczorek (1982) mentions, without reference, that deadbeat control of bilinear and nonlinear systems has received considerable attention, with no solutions given.

The purpose of the chapter is to design a general controller for a class of nonlinear systems described by the Hammerstein model. The implementation of a deadbeat controller, as an example, is studied with computer simulation.

6.2 HCARIMA model and HCARMA model

The Hammerstein model defined in eqn(5.2.1) is written in the alternative form:

$$Ay(t) = Bx(t-1) + \varepsilon(t) \tag{6.2.1}$$

where the polynomials A and B are defined as,

$$A = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na}$$

$$B = b_0 + b_1 q^{-1} + \cdots + b_{nb} q^{-nb}$$

and

$$x(t) = r_0 + r_1 u(t) + \cdots + r_{nr} u^{-nr}(t) = \sum_{i=0}^{nr} r_i u^i(t) \tag{6.2.2}$$

where t=0, 1, 2, etc. are sampling instants, q^{-1} is the backward shift operator such that $q^{-i} y(t) = y(t-i)$, u(t) is the plant input or controller output, y(t) is the measured variable or system output, $\varepsilon(t)$ is a either correlated or uncorrelated disturbance, and x(t) is an intermediate variable which is the nonlinear static element

output or equal to the linear dynamic element input in the plant model.

Hammerstein-like CARIMA (HCARIMA) and Hammerstein-like CARMA (HCARMA) models to be presented are alternative expressions of eqn(6.2.1) with suitable modifications. Eqn(6.2.1) may be written in vector form

$$AY = BX + \varepsilon \tag{6.2.3}$$

where

$$A_{1*(na+1)} = [1\ a_1\ \cdots\ a_{na}]$$

$$Y^T{}_{1*(na+1)} = [y(t)\ y(t-1)\ \cdots\ y(t-na)]$$

$$B_{1*(nb+1)} = [b_0\ b_1\ \cdots\ b_{nb}]$$

$$X^T{}_{1*(nb+1)} = [x(t-1)\ x(t-2)\ \cdots\ x(t-nb-1)] \tag{6.2.4}$$

and from eqn(6.2.2), x(t) may be written as a function of the vector U(t) as

$$x(t) = RU(t) \tag{6.2.5}$$

where

$$R = [r_0\ r_1\ \cdots\ r_{nr}]$$

$$U^T(t) = [1\ u(t)\ \cdots\ u^{nr}(t)] \tag{6.2.6}$$

Hence

$$X = [RU(t-1)\ RU(t-2)\ \cdots\ RU(t-nb-1)] \tag{6.2.7}$$

Substituting eqn(6.2.7) into BX in eqn(6.2.3), gives

$$BX = \beta U \tag{6.2.8}$$

where

$$\beta = [\beta_{00}\ \beta_0\ \cdots\ \beta_{nb}]$$

$$U = [1\ U(t-1)\ \cdots\ U(t-nb-1)] \tag{6.2.9}$$

Furthermore eqn(6.2.9) may be expressed in the form

$$\beta_{00} = r_0 \sum_{i=0}^{nb} b_i$$

$$\beta_0 = [r_1\ r_2\ \cdots\ r_{nr}]b_0 = [\beta_{01}\ \cdots\ \beta_{0nr}]$$

$$\beta_1 = [r_1\, r_2\, \cdots\, r_{nr}]b_1 = [\beta_{11}\, \cdots\, \beta_{1nr}]$$

$$\beta_{nb} = [r_1\, r_2\, \cdots\, r_{nr}]b_{nb} = [\beta_{nb1}\, \cdots\, \beta_{nbnr}]$$

$$U^T(t-1) = [u(t-1)\, u^2(t-1)\, \cdots\, u^{nr}(t-1)]$$

$$U^T(t-2) = [u(t-2)\, u^2(t-2)\, \cdots\, u^{nr}(t-2)]$$

$$U^T(t-nb-1) = [u(t-nb-1)\, u^2(t-nb-1)\, \cdots\, u^{nr}(t-nb-1)] \quad (6.2.10)$$

The vector expression relating the system output Y, the controller output U (instead of intermediate variable X) and disturbance ε is built up through eqn(6.2.3) to eqn(6.2.10), to give

$$AY = \beta U + \varepsilon \quad (6.2.11)$$

The HCARIMA or HCARMA model may be readily developed from eqn(6.2.11), in which the corresponding polynomial form is

$$Ay(t) = \beta(z^{-1})U(t-1) + \beta_{00} + \varepsilon(t) \quad (6.2.12)$$

where A, y(t), U(t-1), β_{00} and ε have been defined above, and

$$\beta(z^{-1}) = \beta_0 + \beta_1 z^{-1} + \cdots + \beta_{nb} z^{-nb} \quad (6.2.13)$$

z^{-1} is a new defined vector backward shift operator, possessing the following properties

$$z^{-1}U^T(t) = U^T(t-1) = [u(t-1)\, u^2(t-1)\, \cdots\, u^{nr}(t-1)]$$

$$(1 - z^{-1})U^T(t) = U^T(t) - U^T(t-1)$$

$$= [u(t) - u(t-1)\;\; u^2(t) - u^2(t-1)\, \cdots\, u^{nr}(t) - u^{nr}(t-1)]$$

$$E(q^{-1})[\beta(z^{-1})U(t)] = E\,\beta(z^{-1})U(t),\;\; q^{-i}z^{-j} = z^{-(i+j)}$$

$$\beta(z^{-1})[E(q^{-1})u(t)] = \beta(z^{-1})U_E(t) \quad (6.2.14)$$

where

$$U_E^T = [E(q^{-1})u(t)\; (E(q^{-1})u(t))^2\, \cdots\, (E(q^{-1})u(t))^{nr}]$$

and $z^{-1} = q^{-1}$ for linear model.

The HCARIMA model is defined when the disturbance $\varepsilon(t) = C\xi(t)/\delta$

$$Ay(t) = \beta(z^{-1})U(t-1) + \beta_{00} + C\xi(t)/\delta \tag{6.2.15}$$

where

$$C = 1 + c_1 q^{-1} + \cdots + c_{nc} q^{-nc}$$

$$\delta = 1 - q^{-1} \tag{6.2.16}$$

and $\xi(t)$ is a uncorrelated random disturbance.

Similarly HCARMA model is defined to be

$$Ay(t) = \beta(z^{-1})U(t-1) + \beta_{00} + C\xi(t) \tag{6.2.17}$$

The model is linear in the parameters and nonlinear in input and output signals. Parameter estimates may be then handled easily by the ERLS algorithm given in chapter five.

6.3 NLDBC and its STC implementation

6.3.1 Nonlinear feedback controller based on HCARMA model

The basic design of the NLDBC originates with the design of DBC in linear systems. Consider a CARMA model

$$Ay(t) = q^{-k}Bu(t) + C\xi(t) \tag{6.3.1}$$

where $k \geq 1$ is the integer part of the transport delay.

Clarke (1982) presented a general form of feedback controller for linear plant described by CARMA model, it has

$$u(t) = \frac{G}{F}[Vw(t) - y(t)] \tag{6.3.2}$$

where

$$G = g_0 + g_1 q^{-1} + ... + g_{ng} q^{-ng}$$

$$F = 1 + f_1 q^{-1} + \cdots + f_{nf} q^{-nf}$$

$$V = v_0 + v_1 q^{-1} + \cdots + v_{nv} q^{-nv} \tag{6.3.3}$$

w(t) is a reference input or so called setpoint sequence. The polynomial V is often set equal to a constant, i.e. $V = v_0$.

The controller parameters are given by

$$G = A$$

$$F = S - q^{-k}B \tag{6.3.4}$$

The polynomial S is assigned according to various performance.

Warwick (1986) presented several typical DBC design schemes by selecting polynomial S, one of his selections was

$$V = 1,\ G = A,\ F = B(1) - q^{-k}B \tag{6.3.5}$$

the resultant controller is given by

$$u(t) = \frac{B}{B(1)}u(t-k) + \frac{A}{B(1)}[w(t) - y(t)] \tag{6.3.6}$$

and the system output response is determined by

$$y(t) = q^{-k}\frac{BV}{S}W(t) + \frac{(S - q^{-k}B)}{SA}[C\xi(t) + \beta_{00}]$$

$$= \frac{B}{B(1)}w(t-k) + \frac{(B(1) - q^{-k}B)}{B(1)A}C\xi(t) \tag{6.3.7}$$

As stated in the introduction, the method of controller design based on a linear model may be applied for nonlinear model given a suitable plant structure. Consider the HCARMA model

$$Ay(t) = z^{-k}\ \beta(z^{-1})U(t-1) + \beta_{00} + C\xi(t) \tag{6.3.8}$$

A kind of general feedback controller is proposed for nonlinear plant described by the HCARMA model in eqn(6.3.8), as follows

$$U(t) = \frac{G}{F}\ [VW(t) - y(t)] \tag{6.3.9}$$

where G, V, and W(t) are the same as for the linear system, U(t) is defined in eqn(6.2.10), and F is defined to be

$$F = f_0 + f_1 z^{-1} + \cdots + f_{nf} z^{-nf}$$

$$f_0 = [f_{01}\ f_{02}\ \cdots\ f_{0nr}] = [1\ 1\ \cdots\ 1]$$

$$f_1 = [f_{11} f_{12} \cdots f_{1nr}]$$

$$f_{bf} = [f_{nf1} f_{nf2} \cdots f_{nfnr}] \tag{6.3.10}$$

Note the new backward shift operator z^{-1} has been introduced. Expanding eqn(6.3.9) with the substitutions in eqn(6.3.11), gives

$$u(t) + u^2(t) + \cdots + u^{nr}(t) = -\sum_{i=1}^{nf} f_i U(t-i) + G[VW(t) - y(t)] \tag{6.3.11}$$

Let

$$\alpha = -\sum_{i=1}^{nf} f_i U(t-i) + G[VW(t) - y(t)] \tag{6.3.12}$$

Eqn(6.3.11) becomes

$$u(t) + u^2(t) + \cdots + u^{nr}(t) - \alpha = 0 \tag{6.3.13}$$

α is known, which is thought of an innovation variable, therefore the controller output u(t) may be determined from one of the real roots, usually the minimum amplitude one, of eqn(6.3.13). The fast recursive root-solving routine has been developed to overcome the problem of no real root and to speed up root-solving for convenience of on-line application. Based on the direct controller in eqn(6.3.9), typical NLDBC designs are readily implemented.

6.3.2 Nonlinear deadbeat controller

An example is selected to show the design procedure of the nonlinear deadbeat controller. Let

$$V = 1, \quad G = A$$

$$F = S - z^{-k}\beta(z^{-1}) = \beta(1) - z^{-k}\beta(z^{-1}) \tag{6.3.14}$$

the NLDBC output u(t) is then obtained from

$$\alpha_1 u(t) + \alpha_2 u^2(t) + \alpha_{nr} u^{nr}(t) - \alpha_0 = 0 \tag{6.3.15}$$

where

$$\alpha_0 = \sum_{j=0}^{nb} \beta_j U(t-k-j) + \sum_{j=0}^{na} a_j [w(t-j) - y(t-j)]$$

$$\alpha_1 = \sum_{j=0}^{nk} \beta_{j1}$$

$$\alpha_2 = \sum_{j=0}^{nk} \beta_{j2}$$

$$\alpha_{nr} = \sum_{j=0}^{nk} \beta_{jnr} \qquad (6.3.16)$$

Consequently, the system output response is obtained from

$$\alpha_1 y(t) + \alpha_2 y^2(t) + \alpha_{nr} y^{nr}(t) - \alpha_0 = \sum_{j=0}^{nk} \beta_j W(t-k-j) - \sum_{j=0}^{nk} \beta_j E(t-k-j) + \beta(1)E(t) \qquad (6.3.17)$$

where

$$W^T(t) = [w(t)\ w^2(t)\ \cdots\ w^{nr}(t)]$$

$$E^T(t) = [\frac{C}{A}\xi(t)\ \frac{C}{A}\xi(t)^2\ \cdots\ \frac{C}{A}\xi(t)^{nr}] \qquad (6.3.18)$$

6.3.3 NLDBC STC implementation

The steps required to employ a self-tuning NLDBC on a plant described by HCARMA model are summarised as follows

At each sampling instant t,

Step 1 Sample the system output.

Step 2 Update the plant model parameter estimates by ERLS or ERML.

Step 3 Calculate the controller output from eqn(6.3.10) with aid of a root-solving routine.

Step 4 Apply the controller output to the plant input.

Step 5 Update the input and output vector of the plant.

Step 6 Wait for the next sampling instant before returning to step 1.

6.4 Simulation results

The simulation is directed towards studying the direct designed NLDBC, and features of its behaviour such as controller feasibility, reference input tracking, suppression of disturbances. The efficiency of the root-solving routine presented in chapter five is also considered.

Two different polynomials are chosen for the static nonlinearities. These are same as those in chapter five, namely

$$NL1:\ x(t) = 1 + u(t) - u^2(t) + 0.2u^3(t)$$

that is

$$NL1:\ r_0 = 1, r_1 = 1, r_2 = -1, r_3 = 0.2 \tag{6.4.1}$$

and

$$NL2:\ x(t) = u(t) - u^3(t)$$

that is

$$NL2:\ r_0 = 0, r_1 = 1, r_2 = 0, r_3 = -1 \tag{6.4.2}$$

the nonlinear characteristics are shown as in Fig. 5.4.1.

There different linear dynamic systems L1, L2, and L3 are chosen, L1 (Clarke, Mohtadi, and Tuffs 1987) is a non-minimum phase plant, L2 is a first order stable plant, L3 (Kurz, Isermann, and Schumann 1980) is a high order plant with low pass behaviour and one zero outside of the unit circle of the z-plane.

$$L1:\ A = 1 - 0.9q^{-1},\ B = 1 + 2q^{-1}$$

$$L2:\ A = 1 - 0.9q^{-1},\ B = 1 - 0.5q^{-1}$$

$$L3:\ A = 1 - 1.7063q^{-1} + 0.958q^{-2} - 0.1767q^{-3}$$

$$B = 1.86 + 4.86q^{-1} + 0.78q^{-2} \tag{6.4.3}$$

In order to overcome the large initial input signal deviations which occur during tuning, a relay providing unity magnitude bang-bang control is employed for the first 10 samples. It is, further, assumed within the estimator that the model structure is known. An ERLS estimator is used with fixed forgetting factor 0.9 to

speed up parameter estimates, and is initialized with parameters (1,0...0).

In order to consider transient behaviour, a sequence of set-point value is assigned as follows,

Samples	set point values
1-10	20
11-30	20
31-50	60
51-70	20
71-90	0

The cycle from 11-90 samples is then repeated periodically and 200 samples in total are taken for every experiment. In each of Fig. 6.4.1 to Fig. 6.4.6, the plots in 'a' show the set-point signal w(t) in a continuous line with the actual system output signal y(t) shown by a broken line. The plots in 'b' show the control input signal u(t) as a continuous line with the innovation variable $\alpha(t)$ shown as a broken line.

The plots show that the NLDBC developed has the same properties as the linear DBC. The efficiency of the root solver is demonstrated again.

The overshoots in Fig. 6.4.3(a) and Fig. 6.4.4(a) come from the fact $|\beta(1)| < |\beta_0|$ in nonlinear system similar to $B(1) < b_0$ in linear system. This can be understood from eqn(6.3.7) by letting $\xi(t) = 0$, $B = b_0 + b_1 q^{-1}$, and considering the linear system

$$y(t) = \frac{B}{B(1)} w(t-k) \qquad (6.4.4)$$

Before the set-point variation, we have w(t-k-1) = w(t-k-2)...= 0. There is a step input at time t-k, and the corresponding system output y(t) is determined by

$$y(t) = [b_0 w(t-k) + b_1 w(t-k-1)]/B(1)$$

$$= \frac{b_0}{b_0 + b_1} w(t-k) \qquad (6.4.5)$$

If $b_0 > b_0 + b_1$, then y(t) > w(t-k), which implies the presence of an overshoot.

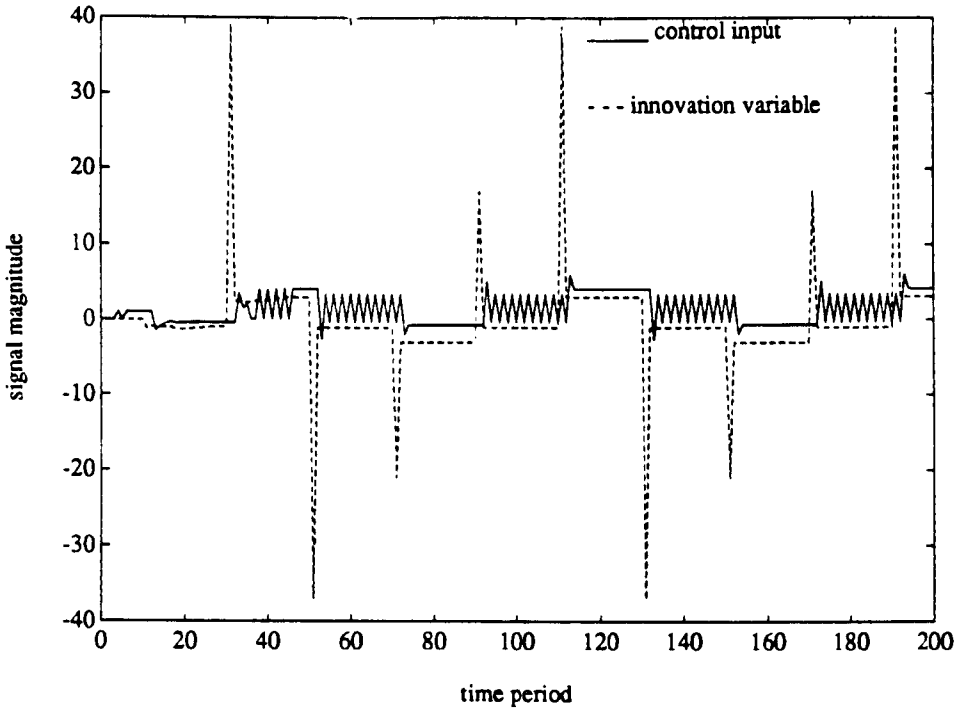


Figure 6.4.1a NLDBC operating on the plant L1+NL1

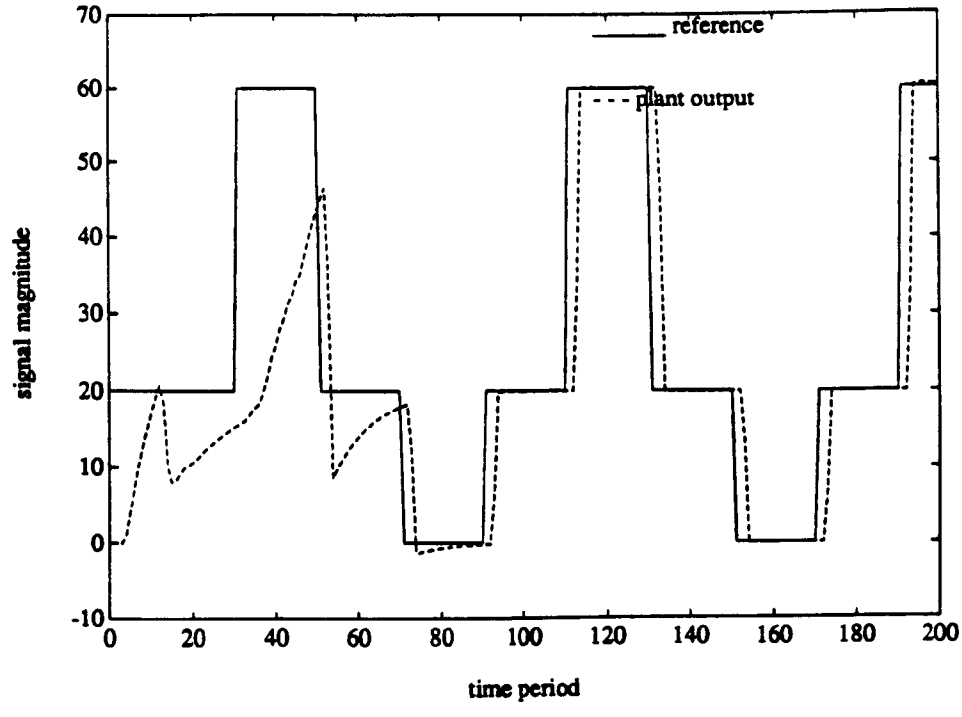


Figure 6.4.1b NLDBC operating on the plant L1+NL1

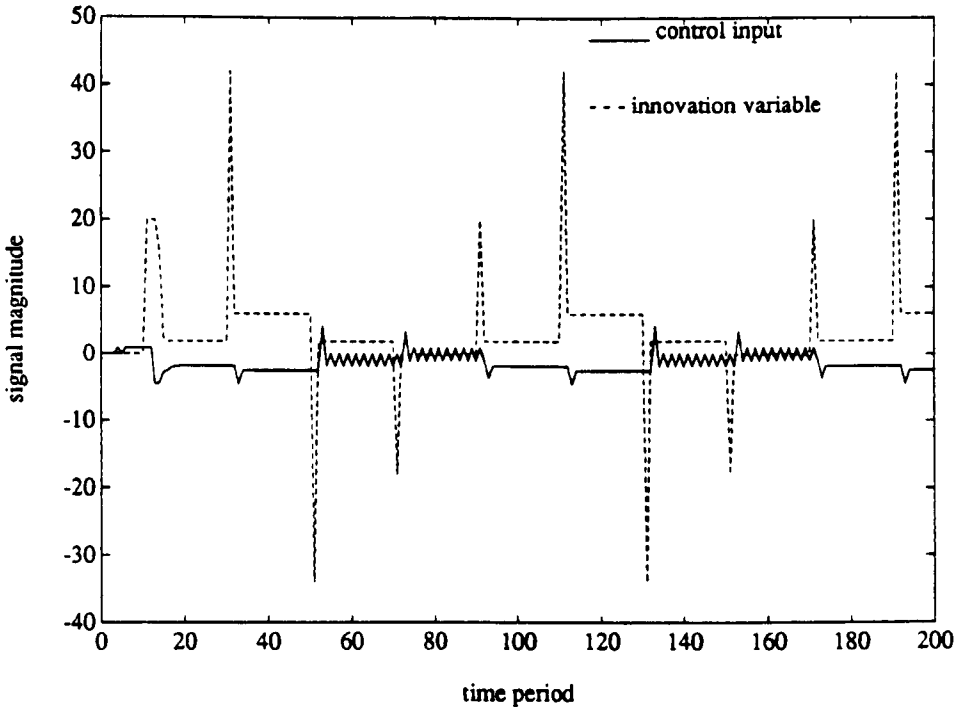


Figure 6.4.2a NLDBC operating on the plant L2+NL2

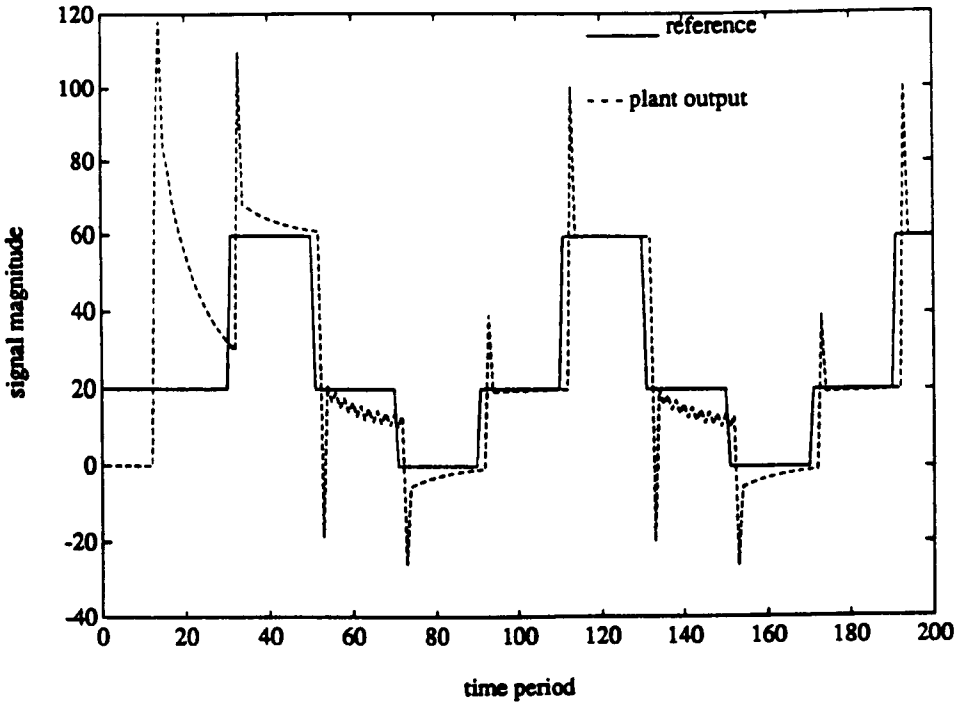


Figure 6.4.2b NLDBC operating on the plant L2+NL2

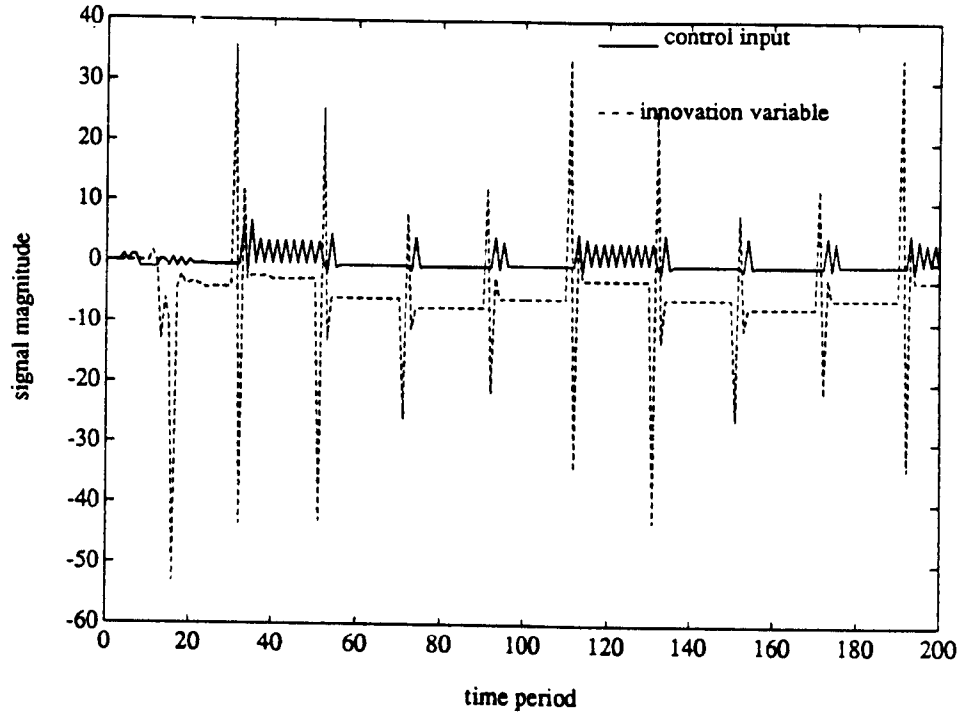


Figure 6.4.3a NLDBC operating on the plant L3+NL1

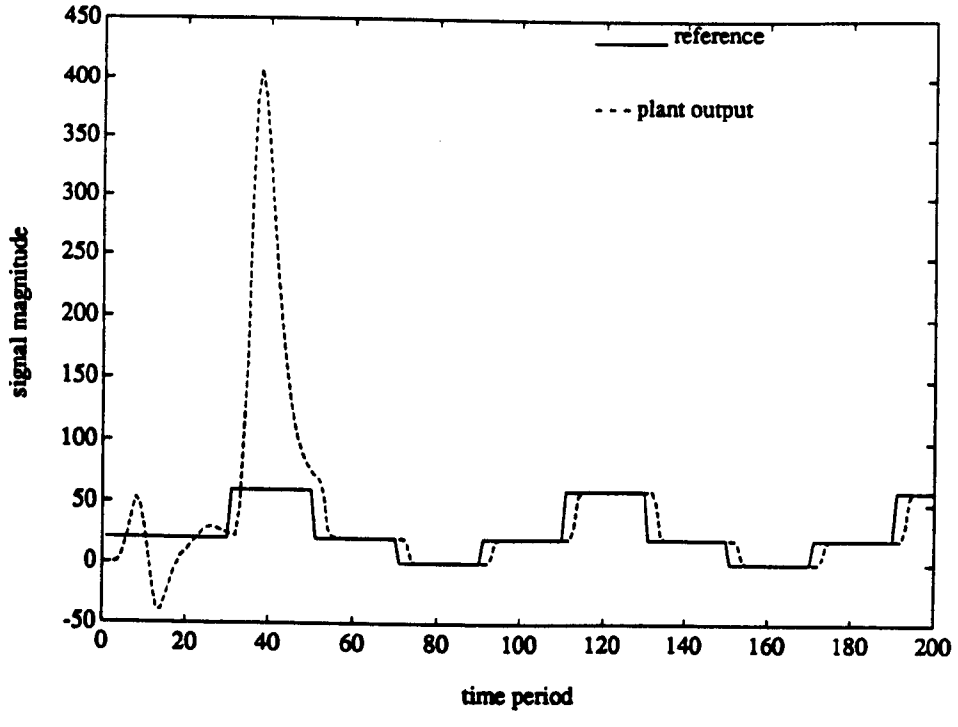


Figure 6.4.3b NLDBC operating on the plant L3+NL1

6.5 Conclusions

The introduction of a vector backward shift operator makes the HCARIMA and HCARMA models resemble their corresponding linear ones. The nonlinear controller design closely follows the linear controller design strategy. The new controller design scheme generalises the linear controller design developed by Clarke (1982).

The second advantage of the models is that the requirement for intermediate variable estimation can be removed. However it should be noticed that a root solver is still required whether indirect or direct design method is used. This is a characteristic in nonlinear system controller design. The NLDBC was selected as an example to show the model validity and efficiency.

Based on the developed models, some typical self-tuning controllers, such as minimum variance, pole placement, and alternative forms, may be realized easily. This will be reported in the near future. For some complicated self-tuning controllers like general predictive controller and general pole-placement controller, the model is still suitable. However the final controller output calculation involves the solution of a set of highly complicated nonlinear equations.

6.6 Appendix

In order to explain the procedure of NLDBC design with the direct method, a simple NLDBC design by hand is selected by considering following model

$$Ay(t) = Bx(t-1) + \xi(t)$$

$$A = 1 - 0.9\,q^{-1}$$

$$B = 1 + 2q^{-1}$$

$$x(t) = 1 + u(t) - u^2 + 0.2u^3(t) \qquad \text{(A6.1)}$$

the corresponding HCARMA model is

$$Ay(t) = \beta(z^{-1})U(t-1) + \beta_{00} + \xi(t) \qquad \text{(A6.2)}$$

where

$$\beta(z^{-1}) = \beta_0 + \beta_1 z^{-1}$$

$$\beta_0 = [r_1\ r_2\ r_3]b_0 = [1\ -1\ 0.2]$$

$$\beta_1 = [r_1\ r_2\ r_3]b_1 = [2\ -2\ 0.4]$$

$$\beta_{00} = r_0 \sum_{i=0}^{1} b_i = 1.2$$

$$U^T(t-1) = [u(t-1)\ u^2(t-1)\ u^3(t-1)]$$

$$U^T(t-2) = [u(t-2)\ u^2(t-1)\ u^3(t-2)] \qquad \text{(A6.3)}$$

According to eqn(6.3.9), design the NLDBC by letting V=1, G=1, and $F = \beta(0) - z^{-1}\beta(z^{-1})$

$$\alpha_1 u(t) + \alpha_2 u^2(t) + \alpha_3 u^3(t) - \alpha_0 = 0 \qquad \text{(A6.4)}$$

where

$$\alpha_1 = b_0 r_1 + b_1 r_1 = 3$$

$$\alpha_2 = b_0 r_2 + b_1 r_2 = -3$$

$$\alpha_3 = b_0 r_3 + b_1 r_3 = 0.6$$

$$\alpha_0 = \sum_{j=0}^{1} \beta_j U(t-1-j) + \sum_{j=0}^{1} a_j [w(t-j) - y(t-j)]$$

$$= [u(t-1) - u^2(t-1) + 0.2u^3(t-1)]$$

$$+ [2u(t-2) - 2u^2(t-2) + 0.4u^3(t-2)]$$

$$+ [W(t) - y(t)] - 0.9[W(t-1) - y(t-1)] \qquad \text{(A6.5)}$$

Accordingly

$$3u(t) - 3u^2(t) + 0.6u^3(t) - \alpha_0 = 0 \qquad \text{(A6.6)}$$

u(t) may be calculated by root-solving routine from eqn(A6.6) and will change with the innovation variable α_0.

Section 4 Conclusions and bibliography

S4.1 Overall conclusions

This thesis has studied some problems in signal processing and nonlinear system identification and control, leading to novel contributions in several areas including: the effect of missing data position to spectrum analysis and frequency response characteristic estimation; the exploration, using a geometric method, of nonlinear system structure detection and parameter estimation; the importance of signal amplitude selection in nonlinear system identification and the consideration of nonlinear system self-tuning controller design with emphasis on the nonlinear systems described by the Hammerstein model.

In summary:

In section one, a relative simple method has been presented for handling data records with missing points, useful for the estimation of the Fourier transform, power spectrum density and frequency response functions. Unlike previous techniques, the new method demonstrates the importance of the position of missing points in the estimation process.

The effect of the periodic occurrence of missing points, leading to a periodic error spectrum, has been predicted and observed in experiments.

Areas for further research include study of the effect of various distributions of missing data position in signal analysis, investigation of parametric model estimation from incomplete time series, and estimation of missing values.

In section two, the consideration of signal amplitude in nonlinear system identification and parameter estimation has led to two novel results. One is the technique of signal amplitude quantisation to detect the system structure and to estimate the system parameters. Another is the VWLS algorithm, chosing weightings by state amplitude distance. The 3D space has displayed rich information about system characteristics.

Areas for further research include automatic determination of the amplitude range of validity of a linear model for the given system, and application of the

quantisation technique for identifying various nonlinear systems.

In section three, the studies, both theoretical and experimental, of nonlinear STC controller design indicate that the attention should be paid to the technique in model treatment, that is to give a suitable description form based on original model, in order to facilitate the implementation of control schemes.

Areas for further research include a study of the validity of designing other type of controllers with the HCARMA model, to borrow the idea developed in section two to investigate the possibility of simplifying nonlinear controller design.

The computer software package Matlab significantly simplifies program design.

As stated at the beginning of the thesis, this research work emphasizes the development of new concepts and exploration of potential applications, therefore much more work must be carried on in the future in order to make it complete.

BIBLIOGRAPHY

1. Agarwal, M. and D.E. Seborg, "Self-tuning controller for nonlinear systems," Automatica, vol. 23, pp. 209-214, 1987.

2. Anbumani, K., L.M. Patnaik, and I.G. Sarma, "Self-tuning minimum variance control of nonlinear systems of the Hammerstein model," IEEE Trans. Aut. Control, vol. AC-26, pp. 959-961, 1981.

3. Astrom, K. J., "Introduction to stochastic control theory," Mathematics in science and enginnering, vol. 70, Academic Press, New York, 1970.

4. Astrom, K.J. and P. Eykhoff, "System identification -- a survey," Automatica, vol. 7, pp. 123-133, 1971.

5. Astrom, K.J. and B. Wittenmark, "On self-tuning regulators," Automatica, vol. 9, pp. 185-199, 1973.

6. Astrom, K.J. and B. Wittenmark, "Self-tuning controllers based on pole-zero placement," Proc. IEE, vol. 127, pp. 120-130, 1980.

7. Astrom, K.J., "Theory and applications of adaptive control-- A servey," Automatica, vol. 19, pp. 471-486, 1983.

8. Atherton, D.P., Nonlinear control engineering, Van Nostrand Reinhold, 1975.

9. Bard, A., Nonlinear parameter estimation, Academic Press , New York and London, 1974.

10. Beelen, Th. and P. V. Dooren, "A numerical method for deadbeat control of generalized state-space systems," System & Control Letters, vol. 10, pp. 225-234, 1988.

11. Bellman, R., Dynamic programming, Princeton University Press, 1957.

12. Bellman, R., Adaptive process -- a guide tour, Princeton University Press, 1961.

13. Billings, B.S. and S.Y. Fakhouris, "Theory of separable processes with applications to the identification of nonlinear systems," Proc. IEE, vol. 125, pp. 1051-1058, 1978.

14. Billings, S.A. and S.Y. Fakhouri, "Nonlinear system identification using the Hammerstien model," Int. J. Systems Sci., vol. 10, pp. 567-578, 1979.

15. Billings, S.A., "Identification of nonlinear systems -- a survey," IEE Proc., vol. 127, pp. 272-285, 1980.

16. Billings, S.A., M.B. Fadzil, J.L. Sulley, and P.M. Johnson, "Identification of a nonlinear difference equation model of an idustrial diesel generator," Mechanical Systems and Signal Processing, vol. 2(1), pp. 59-76, 1988.

17. Bradshow, A. and B. Pooter, "Nilpotency properties of linear multivariable discrete-time tracking systems," Int. J. Control, vol. 24, pp. 573-583, 1976.

18. Chen, B. S., C. C. Chiang, and F. H. Hsiao, "Dead-beat controller synthesis: multivariable case," Int. J. Control, vol. 40, pp. 1207-1213, 1984.

19. Chen, S. and S.A. Billings, "Prediction error estimation algorithm for nonlinear output affine systems," Int. J. Control, vol. 47, pp. 309-332, 1988.

20. Clarke, D.W. and P.J. Gawthrop, "Self-tuning controller," Proc. IEE, vol. 122, pp. 929-934, 1975.

21. Clarke, D.W. and P.J. Gawthrop, "Self-tuning control," Proc.IEE, vol. 126, pp. 633-640, 1979.

22. Clarke, D.W., "Model following and pole-placement self-tuners," Optimal control applications and methods, vol. 3, pp. 323-335, 1982.

23. Clarke, D.W., C. Mohtadi, and P.S. Tuffs, "General predictive control Part 1 and Part 2," Automatica, vol. 23, pp. 137-148 and 149-160, 1987.

24. Clarke, D.W. and C. Mohtadi, "Properties of general predictive control," Report OUEL 1721, Dept. of Engineering Science , Univ. of Oxford, 1988.

25. Cook, P., Nonlinear dynamical systems, Prentice-Hall Int., 1986.

26. Douce, J. L., An introduction to the mathematics of servomechanisms, E. U. P., 1963.

27. Douce, J. L., "Identification of a class of non-linear systems," 4th IFAC Symp. Ident. & Syst.

Par. Est., pp. 1-16, Tbilisi, 1976.

28. Douce, J.L. and Q.M. Zhu, "Spectral analysis of records with missing data," 8th IFAC Symp. Ident. & Syst. Par. Est., vol. 3, pp. 1359-1363, Beijing, 1988.

29. Douce, J. L. and Q. M. Zhu, "Spectrum estimation and system identification from incomplete data," Internationl Conference on System Science, Wroclaw, Poland, 1989.

30. Falkner, A.H., "Iterative technique in the identification of a nonlinear system," Int. J. Control, vol. 48, pp. 385-396, 1988.

31. Fnaiech, F. and L. Ljung, "Recursive identification of bilinear systems," Int. J. Control, vol. 45, pp. 453-470, 1987.

32. Gallmann, P.G., "A comparison of two Hammerstein model identification algorithms," IEEE Trans. Aut. Control, vol. AC-21, pp. 124-126, 1976.

33. Gerald, C. F., Applied numerical analysis, Addison-Wesley, 1978.

34. Godfrey, K. and P. Jones, Signal processing for control, Spring-Verlag , Berlin, 1986.

35. Goodwin, G. C. and R. L. Payne, Dynamic system identification: experiment design and data analysis, Academic Press, New York, 1977.

36. Goring, B. and H. Unbehauen, "Application of different statistical tests for the determinationof the most accurate order of the model in parameter estimation," 3rd IFAC Symp. Ident. & Syst. Par. Est., pp. 917-928, 1973.

37. Grimble, M.J., "Observations-weighted minimum-variance control of linear and nonlinear systems," Int. J. Systems Sci., vol. 16, pp. 1481-1492, 1985.

38. Gupta, M.M(Editor)., Adaptive methods for control system design, IEEE Press, 1986.

39. Harris, R.W., "Spectra from data with missing values," Mechnical Systems and Signal Processing, vol. 1(1), pp. 94-104, 1987.

40. Ichikawa, K., "Deadbeat characteristic of one-step ahead control," Int. J. Control, vol. 49, pp. 25-

31, 1989.

41. Iserman, R., Digital control systems, Sprige-Verlag, Berlin, 1981.

42. Isermann, R., "Parameter adaptive control algorithms-- A tutorial," Automatica, vol. 18, pp. 513-528, 1982.

43. Jarrett, R. G., "The analysis of designed experiments with missing observations," Appl. Statist., vol. 27, pp. 38-46, 1978.

44. John, J. A. and P. Prescott, "Estimating missing values in experiments," Appl. Statist., vol. 24, pp. 190-192, 1975.

45. Jones, R.H., "Spectral analysis with regularly missed observations," Ann. Math. Stat., vol. 32, pp. 455-461, 1962.

46. Jordan, D. and J. Korn, "Deadbeat algorithms for multivariable process control," IEEE Trans., vol. AC25, pp. 486-491, 1980.

47. Kaczorek, T., "Deadbaet control in multivariable systems," Int. J. Control, vol. 35, pp. 653-663, 1982.

48. Kalman, R.E., "Design of a self-optimizing control system," Trans. ASME, vol. 80, pp. 468-478, 1958.

49. Kortmann, M. and H. Unbehauen, "Identification methods for nonlinear MISO systems," 10th IFAC World Congress, pp. 225-230, Munich, 1987.

50. Kou, B. C., Digital control systems, Holt, Rinehart and winston, New York, 1980.

51. Kucera, V., "A deadbeat servo problem," Int. J. Control, vol. 32, pp. 107-113, 1980.

52. Kurz, H., R. Isermann, and R. Schumann, "Experimental comparasion and application of various parameter-adaptive control algorithms," Automatica, vol. 16, pp. 117-133, 1980.

53. Lachmann, K.H., "Parameter adaptive control of a class of nonlinear processes," 6th IFAC Symp. Ident. Syst. Param. Est., pp. 372-378, Alington, 1982.

54. Lelic, M.A. and M.B. Zarrop, "General pole-placement self-tuning controller , Part 1 Basic

algorithm," Int. J. Control, vol. 46, pp. 547-568, 1987a.

55. Lelic, M.A. and P.E. Wellstead, "General pole-placement self-tuning controller , Part 2 Application to robot manipulator control," Int. J. Control, vol. 46, pp. 569-601, 1987b.

56. Leontaritis, I.J. and S.A. Billings, "Input-output parametric models for nonlinear systems," Int. J. Control, vol. 41, pp. Part 1 303-328 , Part 2 329-344, 1985.

57. Ljung, L., System identification--Theory for the user, Prentice Hall Englewood Cliffs , New Jersey, 1987.

58. McGiffin, P.B. and D.N.P. Murthy, "Parameter estimation for auto-regressive systems with missing observations," Int. J. Systems Sci., vol. 11, pp. 1021-1034, 1980.

59. McGiffin, P.B. and D.N.P. Murthy, "Parameter estimation for auto-regressive systems with missing observations - Part 2," Int. J. Systems Sci., vol. 12, pp. 657-663, 1981.

60. Middleton, R.H., G.C. Goodwin, D.J. Hill, and D.Q. Mayne, "Design issues in adaptive control," IEEE Trans. Aut. Control, vol. 33, pp. 50-58, 1988.

61. Moler, C., J. Littel, and S. Bangert, PRO-MATLAB for Sun Workstations, The MathWorks, Inc., 1987.

62. Nahi, N., "Optimal recursive estimation with uncertain observation," IEEE Trans. Inf. Theory, vol. IT-15, pp. 457-462, 1969.

63. Narendra, K.S. and P.G. Gallmann, "An iterative method for the identification of nonlinear systems using a Hammerstein model," IEEE Trans. , Aut. Control, vol. AC-11, pp. 546-550, 1966.

64. Norton, J. P., An introduction to identification, Academic Press New York, 1986.

65. Owens, D. H. and K. Warwick, "Extended predictive control," Proc. IEE, International Conference, Control 88,, pp. 622-627, Oxford, 1988.

66. Parzen, E., "On spectral analysis with missing obsservations and amplitude modulation," Indian Journal of Statistics, vol. A25, pp. 383-392, 1963.

67. Peterka, V., "Predictor-based self-tuning control," Automatica, vol. 20, pp. 39-50, 1984.

68. Roberts, J.B. and M. Gaster, "On the estimation of spectra of randomly sampled signals : A method reducing variability," Proc. R. Sco. Lond., vol. A371, pp. 235-258, 1980.

69. Robinson, P. M., "Review of various approaches to power spectrum estimation," Time series in the frequency domain, edited by D. R. Brillinger and P. R. Krishnaiah, pp. 343-368, North-Holland, 1983.

70. Salgado, M. E., G. C. Goodwin, and R. H. Middleton, "Modified least squares algorithm incorporating exponential resetting and forgetting," Int. J. Control, vol. 47, pp. 477-491, 1988.

71. Schurmann, B. and H. Rake, "Frequency response estimation based on randomly sampled signals," Preprints of 10th IFAC World Congress, pp. 254-259, 1987.

72. Seborg, D. E., T. F. Edgar, and S. L. Shah, "Adaptive control stratedgies for process control: A survey," AIChE, vol. 32, pp. 881-913, 1986.

73. Smith, P. L., "The use of analysis of covariance to analyse data from designed experiments with missing or mixed-up values," Appl. Statist., vol. 30, pp. 1-8, 1981.

74. Soderstrom, T., L. Ljung, and I. Gustavsson, "A comparative study of recursive identification methods," Report 7427, Dept. of Automatic control , Lund Institute of Technology, 1976.

75. Svoronos, S., G. Stephanopoulos, and R. Aris, "On bilinear estimation and control," Int. J. Control, vol. 34, pp. 651-684, 1981.

76. Tou, J. T., Modern control theory, McGraw-hill, New York, 1964.

77. Tsypkin, Y.Z., Adaptation and learning in automatic systems, Academic Press , New York, 1971.

78. Tsypkin, Y.Z., Foundations of the theory of learning systems, Academic Press , New York, 1973.

79. Tuffs, P. S. and D. W. Clarke, "Self-tuning control of offset: a unified approach," Proc. IEE, vol. 132, pp. 100-110, 1985.

80. Volterra, V., Theory of functionals, Blackie , Glasgow, 1930.

81. Warwick, K., "Adaptive deadbeat control of stochastic systems," Int. J. Control, vol. 44, pp. 651-663, 1986.

82. Warwick, K. and D. W. Clarke, "Weighted input predictive controller," Proc. IEE, vol. 135, pp. 16-20, 1988.

83. Warwick, K., Q. M. Zhu, and J. L. Douce, "An adaptive controller for nonlinear systems," IFAC Symposium on Adaptive Control & Signal Processing, Glasgow, UK,, 1989.

84. Wellstead, P.E., D. Prager, and P. Zanker, "Pole assignment self-tuning regulator," Proc. IEE, vol. 126, pp. 781-790, 1979.

85. West, J. C., "Nonlinear signal distortion correlation," Int. J. Control, vol. 2, pp. 529-538, 1965.

86. Wiener, N., Nonlinear problems in random theory, Wiley , New York, 1958.

87. Zhang, J. and S. Lang, "Indirect adaptive suboptimal control for liner dynamic systems having polynomial nonlinearities," IEEE Trans. Aut. Control, vol. 33, pp. 389-392, 1988.

88. Zhao, Y. and H. Kimura, "Deadbeat control with robustness," Int. J. Control, vol. 43, pp. 1427-1440, 1986.

Section 5 Appendix

This appendix includes the following two parts:

1. Papers published

2. Computer Programs

S5.1 Papers published

This part contains copies of the following papers that are related to the thesis and have been or will be presented at confrences.

1 Douce, J.L., Zhu, Q.M., Spectral analysis of records with missing data, IFAC Symposium on identification & system parameter estimation, 1988, Beijing, China.

2 Warwick, K., Zhu, Q.M., Douce, J.L., An adaptive controller for nonlinear systems, IFAC Symposium on adaptive control & signal processing, 1989, Glasgow, UK.

3 Douce, J.L., Zhu, Q.M., Spectrum estimation and system identification from incomplete data, International conference on system science, 1989, Wroclaw, Poland.

8th IFAC/IFORS Symposium on identification & system parameter estimation, August 27-31,1988, Beijing, China.

SPECTRAL ANALYSIS OF RECORDS WITH MISSING DATA (Paper 145)

J.L. Douce

Engineering Department, University of Warwick, Coventry CV4 7AL, U.K.

Q.M. Zhu

Control Engineering Department, Qiqihaer Light Industry Institute, China, (presently at the University of Warwick).

Abstract. This paper considers the spectrum analysis of finite-duration data where the available data contains one or more uncertain observations or missing points. A new recursive method is developed using a mean-square error criterion to obtain estimates of the Fourier transform and the power spectrum of the complete data record. This technique is applied to the estimation of the frequency response of linear systems for the case in which the output data contains missing data.

It is shown that it is important to consider the position through the record of the missing points.

A comparison is presented of the results obtained using this new method and traditional methods, demonstrating the improvements obtained.

Keywords. Spectral analysis, data reduction, missing values, system identification.

NOTATION

u_i $i = 1, N$	Time series, system input.
x_i	Time series, system output, no missing observations.
y_i	Time series, system output with missing observations.
$X(\Omega), S_{xx}(\Omega)$, etc.	Amplitude and power spectra.
$\hat{X}(\Omega)$	Estimate of $X(\Omega)$, based on $Y(\Omega)$.
M	Number of missing points.
σ_x^2	Variance of specified signal.
$\bar{S}_{yy}(\Omega)$	Smoothed value of $S_{yy}(\Omega)$.
$G(\Omega)$	System frequency response function.

INTRODUCTION

In time series modelling and parameter estimation, the situation is considered in which the measured sequence is not a complete set of the observations, but the measurements corresponding to some time instants are missing, not known or unreliable.

The pattern of the uncertain observation may be in one of two categories, one deterministic or periodic, as in the case, for example, of a single sensor which is time shared to measure and record different processes, or a random or aperiodic phenomenon, as in the case of an unreliable sensor which fails intermittently.

One solution to the problem of uncertain observations is interpolation, which estimates uncertain values using the known values and then reconstructs the time series. However, the interpolating method has some disadvantages (Harris, 1987) to use. Another more general solution is more straightforward and compensates the estimated values directly without reconstruction of the time series. Attention will be paid to techniques in the second category in this paper.

Since 1962 (Jones,1982; Parzen,1983), especially 1969 (Nahi), some useful results (Harris,1987; Roberts,1980; McGiffin,1981) have been obtained in signal modelling and parameter estimation from time series with uncertain observations. A good survey of some representative methods can be found in McGiffin (1980).

The Power Spectral Density (PSD) is an important parameter in the description of random processes. There are two general methods currently available for PSD estimation with uncertain observations. We call one the Covariance Method (CM), the alternative technique the Periodogram Method (PM). A brief introduction to CM and PM will be presented after a definition of the problem.

The time series, assumed normally distributed, with no missing observations is written

$$x_i = x_1, \ldots x_i \ldots x_N \quad i = 1 \text{ to } N .$$

With missing observations, the series is written

$$y_i = x_i g_i \tag{1.1}$$

where $g_i = 0$ for a missing point, and unity otherwise.

The missing data consists of M missing points, a member of this set being x_m.

The discrete Fourier transform of x_i is

$$X(\Omega) = \sum_{i=1}^{N} x_i (\cos i\Omega - j \sin i\Omega) = X_R + jX_I \tag{1.2}$$

where $\Omega = 2\pi k/N$ with k integer.

The power spectral density is

$$S_{xx}(\Omega) = \frac{1}{N} X(\Omega).X^*(\Omega) .$$

Alternatively, the power spectral density can be expressed in terms of the measured autocovariance function according to

$$S_{xx}(\Omega) = R_{xx}(0) + 2\sum_{\tau=1}^{N} R_{xx}(\tau) \cos(\Omega\tau) \tag{1.3}$$

where $$R_{xx}(\tau) = \frac{1}{N-\tau} \sum_{i=0}^{N-\tau} x_i \, x_{i+\tau} \quad \text{for } \tau \geq 0.$$

Estimates of $X(\Omega)$, $S_{xx}(\Omega)$, etc. based on y_i, the data with missing points, are written $\hat{X}(\Omega)$, $\hat{S}_{xx}(\Omega)$, etc.

The covariance method for spectral analysis with missing points (Jones,1962; McGiffin,1980; Parzen,1963) uses equation (1.1) to give

$$R_{yy}(\tau) = R_{gg}(\tau).R_{xx}(\tau) .$$

Knowing the positions of the missing points, $R_{gg}(\tau)$ can be calculated, to give the estimate

$$\hat{R}_{xx}(\tau) = R_{yy}(\tau)/R_{gg}(\tau)$$

where $R_{yy}(\tau) = \sum_{i=0}^{N-\tau} y_i y_{i+\tau}$.

Using equation (1.3), the spectral density of $x(t)$ can be estimated.

An alternative approach termed the periodogram method (Harris,1987) is based directly on the measured power spectrum $S_{yy}(\Omega)$. This is defined by

$$\hat{S}_{xx}(\Omega) = \frac{N}{N-M} S_{yy}(\Omega)$$

where $(N-M)$ is the number of non-zero terms in $y(t)$.

For white noise, this estimator is unbiassed and minimum variance. However, as shown below, an improved estimator can be found when the original process is non-white.

THE NEW ESTIMATORS

The Fourier transform of y_t may be written

$$Y(\Omega) = \sum_{i=1}^{N} x_i \cos i\Omega - \sum_{M} x_m \cos m\Omega$$
$$= Y_R + j Y_I .$$

The estimate of $X(\Omega)$ introduced in this paper is

$$\hat{X}(\Omega) = k_1 Y_R + k_2 j Y_I , \qquad (2.1)$$

where k_1 and k_2 are chosen to minimise the mean square error, equal to $E\{(X-\hat{X}).(X-\hat{X})^*\}$. Both k_1 and k_2 are functions of the frequency Ω.

The mean square error of the estimate is

$$E\{(X_R - k_1 Y_R)^2 + (X_I - k_2 Y_I)^2\} .$$

Differentiating with respect to k_1, and setting the result to zero gives

$$2.E[X_R Y_R] = 2 k_1 . E[Y_R^2] .$$

That is

$$k_1 = \frac{E[X_R Y_R]}{E[Y_R^2]}$$

and similarly

$$k_2 = \frac{E[X_I Y_I]}{E[Y_I^2]} .$$

Appendix 1 derives the expression for k_1 on the two assumptions:-

(a) The missing points are separated such that the cross-correlation between values at missing positions may be neglected, and
(b) The missing points are not too near the end of the record.

The essential novel feature of the new method is that the values of factors k_1 and k_2 depend on the position of the missing point(s) in the record. The need for this dependence can be demonstrated from the defining equation (1.2) by considering a single missing point located at the end of the record, that is with $i = N$. By inspection, the value of x_N has zero effect on $X_I(\Omega)$ for all values of Ω, and a maximum effect on $X_R(\Omega)$ since $\cos N\Omega = 1$.

From Appendix 1,

$$k_1 = \frac{f_1 (N - 2\sum_{M} \cos^2 m\Omega)}{f_1 (N - 4\sum_{M} \cos^2 m\Omega) + 2\sum_{M} \cos^2 m\Omega} . \qquad (2.2)$$

The factor f_1 is the normalised spectrum, which is in general not known a priori, given by

$$f_1 = \frac{2}{\sigma^2} . E(X_R^2) = \frac{2}{\sigma^2} . E\{k_i^2 (\Sigma y_i \cos i\Omega)^2\} \qquad (2.3)$$

For white noise, f_1 is unity at all frequencies Ω, so that $k_1 \equiv k_2 \equiv 1$. This shows that a minimum mean-square error estimate of the Fourier transform of a white process with missing observations is $X(\Omega) = \Sigma y_i (\cos i\Omega + j \sin i\Omega)$.

The results presented later in the paper consider highly coloured noise with no a priori information. In these cases an iterative process is used, in which f_1 is updated from the current estimate of k_1 (with the initial value $k_1 = 1$ assumed), according to equation (2.3) and equation (2.2) then used to give k_1.

In a similar manner, the power spectrum estimate $S_{xx}(\Omega)$ may be written in terms of the measured Fourier components

$$\hat{S}_{xx}(\Omega) = \frac{1}{N} [k_3 Y_R^2 + k_4 Y_I^2] .$$

The result is derived in Appendix 2, giving

$$k_3 = \frac{1}{3} \frac{E(x_R^2)}{E(Y_R^2)} + \frac{2}{3} k_1^2$$
$$= \frac{N f_1 - \frac{4}{3}\sum_{M} \cos^2 m\Omega}{f_1 (N - 4\sum_{M} \cos^2 m\Omega) + 2\sum_{M} \cos^2 m\Omega}$$

Similarly for k_4, with $\cos^2 m\Omega$ replaced by $\sin^2 m\Omega$ throughout.

The frequency response of a system is estimated from (smoothed) values of the input spectral density $S_{uu}(\Omega)$ and $S_{u\hat{x}}(\Omega)$ according to

$$\hat{G}(\Omega) = \frac{\bar{S}_{u\hat{x}}(\Omega)}{S_{uu}(\Omega)} .$$

The estimate of $\hat{X}(\Omega)$ is chosen to minimise the mean squared error of the numerator term.

Let $\hat{X} = k_5 Y_R + j K_6 Y_I$ where k_5 and k_6 are chosen to minimise

$$E\{(U^*X - U^*\hat{X})(U^*X - U^*\hat{X})^*\}$$

Minimising with respect to k_5 gives

$$k_5 = \frac{E[U_R^2 X_R Y_R + U_I^2 X_R Y_R]}{E[U_R^2 Y_R^2 + U_I^2 Y_R^2]}$$

Following the same procedures as in appendices 1 and 2 gives

$$k_5 = \frac{\left[f_1 . E(S_{uu}) + N\sigma_u^2 (f_3^2 + f_5^2)\right] . \left[\frac{N}{2} - \sum_{M} \cos^2 m\Omega\right]}{E(s_{uu}) . \left[f_1 \left[\frac{N}{2} - 2\sum_{M} \cos^2 m\Omega\right] + \Sigma \cos^2 m\Omega\right] + 2\sigma_u^2 \left[f_3^2 + f_5^2\right] \left[\frac{N}{2} - \Sigma \cos^2 m\Omega\right]^2}$$

where

$$f_1 = \frac{2}{\sigma_y^2} E\{(k_5 Y_R)^2\}$$

$$f_3 = \frac{2}{\sigma_u \sigma_y} E\{k_s\ U_R\ Y_R\}$$

$$f_s = \frac{2}{\sigma_u \sigma_y} E\{k_s\ U_I\ Y_R\}\ .$$

Similarly for k_c, with $\cos^2 m\Omega$ replaced by $\sin^2 m\Omega$ throughout.

EXPERIMENTAL RESULTS

Two systems have been studied to compare the traditional and new methods for spectral analysis and for system identification. Experiments have been completed to examine the effect of smoothing over several blocks and over adjacent frequencies on the resulting estimates.

For spectral density estimates, two quantitative measures of performance have been used as comparators of the two methods. In the first, the sum over all frequencies of the sum of error squared between the true spectrum with no missing points and the estimated spectrum with missing points using the PM, CM and new technique are compared. Secondly, a linear regression of the true spectrum on the estimated spectrum again over all frequencies has been obtained, to compare the bias of the two methods.

In all experiments, a block length of 128 points has been used, and the missing points have been introduced at positions 10, 20, 30 ..., 120. In those experiments each involving one block, smoothing over four adjacent frequencies has been used to estimate the power spectrum for the calculation of k. For experiments in each involving more blocks, the spectrum is smoothed over the blocks for this calculation. Lag windows are used to smooth the covariance functions when the correlation method is used to estimate the frequency response.

First-order System

The system considered is

$$x_k = 0.9\ x_{k-1} + w_k$$

in which w_k is a white noise normally distributed signal of unity variance and zero mean value.

Figure 1 compares the squared-error in the power spectrum estimates in an experiment consisting of 1000 blocks using the traditional and new methods. The maximum error in the traditional method occurs at zero frequency. This error is equal to approximately 10% of the true value.

Ensembles of experiments, each of duration equal to one, two, ten and one hundred blocks have been completed to investigate the ensemble average of the sum of error squared over frequency of the power spectrum estimates. Table 1 summarises the results and shows the significant reduction in error resulting from the new method. This Table also lists the ensemble average of the linear regression of the true on the estimated spectrum. The new method substantially reduces the deviation of this factor from unity.

TABLE 1 Errors in Spectral Estimates

Length of Data	Periodogram		Correlation		New Method	
	Error	Best fit	Error	Best fit	Error	Best fit
128	777	1.118	573	1.086	137	1.017
128x5	605	1.099	480	1.079	462	1.079
128x10	219	1.108	115	1.032	78	1.024
$128x10^2$	179	1.106	36	1.018	19	1.004

Fourth-order System

The signal is produced by passing the previously defined white noise through the process described by Harris (1987)

$$y_{k-3} = 1.7143y_{k-4} - 0.9048y_{k-5} + w_k$$

$$y_k = 1.0732y_{k-1} - 0.9512y_{k-2} + y_{k-3}\ .$$

This produces a power spectrum with two pronounced resonances.

Data has been analysed for 128 x 10^3 points. For the periodogram method, this has been analysed using blocks of 128 points, whilst for the correlation method a triangular window of length 128 points is introduced.

The periodogram method is again inferior to the correlation method, both being worse than the new method. Figure 2 shows how the errors in spectral estimation vary with frequency, the frequencies of the maxima corresponding to the resonance frequencies of the system. The maximum error is approximately 10% of the true value. Numerical results of the from of Table 1 demonstrate very similar quantitative advantages of the new method.

Input-output data of the two systems as described above has been used for frequency response estimation. The new method again shows significantly superior results compared with the earlier methods. However in this case the periodogram method is superior to the correlation technique.

TABLE 2 Errors in Frequency Response Estimates

Length of Data	Periodogram		Correlation		New Method	
	Error	Best fit	Error	Best fit	Error	Best fit
128	133	1.02	199	0.86	133	1.02
128x5	39	1.004	112	0.86	38	1.003
128x10	17	1.003	108	0.88	16	1.003
$128x10^2$	1.8	1.003	95	0.90	1.7	1.002

Table 2 gives a quantitative comparison of the three methods for the fourth order system. Figure 3 shows, for the same system with 128 x 10^2 data points, the variation of magnitude of error squared of the frequency response estimate with frequency.

CONCLUSIONS

This paper has drawn attention to the effect of the location of missing points on the resulting spectral estimates. This effect has not been considered in detail in earlier papers, but it has a substantial influence on the errors introduced by these missing points. Simulation studies have demonstrated the improvement in estimates of power spectral density and of frequency response which follow when the position of the missing points is included in the analysis.

REFERENCES

Parzen, E. (1963). On spectral analysis with missing observations and amplitude modulation. *Indian Journal of Statistics*, A25, pp. 383-392.

Faridani, H.M. (1986). Performance of Kalman filter with missing measurements. *Automatica*, 22, pp.117-120.

Harris, R.W. (1987). Spectra from data with missing values. *Mechanical Systems and Signal Processing*, 1(1), pp. 94-104.

Jones, R.H. (1962). Spectral analysis with regularly missed observations. *Ann. Math. Stat.*, 32, pp. 455-461.

Roberts, J.B. (1980). On the estimation of spectra of randomly sampled signals: a method of reducing variability. *Proc. R. Soc.* Lond. A371, pp. 235-258.

McGiffin, P.B. (1980). Parameter estimation for auto-regressive systems with missing observations. *Int. J. Systems Sci.*, 11, pp. 1021-1034.

McGiffin, P.B. (1981). Parameter estimation for auto-regressive systems with missing observations – part II. *Int. J. Systems Sci.*, 12, pp. 657–663.

Nahi, N. (1969). Optimal recursive estimation with uncertain observations. *IEEE Trans. Inf. Theory*, IT–15, pp. 457–462.

APPENDICES

Appendix 1

DERIVATION OF THE AMPLITUDE SPECTRUM ESTIMATE

From equation (2.1), the estimate of the real part of the amplitude spectrum is

$$X_R(\Omega) = k_1 \, Y_R(\Omega)$$

where $k_1 = \dfrac{E[X_R Y_R]}{E[Y_R^2]}$

$$X_R Y_R = (\Sigma x_i \cos i\Omega).(\Sigma x_i \cos i\Omega - \sum_M x_m \cos m\Omega).$$

$$\therefore E(X_R Y_R) = E((\Sigma x_i \cos i\Omega)^2) - E(\Sigma x_i \cos i\Omega \sum_M x_m \cos m\Omega)$$

$$= \frac{N}{2} \bar{S}_{xx} - \sum_M \cos^2 m\Omega \, E(\sum_i x_i \, x_m \cos (i-m)\Omega)$$

$$= f_1 \sigma^2 \left[\frac{N}{2} - \sum_M \cos^2 m\Omega \right] .$$

The factor f_1 is equal to the ratio (power spectral density at the frequency Ω) ÷ (variance of the process x_i).

Similarly $Y_R^2 = (\Sigma x_i \cos i\Omega - \sum_M x_m \cos m\Omega)^2$.

Again, assuming that the missing points are adequately separated such that

$$E((\sum_M x_m \cos m\Omega)^2) = \sigma^2 \sum_M \cos^2 m\Omega ,$$

$$E(Y_R^2) = \sigma^2 \left[f_1 \left[\frac{N}{2} - 2\sum_M \cos^2 m\Omega \right] + \sum_M \cos^2 m\Omega \right].$$

An identical derivation gives k_2 with (sin $m\Omega$) replacing (cos $m\Omega$) throughout.

Appendix 2

DERIVATION OF THE POWER SPECTRAL DENSITY ESTIMATE

The true and estimated power spectra are

$$S_{xx}(\Omega) = \frac{1}{N} \left[X_R^2 + X_I^2 \right]$$

$$\hat{S}_{xx}(\Omega) = \frac{1}{N} \left[k_3 \, Y_R^2 + k_4 \, Y_I^2 \right] .$$

k_3 and k_4 are chosen to minimise separately the expected value of

$$(X_R^2 - k_3 \, Y_R^2)^2 \quad \text{and} \quad (X_I^2 - k_4 \, Y_I^2)^2 .$$

This gives

$$k_3 = \frac{E(X_R^2 \, Y_R^2)}{E(Y_R^4)}$$

and $k_4 = \dfrac{E(X_I^2 \, Y_I^2)}{E(Y_I^4)}$.

Since X_R and Y_R are zero mean normal variables,

$$E(X_R^2 \, Y_R^2) = E(X_R^2).E(Y_R^2) + 2.[E(X_R \, Y_R)]^2$$

and $E(Y_R^4) = 3.[E(Y_R^2)]^2$.

This gives $k_3 = \dfrac{1}{3} \dfrac{E(X_R^2)}{E(Y_R^2)} + \dfrac{2}{3} k_1^2$.

With $E(X_R^2) = \dfrac{N}{2} f_1 \sigma^2$ and using the expressions given in Appendix 1, the required result is obtained. An identical derivation applies for k_4.

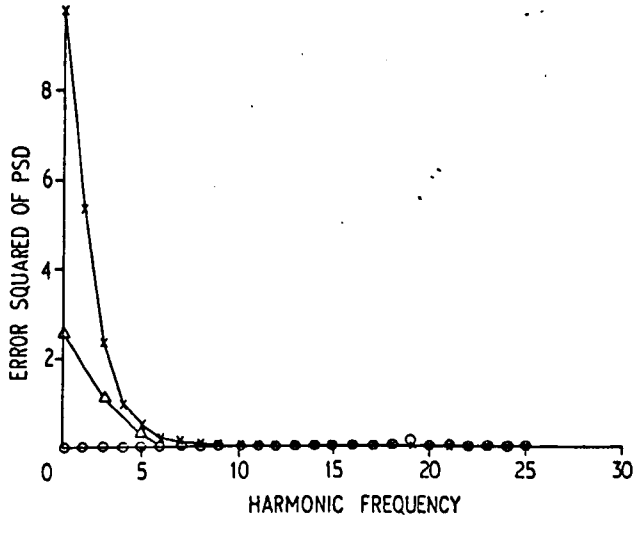


Fig. 1. Error squared as function of frequency, first order system.

X Periodogram method.

Δ Correlation method.

O New method.

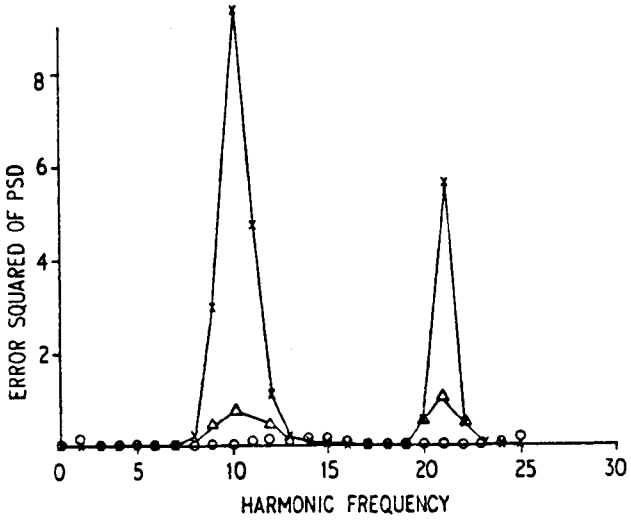


Fig. 2. Error squared for fourth order system.
X Periodogram method.
Δ Correlation method.
O New method.

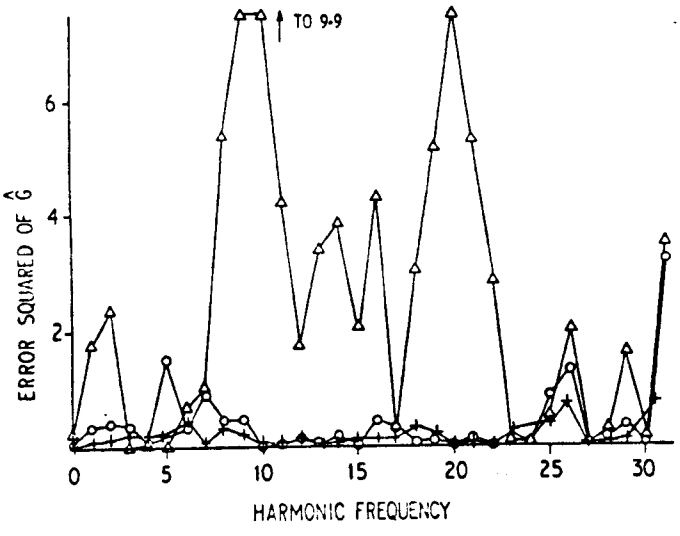


Fig. 3. Error in estimates of G.
O Periodogram method.
Δ Correlation method.
+ New method.

IFAC ACASP 89 Symposium on adaptive control & signal processing, April 19-21, 1989, Glasgow, UK.

AN ADAPTIVE CONTROLLER FOR NON-LINEAR SYSTEMS

K. Warwick

Department of Cybernetics, University of Reading, Whiteknights, Reading, RG6 2AL, U.K.

Q-M. Zhu and J.L. Douce

Department of Engineering, University of Warwick, Coventry, CV4 7AL, U.K.

Abstract. A technique is described for the adaptive control of nonlinear systems. The method is based on a combined Hammerstein + ARMA system model, and is centred on a General Predictive Control Scheme. The overall procedure is termed Nonlinear General Predictive Control and employs a recursively calculated control algorithm which is aimed at making the technique applicable in a practical sense.

In the Nonlinear General Predictive Controller described, the nonlinear and linear system characteristics are treated separately. To this end a simple root solving exercise, dependent on operation within a recursive, tuning algorithm, is employed to find the inverse Hammerstein characteristic parameters which form the nonlinear part. It is shown that the same Hammerstein procedure can be used to form a Nonlinear Dead Beat Controller, and the results of such a controller in operation are compared with those of the Nonlinear General Predictive Controller.

Keywords. Adaptive control; nonlinear control systems; predictive control: discrete time systems; self-adjusting systems.

INTRODUCTION

As the employment of computer based adaptive control schemes becomes more widely acceptable, in a practical sense, so controllers must become more versatile in terms of the types of plant on which they can be operated. In particular, self-tuning control techniques are based on the philosophy that the system to be controlled can be regarded essentially, as being linear. For a large number of systems such an approach is acceptable, any relatively 'small' nonlinearities, such as stiction and minor hysteresis, being effectively linearized by the controller. This results in a performance which is deemed to be all right, but which is nevertheless well below that which would be expected, had the plant been perfectly linear. In order to arrive at much improved control over systems with small nonlinearities or to deal efficiently with strongly nonlinear systems, whilst reaping the benefits of a self-tuning control algorithm, the nonlinearities must be taken care of in an appropriate fashion.

A large number of different types of nonlinearity can occur in practice (Atherton, 1975; Cook, 1988) which means that the extension of a basic linear control scheme to account for all possibilities is not a realistic proposition and would necessarily result in a bank of possible controllers which require an element of operator selection and hence a reasonable depth of plant knowledge. A more sensible approach to the problem is to employ a framework with which a large number of nonlinear plants can be adequately modelled. It has been found that the most suitable solution within the field of self-tuning control is the use of a parametric plant description via a Hammerstein model (Anbumani and Co-Workers, 1981; Lachmann, 1982; Agarwal and Seborg, 1987). This model basically constitutes a linear ARMA system model coupled with a polynomial expressed in powers of the control input. By this means the plant to be controlled is thought of as consisting of a linear part cascaded with a nonlinear part. The ARMA model is then used to represent the linear part, whereas the extra power polynomial approximates the nonlinear part. In fact such an underlying philosophy has been shown as sensible within an adaptive control scheme (Anbumani and Co-Workers, 1981; Lachmann, 1982).

An adaptive nonlinear control scheme is described in this paper with the Hammerstein model as a central theme in a control algorithm which is a nonlinear form of Generalised Predictive Control. The resultant controller retains the flexibility of the linear system General Predictive Control form (Clarke and Co-Workers, 1987), and yet also has the ability to deal effectively with a large number of nonlinear systems. The overall controller design procedure involves two distinct parts, such that the method is indirect, one dealing with the linear elements and the other with the nonlinear elements. On the assumption that the computer based model of the plant is a good representation of the actual plant, a linear controller is designed to provide self-tuning General Predictive Control for the linear part plant model relating pseudo-input x(t) to actual plant output y(t). The inverse function of the nonlinear part is then found, in order to obtain the actual plant input signal u(t). The intermediate pseudo-input x(t), is the input to the linear part system and also the output of the nonlinear part system, it is therefore not a measureable quantity, although it can be estimated within the model due to the process separability (Billings and Fakhouri, 1978). It is worth noting that if the actual system proves to be, to all intents and purposes, linear,so the overall control problem reduces to a linear General Predictive Control requirement, and this reduction property is also exhibited by the controller described here.

The feasibility of Hammerstein model based adaptive controllers has been studied to an extent (Grimble, 1985), and this has highlighted a major problem in the use of such models as being the necessity for the on-line computation of a root solving algorithm

every recursion of the controller updating procedure. Not only is this an extremely time consuming process, which can be a particular problem when the sampling period is small, but also the accuracy of the root solving routine, which is needed to obtain a unique solution to the nonlinear plant model polynomial, can be poor, can produce stability problems and usually requires an odd number of polynomial roots.

In this paper a novel approach is taken by the employment of a simple one step Newton-Raphson iteration, every recursion of the overall algorithm, in place of a complex root solving routine. The technique makes use of signal values from the previous recursion of the algorithm such that on-line computation time is reduced to that necessary for a few multiplications and additions. The end result is that, because each input signal applied is based on its previously employed value, the transmitted signal is filtered as it is applied to the plant, thus input signal variations, from one sample period to the next, are reduced. A positive feature of the controller is therefore the reduction of control input swings, which are often not realisable in practice due to actuator limitations (Payne, 1986; Warwick, 1988).

Details of an implementation study are given to show how the nonlinear adaptive General Predictive Controller can be realised, in a practical sense, and performance results indicate how the method compares with another adaptive control scheme based on the same approach to dealing with nonlinear tendencies but with a different central control objective. An adaptive nonlinear Deadbeat Controller was chosen for this purpose, in order to show the flexible nature of the control approach.

PLANT MODEL

It is assumed that the plant to be controlled can be adequately represented by the SISO discrete-time Hammerstein model of the form:

$$Ay(t) = Bx(t - 1) + Ce(t) \tag{1}$$

where the polynomials A, B and C are defined as,

$$\begin{aligned} A &= 1 + a_1 q^{-1} + \ldots\ldots + a_{n_a} q^{-n_a} \\ B &= b_0 + b_1 q^{-1} + \ldots\ldots + b_{n_b} q^{-n_b} \\ C &= 1 + c_1 q^{-1} + \ldots\ldots + c_{n_c} q^{-n_c} \end{aligned} \tag{2}$$

in which q^{-1} is the unit backward shift operator, such that $q^{-i}y(t) = y(t - i)$. Also $\{y(t) : t\in\tau\}$ is the system output sequence and $\{e(t) : t\in\tau\}$ is a disturbance affecting the system whereby $\delta e(t) = \epsilon(t)$, $\delta = 1 - q^{-1}$, which allows for non-zero offset on a zero mean, white noise signal, Tuffs and Clarke (1985).

The intermediate variable, x(t), is the nonlinear element output and the linear element input, and is defined by

$$x(t) = \sum_{i=0}^{n_\gamma} \gamma_i u^i(t) \tag{3}$$

where $\{u(t) : t\in\tau\}$ is the system input sequence, and is the sequence actually applied to the plant

It can be noted that the transport delay, relating input to output in the system model of eq. (1), is given as unity - this is done only to ease the following explanation of the controller, in general the technique described works with any specific transport delay.

In the adaptive controller described in this paper, the parameters γ_i, a_i, b_i and indeed c_i can be estimated by an enhanced recursive least squares procedure (Kortmann and Unbehauen, 1987; Shah and Cluett, 1988), when the plant model is continuously updated within a self-tuning algorithm. It is perhaps easier however, for the reader to initially consider these parameters as both fixed and known or identified values, the requirement for them to be recursively estimated can then be reintroduced at a later time, specifically for adaptive controller purposes. To simplify the explanation though, it will be considered that C = 1, i.e. $c_i = 0 : i = 1, \ldots, n_c$, noting that the method can readily cope with colored noise, should this occur. Naturally, this means that when the recursive estimator is employed, the c_i parameters need not be included in the estimation procedure.

PREDICTIVE CONTROL

Although the intention of this paper is to primarily put forward the concept of a particular adaptive controller for nonlinear systems, nevertheless a linear adaptive control algorithm is central to the theme. Mainly because of its positive attributes and widespread application possibilities, a predictive control method has been selected in the first instance, and this is now described.

A prediction of the plant output signal is required as a fundamental aspect of the controller considered, and this is based on information available at a particular time instant. By considering merely the linear part of the plant represented in eq. (1), a prediction of the output signal at time instant t + k is directly obtained, based on information available at time instant t, where the index k indicates k sample periods in the future. For this purpose a diophantine identity is introduced (Owens and Warwick, 1988).

$$1 = E_k A\delta + q^{-k}F_k \tag{4}$$

where $k \geq 1$ and E_k, F_k are polynomials which are unique for any given prediction horizon k and plant polynomial A, such that

$$\begin{aligned} F_k &= f_0 + f_1 q^{-1} + \ldots\ldots + f_{n_a} q^{-n_a} \\ E_k &= 1 + e_1 q^{-1} + \ldots\ldots + e_{k-1} q^{-(k-1)} \end{aligned} \tag{5}$$

If now eq. (1) is multiplied throughout by $q^k E_k \delta$ and the diophantine identity, eq. (4), is made use of, it follows that

$$y(t + k) = BE_k \delta x(t + k - 1) + F_k y(t) + E_k \epsilon(t + k) \tag{6}$$

in which the disturbance terms $E_k\epsilon(t + k)$ are all future values, from time t, due to the fact that the E_k polynomial is of order k - 1.

The optimal prediction of the output signal at time instant t + k, made at time instant t, is therefore

$$\hat{y}(t + k/t) = BE_k \delta x(t + k - 1) + F_k y(t) \tag{7}$$

where $\hat{y}(t + k/t)$ indicates the predicted value of the actual output signal y(t + k), given the information available up to and including that at time instant t.

```
function [rc,xm1]=dbc2(tempy,pm,phi,c)
format compact
clc;clg;
% the function started 20/3/1989.
% this function is used to design directly
% a deadbeat controller with an algorithm given by
%
%         B(1)U(t)=B*U(t-k) + A*[W(t)-Y(t)]
n=length(pm); rc=0;
nb=c(1);na=c(2);
am(1)=1;am(2:na+1)=pm(n-na+1:n,1)';
for i=1:3
    rc=[rc,sum(pm(2+(i-1)*(nb+1):1+i*(nb+1),1)')];
end
xm1=pm(2:n-na,1)'*phi(2:n-na,1)+am*tempy';
```

As an aside, it is easy to see from eqs. (6) and (7), that the actual system output can be regarded as

$$y(t+k) = \hat{y}(t+k/t) + E_k\varepsilon(t+k) \qquad (8)$$

It is though, possible at any time instant t, to select a range of values k for which an output prediction can be made. Assuming that a set of output predictions are to be made, from k = 1 to k = N, where N ≥ 1, then eq. (6) can be written in vector form as

$$\underline{y} = G\underline{\delta x} + \underline{f} + \underline{\varepsilon} \qquad (9)$$

in which $\underline{y}^T = [y(t+1), \ldots\ldots, y(t+N)]$

$$\underline{\delta x}^T = [\delta x(t), \delta x(t+1), \ldots, \delta x(t+N-1)] \qquad (10)$$

$$\underline{f}^T = [f(t+1), f(t+2), \ldots, f(t+N)]$$

and $\underline{\varepsilon}^T = [E_1\varepsilon(t+1), E_2\varepsilon(t+2), \ldots\ldots, E_N\varepsilon(t+N)]$

noting that $\underline{y}^T$ indicates the transpose of $\underline{y}$.

In eq. (9), the matrix G is of dimension N x N and has elements such that

$$G = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \ddots & \vdots \\ \vdots & g_1 & \ddots & 0 \\ g_{N-1} & g_{N-2} & \cdots & g_0 \end{bmatrix} \qquad (11)$$

where the g_i parameters are obtained from the integrated plant step-response, i.e. $g_i = h_i$:

$$H = (\delta A)^{-1}B = h_0 + h_1q^{-1} + h_2q^{-2} + \ldots\ldots \qquad (12)$$

The vector $G\underline{\delta x}$ in eq. (9) represents a set of unknown values, due to the $\underline{\delta x}$ vector, at time instant t - the elements in G are themselves assumed to be known, if the plant is known or available from parameter estimations in the adaptive case. Note also that the signal x(t), and therefore δx(t), is unknown until it has been calculated by means of the control algorithm and subsequently applied to the plant, thus at the instant that the output predictions are made, it is an unknown signal

The vector $\underline{f}$ in eq. (9) represents a set of known values at time instant t, such that the individual terms in $\underline{f}$ are given by

$$f(t+k) = F_k y(t) + \bar{h}_k \qquad (13)$$

where

$$\bar{h}_k = (\delta E_k - [h_0 + h_1q^{-1} + \ldots\ldots + h_{k-1}q^{-(k-1)}]) \cdot \delta x(t+k-1) \qquad (14)$$

So, the vector of future output signals $\underline{y}$, can be considered, from eq. (9), to consist at time instant t of a vector of unknown signals $G\underline{\delta x}$, a vector of known signals $\underline{f}$, and a vector of noise signals $\underline{\varepsilon}$.

CONTROLLER DESIGN

The overall control algorithm design is made up of two distinct parts, connected by means of the intermediate variable x(t), and relating to firstly the linear system part and secondly the nonlinear system part. The linear part is considered first, and for predictive control a finite horizon cost function can be written as:

$$J = \sum_{k=1}^{N} [y(t+k) - v(t+k)]^2 + \sum_{k=1}^{N} \lambda(k)[\delta x(t+k-1)]^2 \qquad (15)$$

where N is the maximum prediction horizon, v(t) is a reference input signal, applied at time instant t, and λ(k) is a costing applied to the control inputs. It is worth noting that the output and control input horizons have both been selected as N, this is by no means a necessary requirement, in fact other control horizon selections for linear General Predictive Controllers have been investigated elsewhere (Clarke and Co-Workers, 1987; Warwick and Clarke, 1988).

The control objective is to obtain a vector of future control inputs $\underline{\delta x}$ which will minimize the expected value of the cost function (15). Now the expected value of J is

$$E\{J\} = E\{(G\underline{\delta x} + \underline{f} + \underline{\varepsilon} - \underline{v})^T(G\underline{\delta x} + \underline{f} + \underline{\varepsilon} - \underline{v}) + \underline{\delta x}^T\lambda\underline{\delta x}\} \qquad (16)$$

where E{.} signifies the expected value, and λ is an N x N diagonal matrix whose N diagonal elements are: λ(1), ... λ(N). Also

$$\underline{v}^T = [v(t+1), v(t+2), \ldots\ldots, v(t+N)] \qquad (17)$$

denotes the future set of reference input values, i.e. the required trajectory.

By differentiating eq. (16) with respect to $\underline{\delta x}$ and equating the result to zero, a cost function minimum is given by the optimal control

$$\underline{\delta x} = (G^TG + \lambda)^{-1}G^T(\underline{v} - \underline{f}) \qquad (18)$$

such that the first row of this equation can also be written as

$$x(t) = x(t-1) + \bar{g}^T(\underline{v} - \underline{f}) \qquad (19)$$

where $\bar{g}^T$ is the first row of $(G^TG + \lambda)^{-1}G^T$.

The control signal x(t) is the signal which, if the system was linear, would be applied as the control input. It is based on a set of known future reference signals $\underline{v}$ along with the known vector $\underline{f}$, as can be seen from eq. (19). In fact the signal $\underline{x}(t)$, obtained in eq. (18), is an intermediate variable, found as a solution to the linear predictive controller problem. It remains for the nonlinear part of the controller problem to be solved, which is done directly, remembering that the control signal applied to the actual (nonlinear) plant, u(t) is related to x(t) by means of eq. (3)

The nonlinear problem, can in this case, be stated as: given any signal x(t), at time instant t, and the appropriate coefficients $\gamma_i : i = 0, \ldots, n_\gamma$, find the control input signal u(t).

From eq. (3) it can be noted that

$$x(t) = \phi\{u(t)\} \qquad (20)$$

where φ{.} denotes a functional operator, in this case a Hammerstein model polynomial.

However, we actually require:

$$u(t) = \phi\{x(t)\} \qquad (21)$$

which means that a root of the Hammerstein model polynomial must be found in order to produce a possible signal u(t).

A simple solution to the polynomial can in fact

be found by the Newton-Raphson recursive method (Gerald, 1978), whereby

$$u_{n+1}(t) = u_n(t) - \frac{[\phi(u_n(t)) - x(t)]}{\phi'(u_n(t))} \qquad (22)$$

in which the subscript n denotes the order of iteration, such that the (n+1)th iteration is obtained from the nth iteration, n ≥ 0, and

$$\phi'(u_n(t)) = \frac{d\phi}{du_n}(u_n(t)) \quad .$$

A discussion of implementation policies with regard to this algorithm, including initial value selection and avoiding problem areas, is carried out in the following section.

CONTROLLER IMPLEMENTATION

The adaptive discrete-time controller described in this paper requires the following course of action every sample period:

1. Update a recursively estimated model of the plant.
2. Calculate x(t) via eq. (19), using the estimated model parameters.
3. Calculate u(t) to be applied as the (n+1)th iteration of eq. (22). It is suggested here that n = 0 will suffice.
4. Apply u(t).

However, several problems occur when applying eq. (22), the first being that $\phi'(u_n(t)) = 0$ in the neighbourhood of a solution. This is an extremely critical point because in practice it cannot be guaranteed that the calculated function derivative will not approximately equate with zero after any particular recursion, due to model variation, estimation error and even an unsuitable initial value. Another problem is the possibility of no real root of the polynomial existing, thus causing algorithm breakdown. In order to overcome these problems, whilst retaining stability, the following possibilities arise.

When $\phi'(u_n(t)) = 0$ it is either the case that $u_n(t)$ is a root which satisfies the polynomial or it is not. This can easily be checked by taking account of $\phi(u_n(t)) - x(t)$, which will be within a preset small value, i.e. approximately zero, if $u_n(t)$ is a root. However, if $u_n(t)$ is not a root then a new alternative initial value is employed and the recursion process is repeated. If no real root exists and/or if several searches have been carried out with alternative initial values, a monitoring loop instructs the root solver to stop and a default is taken such that $u(t) = x(t)/\alpha$, where α is a positive constant.

The polynomial order can, if it is so desired, be restricted to an odd number, thereby ensuring that there is at least one real root (Anbumani and Co-Workers, 1981). This procedure can though introduce modelling errors and certainly restricts the type of nonlinearity which can be considered. In this paper therefore, no such restriction is placed on the method described.

The initial control input signal value for eq.(22) is taken as $u_n(t) = u(t-1)$, i.e. the initial value for the iterative start up at time instant t is equal to the control input actually applied at time instant t - 1. This is an extremely suitable choice when reference input changes are either small or infrequent and/or when the signal to noise ratio of the plant is high. In general it is found that with this initialisation procedure, only one or two iterations of eq. (22) are normally required before a good approximation of the solution root is obtained. In the algorithm presented in this paper, the root solving procedure employed involves only a single iteration of eq. (22) each time a new solution is required. However, if the solution, at a particular time instant, does not fit well the polynomial in question, a further preset number of iterations can be carried out, the actual number depending on the inter-sample period available. If the solution is still not good, an alternative initial value is applied, as described earlier.

To summarize, the nonlinear General Predictive Controller described here consists of the following sequence of events during every sample period, the plant under control being periodically sampled.

1. Sample plant output (at time instant t).
2. Update plant model coefficient estimates using enhanced recursive least squares. Note: both the linear part coefficients a_i, b_i and the nonlinear part coefficients γ_i are updated.
3. Using the estimated linear model coefficients a_i, b_i, calculate the linear intermediate signal x(t) from eq. (19).
4. Using the single-iteration Newton-Raphson method eq. (22), calculate the actual plant control input u(t) from the estimated nonlinear model coefficients γ_i and x(t).
5. Apply u(t) to the plant input.
6. Store all appropriate data and coefficient estimate values.
7. Wait for sampling clock pulse then go to 1.

IMPLEMENTATION STUDIES

In order to investigate the usefulness of the nonlinear General Predictive Controller, described in the previous section, the control of a plant with distinct linear and nonlinear parts was simulated. The same plant was also placed under the control of a nonlinear Deadbeat controller, based on the same nonlinear part design features, in order to show the general applicability of the method.

With reference to eqs. (1) and (2) the linear system part was regarded as consisting of:

$$A = 1 - 0.9\,q^{-1} \quad \text{and} \quad B = 1 + 2\,q^{-1}$$

whilst the nonlinear part was regarded as consisting of:

$$x(t) = 1 + u(t) - u^2(t) + 0.2\,u^3(t)$$

such that the linear part is open-loop stable and non-minimum phase.

An enhanced nonlinear system RLS parameter estimation procedure was employed, with a fixed forgetting factor of 0.95, also no noise was introduced into the system, e(t) = 0. Further, it was assumed within the parameter estimator, that the model structure was known.

In order to consider transient behaviour, a sequence of set-point reference values was assigned as follows:

Samples	Reference
1-10	20
11-30	20
31-50	60
51-70	20
71-90	0

The cycle 11-90 samples was then repeated periodi-

cally.

Figures 1 and 2 show the behaviour of a nonlinear General Predictive Controller with N = 10 and W(k) = 0. for all k. The predictive nature of the controller can be clearly seen in the plots, where advance knowledge of a reference value change has allowed for the actual output signal to commence its distinct set value variation before the actual alteration in reference value has occurred. This has resulted in an overshoot free output response once the initial tuning in period has been passed.

Figures 3 and 4 show the behaviour of a nonlinear Deadbeat Controller when operating on the plant model described. It can be seen that following a change in reference signal, the system output reaches the new set-point after only 2 sample periods and no overshoot occurs. These results are those which would be expected from a Deadbeat Controller operating on a purely linear plant (Warwick, 1986), the 2 sample periods corresponding to the sum of the dead time and the order of the B polynomial, both of which are unity.

CONCLUSIONS

Adaptive control has been shown to be an extremely useful tool for certain time varying systems, these systems have, in the main, been regarded as linear. In this paper a self-tuning control algorithm has been presented, which, is based on either General Predictive or Deadbeat control, and which deals with many types of nonlinear system, i.e. those systems with nonlinearities which can be adequately represented within a Hammerstein model. The major computational root-solving problems that are inherent with such an algorithm were avoided by the use of a simple one step Newton-Raphson root solver. In fact such an approach can be taken with the employment of different types of linear part control algorithms, e.g. pole placement or optimal. The characteristics of any particular linear control objective, e.g. General Predictive Control, are retained in the nonlinear controller and hence much more attention can be paid to the nonlinear modelling technique.

The adaptive controller described in this paper operates efficiently whether the system under control is linear or nonlinear, however a detailed theoretical analysis of controller behaviour, particularly transient behaviour, is of little value except in some very specific cases - this is a common point with self-tuning control algorithms. The feasibility of the nonlinear General Predictive Controller was therefore investigated alongside a nonlinear Deadbeat Controller, by means of some simulation studies, results from these and others indicate that both controllers can not only be very successfully applied to nonlinear systems, but also that they can be applied in a relatively simple fashion.

REFERENCES

Agarwal, M., and D.E. Seborg (1987). Self-tuning contollers for nonlinear systems. Automatica, 23, 204-214.

Anbumani, K., L.M. Patnoik, and I.G. Sarma (1981). Self-tuning minimum variance control of nonlinear systems of the Hammerstein model. IEEE Transactions, AC-26, 959-961.

Atherton, D.P. (1975). Nonlinear control engineering. Van Nostrand Reinhold.

Billings, S.A., and S.Y. Fakhouri (1978). Theory of separable processes with applications to the identification of nonlinear systems. Proc. IEE, 125, 1051-1058.

Clarke, D.W., C. Mohtadi, and P.S. Tuffs (1987). General predictive control. Automatica, 23, Parts I and II, 137-160.

Cook, P. (1986). Nonlinear dynamical systems. Prentice-Hall Int.

Gerald, C.F. (1978). Applied numerical analysis. Addison-Wesley.

Grimble, M.J. (1985). Observations-weighted minimum - variance control of linear and nonlinear systems. Int. J. Systems Science, 16, 1481-1492.

Kortmann, M., and H. Unbehauen (1987). Identification methods for nonlinear MISO systems. Proc. IFAC World Congress, Munich, 225-230.

Lachman, K.H. (1982). Parameter adaptive control of a class of nonlinear processes. Proc. 6th IFAC Symposium on Identification and System Parameter Estimation, Washington DC, 372-378.

Owens, D.H. and K. Warwick (1988). Extended predictive control. Proc. IEE International Conference Control 88, Oxford, 622-627.

Payne, A.N. (1986). One-step-ahead control subject to an input-amplitude constraint. Int.J. Control, 43, 1257-1269.

Shah, S.L. and W.R. Cluett (1988). RLS based estimation schemes for self-tuning control. In K. Warwick (Ed.), Implementation of Self-Tuning Controllers, Peter Peregrinus Ltd., 23-40.

Warwick, K. (1986). Adaptive deadbeat control of stochastic systems. Int.J. Control, 44, 651-664.

Warwick, K. and D.W. Clarke (1988). Weighted input predictive controller. Proc. IEE, 135, Pt. D, 16-20.

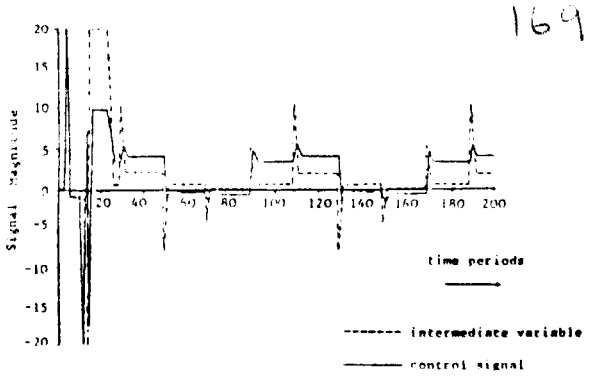


Fig. 1. Nonlinear General Predictive Control

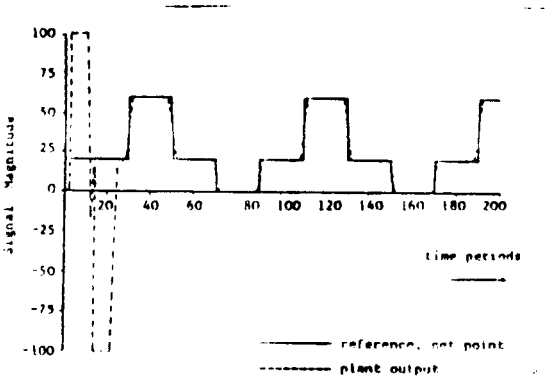


Fig. 2. Nonlinear General Predictive Control.

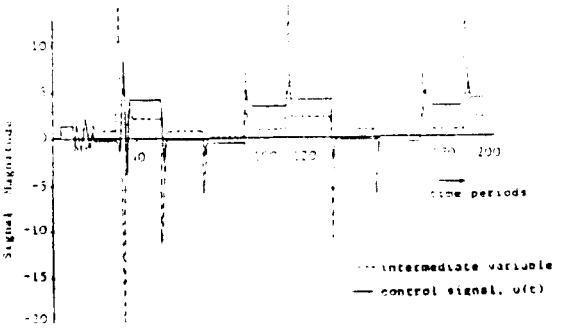


Fig. 3. Nonlinear deadbeat control

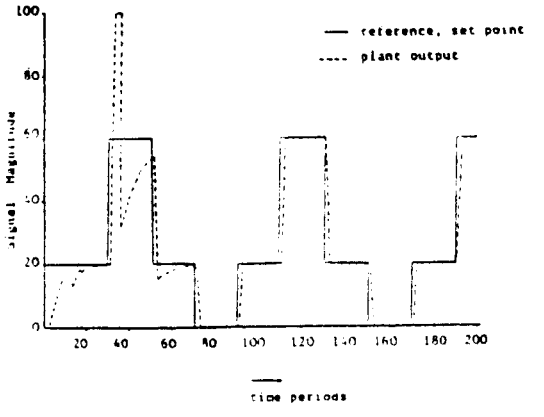


Fig. 4. Nonlinear deadbeat control

International conference on system science, September 19-22, 1989, Wroclaw, Poland.

SPECTRUM ESTIMATION AND SYSTEM IDENTIFICATION FROM INCOMPLETE DATA

Professor J.L. Douce and Q.M. Zhu
Department of Engineering
University of Warwick
Coventry CV4 7AL.

ABSTRACT

This paper considers the spectrum analysis of finite duration records where the available data contains one or more uncertain observations or missing points. A new direct method is developed using an unbiased or minimum mean square error criterion to obtain estimates of the Fourier transform and the power spectrum density of the complete data record. The analysis is extended to include frequency response identification of a wide range of systems with missing data at the output.

It is shown that it is important to consider the position within the record as well as the total number of the missing points. One result of a periodic distribution of the missing points is the production of a spurious periodicity in the measured power spectrum. It is believed that this phenomenon has not been reported previously.

A comparison of the new method with traditional techniques is presented, and simulation results demonstrate the resulting improved performance.

Keywords

Spectral analysis, Data reduction, Missing values, Frequency response.

Notation

u	system input, no missing observations
x	system output, no missing observations
y	system output, with missing observations
$X(\Omega), S_{xx}(\Omega)$, etc.	amplitude and power spectra
$\hat{X}$	estimate of $X(\Omega)$ based on $Y(\Omega)$
M	number of missing points
σ_x^2	variance of specified signal
$\bar{S}_{yy}(\Omega)$	smoothed value of $S_{yy}(\Omega)$
$G(\Omega)$	system frequency response
E[.]	expectation
X^*	complex conjugate of X
CM	Covariance Method
PM	Periodogram Method
UNB	UNBiased or UNB criterion
MSE	Mean Square Error or MSE criterion
Ω	harmonic frequency

1. Introduction

In time series modelling and parameter estimation, a common situation is met in which the measured sequence is not a complete set of the observations, but the measurements corresponding to some time instants are missing, not known or unreliable.

Missing data can arise from a number of causes, such as failure of recording equipment, clerical errors, rejection of outliers, or because of an inability to observe the phenomenon at certain times. The pattern of the missing data, i.e. the distribution of the missing data position, may be in one of two categories, one deterministic or periodic, as in the case, for example, of a single sensor which is time shared to measure and record different processes, or a random or aperiodic phenomenon as in the case of an unreliable sensor which fails intermittently.

One solution to the problem of uncertain observation is interpolation (Bard 1974; John and Prescott 1975; Jarrett 1978; Smith 1981), which estimates uncertain values using the known values and then reconstructs the time series. However, all the authors rely on parametric models of the process, which is not always desirable in frequency domain analysis. The methods become quite complicated when adjacent missing values are encountered.

Other preferred methods do not reconstruct the original time series from the given observations, but introduce direct compensation of the measured statistics. These require less a priori information, and may require less computation effort than interpolation techniques. Notable contributions include Jones (1962), Parzen (1963), Nahi (1969), Robert and Gaster (1980), McGiffin and Murthy (1981), and Harris (1987). A good survey can be found in McGiffin and Murthy (1980). The earlier methods are outlined to permit quantitative comparisons to be made with the new technique introduced in this paper.

There are two general methods currently available for Power Spectral Density (PSD) estimation with uncertain observations. We call one the Covariance Method (CM), the alternative technique the Periodogram Method (PM). A brief introduction to these will be presented after a definition of the problem.

The time series, assumed normally distributed, with no missing observations is written

$$x = x_1, \ldots, x_i, \ldots, x_N , \qquad i = 1,N.$$

With missing observations, the series is written

$$y = x_i \cdot g_i \tag{1.1}$$

where $g_i = 0$ for a missing point and is unity otherwise. The missing data consists of M missing points, a member of this set being x_m.

The discrete Fourier transform of x is

$$X(\Omega) = \sum_{i=1}^{N} x_i(\cos i\Omega - j \sin i\Omega) = X_R + jX_I$$

where $\Omega = \frac{2\pi k}{N}$ with k integer.

The power spectral density is

$$S_{xx}(\Omega) = \frac{X(\Omega)X^*(\Omega)}{N}$$

Alternatively, the power spectral density can be expressed in terms of the measured autocovariance function according to

$$S_{xx}(\Omega) = R_{xx}(0) + 2\sum_{\tau=1}^{N} R_{xx}(\tau)\cos(\Omega\tau) \qquad (1.2)$$

where

$$R_{xx}(\tau) = \frac{1}{N}\sum_{i=1}^{N-\tau} x_i x_{i+\tau} \qquad \text{for } \tau \geq 0.$$

The covariance method for spectral analysis with missing points (Jones, 1962; Parzen, 1963; McGiffin and Murthy, 1980) uses eqn.(1.1) to give

$$R_{yy}(\tau) = R_{gg}(\tau).R_{xx}(\tau).$$

Knowing the positions of the missing points, $R_{gg}(\tau)$ can be calculated to give

$$R_{yy}(\tau) = \sum_{i=1}^{N-\tau} y_i y_{i+\tau}$$

$$R_{gg}(\tau) = \sum_{i=1}^{N-\tau} g_i g_{i+\tau}$$

$$\hat{R}_{xx}(\tau) = \frac{R_{yy}(\tau)}{R_{gg}(\tau)} \quad . \qquad (1.3)$$

By eqn.(1.3) computing N + 1 covariances, consistent estimates can be obtained as $\hat{R}_{xx}(\tau)$ converges to the true value as $N \to \infty$ (McGiffin and Murthy, 1980), then using eqn.(1.2) the spectral density of x(t) can be estimated.

The main drawback of this method is that different covariances are computed with different accuracy since the variance of the estimate $\hat{R}_{xx}(\tau)$ increases with increasing τ, and for a given τ it decreases with increasing value of the denominator of eqn.(1.3) for $\hat{R}_{xx}(\tau)$, thus for a given sequence with a particular pattern of missing observations, determining the optimal τ's to be used in computing $\hat{R}_{xx}(\tau)$ is a difficult problem and has not yet been studied (McGiffin and Murthy, 1980). Another disadvantage of this method is that the estimate is not statistically efficient (McGiffin and Murthy, 1980).

An alternative approach termed the periodogram method, denoted PM, (Harris, 1987) is based directly on the measured power spectrum $S_{yy}(\Omega)$. This is defined by

$$\hat{S}_{xx}(\Omega) = \frac{N}{N - M} S_{yy}(\Omega)$$

where (N − M) is the number of non−zero terms in y(t).

In general, both the above estimators are sub−optimal, as discussed later in this paper.

The new estimator of the property θ_x (for example the power spectral density) of the process giving the sequence x, is derived from the estimate of θ_y from the sequence y according to

$$\hat{\theta}_x = k\,\theta_y$$

where k is a frequency−dependent factor chosen to satisfy a criterion such as zero bias or minimum mean square error. This real gain factor k depends on the true value of θ_x. A recursive method was introduced to determine its value in the authors' previous work (Douce and Zhu, 1988), in which simulation studies showed that the recursion introduced converges rapidly for the wide range of examples considered. A direct method will be

presented to obtain the same results with significant reduction of computing time in this paper, accompanied by revised criteria for the specification of the gain factor k.

2. Estimator for Fourier Transform

The Fourier transform X is conveniently estimated from the observed data y using the MSE criterion. Writing $X = (X_R + j X_I)$ the estimate is $\hat{X} = k_1 Y_R + k_2 j Y_I$ where k_1 and k_2 are chosen to minimise

$$E [(X - \hat{X}) (X - \hat{X})^*] .$$

Differentiating with respect to k_1 and k_2, and setting the differentials to zero gives

$$k_1 = \frac{E[X_R Y_R]}{Y_R^2} \quad \text{and} \quad k_2 = \frac{E[X_I Y_I]}{Y_I^2} \tag{2.1}$$

The appendix derives expressions for k_1 and k_2 on the two assumptions, used throughout the paper:

(a) The missing points are separated such that the cross-correlation between values at missing positions may be neglected, and

(b) The missing points are not too near the end of the record.

$$k_1 = 1 + \left[\frac{Y_R^2 + Y_I^2 - (N-M)\sigma^2}{(N - 2M) Y_R^2} \right] \Sigma \cos^2 m\Omega$$

where the summation is over the missing points. A similar expression holds for k_2, with Y_R^2 replaced by Y_I^2 and $\cos^2 m\Omega$ by $\sin^2 m\Omega$.

3. Estimator for PSD

The reason for the introduction of two gains in the Fourier transform estimator is that there are two independent variables, the real part and imaginary part, in the criterion. However, the PSD estimator is a real variable, hence just one gain k_3 is required. First, the UNB estimator is developed to satisfy

$$E[(X_R^2 + X_I^2) - k_3(Y_R^2 + Y_I^2)] = 0 .$$

Using the results developed in the Appendix, the gain k_3 is

$$k_3 = \frac{E[X_R^2 + X_I^2]}{Y_R^2 + Y_I^2} = \frac{N}{N - 2M} \left[1 - \frac{M\sigma^2}{Y_R^2 + Y_I^2} \right] .$$

Secondly, the MSE estimator is developed to minimise

$$E\{[(X_R^2 + X_I^2) - k_3(Y_R^2 + Y_I^2)]^2\} .$$

Differentiating with respect to k_3 and setting the result to zero gives

$$k_3 = \frac{E[(X_R^2 + X_I^2)(Y_R^2 + Y_I^2)]}{(Y_R^2 + Y_I^2)^2} .$$

Evaluation of this expression involves the fourth order moments of the normal zero mean random processes X_I, X_R, etc. The general result

$$E[A^2B^2] = E[A^2].E[B^2] + 2(E[AB])^2$$

is used extensively. The method is outlined in the Appendix. In terms of the known second moments, the result is

$$k_3 = \frac{(E[X_R^2]+E[X_I^2])(Y_R^2+Y_I^2)+2((E[X_RY_R])^2+(E[X_IY_I])^2+(E[X_RY_I])^2+(E[X_IY_R])^2)}{3\,[Y^2+Y^2]^2} .$$

4. A Property of the PSD Estimation

The estimators developed in this paper give an estimated power spectrum of magnitude proportional to the measured power spectrum at the same frequency. However, missing points also give rise to spectral components at frequencies differing from the components present in the original time series. In particular, periodic missing points lead to a spurious component in the measured power spectrum which is periodic with respect to frequency.

One qualitative explanation of this phenomenon considers the periodic missing points (of period P points) as a sampled version of the original sequence x(i). This sampled sequence is subtracted from x(i) to produce the observed sequence y(i). The sampled sequence, sampled at the low frequency ($^1/_P$), introduces aliasing of the original signal, so that the original term at zero frequency produces alias terms at the frequencies $^1/_P$, $^2/_P$, etc. This is a periodic phenomenon, giving a repetitive power spectrum, repeating over ($^N/_P$) harmonic frequencies. This is clearly seen in figure 2, at high frequencies where the true power spectral density is small.

5. Frequency Response Estimation

The importance of the frequency response characteristics is well known for system identification and controller design. The essence of frequency response estimation is the estimation of the cross−spectrum between input and output and input auto−spectrum of a system. This part studies the estimation of the frequency response function from input and output data of a system in which there exists missing points at the output. Formally speaking, as far as authors know, there had not been any published result until the authors' publication (Douce and Zhu, 1988) even though one can naturally borrow ideas and techniques from power spectral density estimation with missing data.

The definition of the frequency response estimation is given by, without missing data,

$$G = \bar{S}_{ux}/\bar{S}_{uu}$$

where S_{ux} is the measured cross−spectrum between input and output of the system and S_{uu} is the measured auto−spectrum of the system input.

With missing points, the cross spectrum has to be computed using an estimated value for X based on Y. The estimate is selected to give an unbiased estimate of the cross spectrum or to give a minimum mean squared error in the resulting cross−spectrum estimate.

Considering first the UNB estimator, the parameters k_4 and k_5 are chosen to separately ensure that real and imaginary parts of the error in the cross spectrum estimate are zero, that is

$$E[Re(U^*X) - k_4Re(U^*Y)] = 0 = E[(U_RX_R+U_IX_I) - k_4(U_RY_R+U_IY_I)]$$

$$E[Im(U^*X) - k_5Im(U^*Y)] = 0 = E[(U_RX_I-U_IX_R) - k_5(U_RY_I-U_IY_R)] .$$

Define f_3 from

$$Nf_3 = E[U_RX_R + U_IX_I] = N \Sigma E[u_ix_j] \cos(i-j).$$

Then $E[U_R Y_R + U_I Y_I] = E[U_R(X_R - M_R) + U_I(X_I - M_I)]$

$= Nf_3 - \Sigma\Sigma\, E[u_i m_j].(\cos i\Omega \cos j\Omega + \sin i\Omega \sin j\Omega)$

$= Nf_3 - \Sigma\Sigma\, E[u_i m_j].\cos(i-j)\Omega$

$= Nf_3 - Mf_3.$

By substitution, $k_4 = \frac{N}{N - M}$, and similarly for k_5.

This important result demonstrates that the PM, derived on an ad-hoc basis for spectral analysis, is, when the stated assumptions are satisfied, an unbiased estimate of the cross-spectrum. It therefore leads to unbiased estimates of the system frequency response.

Second, consider the MSE estimator, in which the function

$$E[(U^*X - U^*\hat{X})\ (U^*X - U^*\hat{X})^*]$$

with $\hat{X}_R = k_4 Y_R$ and $\hat{X}_I = k_5 Y_I$ is minimised with respect to k_4 and k_5.

Differentiating with respect to k_4 and k_5 leads to

$$k_4 = \frac{E[U_R^2 X_R Y_R + U_I^2 X_R Y_R]}{U_R^2 Y_R^2 + U_I^2 Y_R^2}$$

and

$$k_5 = \frac{E[U_R^2 X_I Y_I + U_I^2 X_I Y_I]}{U_R^2 Y_I^2 + U_I^2 Y_I^2}$$

As previously noted, these higher order moments can be expressed in terms of second order moments, and the details proceed as before. Lengthy expressions result, which are not reproduced here.

6. Experiment Results

Two systems excited by Gaussian white noise have been studied to compare the traditional and new methods for spectral analysis. The experiments have been completed to examine the effect of smoothing over several blocks and over adjacent frequencies on the resulting estimates.

In all experiments a block length of 128 points has been used. When the record contains more than one block, spectral estimates are obtained by smoothing over all blocks. When only one block is involved, spectral estimates are smoothed over adjacent frequencies, using two frequencies for system 1 (defined below) and four for system 2. When analysing the data using correlation function techniques, the whole record is used, with a Bartlett window.

Three quantitative measures of performance have been used for the comparison. The first one is the sum over all frequencies of the error squared between the true spectrum with no missing points and the estimated spectrum with missing points. This is a measure of bias and is given by

$$e1 = \frac{\sum_{\Omega=0}^{\pi} (E(A) - E(\hat{A}))^2}{N}\ .$$

The second one is the sum over all frequencies of the MSE, that is

$$e2 = \frac{\sum_{\Omega=0}^{\pi} E(A-\hat{A})^2}{N} .$$

The third one is a linear regression of the spectrum on the estimated again over all frequencies, which is given by

$$fit = \frac{\sum_{\Omega=0}^{\pi} E(A)E(\hat{A})}{\sum_{\Omega=0}^{\pi} (E(\hat{A}))^2}$$

The first order system (SYS 1) considered is

$$x_k = 0.9\, x_{k-1} + u_k$$

in which u_k is a white noise normally distributed signal of unity variance and zero mean value.

The fourth order system (SYS 2) is the process described by Harris (1987), giving an output produced by passing the previously defined white noise through the process described by

$$y_{k-3} = 1.7143\, y_{k-4} - 0.9048\, y_{k-5} + u_k$$

$$y_k = 1.0732\, y_{k-1} - 0.9512\, y_{k-2} + y_{k-3} .$$

This produces a power spectrum with two pronounced resonances.

A wide range of experiments have been undertaken to test the improvements of the proposed methods. In this printed paper, a small selection of typical results is presented.

Table 1 shows the results for power spectrum estimation with 10% missing points, regularly spaced. The columns list the values of the criteria defined above for four techniques. The rows are for increasing lengths of record for the two systems.

In all experiments, it is noted that

(a) The UNB method has the smallest measured bias (e1), and the fit is closest to unity, and

(b) The MSE has, as predicted, the smallest measured mean squared error (e2).

(c) Both methods are, on all criteria presented, superior to the traditional CM and PM techniques.

In frequency response estimation, it has been shown that the PM method, previously used only for spectral analysis, corresponds to the UNB technique, and hence only three techniques have been compared. Table 2 demonstrates the numerical comparison in tests with 20% missing points. The new methods demonstrate their expected advantages, with both greatly superior to the CM method.

For system 2, Figure 1 shows the various estimates, using a record length of 1000 blocks. It is notable how the CM method is unique in introducing very large errors at low frequencies.

Figure 2 refers to the same experiment, in which the error squared for each method is plotted against frequency. This demonstrates the previously mentioned large errors of the CM, and also shows clearly the periodic errors due to periodic missing points.

7. Conclusions

This paper presents a relatively simple method for handling data records with missing points, useful for the estimation of Fourier transforms, power spectral densities and frequency response functions. Unlike previous techniques, the new method demonstrates the importance of the position of missing points in the estimation process.

The effect of the periodic occurrence of missing points, leading to a periodic error spectrum, has been predicted and observed in experiments.

Extensive simulation studies show the superiority of the method compared with two previous techniques.

8. Bibliography

Bard, A., Nonlinear parameter estimation, Academic Press, New York and London, 1974.

Douce, J.L. and Zhu, Q.M. "Spectral analysis of records with missing data", Preprints of IFAC Symp. Syst. Ident. Param. Est., Vol.3, pp.1359−1363, Beijing, 1988.

Harris, R.W., "Spectra from data with missing values", Mechanical Systems and Signal Processing, Vol.1(1), pp.94−104, 1987.

Jarrett, R.G., "The analysis of designed experiments with missing observations", Appl. Statist., Vol.27, pp.38−46, 1978.

John, J.A. and P. Prescott, "Estimating missing values in experiments", Appl. Statist., Vol.24, pp.190−192, 1975.

Jones, R.H., "Spectral analysis with regularly missed observations", Ann. Math. Stat., Vol.32, pp.455−461, 1962.

McGiffin, P.B. and Murthy, D.N.P. "Parameter estimation for auto−regressive systems with missing observations", Int. J. Systems Sci., Vol.11, pp.1021−1034, 1980.

McGiffin, P.B. and Murthy, D.N.P. "Parameter estimation for auto−regressive systems with missing observations − Part 2", Int. J. Systems Sci., Vol.12, pp.657−663, 1981.

Nahi, N., "Optimal recursive estimation with uncertain observation", IEEE Trans. Inf. Theory, Vol.IT−15, pp.457−462, 1969.

Parzen, E., "On spectral analysis with missing observations and amplitude modulation", Indian Journal of Statistics, Vol.A25, pp.383−392, 1963.

Roberts, J.B. and Gaster, M. "On the estimation of spectra of randomly sampled signals: A method reducing variability", Proc. R. Sco. Lond., Vol.A371, pp.235−258, 1980.

Smith, P.L., "The use of analysis of covariance to analyse data from designed experiments with missing or mixed−up values", Appl. Statist., Vol.30, pp.1−8, 1981.

APPENDIX

For convenience, define

$$E[X_R^2 + X_I^2] \approx k_3 (Y_R^2 + Y_I^2) = Nf_1 \qquad \text{(A1.1)}$$

and let $X_R = Y_R + M_R$

where M_R is the real part of the transform of the missing points.

From the definition,

$$X_R^2 = \Sigma\Sigma\, x_i\, x_j \cos i\Omega \cos j\Omega \ .$$

Throughout the paper, summations with respect to i and j are taken over the whole record, i, j = 1 to N. Summations with respect to m involve only the M missing points.

$$X_R^2 = \Sigma \cos^2 j\Omega \ \Sigma\, x_i x_{i-j} \cos(i-j)\Omega - \tfrac{1}{2} \Sigma \sin 2j\Omega \ \Sigma\, x_i x_{i-j} \sin(i-j)\Omega \ .$$

Taking expectations, the second term is negligible, since $E[x_i x_{i-j}]$ is even, whilst $\sin(i-j)$ is odd. Similarly

$$E[X_I^2] \approx \Sigma \sin^2 j\Omega \ \Sigma\, E[x_i x_{i-j}] \cos(i-j)\Omega$$

$$\therefore \quad f_1 = \Sigma\, E[x_i x_{i-j}] \cos(i-j)\Omega \ .$$

Also, $E[X_R^2] = Y_R^2 + E[2X_R M_R - M_R^2]$

with $E[X_R M_R] = \Sigma\Sigma[x_i x_m] \cos i\Omega \cos m\Omega \approx f_1 \Sigma \cos^2 m\Omega \ .$

Similarly, assuming adequate separation between missing points,

$$E[M_R^2] \approx \sigma^2 \Sigma \cos^2 m\Omega .$$

Similar results hold for $E[X_I M_I]$ etc., with $\sin^2 m\Omega$ replacing $\cos^2 m\Omega$.

Combining the above equations

$$Nf_1 = \frac{Nf_1}{k_3} + 2Mf_1 - M\sigma^2 \qquad \text{or} \qquad k_3 = \frac{Nf_1}{f_1(N-2M) + M\sigma^2} \ .$$

Substituting for f_1 from (A1.1)

$$k_3 = \frac{N}{N-2M} \left[1 - \frac{M\sigma^2}{Y_R^2 + Y_I^2} \right] \ .$$

To determine k_1 and k_2, note that

$$E[X_R Y_R] = Y_R^2 + E[X_R M_R] - E[M_R^2] \ .$$

Substituting these terms into equation (2.1) elimination of f_1 and k_3 gives the required result.

data length (10% missing)		PM			CM			UNB			MSE		
		e1	e2	fit	e1	e2	fit	e1	e2	fit	e1	e2	fit
sys1	128	50.39	105.9	1.113	46.88	83.01	1.093	12.01	22.98	1.069	13.17	21.12	1.077
sys2	128	2600	5942	1.266	2507	5020	1.085	1871	4191	1.041	2009	4001	1.079
sys1	128*5	5.031	10.36	1.108	4.912	9.957	1.088	1.934	8.248	1.064	2.852	8.201	1.084
sys2	128*5	553.0	1261	1.102	370.1	1220	1.062	184.3	1032	1.039	317.9	1022	1.045
sys1	1280	3.679	7.847	1.097	3.552	6.594	1.070	0.993	3.760	1.010	1.687	3.530	1.014
sys2	1280	284.2	1233	1.064	190.2	1187	1.040	167.4	945.1	0.980	170.2	900.8	0.971
sys1	12800	2.7982	5.982	1.065	0.941	4.102	1.025	0.137	2.085	1.007	0.155	1.751	1.012
sys2	12800	238.8	1197	1.038	166.7	900.9	1.031	21.40	709.7	1.001	34.01	668.1	1.010

Table 1 Errors in spectrum estimation

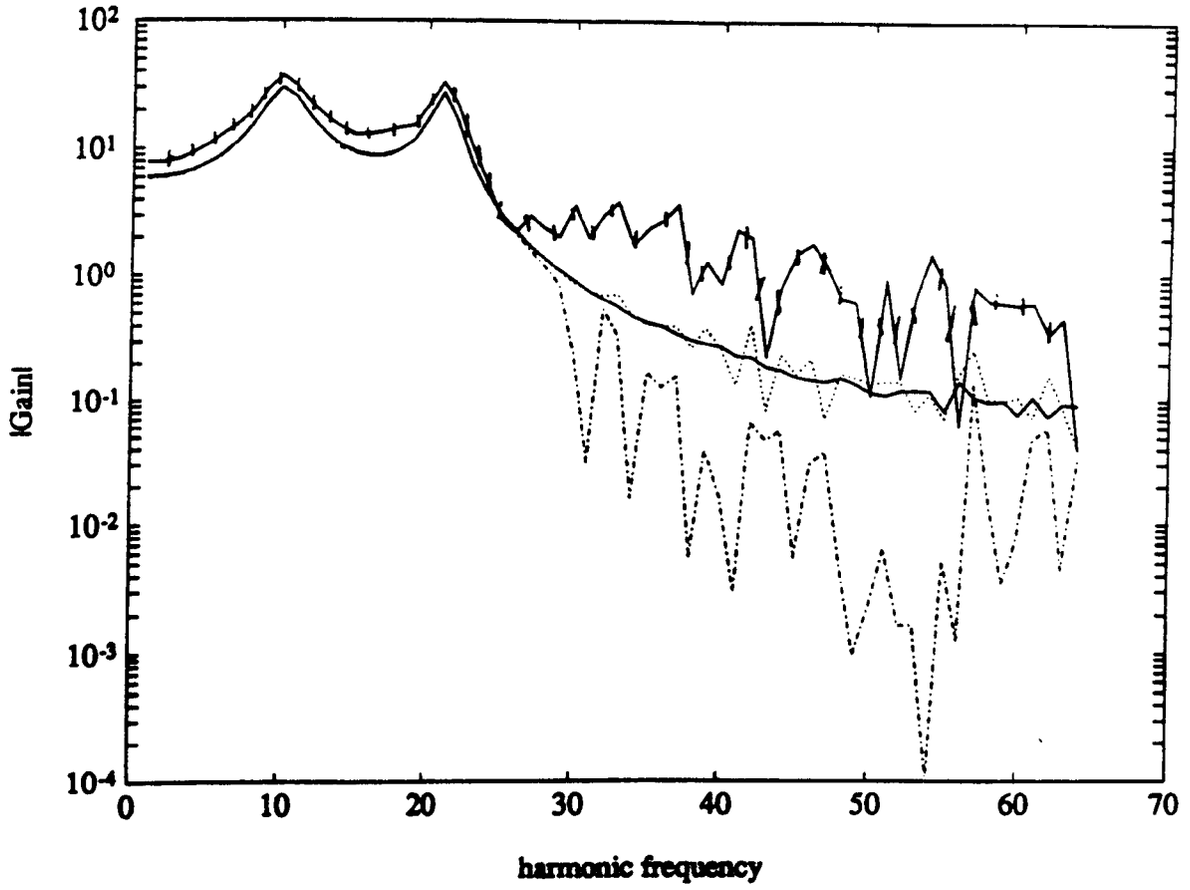


Figure 1. **Frequency response estimates.**

——— with no missing points
–/–/–/–/– covariance method
. . . . . minimum bias
–.–.–.–.– minimum mean square error

data length (20% missing)		CM			UNB			MSE		
		e1	e2	fit	e1	e2	fit	e1	e2	fit
sys1	128	18.17	52.19	0.323	2.872	5.137	0.460	2.929	4.997	0.456
sys2	128	183.4	372.4	0.242	14.43	19.28	0.604	15.84	18.55	0.584
sys1	128*5	7.694	20.66	0.707	0.275	1.417	0.946	0.278	1.175	0.945
sys2	128*5	88.67	143.2	0.647	4.306	19.47	0.969	4.542	18.15	0.963
sys1	1280	5.404	12.93	0.796	0.120	1.322	0.979	0.131	1.311	0.976
sys2	1280	18.30	41.45	0.723	1.417	8.267	0.974	1.646	5.616	0.971
sys1	12800	3.611	8.317	0.797	0.026	1.177	1.002	0.028	1.128	1.003
sys2	12800	14.29	25.71	0.7405	0.038	7.625	1.003	0.057	5.007	1.004

Table 2 Errors in frequency response estimation

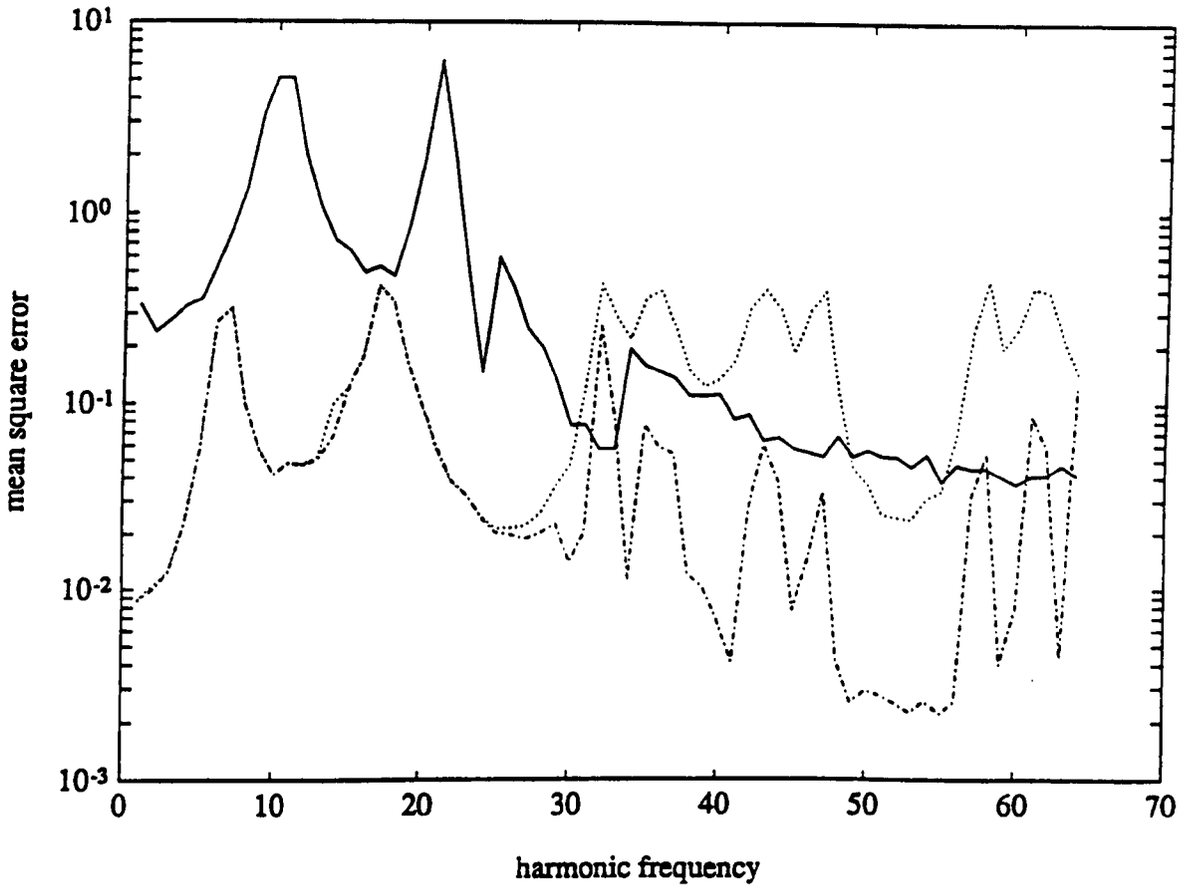


Figure 2. Error squared as a function of harmonic.

——— covariance method

. . . . . minimum bias

-.-.-.-.- minimum mean square error

S5.2 Computer programs

S5.2.1 Computer and program language:

The computers used for the work in this thesis were the Prime computer installation and the Sun Workstation network in the Engineering Department, University of Warwick.

The programs originally written in Fortran 77 were translated into MATLAB. The final version of the all programs was written in MATLAB.

S5.2.2 What is MATLAB?

MATLAB (Moler, Little and Bangert 1987) is an interactive program to help with scientific and engineering numeric calculations. The name MATLAB stands for matrix laboratory. Originally written in Fortran for mainframe computers, it provides easy access to matrix software developed by the LINPACK and EISPACK projects. Together, LINPACK and EISPACK represent the state of the art in software for matrix computation.

MATLAB is an interactive system whose basic data element is a matrix that does not require dimensioning. This allows solution of many numeric problems in a fraction of the time it would take to write a program in a language like Fortran, Basic, or C. Furthermore, problem solutions are expressed in MATLAB almost exactly as they are written mathematically.

MATLAB has evolved over more than half a decade with input from many users. In university environments it has become the standard instructional tool used in introductory courses in applied linear algebra, as well as advanced courses in other areas. In industrial settings, MATLAB is used for research, and to solve practical engineering and mathematical problems. Typical uses include general purpose numeric computation, algorithm prototyping, and solving the special problems with matrix formulations that arise in disciplines like automatic control theory, statistics, and digital signal processing (time-series analysis).

The highly optimized, second generation MATLAB that runs on IBM and other MS-DOS compatible personal computers is called PC-MATLAB. On larger computers, like Sun Workstations and VAX computers, the modern version of MATLAB is called PRO-MATLAB. On the Macintosh, it is MacMATLAB. Entirely written in the C language, MATLAB is a complete "integrated" system, including graphics, programmble macros, IEEE arithmetic, a fast interpreter, and many analytical commands.

S5.2.3 MATLAB files or M-files

An M-file consists of a sequence of normal MATLAB statements, possibly including references to other M-files. One use of M-file is automate long sequences of commands. Such file are called **script files** or just **scripts.** A second type of M-file provides extensibility to MATLAB. Called **function files** or just **functions,** they allow new functions to be added to the existing functions. Much of the power of MATLAB derives from this ability to create new functions that solve user-specific problems.

S5.2.4 Interactive simulation experiment

All the simulation experiments follow the same interactive structure as shown in Fig. 5.2.1. The tasks in either script or function are explained at the beginning and suitable comments are given for some critical lines.

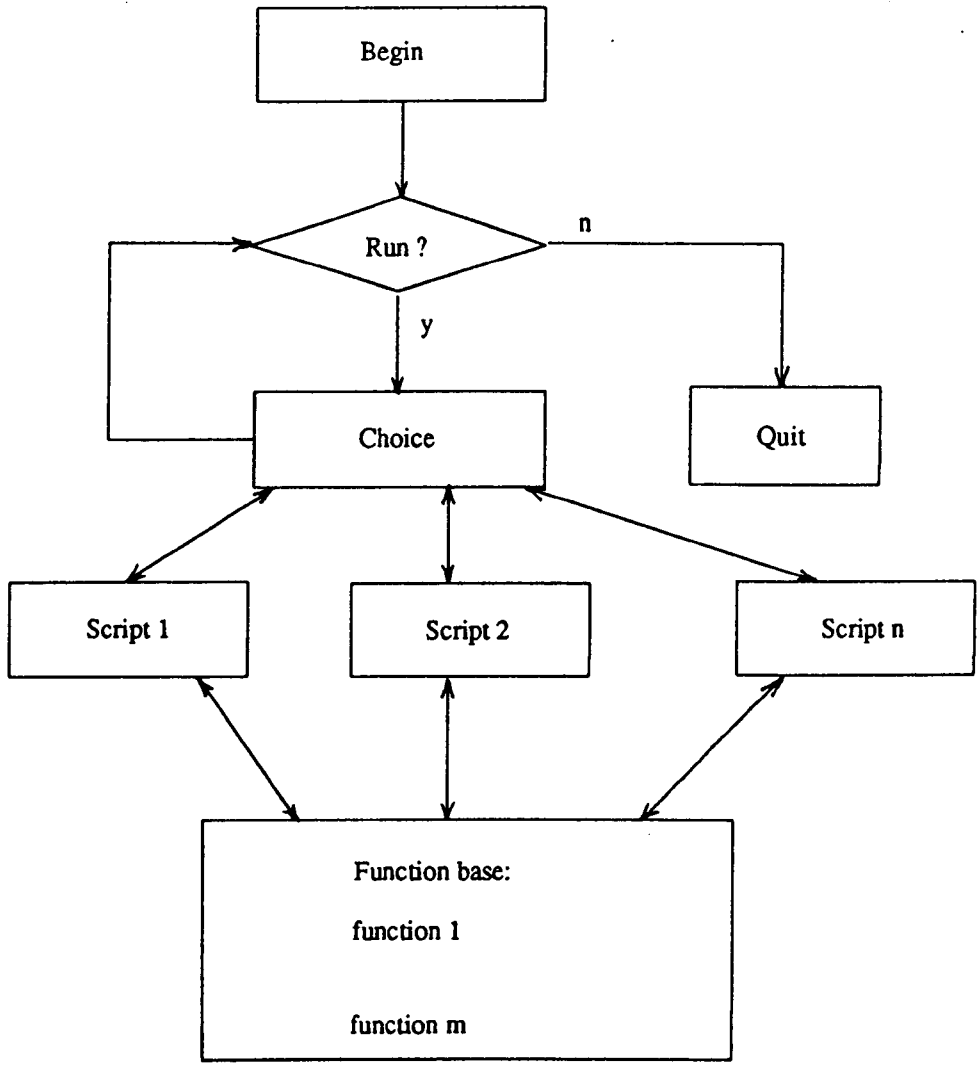


Figure 5.2.1 Flowe chart for interactive simulation

S5.2.5 List of computer programs

Section one

Scripts:	Comments
miss1:	PSD and frequency response estimations by PM, CM, UNB(recursive), and MSE(recursive).
miss2:	PSD and frequency response estimations by PM, IM, UNB(direct), and MSE(direct).

Section two

Scripts:

nondet:	system identification by amplitude quantisation (aq).
testn1:	parameter estimation of system one by aq.
testn2:	parameter estimation of system two by aq.
nident:	parameter estimation by VWLS algorithm.
jump:	frequency response of a nonlinear system consisting of a second order linear dynamic and a saturation static.
jmp:	frequency response prediction of the system given in the program "jump" by VWLS algorithm.

Functions:

ingen:	generate inputs.
sysout:	generate outputs from given inputs and plants.
model:	generate model responses.
fit:	parameter estimatior with VWLS.
jumod:	jump resonant model.

jfit2: parameter estimator of jump resonant model.

Section three

Scripts:

nlgpc: nonlinear general predictive controller (gpc).

nldbc: nonlinear deadbeat controller (dbc).

Functions:

plant: plants.

paraest: parameter estimator.

root: root solver.

diopeq: Diophantine equation solver.

condes: gpc designer.

dbc1: indirect dbc designer.

dbc2: direct dbc designer.

```
format compact
clc;clg; % clear windows
% the script started 18/10/1988.
% this is used to analyses Power Spectrum Density (PSD)
%      with missing data in a time series and to
%      estimate FREquency reSPonse (FRESP)
%      with missing data in output series.
% two sorts of estimators, UNBised and Mean Square Error,
%      are built up to compare with traditional ones such
%      as Periodogram Method and Covariance Method .
% square bias, MSE, and linear FIT are calculated as
%      measurements for the comparison.
%
% x,  input
% y,  ouput without missing point
% ym, ouput with missing points
%
% psdy,  psd from y
% psdum, -------- ym by PM
% psdyc, -------------- CM
% psdyi, -------------- UNB
% psdyn, -------------- MSE
%
% fr1, an1, FRESP (amplitude and phase ) from y
% fr2, an2, ---- from ym by PM
% fr3, an3, ---------------- UNB
% fr4, an4, ---------------- MSE
% fr5, an5, ---------------- CM

load batmis1 % consideration of batch job
while 1
   clc;
   n=input('n>0 enter,n<=0 quit');
   if n<=0;break,end
   '1,initialisation'
   '2,generate input and output data fft, psd,weights'
   '3,FRESP,MSE,FIT calculation'
   '4,plots'
   i=input('your choice');
   if i==1
      clc;clg;
      istart=1;
      ifin=input('length of one block');
      nbl=input('no. of block');
      m=input('no. of missing points');
      misp=input('distr. of mp., unif.=1; arb.=def.');
      clear a
      for t=1:4
          a(t)=input('y(t)=a(1)*y(t-1)+ ...
          a(2)*y(t-2)+y(t-3),y3=a3*y4+a4*y5+x')
      end
      w=input('window for CM, dn.=0; Bartlett=1');
      wl=input('lag length of window');
      if nbl==1
         mv=input('1<order of moving aver. filter<5');
      end
```

```
      ps=input('open loop=1, closed1=2, closed2=3');
      clear wf wq wind
      for j=1:wl
          for i=1:ifin
              wf(i,j)=cos(2*pi*(i-1)*(j-1)/ifin);
              wq(i,j)=sin(2*pi*(i-1)*(j-1)/ifin);
          end
          wind(j,1)=1-(j-1)*w/wl;
      end
   elseif i==2
      clc;clg;
      pl=input('batch job? y=1 n=default');
      if pl~=1
         k=0;clear mp; % locate missing points
         yp=ones(ifin,1);
      for t=1:m
          if misp==1
             k=k+floor(ifin/m);
             mp(t)=k;
          else
             mp(t)=input('missing position= ');
          end
          yp(mp(t),1)=0;
      end
      rm=xcorr(yp); % Parzen weigth for CM
      clear cw sw nl f wl ;
      cw(ifin,1)=0;sw(ifin,1)=0;cs=cw;
      for t=1:ifin
          wl(t,1)=ifin/(ifin-m);% traditional weight
                                % as initial value
                                % of new weight
          nl(t,1)=ifin;
          f(t,1)=(t-1)/ifin;
          for j=1:m
              c1=cos(2*pi*mp(j)*(t-1)/ifin);
              s1=sin(2*pi*mp(j)*(t-1)/ifin);
              cw(t,1)=cw(t,1)+c1^2;
              sw(t,1)=sw(t,1)+s1^2;
              cs(t,1)=cs(t,1)+c1*s1;
          end
      end
      w3=wl; w2=wl;w4=wl;w5=wl; g3=wl; g4=wl;g5=wl;
      g1=wl;g2=wl;
      cw1=nl/2-2*cw;cw2=nl/2-cw;sw1=nl/2-2*sw;sw2=nl/2-sw;
      sw3=nl/2+sw;
      rand('normal')
         clear u ucm x xcm xu xx y y1 yi ym ys ycm ycp xcm;
               y1(ifin+5,1)=0;yh=y1;
         clear tr1 tr2 tr3 tr4 fr1 fr2 fr3 fr4 ;
         clear psdx psdy psdym psdyi psdyn f1 f2 f3 f4 ;
         tr1(ifin,1)=0;tr2=tr1;tr3=tr1;tr4=tr1;
         fr1=tr1;fr2=tr1;fr3=tr1;fr4=tr1;fr5=tr1;
         an1=tr1;an2=tr1;an3=tr1;an4=tr1;an5=tr1;
         psdx=tr1; psdy=tr1; psdym=tr1; psdyi=tr1;
         psdyn=tr1; psdyc=tr1;
         psdxc=tr1; temp=[an1 an2 an3]; xu=tr1;
```

```
cv=0;cvx=0;stx=0;sty=0;pyc=tr1;amp5=tr1;
csp=tr1;qsp=tr1;
f1=tr1;f2=f1;f3=f1;f4=f1;gf1=f1;gf2=f1;gf3=f1;
gf4=f1;
pmse1=f1;pmse2=f1;pmse3=f1;pmse4=f1;
fmse1=f1;fmse2=f1;fmse3=f1;fmse4=f1;
phmse1=f1;phmse2=f1;phmse3=f1;phmse4=f1;
for t1=1:nbl
if ps==1
    for t=6:ifin+5
        x(t-5,1)=rand;
        u(t-5,1)=x(t-5,1);
 y1(t-3,1)=a(3)*y1(t-4,1)+a(4)*y1(t-5,1)+x(t-5,1);
 y1(t,1)=a(1)*y1(t-1,1)+a(2)*y1(t-2,1)+y1(t-3,1);
        y(t-5,1)=y1(t,1);  % without missing point
        ym(t-5,1)=y1(t,1); % traditional
        yi(t-5,1)=y1(t,1); % interpolation
    end
    for t=1:m
        ys(t,1)=ym(mp(t),1); % real missing values
        ym(mp(t),1)=0;
        yi(mp(t),1)=(yi(mp(t)-1,1)+yi(mp(t)+1,1))/2;
        ys(t,2)=yi(mp(t),1); % interpolate values
    end
elseif ps==2    % feedback without missing point
    for t=6:ifin+5
        x(t-5,1)=rand ;
        yh1(t-5,1)=yh(t,1);
        u(t-5,1)=x(t-5,1)-yh(t,1);
  y1(t-3,1)=a(3)*y1(t-4,1)+a(4)*y1(t-5,1)+u(t-5,1);
  y1(t,1)=a(1)*y1(t-1,1)+a(2)*y1(t-2,1)+y1(t-3,1);
        y(t-5,1)=y1(t,1); % without missing point
                          % with missing point
        ym(t-5,1)=y1(t,1)*yp(t-5,1);
        yh(t+1,1)=.2*yh(t,1)+y1(t,1);
    end
elseif ps==3    % feedback with missing point input
    for t=6:ifin+5
        x(t-5,1)=rand ;
        yh1(t-5,1)=yh(t,1);
        u(t-5,1)=x(t-5,1)-yh(t,1);
   y1(t-3,1)=a(3)*y1(t-4,1)+a(4)*y1(t-5,1)+u(t-5,1);
   y1(t,1)=a(1)*y1(t-1,1)+a(2)*y1(t-2,1)+y1(t-3,1);
        y(t-5,1)=y1(t,1); % without missing point
                            % with missing point
        ym(t-5,1)=y1(t,1)*yp(t-5,1);
                    %feedback with missing point
        yh(t+1,1)=.0*yh(t,1)+ym(t-5,1);
    end
end
    tempx=fft(x); % input x's fft
    rex=real(tempx);
    imx=imag(tempx);
    tempu=fft(u);
    tempy=fft(y); % output y's fft
    rey=real(tempy);
```

```
            imy=imag(tempy);
            tempym=fft(ym); % output ym's fft
            reym=real(tempym);
            imym=imag(tempym);
            tempyi=fft(yi); % output yi's fft

            px=abs(tempx).^2/ifin; % psd estimations
            py=abs(tempy).^2/ifin;
            pym=abs(tempym).^2/(ifin-m);
            pyi=g3.*abs(tempym).^2/ifin;
            pyn=w3.*pym*(ifin-m)/ifin;

            psdx=(t1-1)*psdx/t1+px/t1;
            psdy=(t1-1)*psdy/t1+py/t1;
            psdym=(t1-1)*psdym/t1+pym/t1;
            psdyi=(t1-1)*psdyi/t1+pyi/t1;
            psdyn=(t1-1)*psdyn/t1+pyn/t1;
                                        % fresp estimations
            tr1=(tempy.*conj(tempx))./ifin;
            fr1=(t1-1)*fr1/t1+tr1/t1;

            tr2=(tempym.*conj(tempx))/(ifin-m);
            fr2=(t1-1)*fr2/t1+tr2/t1;

            t3=g4.*reym+sqrt(-1)*g5.*imym;
            tr3=t3.*conj(tempx)/ifin;
            fr3=(t1-1)*fr3/t1+tr3/t1;

            t4=w4.*reym+sqrt(-1)*w5.*imym;
            tr4=t4.*conj(tempx)/ifin;
            fr4=(t1-1)*fr4/t1+tr4/t1;

            if ps==3
               pss=1; % weights for case 3 by cm
            else
                                          %for case 1,2 by cm
               xu=(t1-1)*xu/t1+conj(tempx).*tempu/t1;
            end
            cv1=ifin*cov(ym)/(ifin-m);
            cv=(t1-1)*cv/t1+cv1/t1;

            rif=2*(w1.*reym).*(w2.*imym)/ifin;
            f2=(t1-1)*f2/t1+rif/t1;

            f1=psdyn; yy=psdyn*ifin;yy(1,1)=yy(1,1)/2;

            cf3=(rex.*w4.*reym+imx.*w5.*imym)/ifin;
            f3=(t1-1)*f3/t1+cf3/t1;

            cf4=(imx.*w4.*reym-rex.*w5.*imym)/ifin;
            f4=(t1-1)*f4/t1+cf4/t1; ,

        gf1=psdyi;
gf2=(t1-1)*gf2/t1+2*(g1.*reym).*(g2.*imym)/(ifin*t1);
gf3=(t1-1)*gf3/t1+(rex.*g4.*reym+imx.*g5.*imym)/(ifin*t1);
gf4=(t1-1)*gf4/t1+(imx.*g4.*reym-rex.*g5.*imym)/(ifin*t1);
```

```
        if nbl==1  % moving average filter for one block
           tw=1;
           for j=1:ifin-mv+1
               yy(j,1)=tw*sum(yy(j:j+mv-1,1))/mv;
               f1(j,1)=tw*sum(f1(j:j+mv-1,1))/mv;
               f2(j,1)=tw*sum(f2(j:j+mv-1,1))/mv;
               f3(j,1)=tw*sum(f3(j:j+mv-1,1))/mv;
               f4(j,1)=tw*sum(f4(j:j+mv-1,1))/mv;
               gf1(j,1)=tw*sum(gf1(j:j+mv-1,1))/mv;
               gf2(j,1)=tw*sum(gf2(j:j+mv-1,1))/mv;
               gf3(j,1)=tw*sum(gf3(j:j+mv-1,1))/mv;
               gf4(j,1)=tw*sum(gf4(j:j+mv-1,1))/mv;
           end
        end

                                  % new weight (mse) for psd
            ymr=f1.*cw1+cv*cw+2*f2.*cs;
            ymi=f1.*sw1+cv*sw-2*f2.*cs;
            ymi(1,1)=0; ymr(1,1)=f1(1,1)*(ifin/2-m)+cv*m/2;
            yrymr=f1.*cw2+f2.*cs; yiymi=f1.*sw2-f2.*cs;
            yiymi(1,1)=0; yrymr(1,1)=f1(1,1)*(ifin/2-m/2);
            yrymi=f2.*sw3-f1.*cs; yiymr=f2.*cw2-f1.*cs;
            yrymi(1,1)=0;yiymr(1,1)=0;
            ymrymi=f2.*(ifin/2-cw+sw)-2*f1.*cs+cv*cs;
            ymrymi(1,1)=0;
            c3=4*yrymi.^2;

    w3=yy.*(ymr+ymi)+2*(yrymr.^2+yiymi.^2+yrymi.^2+yiymr.^2);
            w7=3*(ymr+ymi).^2-4*ymr.*ymi+4*ymrymi.^2;
            w3=w3./w7;  w3(ifin/2,1)=w3(1,1);

                                   % new weight (unb) for psd
            g3=ifin*gf1./(gf1*(ifin-2*m)+cv*m);
                     % new weight (mse) for Fourie transform
            rn=(f1.*cw2+f2.*cs);rd=(f1.*cw1+cv*cw+2*f2.*cs);
            in=(f1.*sw2-f2.*cs);
            id=(f1.*sw1+cv*sw-2*f2.*cs+.0001);
            w1=rn./rd;w2=in./id;

                      % new weight(mse) for fresp real part
            uryr=ifin*f3/2;
            urymr=f3.*cw2+f4.*cs;
            urymr(1,1)=f3(1,1)*(ifin/2-m/2);
            uiyr=ifin*f4/2;
            uiymr=f4.*cw2-f3.*cs;
            c1=urymr.^2+uiymr.^2;

            w4=psdx.*rn+2*(uryr.*urymr+uiyr.*uiymr);
            w7=psdx.*rd+2*c1;
            w4=w4./w7; w4(ifin/2,1)=w4(1,1);

                     % new weight(mse) for fresp imag. part
            uryi=-ifin*f4/2;
            urymi=-f4.*sw2-f3.*cs;
            uiyi=ifin*f3/2;
```

```
            uiymi=f3.*sw2-f4.*cs;
            c2=urymi.^2+uiymi.^2;

            w5=psdx.*in+2*(uryi.*urymi+uiyi.*uiymi);
            w7=psdx.*id+2*c2;
            w7(1,1)=1; w7(ifin/2,1)=1;
            w5=w5./w7;
            w5(1,1)=1; w5(ifin/2,1)=1;

            rn=(gf1.*cw2+gf2.*cs);
            rd=(gf1.*cw1+cv*cw+2*gf2.*cs);
            in=(gf1.*sw2-gf2.*cs);
            id=(gf1.*sw1+cv*sw-2*gf2.*cs+.0001);
            g1=rn./rd;g2=in./id;

               % new weight(unb) for fresp real and imag. parts
            g4=ifin/(ifin-m);
            g5=g4;

            if nbl==1
               psdyi=g3.*abs(tempym).^2/ifin;
               psdyn=w3.*abs(tempym).^2/ifin;
           fr3=(g4.*reym+sqrt(-1)*g5.*imym).*conj(tempx)/ifin;
           fr4=(w4.*reym+sqrt(-1)*w5.*imym).*conj(tempx)/ifin;
            end

                                          % mse calculation
pmse1=(t1-1)*pmse1/t1+(py-pym).^2/t1;  % psd
pmse2=(t1-1)*pmse2/t1+(py-pyi).^2/t1;
pmse3=(t1-1)*pmse3/t1+(py-pyn).^2/t1;
pmse4=(t1-1)*pmse4/t1+(py-pyc).^2/t1;
                                          % amplitude (fresp)
fmse1=(t1-1)*fmse1/t1+(tr1-tr2).*conj(tr1-tr2)/t1;
fmse2=(t1-1)*fmse2/t1+(tr1-tr3).*conj(tr1-tr3)/t1;
fmse3=(t1-1)*fmse3/t1+(tr1-tr4).*conj(tr1-tr4)/t1;
fmse4=(t1-1)*fmse4/t1+(tr1-amp5).*conj(tr1-amp5)/t1;
                                                    % phase
phmse1=(t1-1)*phmse1/t1+(angle(tr1)-angle(tr2)).^2/t1;
phmse2=(t1-1)*phmse2/t1+(angle(tr1)-angle(tr3)).^2/t1;
phmse3=(t1-1)*phmse3/t1+(angle(tr1)-angle(tr4)).^2/t1;
phmse4=(t1-1)*phmse4/t1+(angle(tr1)-phase5).^2/t1;

            xcm((t1-1)*ifin+1:t1*ifin,1)=x;
            ucm((t1-1)*ifin+1:t1*ifin,1)=u;
            ycm((t1-1)*ifin+1:t1*ifin,1)=ym;
            ycp((t1-1)*ifin+1:t1*ifin,1)=yp;
         end

            rm=xcorr(ycp);
            xy=[xcm ycm];
            xuc=[xcm ucm];
            xx=[xcm xcm];
            wfq=[wf wq]; l=ifin*nbl;
            [pyc,temp]=frsp(l,wind,wfq,wl,rm,xy);
            psdyc=pyc;
            fr5=temp(:,1);an5=atan2(temp(:,3),temp(:,2));
```

```
            [pyc,temp]=frsp(1,wind,wfq,wl,rm,xuc);
            fxu=temp(:,1);axu=atan2(temp(:,3),temp(:,2));
            [pyc,temp]=frsp(1,wind,wfq,wl,rm,xx);
            fxx=temp(:,1);axx=atan2(temp(:,3),temp(:,2));

            i=sqrt(-1);
            fxu=fxu.*exp(i*axu);fxx=fxx.*exp(i*axx);
            fr5=fr5.*exp(i*an5);
      else
         save batmis1
      end

   elseif i==3
      clc; clg % fresp estimations, mse, fit

                                    % amplitude squared bise
            ef1=((fr1-fr2).*conj(fr1-fr2))/ifin;
            ef2=((fr1-fr3).*conj(fr1-fr3))/ifin;
            ef3=((fr1-fr4).*conj(fr1-fr4))/ifin;
            ef4=((fr1-fr5).*conj(fr1-fr5))/ifin;
            e1=sum(ef1), e2=sum(ef2), e3=sum(ef3), e4=sum(ef4)

                                    % fresp
      if ps==3
         fr1=fr1./(psdx-fr1);fr2=fr2./(psdx-fr2);
         fr3=fr3./(psdx-fr3);fr4=fr4./(psdx-fr4);
         fr5=fr5./(psdx-fr5);
      else
         fr1=fr1./xu;fr2=fr2./xu;
         fr3=fr3./xu;fr4=fr4./xu;
         fr5=fr5./xu;
      end
                                    % phase
      an1=angle(fr1);an2=angle(fr2);an3=angle(fr3);
      an4=angle(fr4);an5=angle(fr5);

                                    % amplitude
      fr1=abs(fr1);fr2=abs(fr2);fr3=abs(fr3);
      fr4=abs(fr4);fr5=abs(fr5);

                                    % psd squard bise
      bis1=((psdy-psdym).^2)/ifin;
      bis2=((psdy-psdyi).^2)/ifin;
      bis3=((psdy-psdyn).^2)/ifin;
      bis4=((psdy-psdyc).^2)/ifin;
      bs1=sum(bis1);bs2=sum(bis2);
      bs3=sum(bis3);bs4=sum(bis4);

                                    % psd mse
      pm1=sum(pmse1)/ifin;pm2=sum(pmse2)/ifin;
      pm3=sum(pmse3)/ifin;pm4=sum(pmse4)/ifin;

                                    % amplitude mse
      fm1=abs(sum(fmse1))/ifin, fm2=abs(sum(fmse2))/ifin
      fm3=abs(sum(fmse3))/ifin, fm4=abs(sum(fmse4))/ifin
```

```
                                        % phase squared bise
      ea1=((an1-an2).^2)/ifin; ea2=((an1-an3).^2)/ifin;
      ea3=((an1-an4).^2)/ifin; ea4=((an1-an5).^2)/ifin;
      ee1=sum(ea1),ee2=sum(ea2),ee3=sum(ea3),ee4=sum(ea4)

                                       % phase mse
       ph1=sum(phmse1)/ifin;ph2=sum(phmse2)/ifin;
       ph3=sum(phmse3)/ifin;ph4=sum(phmse4)/ifin;

                                         % fit psd
      plm=(psdy'*psdym)/(psdym'*psdym);
      pli=(psdy'*psdyi)/(psdyi'*psdyi);
      pln=(psdy'*psdyn)/(psdyn'*psdyn);
      plc=(psdy'*psdyc)/(psdyc'*psdyc);

                                         % fit amplitude
      amlm=(fr1'*fr2)/(fr2'*fr2)
      amli=(fr1'*fr3)/(fr3'*fr3)
      amln=(fr1'*fr4)/(fr4'*fr4)
      amlc=(fr1'*fr5)/(fr5'*fr5)

                                         % fit phase
      alm=(an1'*an2)/(an2'*an2)
      ali=(an1'*an3)/(an3'*an3)
      aln=(an1'*an4)/(an4'*an4)
      alc=(an1'*an5)/(an5'*an5)
                                  pause
   elseif i==4
      clc,clg
      plot(f,fr1,'-',f,fr2,'*',f,fr3,':',f,fr4,'-.',f,fr5,'+')
      title('fresp amp:  true - ,pm *,unb :,mse,-.,cm +')
      pause
      plot(f,ef1,'*',f,ef2,':',f,ef3,'-.',f,ef4,'+')
      title('amp bis:  pm *, unb :, mse -.,cm +')
      pause
      plot(f,an1,'-',f,an2,'*',f,an3,':',f,an4,'-.',f,an5,'+')
      title('fresp ang:  true -,pm *,unb :,mse -.,cm +')
      pause
      plot(f,ea1,'*',f,ea2,':',f,ea3,'-.',f,ea4,'+')
      title('ang bis:  pm *, unb :, mse -.,cm +')
      pause
 plot(f,psdy,'-',f,psdym,'*',f,psdyi,':',f,psdyn,'-.',f,psdyc,'+')
      title('psd:  true -,pm *,unb :,mse -.,cm +')
      pause
      plot(f,bis1,'*',f,bis2,':',f,bis3,'-.',f,bis4,'+')
      title('psd bis:  pm *,unb :,mse -.,cm +')
      pause
   else
      'wrong number given,reset your choice 1,2,3,4'
   end
end
```

```
format compact
clc;clg; % clear windows
% the script started 18/10/1988.
% this is used to analyses Power Spectrum Density (PSD)
%     with missing data in a time series and to estimate
%     FREquency reSPonse (FRESP) with missing data
%     at output series.
% two sorts of estimators, UNBised and Mean Square Error,
%     are built up to compare with traditional ones such
%     as Periodogram Method and Interpolation Method.
% square bias, MSE, and linear FIT are calculated as
%     measurements for the comparison
% direct estimate and recursive estimate are checked with
%     UNB and MSE.
%
% x,  input
% y,  ouput without missing point
% ym, ouput with missing points
%
% psdy,  psd from y
% psdum, -------- ym by PM
% psdyc, -------------- IM
% psdyi, -------------- UNB
% psdyn, -------------- MSE
%
% fr1, an1, FRESP (amplitude and phase ) from y
% fr2, an2, ---- from ym by PM same as UNB
% fr3, an3, ---------------- UNB
% fr4, an4, ---------------- MSE
% fr5, an5, ---------------- IM

%load batmis3 % consideration of batch job
while 1
   clc;
   n=input('n>0 enter,n<=0 quit');
   if n<=0;break,end
   '1,initialisation'
   '2,generate input and output data fft, psd,weights'
   '3,FRESP,MSE,FIT calculation'
   '4,plots'
   i=input('your choice');
   if i==1
      clc;clg;
      istart=1;
      ifin=input('length of one block');
      nbl=input('no. of block');
      m=input('no. of missing points');
    misp=input('distr. of mp., unif.=1, arb.=def.');
      clear a
      for t=1:4
          a(t)=input('y(t)=a(1)*y(t-1)+ ...
          a(2)*y(t-2)+y(t-3),y3=a3*y4+a4*y5+x')
      end
      if nbl==1
         mv=input('1<order of moving aver. filter<5');
      end
```

```
elseif i==2
   clc;clg;
   pl=input('batch job? y=1 n=default');
   if pl~=1
      k=0;clear mp; % locate missing points
      yp=ones(ifin,1);
   for t=1:m
       if misp==1
          k=k+floor(ifin/m);
          mp(t)=k;
       else
          mp(t)=input('which is missing position');
       end
       yp(mp(t),1)=0;
   end
   rm=xcorr(yp); % Parzen weigth for CM
   clear cw sw nl f w1 ;
   cw(ifin,1)=0;sw(ifin,1)=0;cs=cw;
   for t=1:ifin
       w1(t,1)=ifin/(ifin-m);% traditional weight
                             % as initial value
                             % of new weight
       nl(t,1)=ifin;
       f(t,1)=(t-1)/ifin;
       for j=1:m
           c1=cos(2*pi*mp(j)*(t-1)/ifin);
           s1=sin(2*pi*mp(j)*(t-1)/ifin);
           cw(t,1)=cw(t,1)+c1^2;
           sw(t,1)=sw(t,1)+s1^2;
           cs(t,1)=cs(t,1)+c1*s1;
       end
   end
   w3=w1; w2=w1;w4=w1;w5=w1; g3=w1; g4=w1;g5=w1;
   g1=w1;g2=w1;
   cw1=nl/2-2*cw;cw2=nl/2-cw;sw1=nl/2-2*sw;
   sw2=nl/2-sw;sw3=nl/2+sw;
   rand('normal')
      clear x y y1 yi ym ys ; y1(ifin+5,1)=0;
      clear tr1 tr2 tr3 tr4 fr1 fr2 fr3 fr4 ;
      clear psdx psdy psdym psdyi psdyn f1 f2 f3 f4 ;
      tr1(ifin,1)=0;tr2=tr1;tr3=tr1;tr4=tr1;tr5=tr1;
      fr1=tr1;fr2=tr1;fr3=tr1;fr4=tr1;fr5=tr1;
      an1=tr1;an2=tr1;an3=tr1;an4=tr1;an5=tr1;
      psdx=tr1; psdy=tr1; psdym=tr1; psdyi=tr1;
      psdyn=tr1; psdyc=tr1;
      psdxc=tr1; temp=[an1 an2 an3];
      cv=0;cvx=0;stx=0;sty=0;pyc=tr1;amp5=tr1;csp=tr1;
      qsp=tr1;
      f1=tr1;f2=f1;f3=f1;f4=f1;gf1=f1;gf2=f1;gf3=f1;
      gf4=f1;
      pmse1=f1;pmse2=f1;pmse3=f1;pmse4=f1;
      fmse1=f1;fmse2=f1;fmse3=f1;fmse4=f1;
      phmse1=f1;phmse2=f1;phmse3=f1;phmse4=f1;
      dri=0;drr=0;dii=0;
      for t1=1:nbl
          for t=6:ifin+5
```

```
        x(t-5,1)=rand;
    y1(t-3,1)=a(3)*y1(t-4,1)+a(4)*y1(t-5,1)+x(t-5,1);
    y1(t,1)=a(1)*y1(t-1,1)+a(2)*y1(t-2,1)+y1(t-3,1);
        y(t-5,1)=y1(t,1);  % without missing point
        ym(t-5,1)=y1(t,1); % traditional
        yi(t-5,1)=y1(t,1); % interpolation
    end
    for t=1:m
        ys(t,1)=ym(mp(t),1); % real missing values
        ym(mp(t),1)=0;
        yi(mp(t),1)=(yi(mp(t)-1,1)+yi(mp(t)+1,1))/2;
        ys(t,2)=yi(mp(t),1); % interpolate values
    end
    tempx=fft(x); % input x's fft
    rex=real(tempx);
    imx=imag(tempx);
    tempy=fft(y); % output y's fft
    rey=real(tempy);
    imy=imag(tempy);
    tempym=fft(ym); % output ym's fft
    reym=real(tempym);
    imym=imag(tempym);
    tempyi=fft(yi); % output yi's fft

    px=abs(tempx).^2/ifin; % psd estimations
    py=abs(tempy).^2/ifin;
    pym=abs(tempym).^2/(ifin-m);
    pyi=g3.*pym*(ifin-m)/ifin;
    pyn=w3.*pym*(ifin-m)/ifin;
    pyc=abs(tempyi).^2/ifin; %im

    psdx=(t1-1)*psdx/t1+px/t1;
    psdy=(t1-1)*psdy/t1+py/t1;
    psdym=(t1-1)*psdym/t1+pym/t1;
    psdyi=(t1-1)*psdyi/t1+pyi/t1;
    psdyn=(t1-1)*psdyn/t1+pyn/t1;
    psdyc=(t1-1)*psdyc/t1+pyc/t1; %im
                              % fresp estimations
    tr1=(tempy.*conj(tempx))./ifin;
    fr1=(t1-1)*fr1/t1+tr1/t1;

    tr2=(tempym.*conj(tempx))/(ifin-m);
    fr2=(t1-1)*fr2/t1+tr2/t1;

    t3=g4.*reym+sqrt(-1)*g5.*imym;
    tr3=t3.*conj(tempx)/ifin;
    fr3=(t1-1)*fr3/t1+tr3/t1;

    t4=w4.*reym+sqrt(-1)*w5.*imym;
    tr4=t4.*conj(tempx)/ifin;
    fr4=(t1-1)*fr4/t1+tr4/t1;

    tr5=(tempyi.*conj(tempx))/ifin; %im
    fr5=(t1-1)*fr5/t1+tr5/t1;

                                  % mse calculation
```

```
            cv1=ifin*cov(ym)/(ifin-m);
            cv=(t1-1)*cv/t1+cv1/t1;

            rif=2*(w1.*reym).*(w2.*imym)/ifin;
            f2=(t1-1)*f2/t1+rif/t1;

            f1=psdyn; yy=psdyn*ifin;yy(1,1)=yy(1,1)/2;

            cf3=(rex.*w4.*reym+imx.*w5.*imym)/ifin;
            f3=(t1-1)*f3/t1+cf3/t1;

            cf4=(imx.*w4.*reym-rex.*w5.*imym)/ifin;
            f4=(t1-1)*f4/t1+cf4/t1; ,

        gf1=psdyi;
   gf2=(t1-1)*gf2/t1+2*(g1.*reym).*(g2.*imym)/(ifin*t1);
   gf3=(t1-1)*gf3/t1+(rex.*g4.*reym+imx.*g5.*imym)/(ifin*t1);
   gf4=(t1-1)*gf4/t1+(imx.*g4.*reym-rex.*g5.*imym)/(ifin*t1);

        dri=(t1-1)*dri/t1+(reym.*imym)/t1; %direct operation
        drr=(t1-1)*drr/t1+reym.^2/t1;
        dii=(t1-1)*dii/t1+imym.^2/t1;

        if nbl==1 % moving average filter for one block
           tw=1;
           for j=1:ifin-mv+1
               yy(j,1)=tw*sum(yy(j:j+mv-1,1))/mv;
               f1(j,1)=tw*sum(f1(j:j+mv-1,1))/mv;
               f2(j,1)=tw*sum(f2(j:j+mv-1,1))/mv;
               f3(j,1)=tw*sum(f3(j:j+mv-1,1))/mv;
               f4(j,1)=tw*sum(f4(j:j+mv-1,1))/mv;
               gf1(j,1)=tw*sum(gf1(j:j+mv-1,1))/mv;
               gf2(j,1)=tw*sum(gf2(j:j+mv-1,1))/mv;
               gf3(j,1)=tw*sum(gf3(j:j+mv-1,1))/mv;
               gf4(j,1)=tw*sum(gf4(j:j+mv-1,1))/mv;
           end
        end

                              % new weight (mse) for psd
            ymr=f1.*cw1+cv*cw+2*f2.*cs;
            ymi=f1.*sw1+cv*sw-2*f2.*cs;
            ymi(1,1)=0;
            ymr(1,1)=f1(1,1)*(ifin/2-m)+cv*m/2;
            yrymr=f1.*cw2+f2.*cs; yiymi=f1.*sw2-f2.*cs;
            yiymi(1,1)=0; yrymr(1,1)=f1(1,1)*(ifin/2-m/2);
            yrymi=f2.*sw3-f1.*cs; yiymr=f2.*cw2-f1.*cs;
            yrymi(1,1)=0;yiymr(1,1)=0;
            ymrymi=f2.*(ifin/2-cw+sw)-2*f1.*cs+cv*cs;
            ymrymi(1,1)=0;
            c3=4*yrymi.^2;

     w3=yy.*(ymr+ymi)+2*(yrymr.^2+yiymi.^2+yrymi.^2+yiymr.^2);
            w7=3*(ymr+ymi).^2-4*ymr.*ymi+4*ymrymi.^2;
            w3=w3./w7;  w3(ifin/2,1)=w3(1,1);

                                    % new weight (unb) for psd
```

```
g3=ifin*gf1./(gf1*(ifin-2*m)+cv*m);

            % new weight (mse) for Fourie transform
rn=(f1.*cw2+f2.*cs);rd=(f1.*cw1+cv*cw+2*f2.*cs);
in=(f1.*sw2-f2.*cs);
id=(f1.*sw1+cv*sw-2*f2.*cs+.0001);
w1=rn./rd;w2=in./id;

                % new weight(mse) for fresp real part
uryr=ifin*f3/2;
urymr=f3.*cw2+f4.*cs;
urymr(1,1)=f3(1,1)*(ifin/2-m/2);
uiyr=ifin*f4/2;
uiymr=f4.*cw2-f3.*cs;
c1=urymr.^2+uiymr.^2;

w4=psdx.*rn+2*(uryr.*urymr+uiyr.*uiymr);
w7=psdx.*rd+2*c1;
w4=w4./w7; w4(ifin/2,1)=w4(1,1);

             % new weight(mse) for fresp imag. part
uryi=-ifin*f4/2;
urymi=-f4.*sw2-f3.*cs;
uiyi=ifin*f3/2;
uiymi=f3.*sw2-f4.*cs;
c2=urymi.^2+uiymi.^2;

w5=psdx.*in+2*(uryi.*urymi+uiyi.*uiymi);
w7=psdx.*id+2*c2;
w7(1,1)=1; w7(ifin/2,1)=1;
w5=w5./w7;
w5(1,1)=1; w5(ifin/2,1)=1;

rn=(gf1.*cw2+gf2.*cs);rd=(gf1.*cw1+cv*cw+2*gf2.*cs);
in=(gf1.*sw2-gf2.*cs);
id=(gf1.*sw1+cv*sw-2*gf2.*cs+.0001);
g1=rn./rd;g2=in./id;

   % new weight(unb) for fresp real and imag. parts
g4=ifin/(ifin-m);
g5=g4;

if nbl==1
   psdyi=g3.*abs(tempym).^2/ifin;
   psdyn=w3.*abs(tempym).^2/ifin;
   fr3=(g4.*reym+sqrt(-1)*g5.*imym).*conj(tempx)/ifin;
   fr4=(w4.*reym+sqrt(-1)*w5.*imym).*conj(tempx)/ifin;
   tr3=fr3;tr4=fr4;
end

pmse1=(t1-1)*pmse1/t1+(py-pym).^2/t1;  % psd
pmse2=(t1-1)*pmse2/t1+(py-pyi).^2/t1;
pmse3=(t1-1)*pmse3/t1+(py-pyn).^2/t1;
pmse4=(t1-1)*pmse4/t1+(py-pyc).^2/t1;

                                % amplitude (fresp)
```

```
  fmse1=(t1-1)*fmse1/t1+(tr1-tr2).*conj(tr1-tr2)/t1;
  fmse2=(t1-1)*fmse2/t1+(tr1-tr3).*conj(tr1-tr3)/t1;
  fmse3=(t1-1)*fmse3/t1+(tr1-tr4).*conj(tr1-tr4)/t1;
  fmse4=(t1-1)*fmse4/t1+(tr1-tr5).*conj(tr1-tr5)/t1;

  phmse1=(t1-1)*phmse1/t1+(angle(tr1)-angle(tr2)).^2/t1; % phase
  phmse2=(t1-1)*phmse2/t1+(angle(tr1)-angle(tr3)).^2/t1;
  phmse3=(t1-1)*phmse3/t1+(angle(tr1)-angle(tr4)).^2/t1;
  phmse4=(t1-1)*phmse4/t1+(angle(tr1)-angle(tr5)).^2/t1;
        end
     else
        save batmis3
     end
   elseif i==3
     clc; clg % fresp estimations, mse, fit
psddu=(drr+dii-cv*m)/(ifin-2*m); % direct operation
d3=psddu-cv;
d1=psddu*ifin.*(drr+dii)+2*((drr+d3.*cw).^2+(dii+d3.*sw).^2+2*(dri+ ..
   d3.*cs).^2);
d2=3*(drr+dii).^2-4*drr.*dii+4*dri.^2;
d1=d1./d2;
psddm=d1.*(drr+dii)/ifin;

                          % amplitude squared bise of fresp
ef1=((fr1-fr2).*conj(fr1-fr2))/ifin;
ef2=((fr1-fr3).*conj(fr1-fr3))/ifin;
ef3=((fr1-fr4).*conj(fr1-fr4))/ifin;
ef4=((fr1-fr5).*conj(fr1-fr5))/ifin;
e1=sum(ef1),e2=sum(ef2),e3=sum(ef3),e4=sum(ef4),

                                  % fresp
     fr1=fr1./psdx;fr2=fr2./psdx;
     fr3=fr3./psdx;fr4=fr4./psdx;
     fr5=fr5./psdx;

                                  % phase
     an1=angle(fr1);an2=angle(fr2);
     an3=angle(fr3);an4=angle(fr4);
     an5=angle(fr5);
                                  % amplitude
     fr1=abs(fr1);fr2=abs(fr2);fr3=abs(fr3);
     fr4=abs(fr4);fr5=abs(fr5);

                                  % psd squard bise
     bis1=((psdy-psdym).^2)/ifin;
     bis2=((psdy-psdyi).^2)/ifin;
     bis3=((psdy-psdyn).^2)/ifin;
     bis4=((psdy-psdyc).^2)/ifin;
     bs1=sum(bis1);bs2=sum(bis2);
     bs3=sum(bis3);bs4=sum(bis4);
                                  % psd mse
     pm1=sum(pmse1)/ifin;pm2=sum(pmse2)/ifin;
     pm3=sum(pmse3)/ifin; pm4=sum(pmse4)/ifin;

                                  % amplitude mse
     fm1=abs(sum(fmse1))/ifin,fm2=abs(sum(fmse2))/ifin
```

```
      fm3=abs(sum(fmse3))/ifin, fm4=abs(sum(fmse4))/ifin
                                    % phase squared bise
      ea1=((an1-an2).^2)/ifin; ea2=((an1-an3).^2)/ifin;
      ea3=((an1-an4).^2)/ifin; ea4=((an1-an5).^2)/ifin;
      ee1=sum(ea1);ee2=sum(ea2);ee3=sum(ea3);ee4=sum(ea4);

                                   % phase mse
       ph1=sum(phmse1)/ifin;ph2=sum(phmse2)/ifin;
       ph3=sum(phmse3)/ifin;ph4=sum(phmse4)/ifin;

                                     % fit psd
      plm=(psdy'*psdym)/(psdym'*psdym);
      pli=(psdy'*psdyi)/(psdyi'*psdyi);
      pln=(psdy'*psdyn)/(psdyn'*psdyn);
      plc=(psdy'*psdyc)/(psdyc'*psdyc);

                                     % fit amplitude
      amlm=(fr1'*fr2)/(fr2'*fr2)
      amli=(fr1'*fr3)/(fr3'*fr3)
      amln=(fr1'*fr4)/(fr4'*fr4)
      amlc=(fr1'*fr5)/(fr5'*fr5)

                                     % fit phase
      alm=(an1'*an2)/(an2'*an2);
      ali=(an1'*an3)/(an3'*an3);
      aln=(an1'*an4)/(an4'*an4);
      alc=(an1'*an5)/(an5'*an5);

   elseif i==4
      clc,clg
      plot(f,fr1,'-',f,fr2,'*',f,fr3,':',f,fr4,'-.',f,fr5,'+')
      title('fresp amp:  true - ,pm *,unb :,mse,-.,cm +')
      pause
      plot(f,ef1,'*',f,ef2,':',f,ef3,'-.',f,ef4,'+')
      title('amp bis:  pm *, unb :, mse -.,cm +')
      pause
      plot(f,an1,'-',f,an2,'*',f,an3,':',f,an4,'-.',f,an5,'+')
      title('fresp ang:  true -,pm *,unb :,mse -.,cm +')
      pause
      plot(f,ea1,'*',f,ea2,':',f,ea3,'-.',f,ea4,'+')
      title('ang bis:  pm *, unb :, mse -.,cm +')
      pause
plot(f,psdy,'-',f,psdym,'*',f,psdyi,':',f,psdyn,'-.',f,psdyc,'+')
      title('psd:  true -,pm *,unb :,mse -.,cm +')
      pause
      plot(f,bis1,'*',f,bis2,':',f,bis3,'-.',f,bis4,'+')
      title('psd bis:  pm *,unb :,mse -.,cm +')
      pause
   else
      'wrong number given,reset your choice 1,2,3,4'
   end
end
```

```
format compact
clc;clg; % clear windows.
% the script started 1/10/1988.
% this detects nonlinear systems structure based on input
%      and output data, by a quantisation algorithm, to
%      give illustrations both within three dimensional
%      space and in particular project planes.
% uncorrelated nomarl noise (0.,vn) can be superposed on
%      input and output if required.
%
%load batdet2 % consideration of batch job.
while 1
   clc;
   n=input('n>0 enter,n<=0 quit ');
   if n<=0;break,end
   '1,initialisation'
   '2,generate ipt'
   '3,generate opt and quantization'
   '4,plot 3D surface and 2D project curve'
   i=input('your choice ');
   if i==1
      clc;clg;
   elseif i==2
      clc;clg;
      ipt=5;
      'set ipt signal'
      '1,sinusoid'
      '2,Gaussian'
      '3,step'
      '4,random amplitude step '
      ipt=input('your choice ');
      istart=input('starting point of data series ');
      ifin=input('final point of data series <4000 ');
      x1=ingen(ipt,istart,ifin);
          x=x1(istart:ifin);
      clc;clg;
      'set sys. para.'
      '1,linear follows nl.'
      '2,closed loop system with jumping effect'
      j=input('your choice');
      if j==1
         '1,first or second order lag saturation'
         '2,- - - - - - - - - - - - - - relay'
         '3,- - - - - - - - - - - - - - output=input^3'
         '4,- - - - - - - - - - - - - - linear'
         istype=input('your choice ');
         c=line1(j);
      elseif j==2
         j1=j;
      else
         c=line3(j);
      end
      clc;clg;
      'set quant. control'
      s1=input('starting point of polt');
      s2=input('final point of plot');
```

```
      nr=input('set amp. quan. range of x(i) and y(i-1)');
      ax=input('set axes control para.');
   elseif i==3
      clc,clg
      clear z1 z2 z3 zn1 zn2 zn3;
      pl=input('batch job? yes=1, no=default');%open-loop sys.
      if j==1
         y1=sysout1(istype,istart,ifin,x,c);
      else
         y1=sysout2(1,istart,ifin,x,c); % closed-loop sys.
      end
      y=y1(istart:ifin);
if pl~=1
          x1=x(s1:s2);
          x2=x(s1-1:s2-1);
          y1=y(s1-1:s2-1);
          y2=y(s1-2:s2-2);
          z=y(s1:s2);
      k1=nr/(max(x1)-min(x1));k2=nr/(max(y1)-min(y1));
      k3=nr/(max(x2)-min(x2));k4=nr/(max(y2)-min(y2));
      i1=round(k1*(x1-min(x1))+1);j1=round(k2*(y1-min(y1))+1);
      j2=round(k3*(x2-min(x2))+1);j3=round(k4*(y2-min(y2))+1);
      for t1=1:nr
          for t2=1:nr
              ind=find(i1==t1&j1==t2);
              z1(t1,t2)=sum(z(ind));zn1(t1,t2)=length(ind);
              if zn1(t1,t2)==0
                 zn1(t1,t2)=1;
              end
              ind=find(i1==t1&j2==t2);
              z2(t1,t2)=sum(z(ind));zn2(t1,t2)=length(ind);
              if zn2(t1,t2)==0
                 zn2(t1,t2)=1;
              end
              ind=find(i1==t1&j3==t2);
              z3(t1,t2)=sum(z(ind));zn3(t1,t2)=length(ind);
              if zn3(t1,t2)==0
                 zn3(t1,t2)=1;
              end
          end
      end
z1=z1./zn1;temp=zeros(2*ax+nr); temp(ax+1:ax+nr,ax+1:ax+nr)=z1;
z1=temp;
z2=z2./zn2;temp(ax+1:ax+nr,ax+1:ax+nr)=z2;z2=temp;
z3=z3./zn3;temp(ax+1:ax+nr,ax+1:ax+nr)=z3;z3=temp;
else
     save batdet
end
   elseif i==4
      clear px0 px1 py1 py2 pd0 pd1;
          k=round(ax+nr/2)+1; n=(max(x)-min(x))/(nr-1);
          px0=z1(ax+1:ax+nr,k);
          hx0=min(x):n:max(x); n=(max(y)-min(y))/(nr-1);
          py1=z1(k,ax+1:ax+nr);
          hy1=min(y):n:max(y);
          py2=z3(k,ax+1:ax+nr);
```

```
        px1=z2(k,ax+1:ax+nr);

    mesh(z1) %coordinate (x,y1,y)
    title('sys. opt y(i) versus ipt x(i) and previous opt y(i-1)')
    pause
    pl=input('do you want to store the plot , yes=1,no=default')
    if pl==1
       meta plotn
    end
    mesh(z1')
    title('sys opt y(i) versus ipt x(i) and previous opt y(i-1)')
    pause
    if pl==1
       meta plotn
    end
    plot(hx0',px0)
    grid
    title('projeting plane of y(i) and x(i)')
    xlabel('x(i)')
    ylabel('y(i)')
    pause
    if pl==1
       meta plotn
    end
    plot(hy1',py1)
    grid
    title('projecting plane of y(i) and y(i-1)')
    xlabel('y(i-1)')
    ylabel('y(i)')
    pause
    if pl==1
       meta plotn
    end
    mesh(z3) %coordinate (x,y2,y)
    title('sys opt y(i) versus ipt x(i) and previous opt y(i-2)')
    pause
    pl=input('do you want to store the plot , yes=1,no=default')
    if pl==1
       meta plotn
    end
    mesh(z3')
    title('sys opt y(i) versus ipt x(i) and previous opt y(i-2)')
    pause
    if pl==1
       meta plotn
    end
    plot(hy1',py2)
    grid
    title('projecting plane of y(i) and y(i-2)')
    xlabel('y(i-2)')
    ylabel('y(i)')
    pause
    if pl==1
       meta plotn
    end
    mesh(z2)%coordinate (x,x1,y)
```

```
      title('sys opt y(i) versus ipt x(i) and previous ipt x(i-1)')
      pause
      pl=input('do you want to store the plot , yes=1,no=default')
      if pl==1
         meta plotn
      end
      mesh(z2')
      title('sys opt y(i) versus ipt x(i) and previous ipt x(i-1)')
      pause
      if pl==1
         meta plotn
      end
      plot(hx0',px1)
      grid
      title('projecting plane of y(i) and x(i-1)')
      xlabel('x(i-1)')
      ylabel('y(i)')
      pause
      if pl==1
         meta plotn
      end
   else
      'wrong number given,reset your choice 1,2,3,4,5'
   end
end
```

```
format compact
clc,
% the script started 15/11/1988 .
% the script is used to eatimate system parameters
% by recursive procedure after
% detection of nonlinear system (sys1) done by 'nondet.m'
load batdet
clear z1 z2 z3 zn1 zn2 zn3
x1=x(s1:s2); %set up coordinates
x2=x(s1-1:s2-1);
y1=y(s1-1:s2-1);
y2=y(s1-2:s2-2);
z=y(s1:s2);
ta1=z+c(4)*y2-c(2)*x2;
tb1=z+c(3)*y1+c(4)*y2;
ta2=z+c(3)*y1-c(2)*x2;
                                    %quant. gains
      k1=nr/(max(x1)-min(x1));k2=nr/(max(y1)-min(y1));
      k3=nr/(max(x2)-min(x2));k4=nr/(max(y2)-min(y2));
      i1=round(k1*(x1-min(x1))+1);j1=round(k2*(y1-min(y1))+1);
      j2=round(k3*(x2-min(x2))+1);j3=round(k4*(y2-min(y2))+1);
      for t1=1:nr
          for t2=1:nr
              ind=find(i1==t1&j1==t2);
              z1(t1,t2)=sum(ta1(ind));zn1(t1,t2)=length(ind);
              if zn1(t1,t2)==0
                 zn1(t1,t2)=1;
              end
              ind=find(i1==t1&j2==t2);
              z2(t1,t2)=sum(tb1(ind));zn2(t1,t2)=length(ind);
              if zn2(t1,t2)==0
                 zn2(t1,t2)=1;
              end
              ind=find(i1==t1&j3==t2);
              z3(t1,t2)=sum(ta2(ind));zn3(t1,t2)=length(ind);
              if zn3(t1,t2)==0
                 zn3(t1,t2)=1;
              end
          end
      end
z1=z1./zn1;temp=zeros(2*ax+nr); temp(ax+1:ax+nr,ax+1:ax+nr)=z1;
z1=temp;
z2=z2./zn2;temp(ax+1:ax+nr,ax+1:ax+nr)=z2;z2=temp;
z3=z3./zn3;temp(ax+1:ax+nr,ax+1:ax+nr)=z3;z3=temp;
save batdet1
```

```
format compact
clc,
% the script started 15/11/1988 .
% the script is used to eatimate system parameters
% by recursive procedure after the
% detection of the nonlinear system (sys2) done by 'nondet.m'
load batdet
clear z1 z2 z3 zn1 zn2 zn3
          x1=x(s1:s2);
          x2=x(s1-1:s2-1);
          y1=y(s1-1:s2-1);
          y2=y(s1-2:s2-2);
          z=y(s1:s2);
      ta1=y1+(c(4)*y2+c(2)*x2)/c(3);
      ta2=y2+(c(3)*y1+c(2)*x2)/c(4);
      tb1=x2+(c(3)*y1+c(4)*y2)/c(2);
                                                %quant. gain
      k1=nr/(max(x1)-min(x1));k2=nr/(max(ta1)-min(ta1));
      k3=nr/(max(tb1)-min(tb1));k4=nr/(max(ta2)-min(ta2));
      i1=round(k1*(x1-min(x1))+1);j1=round(k2*(ta1-min(ta1))+1);
      j2=round(k3*(tb1-min(tb1))+1);j3=round(k4*(ta2-min(ta2))+1);
      for t1=1:nr
          for t2=1:nr
              ind=find(i1==t1&j1==t2);
              z1(t1,t2)=sum(z(ind));zn1(t1,t2)=length(ind);
              if zn1(t1,t2)==0
                 zn1(t1,t2)=1;
              end
              ind=find(i1==t1&j2==t2);
              z2(t1,t2)=sum(z(ind));zn2(t1,t2)=length(ind);
              if zn2(t1,t2)==0
                 zn2(t1,t2)=1;
              end
              ind=find(i1==t1&j3==t2);
              z3(t1,t2)=sum(z(ind));zn3(t1,t2)=length(ind);
              if zn3(t1,t2)==0
                 zn3(t1,t2)=1;
              end
          end
      end
z1=z1./zn1;temp=zeros(2*ax+nr); temp(ax+1:ax+nr,ax+1:ax+nr)=z1;
z1=temp;
z2=z2./zn2;temp(ax+1:ax+nr,ax+1:ax+nr)=z2;z2=temp;
z3=z3./zn3;temp(ax+1:ax+nr,ax+1:ax+nr)=z3;z3=temp;
save batdet2
```

```
format compact
clc;clg; % clear windows.
% the script started 1/7/1988.
% this models nonlinear or linear systems as a linear one ,
%      ym(t)=b0*x(t)+b1*x(t-1)-a1*ym(t-1)-a2*ym(t-2)+c0,
%      by Weighted least Squres (WLS) , which the weights
%      are determined according to nearness of current
%      state to previous states.
% uncorrelated nomarl noise (0.,vn) can be superposed on
%      input if required.
% iprog = 1, one step prediction;
%       = 2, model response.
% plots and estimated parameters show are inclusive either.
%
% both off line and on line algorithms are available.
%
while 1
   clc
   n=input('n>0 enter,n<=0 quit ');
   if n<=0;break,end
   '1,initialisation'
   '2,generate input'
   '3,generate output'
   '4,model estimation'
   '5,plot model response or one step prediction and input'
   i=input('your choice ');
   if i==1
      clc,clg,
   elseif i==2
      clc,clg,
      ipt=5;
      '1,sinusoid'
      '2,Gaussian'
      '3,uniform'
      '4,step'
      '5,random amplitude step '
      ipt=input('your choice ');
      istart=input('starting point of data series ');
      ifin=input('final point of data series <4000 ');
      x1=ingen(ipt,istart,ifin);
          x=x1(istart:ifin);
      noi=input('input noise? y=1, n=default');
      if noi==1
         v=input('variance= '); v=sqrt(v);
         rand('normal')
         z=x+v*rand(1,istart:ifin);
      else
         z=x;
      end
   elseif i==3
      clc,clg,
      '1,linear'
      '2,output=input^2'
      '3,first order lag hysteresis'
      '4,complex nonlinear'
      '5,closed system with jumping effect'
```

```
      istype=input('your choice ');
      [y1,c]=sysout(istype,istart,ifin,z);
          y=y1(istart:ifin);
      noi=input('output noise? y=1, n=default');
      if noi==1
         v=input('variance= '); v=sqrt(v);
         rand('normal')
         y=y+v*rand(1,istart:ifin);
      end
   elseif i==4
      clc,clg,
      '1,one step prediction'
      '2,model response'
      iprog=input('your choice ');
      wc=input('weight choice, lim. data=1, default otherwise');
      if wc==1
         d=input('weight range>4');
      end
      ies=input('starting point of para. est. ');
      ief=input('final point of para. est. ');
      pl=input('batch job? y=1, n=default');
      if pl~=1
         if wc==1
            const=[iprog,d];
            [ym,pm,erols,erwls]=model1(const,ies,ief,x,y);
         else
            [ym,pm,erols,erwls]=model2(iprog,ies,ief,x,y);
                        % store model state and parameter table
             pm=pm(:,ies+3:ief);
             x1=x(ies+3:ief)';y1=y(ies+2:ief-1)';
             x2=x(ies+2:ief-1)';y2=y(ies+1:ief-2)';
             table=[x1,x2,y1,y2];
             save jumtab2 table pm
         end
      else
         save batnid
      end
   elseif i==5
      clc;clg;clear k xp yp ymp;
      s1=input('starting point of plot ');
      s2=input('final point of plot ');
      k=s1:s2;
      xp=x(s1:s2);
      yp=y(s1:s2);
      ymp=ym(s1:s2);
      plot(k,xp','-',k,yp','o',k,ymp','*') % plots
      grid
      title('true y , est. ym , input x versus time t')
      xlabel('time t')
      ylabel('x,y|o,ym|*')
      pause
      pl=input('store the plots? yes=1, no=default ')
      if pl==1
         meta ploide
      end
      'mean square errors'
```

```
      erols,erwls
      pause
   else
      'wrong number given,reset your choice 1,2,3,4,5,6'
   end
end
```

210

```
format compact
clc; clg; % clear windows.
% the script stareted 7/4/1989.
% this studies jumping effect of a  nonlinear system which
%       consists of
%       a second order linear dynamic and a saturation static .
%
while 1
    clc;
    n=input('enter>0,quit<=0');
    if n<=0;break,end
    '1,initialisation ,ipt. and opt. generation'
    '2,plot of opt. amp. versus freq.specified'
    i=input('your choice');
    if i==1
           'set sys. para. ?'
           n1=input('yes=1,no=default');
           if n1==1
              damp=input('damp=');
              b=input('break point of saturation');
              h=input('height of saturation');
              h=h/b; clc
           end
           'set input signal ?'
           n1=input('yes=1,no=default');
           if n1==1
              amp=input('amplitude of sinusoid');
              rg1=input('min freq.');
              rg2=input('max freq.');
              dr=input('dir. of freq. inc=1 dec=-1');
              step=input('step of freq. change');
              sp=input('sampling interval');
              ns=input('no. of samples');
           end
           clear f fr yf yfm w;
           y(2)=0;y1(2)=0;y2=[];
     pl=input('batch job, pl=1, normal pl=default');
     if pl~=1
           t1=1; t2=0;
           if dr==1
              t3=1;
              fr(t3)=rg1-step; f1=rg1; f2=step; f3=rg2;
           else
              t3=ceil((rg2-rg1)/step)+3;
              fr(t3-1)=rg2+step; f1=rg2; f2=-step; f3=rg1;
              step=-step;
           end
           for t4=f1:f2:f3
                 t3=dr+t3;
                 if dr==1
                    fr(t3)=fr(t3-1)+step; f(t3-1)=fr(t3);
                 else
                    fr(t3-1)=fr(t3)+step; f(t3-1)=fr(t3-1);
                 end
                 w=2*pi*f(t3-1);
                       for t=1:ns % one sinewave
```

21

```
                              t2=t2+sp;
                              t1=t1+1;
                              r=amp*cos(w*(t2-sp));
                              x(t1)=r;
                              e1=x(t1)-y(t1-1);
                              e2=e1-damp*y1(t1-1);
                              if abs(e2)>1
                                 u=sign(e2);
                              else
                                 u=e2;
                              end
                              y1(t1)=sp*u+y1(t1-1);
                              y(t1)=sp*y1(t1)+y(t1-1);
                           end
          yfm(t3-1)=(abs(max(y(t1-100+1:t1)))+ ...
                     abs(min(y(t1-100+1:t1))))/(2*amp); %freq
             end
      else
           save bajmp
      end
     elseif i==2
            plot(x)
            xlabel('time t')
            ylabel('sys ipt')
pause
            plot(y)
            xlabel('time t')
            ylabel('sys opt')
pause
            pl=input('store the plot? yes=1,no=default');
            if pl==1
               meta jmp2
            end
            plot(f,yfm)
            grid
            title('sys. freq. respon.')
            xlabel('freq.')
            ylabel('amp.')
            pause
            if pl==1
               meta jmp2
            end
     else
         'wrong number given,reset your choice'
     end
end
```

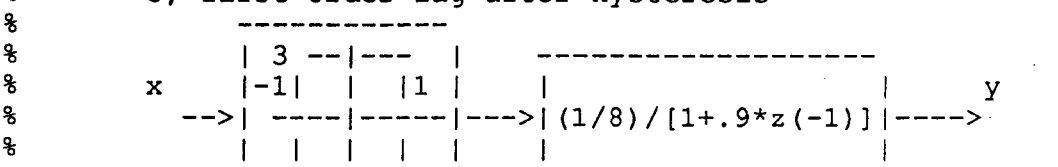
```
format compact
clc; clg; % clear windows.
% the script stareted 7/4/1989.
% this studies jumping effect of a  nonlinear system
%      which consists of
%      a second order linear dynamic and a saturation static .
% a certain length of data (ipt and opt) has been
%      collected from
%      real system excited by Gaussian input in experiment.
% a set of sinewaves is used to predicted the corresponding
%      output by VWLS, which the weights are calculated from
%      the whole data including experimental ones.
while 1
    clc;
    n=input('enter>0,quit<=0');
    if n<=0;break,end
    '1,initialisation ,ipt. and opt. generation'
    '2,plot of opt. amp. versus freq.specified'
    i=input('your choice');
    if i==1
          'set input signal ?'
          n1=input('yes=1,no=default');
          if n1==1
             amp=input('amplitude of sinusoid');
             rg1=input('min freq.');
             rg2=input('max freq.');
             dr=input('dir. of freq. inc=1 dec=-1');
             step=input('step of freq. change');
             sp=input('sampling interval'); %sp=0.5 or 1.
             ns=input('no. of samples'); %ns=200 or 100.
          end
          clear f fr yf yfm w;
     pl=input('batch job, pl=1, normal pl=default');
     if pl~=1
          load jdat2 %experimental data ipt/opt
          es=length(x); %experimental data length
          est=es;t2=0;
          if dr==1
             t3=1;
             fr(t3)=rg1-step; f1=rg1; f2=step; f3=rg2;
          else
             t3=ceil((rg2-rg1)/step)+3;
             fr(t3-1)=rg2+step; f1=rg2; f2=-step; f3=rg1;
             step=-step;
          end
          for t4=f1:f2:f3
                t3=dr+t3;
                if dr==1
                   fr(t3)=fr(t3-1)+step; f(t3-1)=fr(t3);
                else
                   fr(t3-1)=fr(t3)+step; f(t3-1)=fr(t3-1);
                end
                w=2*pi*f(t3-1);
                      for t=1:ns % one sinewave
                          est=est+1;t2=t2+sp;
                          r=amp*cos(w*(t2-sp));
```

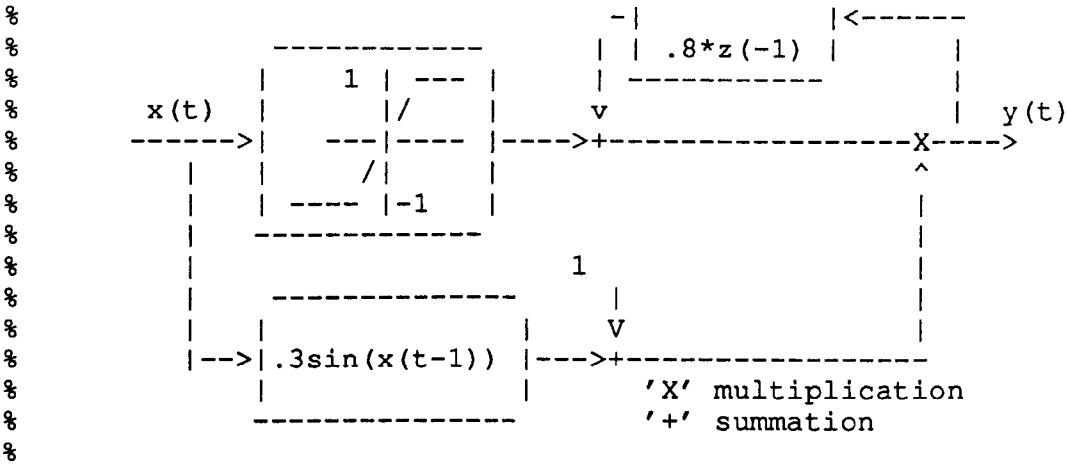
```
                        x(est)=r;
                        [y(est),pm(est-es,:)]=jumod(x,y,est);
                    end
                yfm(t3-1)=(abs(max(y(est-100+1:est)))+ ...
                        abs(min(y(est-100+1:est))))/(2*amp);
            end
     else
          save bajmp
     end
    elseif i==2
           plot(x)
           xlabel('time t')
           ylabel('sys ipt')
pause
           plot(y)
           xlabel('time t')
           ylabel('sys opt')
 pause
           pl=input('store the plot? yes=1,no=default');
           if pl==1
              meta jmp2
           end
           plot(f,yfm)
           grid
           title('sys. freq. respon.')
           xlabel('freq.')
           ylabel('amp.')
           pause
           if pl==1
              meta jmp2
           end
    else
        'wrong number given,reset your choice'
    end
end
```

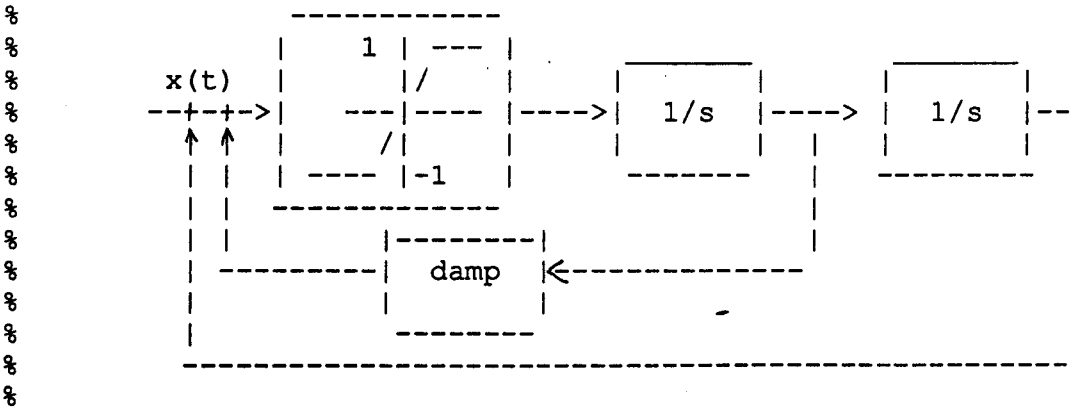
```
function x = ingen(inp,is,ifi)
format compact
clc;clg; % clear windows
% the function started 27/6/1988.
% generate four input signals
% inp = 1, sinusoid;
%     = 2, Gaussian;
%     = 3, uniform;
%     = 4, step;
%     = 5, random amplitude steps.
%
% is  = start input point .
% ifi = final input point .
% input signal is stored in x(:,1).
%
clear x; x(ifi)=0;
if inp==1
   a=input('give amplitude of sinusoid ');
   p=input('give period of sinusoid(>2) ');
   temp=2*pi/p;
   for t=is:ifi
       x(t)=a*sin((t-is)*temp)+x(t);
   end
elseif inp==2
   rand('normal')
   v=input('give variance (>0) ');
   v=sqrt(v);
   for t=is:ifi
       x(t)=v*rand+x(t);
   end
elseif inp==3
   rand('uniform')
   v=input('give variance(>0)');
   v=1*v;
   for t=is:ifi
       x(t)=v*(rand-.5)+x(t);
   end
elseif inp==4
   a=input('give step amplitude ');
   for t=is:is+4
       x(t)=0.+x(t);
   end
   for t=is+5:ifi
       x(t)=a+x(t);
   end
elseif inp==5
   v=-1.0;
   max=-1.0;
   while v<= 0. | max<= 0.
         v=input('give variance of step >0');
         max=input('0< max. integer step interval <50 ');
   end
   v=sqrt(v);
   t3=is;
   for t2=1:ifi-is+1
       a=v*rand;
```

```
        rand('uniform')
        l=max*rand+1;
        rand('normal')
        if l<1
           l=1;
        end
        if l>max
           l=max;
        end
        for t4=t3:t3+l-1
            if t4<= ifi
               x(t4)=a+x(t4);
            end
        end
        t3=t3+l;
    end
else
    'wrong number given , reset input signal type 1,2,3,4'
end
```

```
function [y,c] = sysout(sys,is,ifi,x)
format compact
clc; % clear windows
% the funtion started 28/6/1988.
% generate system output from predetermined input signal and system.
% sys = 1, linear [ b0+b1*z(-1)]/[a0+a1*z(-1)+a2*z(-2)];
%     = 2, output=input^2;
%     = 3, first order lag after hysteresis
%
%
%
%
%
%     = 4, complex nonlinear.
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%     = 5, nonlinear closed-loop system
%
%
%
%
%
%
%
%
%
% is = starting point of data series.
% ifi = final point of data series.
% x(:,1) = input signal generated from function ingen.
% c(:,1) = linear system parameters ,
% c(1,1)=b0,c(2,1)=b1,c(3,1)=a0,c(4,1)=a1,c(5,1)=a2.
```







```
%
% system output is stored in y(:,1).
%
clear y; y(1,ifi)=0.;
if sys==1
   '[b0+b1*z(-1)]/[a0+a1*z(-1)+a2*z(-2)]'
   c(1,1)=input('b0');c(2,1)=input('b1');c(3,1)=input('a0');
                      c(4,1)=input('a1');c(5,1)=input('a2');
   ga=c(1,1)/c(3,1);gb=c(2,1)/c(3,1);gc=c(4,1)/c(3,1);
   gd=c(5,1)/c(3,1);
   y(is)=ga*x(is);
   y(is+1)=ga*x(is+1)+gb*x(is)-gc*y(is);
   for t=is+2:ifi
       y(t)=ga*x(t)+gb*x(t-1)-gc*y(t-1)-gd*y(t-2);
   end
elseif sys==2
   for t=is:ifi
       y(t)=x(t)^2;
   end
elseif abs(sys)==3
   for t=is:ifi
       if abs(x(t))>=1.0 % hysteresis
          sys=3*sign(x(t));
       end
       if t==is
          y(t)=.9*y(t)+sys/8;
       else
          y(t)=.9*y(t-1)+sys/8; % first order lag
       end
   end
elseif abs(sys)==4
   for t=is:ifi
       temp=x(t);
       if abs(temp)>1.
          temp=sign(temp);
       end
       if t==is
          y(t)=(.8*y(t)+temp)*(1.+.3*sin(x(t)));
       else
          y(t)=(.8*y(t-1)+temp)*(1.+.3*sin(x(t-1)));
       end
   end
elseif sys==5
   clear y1;y1(1+is)=0;
   y(1+is)=0;
   damp=input('damp=');
   h=1;
   for t=is+1:ifi
       e1=x(t)-y(t-1);
       e2=e1-damp*y1(t-1);
       if abs(e2)>1
          u=sign(e2);
       else
          u=e2;
       end
       y1(t)=0.5*u+y1(t-1);
```

```
        y(t)=0.5*y1(t)+y(t-1);
    end
else
   'wrong number given,reset system type 1,2,3,4'
end
```

```
function [yml,pm,eo,ew]=model2(iprog,is,ifi,x,y)
format compact
clc;clg; % clear windows.
% the function started 29/6/1988.
% obtain model parameters by a function fit(n,iwp,noise,x,y).
% generate model response
%   ym(t)=b0*x(t)+b1*x(t-1)-a1*ym(t-1)-a2*ym(t-2)+c0,
% or one step prediction
%   ym(t)=b0*x(t)+b2*x(t-1)-a1*y(t-1)-a2*y(t-2)+c0.
% compare WLS with OLS by mean square error between system
% output and model output or predictive output.
%
% iprog = 1 , one step prediction;
%       = 2 , model response.
% is = starting point of data series.
% ifi = final point of data series.
% x = input signal
% y = system output.
%
% model output or one step predictive output by WLS and OLS
%    is stored in ym(:,1) and ys(:,1) respectively .
% estimated parameters by WLS and OLS are stored in pm(1:5,:)
%    and ps(1:5,1) respectively.
%
%                t= 1  2  3 ...k
%
%                   b0 b0 b0...b0                  b0
%                   b1 b1 b1...b1                  b1
%       pm(1:5,t)=  a1 a1 a1...a1       ps(1:5,1)=a1
%                   a2 a2 a2...a2                  a2
%                   c0 c0 c0...c0                  c0
%
clear ym yml; ym(1,4000)=0; yml=ym;
ys=ym; ysl=yml;
temp=[0.,0.,0.,0.,0.]';
pm(:,4000)=temp;
ps= fit (ifi,2,1,x,y); % obtain OLS parameters.
for t=is+3:ifi
    if t==is+3
       for t1=1:2
           ym(t1)=y(t1);yml(t1)=y(t1);
           ys(t1)=y(t1);ysl(t1)=y(t1);
           errsq(t1)=0;err(t1)=0;
       end
       errsq(3)=0;
    end
    pml= fit2(t,1,1,x,y); % obtain model parameters.
    pm(:,t)=pml;
        ym(t)=pm(1,t)*x(t)+pm(2,t)*x(t-1) ...
                -pm(3,t)*y(t-1)-pm(4,t)*y(t-2)+pm(5,t);
        ys(t)=ps(1,1)*x(t)+ps(2,1)*x(t-1)...
               -ps(3,1)*y(t-1)-ps(4,1)*y(t-2)+ps(5,1);
        yml(t)=pm(1,t)*x(t)+pm(2,t)*x(t-1) ...
                -pm(3,t)*yml(t-1)-pm(4,t)*yml(t-2)+pm(5,t);
        ysl(t)=ps(1,1)*x(t)+ps(2,1)*x(t-1)...
               -ps(3,1)*ysl(t-1)-ps(4,1)*ysl(t-2)+ps(5,1);
```

```
    errsq(1)=(y(t)-ys(t))^2+errsq(1);
    errsq(2)=(y(t)-ym(t))^2+errsq(2);
    err(1)=(y(t)-ys1(t))^2+err(1);
    err(2)=(y(t)-ym1(t))^2+err(2);
    errsq(3)=y(t)^2+errsq(3);
end
eo=errsq(1)/errsq(3); % OLS MSE for prediction
ew=errsq(2)/errsq(3); % WLS MSE for prediction
eo1=err(1)/errsq(3); % OLS MSE for model response
ew1=err(2)/errsq(3); % WLS MSE for model response
eo=[eo,eo1];ew=[ew,ew1];
```

```
function p=fit2(n,iwp,noise,x,y)
format compact
clc;clg; % clear windows
% the function started 29/6/1988.
% estimate model parameters(b0,b1,a1,a2,c0) using data up
% to time n-1.
% system is approximated by model
% y(t)=b0*x(t)+b1*x(t-1)-a1*y(t-1)-a2*y(t-2)+c0
% weighted Least Squares or ordinary LS estimator is used.
%
% n = system present time.
% iwp = 1 , weighted LS estimator;
%     = default , ordinary  LS estimator.
% noise = 0 , noise free at input;
%       = 1 , noise superposed on input.
%
% estimated parameters are stored in p(:,1)=[b0,b1,a1,a2,c0]'.
%
% first find weights stored in w(:,1),
% model state s1=x(n),s2=x(n-1),s3=y(n-1),s4=y(n-2)
%
z=[x(3:n-1)',x(2:n-2)',y(2:n-2)',y(1:n-3)'];
s=[x(n),x(n-1),y(n-1),y(n-2)];
tmp=ones(n-3,1);
a1=[z(:,1),z(:,2),-z(:,3),-z(:,4),tmp]';

if iwp==1
   wx=(max(x(1:n))-min(x(1:n)))^2;wx=1/wx; %scaling
   wy=(max(y(1:n-1))-min(y(1:n-1)))^2;wy=1/wy;
   wsc=[wx,wx,wy,wy];ws=tmp*wsc;

   s=tmp*s; e=z-s; e=e.*ws;
   w=sum(e'.^2);
   i=find(w<=.0001); [n1,n2]=size(i);
   for t=1:n2
       w(i(t))=.0001;
   end
   w=tmp'./w;
% for noise free case omit second limit on weight.
    if noise==1
       av=sum(w)/(n-4);
       temp=100*av;
       i=find(w>temp);
       [n1,n2]=size(i);
       for t=1:n2
           w(i(t))=temp;
       end
    end
else
       w=tmp'; % ordinary weights.
end
% build up matrices.
sw=sqrt(w);
tw1=[sw;sw;sw;sw;sw];
b1=a1;
a1=tw1.*a1; a=a1*a1';
```

```
ty=y(3:n-1)'.*w'; b=b1*ty;

p=a\b; % obtain model parameters.
```

```
function [ym,pm]=jumod(x,y,ifi)
format compact
clc % clear windows.
% the function started 29/6/1988.
% obtain model parameters by a function jfit(n,iwp,noise,x,y).
% generate model response
% ym(t)=b0*x(t)+b1*x(t-1)-a1*ym(t-1)-a2*ym(t-2)+c0
% to predict jump resonance characteristics.
% is = starting point of model prediction.
% ifi = final point of model prediction.
% x = input signal
% y = system output.
%
% model output obtained by VWLS is stored in ym.
%
% estimated parameters by VWLS is stored in pm(1:5,:).
%
%
%                t= 1  2  3 ...k
%
%                   b0 b0 b0...b0
%                   b1 b1 b1...b1
%       pm(1:5,t)=  a1 a1 a1...a1
%                   a2 a2 a2...a2
%                   c0 c0 c0...c0
%
for t=ifi:ifi
    pm1=jfit(t,1,x,y); % obtain model parameters.
    pm=pm1';
                % one step prediction.
    ym=pm(1)*x(t)+pm(2)*x(t-1)-pm(3)*y(t-1)-pm(4)*y(t-2)+pm(5);
end
```

```
function p=jfit2(n,noise,x,y)
format compact
clc % clear windows
% the function started 29/6/1988.
% estimate model parameters(b0,b1,a1,a2,c0)
% using data up to time n-1.
% system is approximated by model
% y(t)=b0*x(t)+b1*x(t-1)-a1*y(t-1)-a2*y(t-2)+c0
% Weighted Least Squares or ordinary LS estimator is used.
%
% n = system present time.
% iwp = 1 , weighted LS estimator;
%     = default , ordinary  LS estimator.
% noise = 0 , noise free at input;
%       = 1 , noise superposed on input.
%
% estimated parameters are stored in p(:,1)=[b0,b1,a1,a2,c0]'.
%
% first find weights stored in w(:,1),
% model state s1=x(n),s2=x(n-1),s3=y(n-1),s4=y(n-2)
%
z=[x(3:n-1)',x(2:n-2)',y(2:n-2)',y(1:n-3)'];
s=[x(n),x(n-1),y(n-1),y(n-2)];
tmp=ones(n-3,1);
a1=[z(:,1),z(:,2),-z(:,3),-z(:,4),tmp]';

   s=tmp*s; e=z-s; w=sum(e'.^2);
   i=find(w<=.0001); [n1,n2]=size(i);
   for t=1:n2
       w(i(t))=.0001;
   end
   w=tmp'./w;
% for noise free case omit second limit on weight.
    if noise==1
       av=sum(w)/(n-4);
       temp=100*av;
       i=find(w>temp);
       [n1,n2]=size(i);
       for t=1:n2
           w(i(t))=temp;
       end
    end
% build up matrices.
sw=sqrt(w);
tw1=[sw;sw;sw;sw;sw];
b1=a1;
a1=tw1.*a1; a=a1*a1';
ty=y(3:n-1)'.*w'; b=b1*ty;

p=a\b; % obtain model parameters.
```

```
format compact
clc;clg; % clear windows.
% the script started 15/3/1989.
% this program is used to study General Predictive Control
% in NL system, which the plant can be modelled as
% Hammerstein one. the GPC strategy is shown as follows,
% the scheme is to obtain  x by linear GPC design, and then
% to calculate u by a root-solver, therefore this is
% an indirect design method.
%
%
%      ---->|Con.Desi|<-----| Esti|<--------------------
%      |    |________|      | mat |                |D      |
%      |          |         | or  |               __V__    |
%      |          |         |_____|             |       |   |
%      |          |          ^                  |_____|   |
%      |          |          |                      |       |
%      |         _V_____     |  ____   _______      |       |
%    W |      |           | U  | |    |X |       |   |       |    Y
%   ---|-O-->| Contro.  |----|-| NL  |--| B/A   |--O-------|--->
%       ^    |__________|      |____|   |_______|      |
%       |                                              |
%       |                                              |
%       |                                              |
%       |______________________________________________|
%
while 1
   clc;
   n=input('n>0 enter,n<=0 quit ');
   if n<=0;break,end
   '1,initialisation'
   '2,process running'
   '3,plots'
   i=input('your choice ');
   if i==1
      clc;clg;
      ns=input('lenght of sample');
      'nl part of Hammerstein model,1,2'
      nl=input('nl1=a1+u+a2u^2+a3u^3;nl2=u+a3u^3');
      'linear part of Hammerstein model,a,b'
      na=input('order of a=1+a1q(-1)+...+anaq(-na)');
      for i=1:na+1
          a(i)=input('=')
      end
      nb=input('order of b=b0+b1q(-1)+...+bnbq(-nb)');
      for i=1:nb+1
          b(i)=input('=')
      end
      'settings of ERLS estimator'
      lamd=input('forgetting factor');
      clc,
      temp=input('initial values of normal matrix');
      nm=3*(nb+1)+na+1;
      p=temp*eye(nm);
      v=input('noise variance');
      v=sqrt(v);
```

```
    'settings of controller'
    c1=input('control weighting');
    n2=input('max. output horizon');
    nu=input('control horizon');
    td=input('time delay');
    dn=input('default no. of sample at start >=10');
    'settings of root-solver'
    c3=input('derivative mornitoring factor')
    c4=input('root solution accuracy')
    ite=20*ones(1,20);itd=ite(1:10);
    w(1:10)=itd; %settings of setpoint sequence
    temp(1:20)=ite;temp(21:40)=3*ite;
    temp(41:60)=ite;temp(61:80)=0*ite;
    while length(w)<ns
       w=[w,temp];
    end
    w=w(1:ns+n2);
    crot=[c3,c4];x=zeros(1,4);y=x;du=zeros(1,dn+1);
    ab=[b,a];pm=zeros(1,nm);pm(1)=b(1);pm=pm';
    xm=du; phi=[]; u=zeros(1,4+dn+1);
elseif i==2
    clc;clg;
    u(4)=1;rand('normal');
    for t=4:ns
        cot=[nb,na,t,lamd,nl];cs=[nb,na,n2,nu,c1,t];
        [y1,x1]=plant(u,x,y,ab,cot);
        x(t)=x1;y(t)=y1+v*rand;
           %generate system output y(t) and
           %intermediate variable x(t)
                                % parameter estimates
        [paset,pm,phi,p]=paraest(u,y,pm,p,cot);
        bm=paset(1:nb+1);       % polynomial b
        am=paset(nb+2:nb+2+na); % polynomial a
                                % nonlinear part r
        rm=paset(nb+na+3:length(paset));
        if t>dn
           [g,g1,wf]=diopeq(w,y,du,cs,paset);
           [xm1,du(t)]=condes(g1,xm(t-1),wf,n2,t);
           xm(t)=xm1;
           ut1=u(t);
           ut=root(ut1,xm1,rm,crot);
           u(t+td)=ut;
        else
           if w(t+1)>=y(t)
              u(t+td)=1.0;
           else
              u(t+td)=-1.0;
           end
        end
    end

elseif i==3
    sp=input('length of plot');
    plot([w(1:sp)',y(1:sp)'])
    xlabel('time period'),ylabel('signal magnitude')
    text(.6,.9,'_____ reference','sc')
```

```
      text(.6,.8,'- - - plant output','sc')
      pause
      plo=input('store plot=1, otherwise default');
      if plo==1
         meta chap5
      end
      plot([u(1:sp)',xm(1:sp)'])
      xlabel('time period'),ylabel('signal magnitude')
      text(.6,.9,'_____ control input','sc')
      text(.6,.8,'- - - intermediate variable','sc')
      pause
      if plo==1
         meta chap5
      end
      pause
      'model para.'
      am,bm,rm, pause
   else
      'wrong number given,reset your choice 1,2,3,4,5'
   end
end
```

```
format compact
clc;clg; % clear windows.
% the script started 15/3/1989.
% this program is used to study DeadBeat Control in NL
% system, which the plant can be modelled as Hammerstein
% one. the DBC strategy is shown as follows, the scheme 1
% is to obtain x by linear DBC design, and then to calculate
% u by  a root-solver, therefore this is an indirect design
% method. the scheme 2 is to obtain directly u, i.e.
% so called direct method.
%
%
%      --->|Con.Desi|<-----| Esti|<--------------------
%      |   |_________|      | mat |                |D      |
%      |        |           | or  |               _V__    |
%      |        |           |_____|             |       |   |
%      |        |            ^                  |_____|   |
%      |        |            |                     |      |
%      |      __V_____       |   ____    _______   |      |
%    W |     |          | U  |  |    |X |       |  |      |    Y
%   ---|-O-->| Contro.  |----|-| NL |--| B/A   |--O------|--->
%      ^     |__________|    |  |____|  |_______|         |
%      |                                                 |
%      |                                                 |
%      |                                                 |
%      |_________________________________________________|
%
while 1
   clc;
   n=input('n>0 enter,n<=0 quit ');
   if n<=0;break,end
   '1,initialisation'
   '2,process running'
   '3,plots'
   i=input('your choice ');
   if i==1
      clc;clg;
      ns=input('lenght of sample');
      'nl part of Hammerstein model,1,2'
      nl=input('nl1=a1+u+a2u^2+a3u^3;nl2=u+a3u^3');
      'linear part of Hammerstein model,a,b'
      na=input('order of a=1+a1q(-1)+...+anaq(-na)');
      for i=1:na+1
          a(i)=input('=')
      end
      nb=input('order of b=b0+b1q(-1)+...+bnbq(-nb)');
      for i=1:nb+1
          b(i)=input('=')
      end
      'settings of ERLS estimator'
      lamd=input('forgetting factor');
      clc,
      temp=input('initial values of normal matrix');
      nm=3*(nb+1)+na+1;
      p=temp*eye(nm);
      v=input('noise variance');
```

```
      v=sqrt(v);
      'settings of controller'
      c1=input('controller magnitude clamper');
      td=input('time delay');
      dn=input('default no. of sample at start >=10');
      'settings of root-solver'
      c3=input('derivative mornitoring factor');
      c4=input('root solution accuracy');
      ite=20*ones(1,20);itd=ite(1:10);
      clear w
      w(1:10)=itd; %settings of setpoint sequence
      temp(1:20)=ite;temp(21:40)=3*ite;
      temp(41:60)=ite;temp(61:80)=0*ite;
      while length(w)<ns
         w=[w,temp];
      end
      w=w(1:ns);
      crot=[c3,c4];x=zeros(1,4);y=x;du=zeros(1,dn+1);
      ab=[b,a];pm=zeros(1,nm);pm(2)=b(1);pm=pm';u=zeros(1,4+dn+1);
      xm=du;phi=[];
      db=input('indir. method=1, dir. method=2');
   elseif i==2
      clc;clg;
      u(4)=1;rand('normal');
      for t=4:ns
          cot=[nb,na,t,lamd,nl];
          [y1,x1]=plant(u,x,y,ab,cot);
          x(t)=x1;y(t)=y1+v*rand;
                     %generate system output y(t) and
                     %intermediate variable x(t)
                                % parameter estimates
          [paset,pm,phi,p]=paraest(u,y,pm,p,cot);
          bm=paset(1:nb+1);        % polynomial b
          am=paset(nb+2:nb+2+na); % polynomial a
                                % nonlinear part r
          rm=paset(nb+na+3:length(paset));
          if t>dn
             tempx=xm(t-td:-1:t-nb-td);
             tempy=w(t:-1:t-na)-y(t:-1:t-na);
             if db==1
                xm(t)=dbc1(tempx,tempy,am,bm,c1);
                ut1=u(t);
                ut=root(ut1,xm(t),rm,crot);
             else
                ut1=u(t);
                [rc,xm(t)]=dbc2(tempy,pm,phi,cot);
                ut=root(ut1,xm(t),rc,crot);
                if abs(ut)>c1
                   ut=c1*sign(ut)
                end
             end
             u(t+td)=ut;
          else
             if w(t+1)>=y(t)
                u(t+td)=1.0;
             else
```

```
                u(t+td)=-1.0;
             end
          end
       end

   elseif i==3
      sp=input('length of plot');
      plot([w(1:sp)',y(1:sp)'])
      xlabel('time period'),ylabel('signal magnitude')
      text(.6,.9,'_____ reference','sc')
      text(.6,.8,'- - - plant output','sc')
      pause
      plo=input('store plot=1, otherwise default');
      if plo==1
         meta chap6
      end
      plot([u(1:sp)',xm(1:sp)'])
      xlabel('time period'),ylabel('signal magnitude')
      text(.6,.9,'_____ control input','sc')
      text(.6,.8,'- - - innovation variable','sc')
      pause
      if plo==1
         meta chap6
      end
      pause
      'model para.'
      am,bm,rm, pause
   else
      'wrong number given,reset your choice 1,2,3,4,5'
   end
end
```

```
function [y1,x1]=plant(u,x,y,ab,c)
format compact
clc;clg;
% the function started 15/3/1989
% this function generates an output y(t) of nl system
% and intermediate variable x(t) from input u(t),
% which consists of a nonlinear element single-valued
% followed by a linar dynamic element .
nb=c(1);na=c(2);t=c(3);nl=c(5);
b=ab(1:nb+1);a=ab(nb+2:nb+2+na);
if nl==1
   x(t)=1+u(t)-u(t)^2+.2*u(t)^3;
else
   x(t)=u(t)-u(t)^3;
end
x1=x(t);
tempx=x(t:-1:t-nb);tempy=y(t-1:-1:t-na);
y1=tempx*b'+tempy*a(2:na+1)';
```

```
function [theta,pm,phi,p]=paraest(u,y,pm,p,c)
format compact
clc;clg; % clear windows.
% the function started 15/3/1989.
% This function estimates parameters of Hammerstein
% model by an Enhanced Recursive Least Square method,
% Parameters estimated are stored in am(i),bm(i),
% rm(i) which are separated from original
% parameter vector.
  nb=c(1);na=c(2);t=c(3);lam=c(4);nl=c(5);
  tempu=u(t:-1:t-nb);
  phi=[1,tempu,tempu.^2,tempu.^3,-y(t-1:-1:t-na)]';
  leng=length(pm');
  l=p*phi/(1+phi'*p*phi); %
  p=(p-(p*phi*phi'*p)/(1+phi'*p*phi))/lam;
  pm=pm+l*(y(t)-phi'*pm);
                        % parameter are separated
  bm=pm(2:nb+2,1)';
  am(2:na+1)=pm(2+(nb+1)*3:leng,1)';am(1)=1;
  theta=[];
  for i=2:4
      theta=[theta;pm((i-2)*(nb+1)+2:(i-1)*(nb+1)+1,1)'];
  end
  rm(2:4)=(theta*bm'/(bm*bm'))';
  rm(1)=pm(1)/sum(bm);
  theta=[bm,am,rm];
```

```
function u1=root(u1,xm1,r,c)
format compact
clc;clg;
% the function started 15/3/1989
% this subprogram solves one of a polynomial roots by
% Newton-Raphson method, the root is stored in u1 and u2.
i3=0;i4=0;df=1;
c3=c(1);c4=c(2);x2=1;u2=u1; % best value c3=0.5, c4=1
while df<20
i=0;x2=abs(x2)+c3+1;
    while abs(x2)>c3 % derivative check
          i=i+1;
          temp1=[1,u1,u1^2,u1^3];
          temp2=[1,2*u1,3*u1^2];r2=r(2:4);
          x1=r*temp1'-xm1;x2=r2*temp2';
          if abs(x2)>c3 % derivative check
             u2=u1-x1/x2;
             u1=u2;
          end
          if i==5
             x2=0;
          end
    end
    u1=u2;x1=r*[1,u2,u2^2,u2^3]'-xm1;
    if abs(x1)>c4 % root accuracy check
       u1=xm1/(5+i3); u2=u1;
       i3=i3+5;i4=i4+1;
       if i4==10
          i3=-70;
       end
       if i4==20
          df=30;
          u2=xm1/20;u1=u2;
       end
    else
       df=30;
    end
end
```

```
function [g,g1,wf]=diopeq(w,y,du,c,pa)
format compact
clc;clg;
% the function started 15/3/1989
% this function gives solution of Diophantine equation
% by recursive method . the equation has form like
%
%          1=E*A*Delt + qPow(-j)*F
%
% where Delt = 1-qPow(-1)
%       qPow(-j) means q to the minus j

% the second role is to build up matrix G.
nb=c(1);na=c(2);n2=c(3);nu=c(4);c1=c(5);t=c(6);
bm=pa(1:nb+1);am=pa(nb+2:nb+na+2);
e(1)=1;%initialize polynomial e
adelt(2:na+1)=am(2:na+1)-am(1:na);
adelt(1)=1;adelt(na+2)=-am(na+1);
f=-adelt(2:na+2);%initialize polynomial f
eb=bm;eb(nb+1+n2)=0;
g3=bm(2:nb+1);g3(nb+1+n2)=0;
for j=1:n2
    e(j+1)=f(1); % e
    f(1:na)=f(2:na+1)-adelt(2:na+1)*e(j+1);% f
    f(na+1)=-adelt(na+2)*e(j+1);
    for j1=1:nb+1+j
        if j1>=j+1
           eb(j1)=eb(j1)+f(1)*bm(j1-j);
        end
    end
        g2=eb(j+1:nb+1+j);
        wf(j)=g2*du(t-1:-1:t-nb-1)'+f*y(t:-1:t-na)';
        wf(j)=w(t+j)-wf(j);%difference between setpoint
                           %and predicted step response
end
g=eb(1:n2)'; g3=g; %build up matrix g
for i=1:n2-1
    g4(i)=0;g4(i+1:n2)=g3(1:n2-i);
    g=[g,g4'];
end
g=g(:,1:nu);% consideration for control horizon
                          % build up matrix "g'*g-c1*unit"
unit=eye(nu);g1=g'*g-c1*unit;
g1=inv(g1)*g';
```

```
function[xm1,du]=condes(g1,xm2,wf,n2,t)
format compact
clc;clg;
% the function started 15/3/1989
% the function calculates the intermediate variable
% x(t) by linear gpc design% procedure, which
% will be used to find nlgpc controller output u(t)
% by root-solving routine (root.m).
xm3=g1(1,:)*wf';
xm1=xm2+xm3;
if abs(xm1)>100
   xm1=100*sign(xm1);
end
du=xm1-xm2;
```

```
function xm1=dbc1(x,y,am,bm,c1)
format compact
clc;clg;
% the function started 20/3/1989.
% this function is used to design indirectly
% a deadbeat controller with an algorithm given by
%
%          X(t)=B*X(t-k)/B(1) + A*[W(t)-Y(t)]/B(1)
xm1=bm*x'+am*y';
bs=sum(bm);
if abs(bs)<=0.01
   bs=0.01*sign(bs);
end
xm1=xm1/bs;
if abs(xm1)>=c1
   xm1=c1*sign(xm1);
end
```

```
function [rc,xm1]=dbc2(tempy,pm,phi,c)
format compact
clc;clg;
% the function started 20/3/1989.
% this function is used to design directly
% a deadbeat controller with an algorithm given by
%
%          B(1)U(t)=B*U(t-k) + A*[W(t)-Y(t)]
n=length(pm); rc=0;
nb=c(1);na=c(2);
am(1)=1;am(2:na+1)=pm(n-na+1:n,1)';
for i=1:3
    rc=[rc,sum(pm(2+(i-1)*(nb+1):1+i*(nb+1),1)')];
end
xm1=pm(2:n-na,1)'*phi(2:n-na,1)+am*tempy';
```