MULTIVARIATE STATISTICAL REGRESSION METHODS FOR PROCESS MODELLING AND EXPERIMENTAL DESIGN

BY

BHUPINDER SINGH DAYAL, M. Eng.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree of
Doctor of Philosophy

McMaster University

MULTIVARIATE STATISTICAL REGRESSION METHODS FOR PROCESS MODELLING AND EXPERIMENTAL DESIGN

... to my uncles, Gurdial and Charan.

DOCTOR OF PHILOSOPHY (1996) McMASTER UNIVERSITY

(Chemical Engineering) Hamilton, Ontario

TITLE: Multivariate Statistical Regression Methods for Process Modelling and Experimental Design

AUTHOR: Bhupinder Singh Dayal,
B.A.Sc (University of British Columbia)
M. Eng. (McMaster University)

SUPERVISOR: Professor John F. MacGregor

NUMBER OF PAGES: xvi, 154

Abstract

In this thesis, various multivariate statistical regression methods are investigated for estimating process models from the process input-output data. These identified models are to be used for designing model based controllers and experimental optimisation of multivariate processes. The following issues are explored: (i) identification of finite impulse response models for model based control; (ii) multi-output identification for multivariate processes; (iii) recursive updating of process models for adaptive control and prediction; and (iv) experimental design in latent variables for high dimensional systems.

In the first part of the thesis, various approaches to identifying non-parsimonious finite impulse response (FIR) models are compared on the basis of closeness of fit to the true process, robust stability provided by the resulting model, and the control performance obtained. The major conclusion by all assessments is that obtaining FIR models by first identifying low order transfer function models by prediction error methods is much superior to any of the methods which directly identify the FIR models.

In the second part, the potential of multi-output identification for multivariate processes is investigated via simulations on two process examples: a quality control example and an extractive distillation column. The identification of both the parsimonious transfer function models using multivariate prediction error methods and of non-parsimonious FIR models using multivariate statistical regression methods such as two-block partial least squares (PLS2), canonical correlation regression (CCR), reduced rank regression (RRR) are considered. The multi-output identification methods provide better results when compared to the single-output identification methods based on essentially all comparison criteria. The benefits for using multi-output identification are most obvious when there are limited amount of data and when the secondary output variables have better signal to noise ratios.

In the third part of this thesis, an improvement to the PLS algorithm is made. It is shown that only one of either the **X** or the **Y** matrix needs to be deflated during the sequential process of computing latent vectors. This result then leads to two very fast PLS kernel algorithms. Using these improved kernel algorithms, a new and fast recursive, exponentially weighted PLS algorithm is developed. The recursive PLS algorithm provides much better performance than the recursive least squares algorithm when applied to adaptive control of a simulated 2 by 2 multivariable continuous stirred tank reactor and updating of a multi-output prediction model for an industrial mineral flotation circuit.

Finally, a design methodology similar to the evolutionary operation (EVOP) and the response surface methodology (RSM) for optimisation of high dimensional system is proposed. A variation of the PLS algorithm, called selective PLS, is developed. It can be used to analyse the process data and select meaningful groupings of the process variables in which the EVOP/RSM experiments can be performed.

Acknowledgements

I would like to express my greatest gratitude to my supervisor, Professor John F. MacGregor, for his continuous guidance, interest and kindly encouragement throughout the course of this study. It has been a great pleasure to work with him for the last few years.

I would also like to thank the members of my supervisory committee, Professor Paul Taylor and Professor M. Elbestawi for their many helpful suggestions and insights that enhanced the quality of this work. The financial support from the Natural Science and Engineering Research Council and the Department of Chemical Engineering is greatly appreciated.

During my stay at McMaster University, I had an opportunity to make friends from all corners of the world. Special thanks go to Thanassis Kassidas, Sridhar Sampath, Panos Seferlis, Phil Nelson, Andre Almeida, François Boudreau, Christiane Jaeckle, Ivan Miletic, Tracy Clarke-Pringle and Barbara Owen. I will definitely miss the lively, intelligent and mind provoking discussions on almost any topic.

I am indebted forever to my uncles, Gurdial Dayal and Charan Dayal, and their families who gave me everything but never expected anything in return.

Finally, I would like to express my deepest gratitude to my wife, Ravinder, for her love, support and patience over the last two and half years.

Table of Contents

List of Figures

List of Tables

1. Introduction

In this thesis, various multivariate statistical regression methods are investigated for estimating process models from the process input-output data for designing model based controllers and for experimental optimisation of multivariate processes. The following issues are explored: (i) identification of finite impulse response models for model based control; (ii) multi-output identification for multivariable processes; (iii) recursive updating of process models for adaptive control and predictions; and (iv) experimental design in latent variables for high dimensional systems.

In model predictive control one often needs a finite impulse response (FIR) or step response model of the process. The non-parsimonious FIR models can be estimated directly from the process input-output using methods such as ordinary least squares, biased methods such as regularised least squares and partial least squares. Alternatively, a low order parsimonious rational transfer function model can be fitted to the process input-output data by prediction error methods and then the FIR model can be obtained from it. Although, directly identifying non-parsimonious FIR models have certain advantages (i.e, they can fit any complex linear dynamic system; there is no need for model structure selection), the trade-offs involved in identifying the FIR models directly using various regression methods versus first identifying low order parsimonious models by prediction error methods have not been well documented. In chapter 2, the identification of non-parsimonious FIR models using methods such as ordinary least squares, regularised least squares and partial least squares are compared with each other and with a parametric modelling of parsimonious transfer functions. The comparisons are made on the basis of (i) the closeness of the fit to the true model; (ii) the level of robust stability provided by the identified model; and (iii) the actual control performance obtained using the identified models. Although many of the points raised in chapter 2 are appreciated in a general way by the control community, it tries to quantify them on a reasonable basis.

In model based control of multivariate processes, it has been common practice to identify Multi-Input Single-Output (MISO) model for each output separately and then combine the individual models into a final Multi-Input Multi-Output (MIMO) model. If the models for all outputs are independently parameterised then this approach is optimal. However, if there are common or correlated parameters among models for different output variables and/or correlated noise, then performing identification on all outputs simultaneously can lead to better and more robust models.

In chapter 3, the potential of multi-output identification for multivariate processes is investigated via simulations on two process examples: a quality control example and an extractive distillation column. The identification of both the parsimonious transfer function models using multivariate prediction error methods, and of non-parsimonious finite impulse response models using multivariate statistical regression methods such as two-block partial least squares (PLS2), canonical correlation regression (CCR) and reduced rank regression (RRR) are considered. The multi-output identification results are compared to traditional single-output identification from several points of view: (i) best predictions; (ii) closeness of the model to the true process; (iii) the precision of the identified models in the frequency domain; (iv) stability robustness of the resulting model based control system; and (v) the multivariable control performance.

In chapter 4, an improvement to the PLS algorithms is considered. A proof is given that only one of either the $\mathbf{X}$ or the $\mathbf{Y}$ matrix in PLS algorithms needs to be deflated during the sequential process of computing latent vectors. With the aid of this proof, the original kernel algorithm developed by Lindgren et al. (1993) is modified to provide two faster and more economical algorithms. In the first algorithm, the covariance matrix $\mathbf{X}^T\mathbf{X}$ is not computed and $\mathbf{X}$ is used directly in computations for the loading vectors. In the second algorithm, the covariance matrix $\mathbf{X}^T\mathbf{X}$ is computed once and then used subsequently in the computations for the loading vectors. The performances of these new algorithms are compared to that of De Jong and Ter Braak's (1994) modified kernel

algorithm in terms of speed. Furthermore, the advantages of these new algorithms for performing cross-validation or treating missing data are also discussed.

In chapter 5, a new and fast recursive, exponentially weighted PLS algorithm which provides greatly improved parameter estimates in most process situations is presented. This new recursive algorithm for updating of the PLS regression model is developed by combining the improved kernel algorithm developed in chapter 4 with the recursive updating of the covariance matrices $\left(\mathbf{X}^T\mathbf{X}\right)_t$ and $\left(\mathbf{X}^T\mathbf{Y}\right)_t$. The potential of the recursive PLS algorithm is illustrated with two process examples: (i) adaptive control of a 2 by 2 simulated multivariable continuous stirred tank reactor; and (ii) updating of a prediction model for an industrial flotation circuit. The performance of the recursive PLS algorithm is also compared to that of the recursive least squares algorithm.

In chapter 6, a design methodology similar to the evolutionary operation (EVOP) (Box, 1957) and the response surface methodology (RSM) (Box and Wilson, 1951) for optimisation of high dimensional systems is proposed. A variation of the PLS algorithm, called selective PLS, is developed. It can be used as a tool to select meaningful groupings of variables (latent variables) in which the EVOP/RSM experiments can be performed. It is a new methodology which can be used to select orthogonal variables consisting of linear combinations of alike manipulated variables by analysing the historical plant data. Since most of the process variables are moved in very few directions, therefore, selective PLS could be applied to find these underlying directions. Furthermore, a sequential design methodology in these groupings of variables is also proposed.

2. Identification of FIR Models: Methods and Robustness Issues

2.1 Introduction

In designing model predictive control schemes (e.g., DMC), step or impulse response models of the process relating manipulated variables to controlled variables are required. The non-parsimonious finite impulse response (FIR) models can be estimated directly from the process input-output data using methods such as ordinary least squares (OLS), regularised least squares methods, (e.g., ridge regression (RR)) and partial least squares (PLS). Alternatively, a low order parsimonious rational transfer function model can be fitted to the process input-output data by prediction error methods and then the finite impulse response model can be obtained from it. Identifying parsimonious transfer function models using maximum likelihood estimation or general prediction error methods is a well established field (Box & Jenkins, 1976; Ljung, 1987; and Soderstrom and Stoica, 1989). Although, the open-loop behaviour of most individual unit operations in chemical processes can be modelled sufficiently well with low order transfer functions, the dynamic behaviour of these units coupled with low level controllers, or of a whole section of a process, consisting of interconnected units with recycle, etc., often can not be modelled easily with such low order models. Furthermore, to identify low order transfer function models one must first identify the appropriate model structure (number of zeros and poles, and dead-time) and then check the adequacy of this structure. Directly identifying non-parsimonious FIR models overcomes these problems. With a sufficiently high order, FIR models have sufficient flexibility to model linear systems of any complexity, and other than the total number of terms to include (time to steady state), no structural decisions need to be made. However, as will be shown in this chapter, along with these advantages come

some substantial disadvantages such as poor robustness and performance of the resulting controllers designed using the identified FIR models.

Several articles have been published on the direct estimation of FIR models using methods such as PLS, PCR and regularised least squares (Ricker (1988), MacGregor et al. (1991), Wise and Ricker (1992) and Wise and Ricker (1993)). These methods are well suited to handling the highly correlated structure of the lagged input models. Ricker (1988) investigated the use of PLS and SVD (singular value decomposition) methods for estimation of FIR model coefficients for a simulated process and an anaerobic wastewater treatment process. The SVD method applied by Ricker (1988) is same as principal component regression (PCR). PCR consists of performing a principal component analysis (PCA) on the inputs and then regressing the outputs on the singular vectors or principal components which are linear combinations of the inputs. His conclusions were that PLS performed poorly in determining the dead time of the process. This has also been confirmed by MacGregor et al. (1991). If the dead time of the process is known a priori, then this information could be built into FIR models to improve the estimation of the impulse response coefficients. MacGregor et al. (1991) investigated the use of PLS along with various regularised least squares methods such as ridge regression (RR) and ordinary least squares (OLS) to obtain FIR model coefficients estimates for a simulated MISO process with 5 inputs and a single output under many different conditions (i.e., amount of data, independent and correlated perturbations to the input variables, etc.). Both PLS and RR provided comparable but better estimates than OLS. PLS also had problems determining a good estimate for the process dead time unless extra latent vectors beyond that suggested by cross-validation were computed.

Although, several algorithms have been proposed for the direct identification of non-parsimonious FIR models, the trade-offs involved in using these methods versus first identifying low order parsimonious models by prediction error methods and then obtaining the FIR models from them have not been well documented. In this chapter, the identification of non-parsimonious FIR and ARX models using methods such as OLS,

regularised least squares and PLS are compared with each other and with parametric modelling of parsimonious transfer functions. The comparisons are made on the basis of (i) closeness of fit to the true model; (ii) stability robustness of the resulting controller to identification errors; and (iii) performance of the resulting controller.

2.2 Model Structures for Identification

2.2.1 FIR Models

For a linear process model, the process output variable (y_t) can be expressed as a linear combination of inputs consisting of lagged process input variables (u_t)

$$y_t = \sum_{j=1}^{n_u} \beta_j(z^{-1})\, u_{j,t} + D_t = \sum_{j=1}^{n_u} \beta_{j,1} u_{j,t-1} + \beta_{j,2} u_{j,t-2} + \ldots + \beta_{j,p_j} u_{j,t-p_j} + D_t \qquad (2.1)$$

where $\beta_j(z^{-1}) = \left[\beta_{j,1} \quad \beta_{j,2} \quad \cdots \quad \beta_{j,p_j}\right]^T$ is a vector of process impulse response model coefficients associated with j^{th} input variable. The number of coefficients being estimated in an impulse response model for a given input variable should be such that the process is within about one percent of its settling time when a step change is made to that specific input variable. Due to the process dead-times existing between the input and output variables, the first few impulse coefficients will be actually zero. If the periods of dead-times are known a priori then the data can be shifted to remove the dead times. D_t is the process disturbance which could be either white noise (i.e., $D_t=a_t$) or coloured noise (i.e., represented by an ARIMA model (Box and Jenkins, 1976)).

In the case of D_t being white noise ($D_t=a_t$), one can estimate the impulse response coefficients using Equation (2. 1) with any regression method. However, if D_t is coloured noise, then the disturbance model also needs to be identified along with the process model. The generalised least squares method (Clarke, 1967) is a convenient approach. Now the model being identified becomes

$$y_t = \sum_{j=1}^{n_u} \beta_j(z^{-1})\, u_{j,t} + \frac{1}{\phi(z^{-1})} a_t \qquad (2.\ 2)$$

where $1/\phi(z^{-1})$ is the disturbance model and a_t is the white noise. Equation (2. 2) can be reformulated to facilitate the estimation of the process impulse response model coefficients.

$$\phi(z^{-1})y_t = \sum_{j=1}^{n_u} \beta_j(z^{-1})\phi(z^{-1})u_{j,t} + a_t \qquad (2.\ 3)$$

By letting $y_t^F = \phi(z^{-1})y_t$ and $u_{j,t}^F = \phi(z^{-1})u_{j,t}$ and rewriting Equation (2. 3),

$$y_t^F = \sum_{j=1}^{n_u} \beta_j(z^{-1})u_{j,t}^F + a_t \qquad (2.\ 4)$$

To start the identification process, one can assume an adequate model structure for the disturbance model. Using initial guesses for the parameters in the disturbance model, the process output (y_t) and input (u_t) can be filtered with $1/\phi(z^{-1})$ to obtain y_t^F and u_t^F. Now, y_t^F and u_t^F can be used to obtain estimates of $\beta(z^{-1})$. Once, the estimates of $\beta(z^{-1})$, $b(z^{-1})$, are available, then the estimate of disturbance, $\hat{D}_t$, can be obtained using the following equation:

$$\hat{D}_t = y_t - \sum_{j=1}^{n_u} b_j(z^{-1})u_{j,t} \qquad (2.\ 5)$$

The disturbance model parameters $\phi(z^{-1})$ can then be estimated using OLS or any of the other linear methods on $\phi(z^{-1})\hat{D}_t = a_t$. The new disturbance model parameters are now used to filter the process input and output data and the above procedure can be repeated until the parameters in $\beta(z^{-1})$ and $\phi(z^{-1})$ have converged.

In Equation (2. 1), y_t is expressed as function of lagged values of process input variables (u_t) for one observation. For n observations, the process data can be arranged into matrix form

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon \qquad (2.\ 6)$$

where y is a vector containing the observations for the output, **X** is a matrix containing the input data, β is an unknown vector of regression coefficients to be estimated from the input-output data and ε is a vector of model errors.

$$\mathbf{y}=\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ \vdots \\ y^n \end{bmatrix} \quad \mathbf{X}=\begin{bmatrix} u^1_{1,t-1} & u^1_{1,t-2} & \cdots & u^1_{1,t-p_1} & \cdots & u^1_{n_u,t-1} & \cdots & u^1_{n_u,t-p_{n_u}} \\ u^2_{1,t-1} & u^2_{1,t-2} & \cdots & u^2_{1,t-p_1} & \cdots & u^2_{n_u,t-1} & \cdots & u^2_{n_u,t-p_{n_u}} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ u^n_{1,t-1} & u^n_{1,t-2} & \cdots & u^n_{1,t-p_1} & \cdots & u^n_{n_u,t-1} & \cdots & u^n_{n_u,t-p_{n_u}} \end{bmatrix} \quad \varepsilon=\begin{bmatrix} \varepsilon^1 \\ \varepsilon^2 \\ \vdots \\ \vdots \\ \varepsilon^n \end{bmatrix} \qquad (2.7)$$

$$\beta=\begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,p_1} & \cdots & \beta_{n_u,1} & \cdots & \beta_{n_u,p_{n_u}} \end{bmatrix}^T$$

2.2.2 ARX Models

Alternatively, one can also identify an ARX (autoregressive with exogenous variable) model of the following form:

$$A(z^{-1})y_t = \sum_{j=1}^{n_u} B_j(z^{-1})u_{j,t} + a_t \qquad (2.8)$$

where $A(z^{-1})=(1+a_1z^{-1}+a_2z^{-2}+...+a_{na}z^{-na})$ and $B(z^{-1})=(b_1z^{-1}+b_2z^{-2}+...+b_{nb}z^{-nb})$. This model form offers some flexibility by introducing some structure into the model and requires fewer parameters to be estimated than the FIR model form. Autocorrelated disturbances are effectively modelled as $D_t = A^{-1}(z^{-1})a_t$. Furthermore, ordinary linear regression methods can be used directly on the model form. The impulse response can then be obtained from the identified ARX model.

2.3 Regression Methods

A brief overview of the regression methods to be investigated in this study for estimation of FIR model coefficients is given here. More details can be found in the references.

2.3.1 Ordinary Least Squares

For any linear model in the form of Equation (2. 6), the ordinary least squares (OLS) minimises the following objective function:

$$\min_{\mathbf{b}} (\mathbf{y}-\mathbf{X}\mathbf{b})^{T}(\mathbf{y}-\mathbf{X}\mathbf{b}) \tag{2. 9}$$

Assuming that the inverse of $\mathbf{X}^{T}\mathbf{X}$ exists (i.e., it is of full rank), the least squares estimates are given by

$$\mathbf{b} = \left(\mathbf{X}^{T}\mathbf{X}\right)^{-1}\mathbf{X}^{T}\mathbf{y} \tag{2. 10}$$

If the inputs are independent of the errors, then OLS solution is the unbiased estimator. The expected total mean squares error (MSE) for all the regression coefficient estimates is given by (Myers, 1990),

$$E\left[(\beta-\mathbf{b})^{T}(\beta-\mathbf{b})\right] = \sigma^{2}\text{trace}\left(\mathbf{X}^{T}\mathbf{X}\right)^{-1} = \sigma^{2}\sum_{i=1}^{p}\frac{1}{\lambda_{i}} \tag{2. 11}$$

where λ_i is the i^{th} eigenvalue of $\mathbf{X}^{T}\mathbf{X}$. If, as usual for process data, the columns of $\mathbf{X}$ are highly correlated, then $\mathbf{X}^{T}\mathbf{X}$ is highly ill-conditioned and some of the eigenvalues of $\mathbf{X}^{T}\mathbf{X}$ will be nearly zero. Therefore, it is evident from Equation (2. 11) that the eigenvalues close to zero will inflate the expected value of MSE for the regression coefficients and also for any resulting predictions from the model.

2.3.2 Regularisation or Constrained Least Squares

The inflation of the MSE for the regression coefficient estimates due to ill-conditioning of $\mathbf{X}^{T}\mathbf{X}$ can be reduced by imposing some type of constraint on the regression coefficient estimator $\mathbf{b}$ (Hoerl and Kennard, 1970a). The objective function of such regularised least squares methods is,

$$\min_{\mathbf{b}} (\mathbf{y}-\mathbf{X}\mathbf{b})^{T}(\mathbf{y}-\mathbf{X}\mathbf{b}) + k\,\mathbf{b}^{T}\mathbf{H}\mathbf{b} \tag{2. 12}$$

where k is a scalar and $\mathbf{H}$ is the constraint or penalty matrix for the estimator $\mathbf{b}$. Taking the first derivative of the objective function in Equation (2. 12) and equating it to zero and by solving for $\mathbf{b}$, one obtains the following estimates,

$$\mathbf{b} = \left(\mathbf{X}^T\mathbf{X} + k\,\mathbf{H}\right)^{-1}\mathbf{X}^T\mathbf{y} \qquad (2.\ 13)$$

2.3.2.1 Ridge Regression: a Constraint on Magnitude of the Regression Coefficients

Various versions of regularised least squares arise from different choices of **H**. If one were to choose **H** equal to the identity matrix (**H**=**I**), the objective function in Equation (2. 12) will minimise the sum of squares of the residuals (RSS) subject to a constraint on the magnitude or length of the regression estimates **b**. These estimates are referred to as ridge regression estimates $\mathbf{b}_{RR}$ (Hoerl and Kennard, 1970a) and are given by the following expression,

$$\mathbf{b}_{RR} = \left(\mathbf{X}^T\mathbf{X} + k\,\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} \qquad (2.\ 14)$$

It can be shown that $\mathbf{b}_{RR}$ is a biased estimate of β. The mean square error for the ridge regression estimates is given by (Hoerl and Kennard, 1970a),

$$E\left[\left(\beta - \mathbf{b}_{RR}\right)^T\left(\beta - \mathbf{b}_{RR}\right)\right] = \sigma^2\sum_{i=1}^{p}\frac{\lambda_i}{\left(\lambda_i + k\right)^2} + k^2\beta^T\left(\mathbf{X}^T\mathbf{X} + k\mathbf{I}\right)^{-2}\beta \qquad (2.\ 15)$$

The first term in the above equation is the sum of the variances of the ridge regression estimates. The second term is the bias introduced by the ridge regression. The bias increases with increasing k, however, the sum of the variances usually decreases much more rapidly with k. Therefore, for some positive value of k, the mean square error of ridge regression estimates will be smaller than that of OLS estimates. The ridge parameter, k, can be selected in several ways - using a ridge trace (Hoerld and Kennard, 1970b) or using cross-validation (Myers, 1990).

2.3.2.2 Regularisation with a Constraint on the Change in the Regression Coefficients

For estimating impulse weights a very appealing way to regularise the least squares solution is to penalise the size of the changes in the regression coefficients (MacGregor et al., 1991). For most processes, we expect that the impulse weights will change in a

smooth manner and so the change in two successive impulse response weights should be small.

To achieve this, the penalty or constraint matrix (**H**) in Equation (2. 13) can be taken to be $\mathbf{H}=\mathbf{A}^T\mathbf{A}$ where **A** is the first difference matrix given below:

$$\mathbf{A}=\begin{bmatrix} 1 & 0 & 0 & . & . & 0 \\ -1 & 1 & 0 & . & . & 0 \\ 0 & -1 & 1 & . & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & 0 & . & -1 & 1 \end{bmatrix} \qquad (2.\ 16)$$

The resulting regularised estimates from this method (RRD) are denoted by $\mathbf{b}_{RRD}$.

An additional enhancement to the above regularisation can be achieved if one also expects the changes in the successive impulse weights $(\beta_j - \beta_{j-1})$ to also become progressively smaller with increasing lag j. This is expected of most overdampled systems. Such prior knowledge can be incorporated into the regularisation solution by $\mathbf{H}=\mathbf{A}^T\mathbf{L}\mathbf{A}$ where **L** is a weighting matrix with linearly increasing weights given by (Kozub, 1994),

$$\mathbf{L}=\begin{bmatrix} 1 & 0 & 0 & . & . \\ 0 & 2 & 0 & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & 0 & . & p \end{bmatrix} \qquad (2.\ 17)$$

This method is denoted by RRDlin in the following sections.

2.3.3 Partial Least Squares

Partial Least Squares (PLS) is a multivariable regression method ideally suited to studying the variation in large numbers of highly correlated process variables (**X**) and relating them to a set of output variables (**Y**). PLS handles this by projecting the information in the data down into a low dimensional space defined by a small number of

latent vectors (t_1, t_2, ..., t_A). These new latent vectors summarise the information contained in the original data set. The scaled and mean-centred **X** and **Y** matrices are represented in PLS as

$$\mathbf{X} = \sum_{a=1}^{A} \mathbf{t}_a \mathbf{p}_a^T + \mathbf{E} \tag{2. 18}$$

$$\mathbf{Y} = \sum_{a=1}^{A} \mathbf{t}_a \mathbf{q}_a^T + \mathbf{F} \tag{2. 19}$$

where $\mathbf{t}_a$ are the latent vectors calculated sequentially for each dimension a=1,2,...,A. **E** and **F** are the residual matrices for **X** and **Y**, respectively. For models to be used for predicting **Y**, cross-validation is usually used to select the number of latent vectors (Wold, 1978).

The latent vectors in PLS can be computed by either the classical NIPALS (the Nonlinear Iterative Partial Least Squares) algorithm (Wold, 1982) or a kernel algorithm (Lindgren et al., 1993). The latent vectors are computed in a sequential manner. A typical PLS algorithm is as follows:

(i). Mean-centre and scale **X** and **Y**.

(ii). Compute the following quantities: $\mathbf{w}_a$, $\mathbf{t}_a$, $\mathbf{q}_a$, $\mathbf{u}_a$ and $\mathbf{p}_a$ using either the NIPALS algorithm or a kernel algorithm.

(iii). Deflate **X** and **Y** by subtracting the computed latent vectors from them.

$$\mathbf{X}_{a+1} = \mathbf{X}_a - \mathbf{t}_a \mathbf{p}_a^T \tag{2. 20}$$

$$\mathbf{Y}_{a+1} = \mathbf{Y}_a - \mathbf{t}_a \mathbf{q}_a^T \tag{2. 21}$$

(iv). Go to step (ii) to compute the next latent vector.

The PLS regression can be viewed as maximising the covariance between the linear combinations of $\mathbf{X}_a$ defined by $\mathbf{t}_a=\mathbf{X}_a\mathbf{w}_a$ and the output measurement matrix, $\mathbf{Y}_a$, at each dimension. The vector $\mathbf{w}_a$ is the weight vector for a^{th} dimension whose elements express the contribution of each variable in $\mathbf{X}_a$ toward defining the new latent vector $\mathbf{t}_a$. In univariate PLS (PLS1) , the squared variance of $\mathbf{t}_a$ with $\mathbf{y}_a$ is maximised whereas, in

multivariate PLS (PLS2), the sum of squared covariances of $\mathbf{t}_a$ with the columns of $\mathbf{Y}_a$ is maximised. The objective function to be maximised is

$$\max \ \mathrm{cov}^2(\mathbf{t}_a, \mathbf{Y}_a) \tag{2. 22}$$

Since $\mathbf{t}_a=\mathbf{X}_a\mathbf{w}_a$, therefore,

$$\begin{aligned} &\max \ \mathbf{w}_a^T\mathbf{X}_a^T\mathbf{Y}_a\mathbf{Y}_a^T\mathbf{X}_a\mathbf{w}_a \\ &\text{s.t. } \mathbf{w}_a^T\mathbf{w}_a = 1 \end{aligned} \tag{2. 23}$$

For the above equation to be at its maximum, $\mathbf{w}_a$ must be the eigenvector associated with the largest eigenvalue of $\mathbf{X}_a^T\mathbf{Y}_a\mathbf{Y}_a^T\mathbf{X}_a$. Similarly, the $\mathbf{q}_a$, $\mathbf{t}_a$, and $\mathbf{u}_a$ are the eigenvectors of the matrices $\mathbf{Y}_a^T\mathbf{X}_a\mathbf{X}_a^T\mathbf{Y}_a$, $\mathbf{X}_a\mathbf{X}_a^T\mathbf{Y}_a\mathbf{Y}_a^T$ and $\mathbf{Y}_a\mathbf{Y}_a^T\mathbf{X}_a\mathbf{X}_a^T$, respectively. Once, $\mathbf{w}_a$ has been computed, the remaining latent vectors can be computed as follows:

$$\mathbf{t}_a = \mathbf{X}_a\mathbf{w}_a \tag{2. 24}$$

$$\mathbf{p}_a = \frac{\mathbf{X}_a^T\mathbf{t}_a}{\mathbf{t}_a^T\mathbf{t}_a} \tag{2. 25}$$

$$\mathbf{q}_a = \frac{\mathbf{Y}_a^T\mathbf{t}_a}{\mathbf{t}_a^T\mathbf{t}_a} \tag{2. 26}$$

$$\mathbf{u}_a = \frac{\mathbf{Y}_a\mathbf{q}_a}{\mathbf{q}_a^T\mathbf{q}_a} \tag{2. 27}$$

The vectors $\mathbf{p}_a$ and $\mathbf{q}_a$ are the loading vectors for X and Y-variables, respectively. The vectors $\mathbf{t}_a$ and $\mathbf{u}_a$ are the score vectors for the X and Y, respectively.

There are four major properties of the partial least squares regression method.

1. **W** are mutually orthogonal (i.e., $\left(\mathbf{w}_i, \mathbf{w}_j\right) = \mathbf{w}_i^T\mathbf{w}_j = 0$ for $i \neq j$).
2. **T** are mutually orthogonal (i.e., $\left(\mathbf{t}_i, \mathbf{t}_j\right) = \mathbf{t}_i^T\mathbf{t}_j = 0$ for $i \neq j$).
3. $\mathbf{W}^T\mathbf{P}$ is a lower triangular matrix (i.e., $\mathbf{w}_i^T\mathbf{p}_j = 0$ for $i > j$).
4. $\mathbf{T}^T\mathbf{U}$ is a lower triangular matrix (i.e., $\mathbf{t}_i^T\mathbf{u}_j = 0$ for $i > j$).

The final PLS prediction model can be expressed in the terms of the original **X** variables as

$$\hat{\mathbf{Y}} = \mathbf{Xb} + \mathbf{F} \tag{2. 28}$$

where **b** is given by

$$\mathbf{b} = \mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1}\mathbf{Q}^T \tag{2. 29}$$

An overview of PLS is provided in Geladi and Kowalski (1986).

2.4 Parsimonious Transfer Function Modelling

Instead of directly identifying the impulse response weights by fitting an FIR model, parsimonious transfer function models can be identified, and the impulse weights then obtained from them. In this section, it is shown that the variance-covariance matrix for a parsimonious transfer function model estimates is better conditioned which, in turn, gives smaller confidence intervals than a directly fitted FIR model.

For example, consider the following process model structure,

$$y_t = \delta_1 y_{t-1} + \omega_0 u_{t-1} + a_t \tag{2. 30}$$

where y_t is the process output at lag t; u_{t-1} is the process input at lag t-1 and a_t is white noise with mean zero and variance σ_a^2. The model parameters to be estimated are δ_1 and ω_0. The variance-covariance matrix of the estimates $[\delta_1\ \omega_0]^T$ is

$$\text{var}\begin{pmatrix}\delta_1\\ \omega_0\end{pmatrix} = \frac{\sigma_a^2}{n}\begin{bmatrix} E(y_t^2) & E(y_t u_t) \\ E(y_t u_t) & E(u_t^2)\end{bmatrix}^{-1} \tag{2. 31}$$

where

$$E(u_t^2) = \sigma_u^2 \tag{2. 32}$$

$$E(y_t^2) = \sigma_y^2 = \sigma_u^2\omega_0^2\frac{1+2q}{1-\delta_1^2} + \frac{\sigma_a^2}{1-\delta_1^2} \tag{2. 33}$$

$$E(y_t u_t) = \sigma_u^2\frac{\omega_0}{\delta_1}q \tag{2. 34}$$

$$q = \sum_{i=1}^{\infty}\delta_1^i\rho_i \qquad \rho_i = \frac{E(u_t u_{t-i})}{\sigma_u^2} \tag{2. 35}$$

Further details relating to variance-covariance matrix computation for the parametric model given in Equation (2. 30) can be found in Box and Jenkins (1976).

For a FIR model where p impulse weights are being estimated, the variance-covariance matrix is given below

$$\mathrm{var}(\mathbf{b}) = \sigma_a^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1} = \frac{\sigma_a^2}{n}\begin{bmatrix} E(u_t^2) & E(u_t u_{t-1}) & . & E(u_t u_{t-p}) \\ E(u_{t-1}u_t) & E(u_{t-1}^2) & . & E(u_{t-1}u_{t-p}) \\ . & . & . & . \\ E(u_{t-p}u_t) & E(u_{t-p}u_{t-1}) & . & E(u_{t-p}^2) \end{bmatrix}^{-1} \qquad (2.\ 36)$$

Suppose, a pseudo random binary sequence (PRBS) is used to excite the process input during the open loop experiments to generate process input-output data. For a PRBS signal with a base switching interval of T_{si} sampling periods,

$$E(u_t u_{t-i}) = \begin{cases} \dfrac{(T_{si}-i)}{T_{si}}\sigma_u^2 & i = 0 \text{ to } T_{si} \\ 0 & i \geq T_{si} \end{cases} \qquad (2.\ 37)$$

Therefore, the variance-covariance matrix for the impulse response model estimates **b** for a PRBS with a switching interval of 5 sampling periods and $\sigma_u^2 = 1$ is

$$\mathrm{var}(\mathbf{b}) = \frac{\sigma_a^2}{n}\begin{bmatrix} 1 & 0.8 & 0.6 & 0.4 & 0.2 & 0 & \cdots & 0 \\ 0.8 & 1 & 0.8 & 0.6 & 0.4 & 0.2 & \cdots & 0 \\ 0.6 & 0.8 & 1 & 0.8 & 0.6 & 0.4 & \cdots & 0 \\ 0.4 & 0.6 & 0.8 & 1 & 0.8 & 0.6 & \cdots & 0 \\ 0.2 & 0.4 & 0.6 & 0.8 & 1 & 0.8 & \cdots & 0 \\ 0 & 0.2 & 0.4 & 0.6 & 0.8 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}^{-1} \qquad (2.\ 38)$$

For estimation of p=30 coefficients in the FIR model, the condition number of $\frac{1}{n}\left(\mathbf{X}^T\mathbf{X}\right)$ is approximately 448. The largest and the smallest eigenvalues are 4.91 and 0.0109, respectively. The ten largest eigenvalues account for over 90% of the sum of squares of **X**. The magnitude of the remaining 20 eigenvalues is below 0.3. On the other hand, the

condition number of the variance-covariance matrix for the parameteric model in Equation (2. 30) with ω_0=0.2 and δ_1=0.8 is approximately 2.2. The confidence intervals (i.e., ± one standard deviation) for the impulse weights computed with the variance-covariance matrices for the parsimonious transfer function model and the directly identified FIR model are shown in Figure 2.1. The confidence intervals are computed with n=500 and $\sigma_a^2 = 0.4$. The solid line represents the true impulse weights. The dotted and dashed lines represent the confidence intervals for the impulse weights obtained from the parsimonious transfer function model and the directly fitted FIR model. As is evident, the confidence intervals for the impulse weights obtained from the parsimonious transfer function model are much smaller than those of the directly identified FIR model.

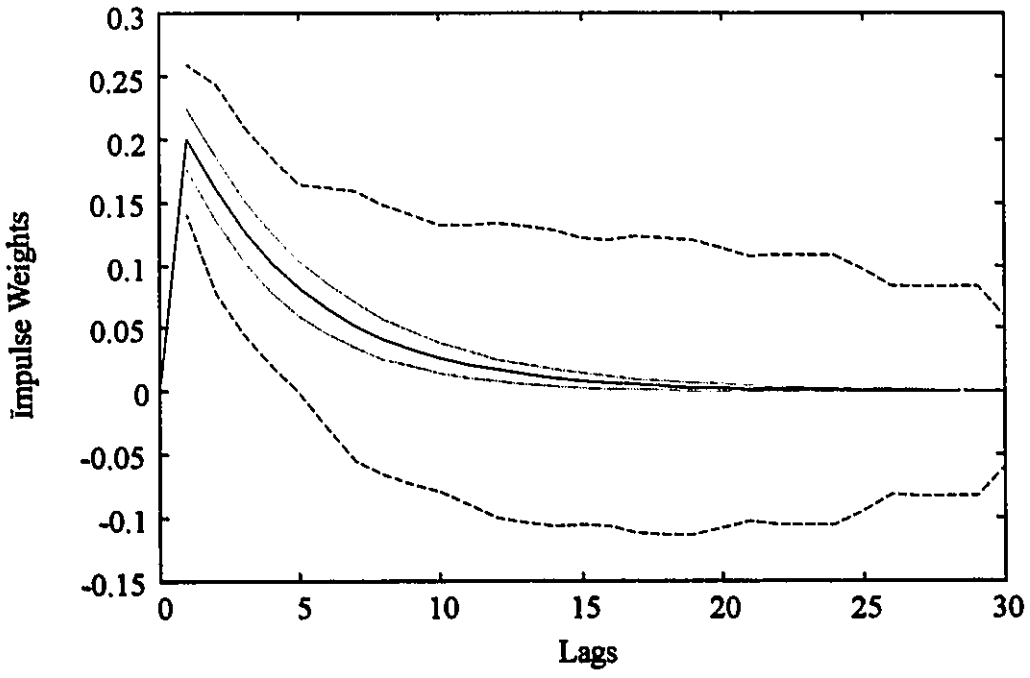

Figure 2.1: First order process impulse response weights and its confidence intervals computed using variance-covariance matrices for the parsimonious transfer function model (dotted line) and the directly fitted FIR model (dashed line).

2.5 Process Example

The example used to generate the process input-output data is a methanol-acetone-water (MAW) extractive distillation column simulation developed using a fundamental tray-by-tray model at McMaster University (Chin, 1989). The column is used to separate a mixture of acetone and methanol. Water is used as a solvent. A schematic of the extractive distillation column is given in Figure 2.2. The output variables are acetone composition in the top product and the methanol composition on a water free basis (MWF) in the bottoms product. The input variables are the steam temperature to the reboiler and the solvent flow rate. The disturbance is the feed flow rate. The process steady state condition number is 72. The process is sampled every 11 minutes. The impulse weights relating the output variables to input and disturbance variables were generated by introducing pulse changes to both inputs and disturbance. These impulse weights, shown as a solid smooth line in Figure 2.3, were treated as the true process and were used to generate the process data. Both inputs were excited with pseudo random binary sequence (PRBS) signals with switching intervals of 5 sampling periods. The PRBS magnitudes were 2.5 °C and 11.125 ml/min for the steam temperature and the solvent flow rate, respectively. The PRBS magnitudes for both inputs were such that they contributed equally to the variances of the output variables. The effects of the random variation in the feed flow rate disturbance and the measurement noise added to both outputs accounted for roughly 12% and 18% of the total variances of the top and bottoms product compositions, respectively.

2.5.1 Process Identification

The following parsimonious transfer function models in the Laplace domain for the extractive distillation column were identified using nonlinear optimisation:

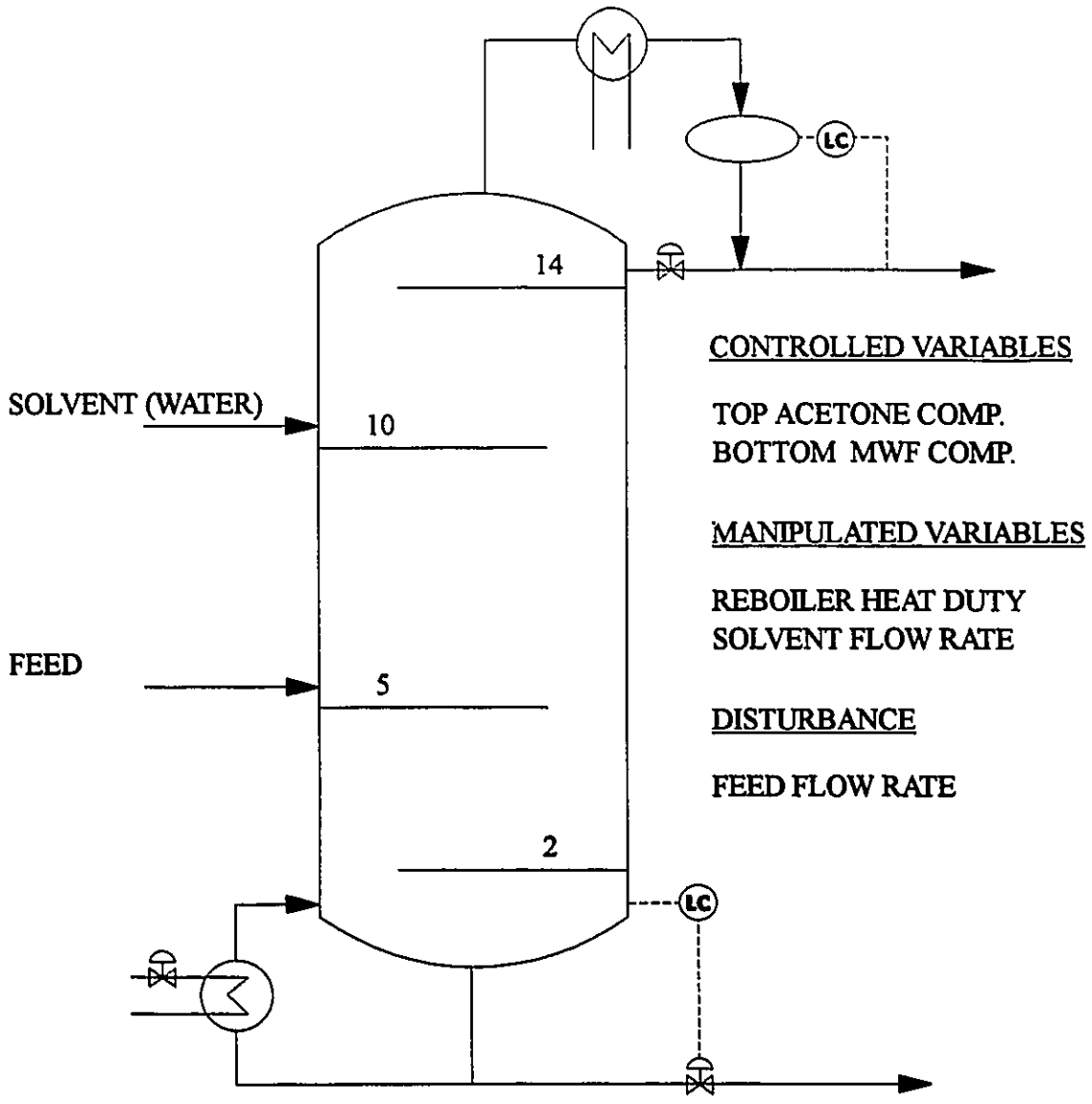

Figure 2.2: A schematic of methanol-acetone-water extractive distillation column.

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{K_1(\tau_1 s+1)}{(\tau_2 s^2+\tau_3 s+1)} & \dfrac{K_2(\tau_4 s+1)}{(\tau_5 s+1)(\tau_6 s^2+\tau_7 s+1)} \\ \dfrac{K_3}{(\tau_8 s+1)(\tau_9 s+1)} & \dfrac{K_4}{(\tau_{10} s+1)(\tau_{11} s+1)} + \dfrac{K_5(\tau_{12} s+1)e^{-6s}}{(\tau_{13} s+1)(\tau_{14} s+1)(\tau_{15} s+1)} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix} + D(s) \qquad (2.39)$$

All parameters in the above models are parameterised independently. The disturbance models were identified in discrete domain using AR(3) structure.

For the FIR and ARX models, the following number of coefficients were estimated:

FIR Models: y_1 model: u_1 (30) , u_2 (40) and AR(3) disturbance model.

y_2 model: u_1 (30), u_2 (50) and AR(3) disturbance model.

ARX Models: y_1 model: y_1 (10), u_1 (10) and u_2 (10).

y_2 model: y_1 (10), u_1 (10) and u_2 (20).

The FIR models were identified with generalised least squares using the following regression methods: OLS, PLS and RRDlin. The ARX models were estimated using OLS. PLS was also used to identify ARX models; however, the results obtained are similar to those of OLS and thus are not presented here. The models for the process example were estimated using a training data set with 500 data points. The number of optimal latent vectors in PLS model and the ridge parameter in the regularisation method were selected by minimising the prediction error of the models on a testing data set.

The FIR models were also estimated using ridge regression (RR) where the magnitudes of the impulse coefficients were penalised with a constant parameter. It gave poor results compared to RRDlin (penalising the linearly weighted changes in the impulse weights) because of the excessive bias in the steady state gains that resulted from penalising the magnitudes of the impulse coefficients.

2.6 Results

The following criteria were used to compare the non-parsimonious FIR and ARX modelling using various regression methods and the parsimonious parametric transfer function modelling:

(i) providing the impulse and step weights closest to the true process;

(ii) giving a model which leads to control systems with best robust stability;

(iii) providing the best control performance

All the results in the following sections are based on 500 Monte Carlo simulations. A different seed was used to generate process input-output data for each simulation.

2.6.1 Closeness of the Fit to the True Model

The first comparison criterion used is the closeness of the fit to the true model. This is measured by amount of departure from the true impulse and step weights of the process example. The sum of the squared deviations from the true impulse response weights (MSE_impulse) and the sum of squared deviations from the true step weights (MSE_step) are used to quantify the closeness of the fit to the true model. For a given process model, these quantities are computed as given below:

$$\text{MSE\_impulse} = \sum_{i=1}^{p_j} \left(\beta_{j,i} - b_{j,i}\right)^2 \tag{2.40}$$

$$\text{MSE\_step} = \sum_{i=1}^{p_j} \left(S_{j,i} - \hat{S}_{j,i}\right)^2 \tag{2.41}$$

Note that the step weights can be computed as the cumulative sum of the impulse weights.

$$S_k = \sum_{i=1}^{k} \beta_i \tag{2.42}$$

The MSE_impulse primarily measures the closeness of the estimated process dynamics to the true process dynamics whereas MSE_step primarily measures the accuracy of the steady state gain estimated by each method. The quantities MSE_impulse and MSE_step

for each regression method for the extractive distillation process example are given in Tables 2.1 and 2.2, respectively. Plots of identified impulse and step weight estimates for one Monte Carlo simulation are shown in Figures 2.3 and 2.4, respectively. In these figures, the solid line represents the true impulse and step weights. In Figures 2.3a-2.3d and 2.4a-2.4d, the dots represent the FIR and step weights generated from the estimated parsimonious transfer function models and the dashed line represents the FIR weights and step generated from the estimated ARX models. In Figures 2.3e-2.3h and 2.4e-2.4h, the dashed line, the dots and the dotted line represent the impulse and step weights estimated by OLS, RRDlin and PLS, respectively.

It is evident from Tables 2.1 and 2.2 that the parsimonious transfer function models provide the best results by these criteria. This is expected since selection of the structured transfer model forces smoothness and a structure on the impulse weights as seen in Figures 2.3a-2.3d. Among the non-parsimonious FIR methods, RRDlin provides the best results and it even provides lower MSE_impulse than the ARX model. PLS gives very poor estimates of the steady state gains thus leading to very large MSE_step. By selecting the number of latent vectors in PLS models so that it provided the minimum prediction error sum of squares (PRESS) on the testing data set, PLS stopped at 4 latent vectors for most of the simulations. This provided good predictions and reasonable estimates of the impulse weights but eliminated too much information on the steady state gains. However, the performance of PLS models with 10 latent vectors improved the results very considerably.

2.6.2 Effects of the Amount of Data

In this section, we investigate the precision of the estimates as a function of the amount of data. The models relating the top acetone composition (y_1) to both input variables (u_1 and u_2) were identified. The plots for MSE_impulse for both process models versus the number of data points are shown in Figure 2.5. The results shown are the averages of 5 different simulations.

		MSE_impulse (Deviations from the true impulse weights)			
Modelling Technique	Regression Method	y_1-u_1 (1e-8)	y_1-u_2 (1e-9)	y_2-u_1 (1e-6)	y_2-u_2 (1e-7)
Parsimonious TF	Nonlinear Opt.	0.46	0.11	0.30	0.14
ARX	OLS	1.20	0.83	1.32	2.04
FIR with Disturbance Model	OLS	7.06	6.00	7.76	9.89
	RRDlin	0.99	0.26	0.69	0.44
	PLS	2.04	1.30	1.53	1.13

Table 2.1: Average results for deviations from the true impulse models given by the estimated models using various regression methods.

		MSE_step (Deviations from the true step weights)			
Modelling Technique	Regression Method	y_1-u_1 (1e-7)	y_1-u_2 (1e-8)	y_2-u_1 (1e-5)	y_2-u_2 (1e-6)
Parsimonious TF	Nonlinear Opt.	4.53	3.57	2.97	2.04
ARX	OLS	6.24	5.23	3.18	4.90
FIR with Disturbance Model	OLS	6.87	4.98	3.78	2.40
	RRDlin	6.35	4.55	3.31	1.90
	PLS	11.17	28.58	9.24	4.60

Table 2.2: Average results for deviations from the true step response models given by the estimated models using various regression methods.

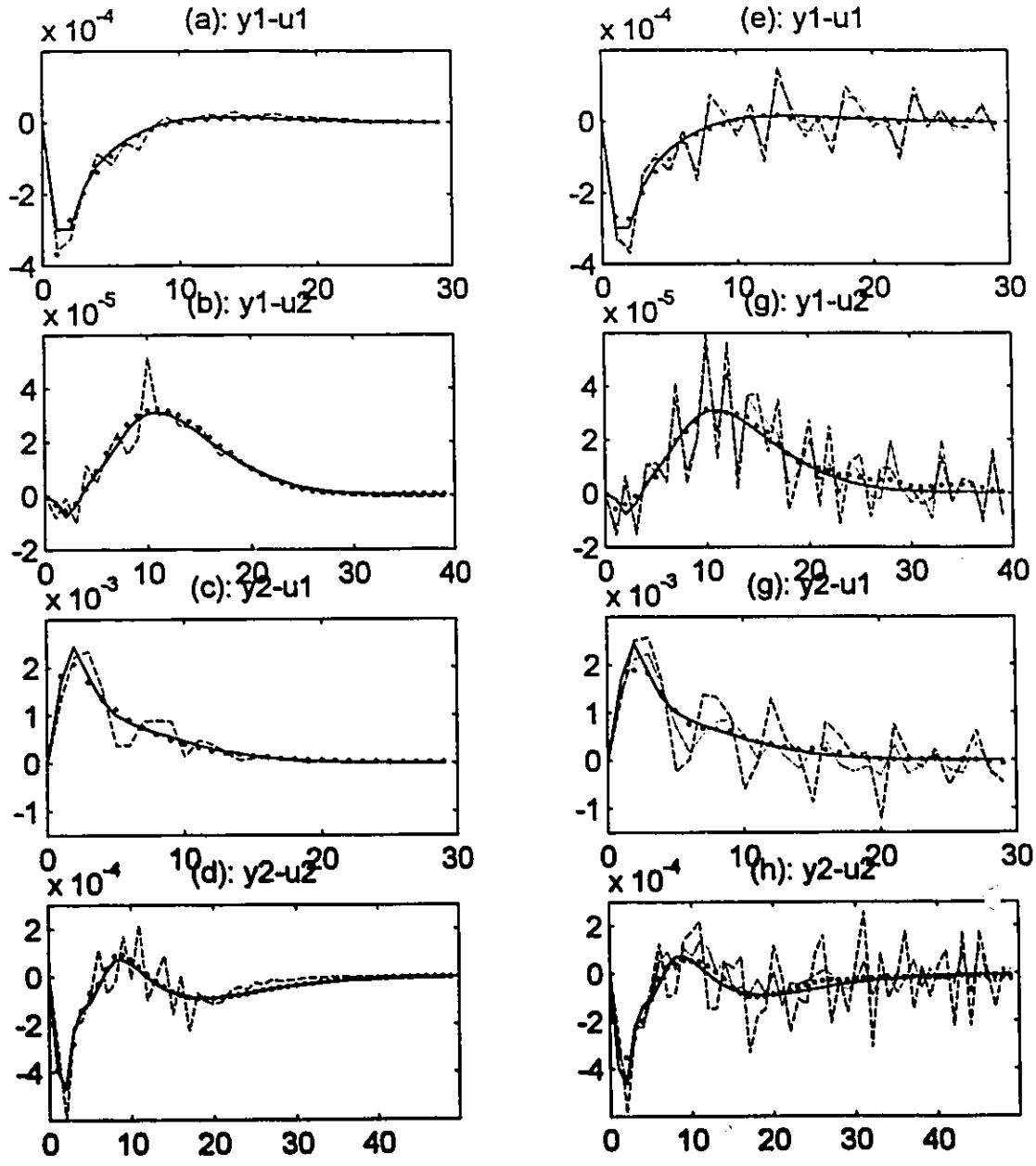

Figure 2.3: Typical impulse response coefficients estimated using various regression methods.
Legends: figures (a) - (d): solid line - true impulse response weights; dots - parsimonious transfer model; dashed line - ARX model.
Figures (e) - (h): solid line - true impulse response weights; dots - RRDlin; dashed line - OLS; dotted line: PLS.

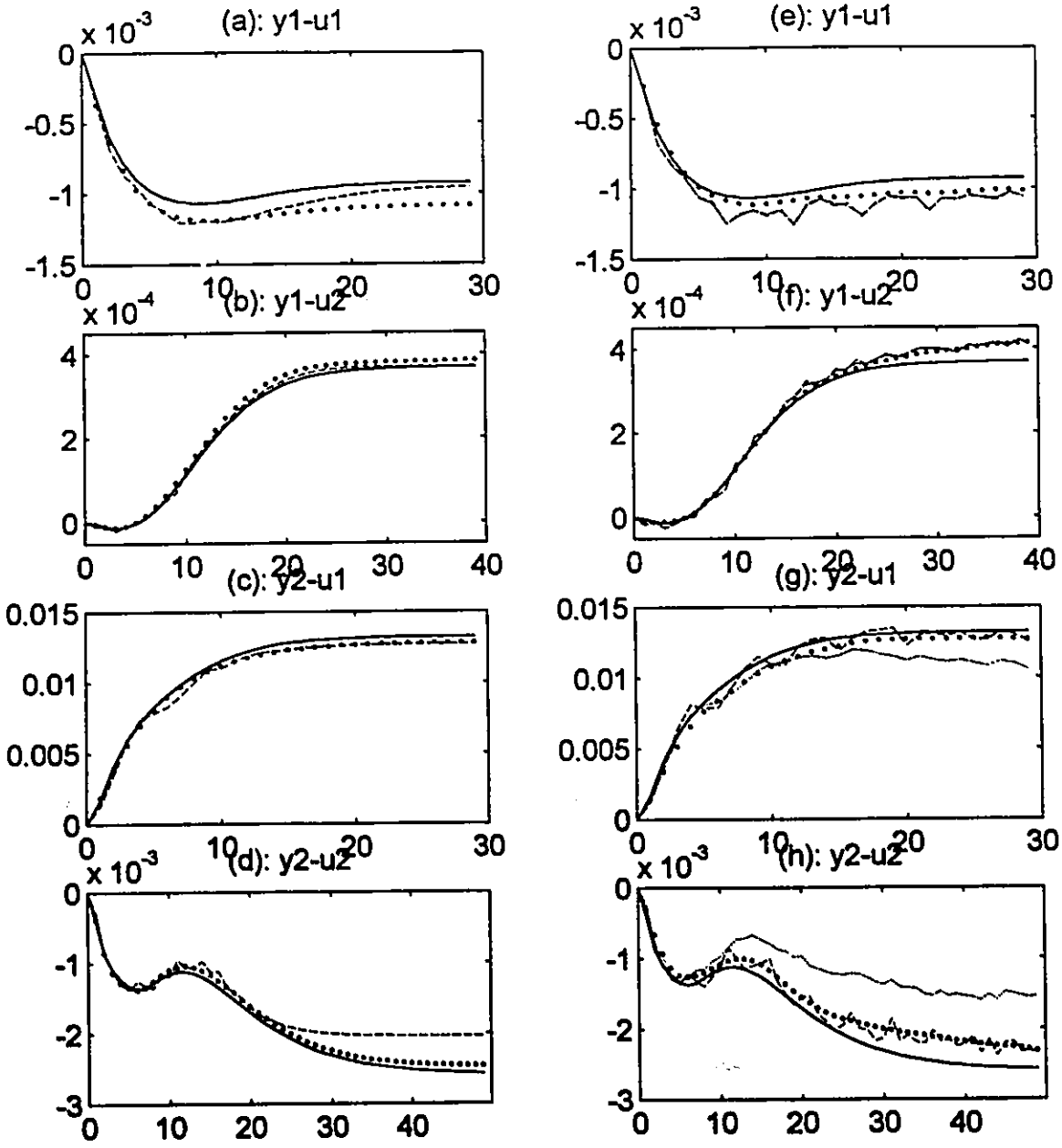

Figure 2.4: Typical step response coefficients estimated using various regression methods. Legends: figures (a) - (d): solid line - true step response weights; dots - parsimonious transfer model; dashed line - ARX model.
Figures (e) - (h): solid line - true step response weights; dots - RRDlin; dashed line - OLS; dotted line: PLS.

Two main points are evident from these plots. First, it is apparent that for small data sets the parsimonious transfer function model fits yield much better results than any of the methods for fitting the non-parsimonious models directly. Furthermore, the impulse weights obtained from the transfer function models continue to provide better results by the MSE_impulse criterion than the next best approach -RRDlin until quite large amounts of data (2000 sampling intervals for y_1-u_1 and 20000 sampling intervals for y_1-u_2). On the other hand, the MSE of the FIR estimates obtained from the transfer function models is eventually limited by any structural bias in the chosen model forms. The total MSE between the true process response and any fitted model can be decomposed into two parts (Goodwin et al., 1992) - a variance component which tends to zero as the amount of data increases, and a squared bias component which is constant and exists whenever the model structure is not structurally rich enough to capture all aspects of the true system response. Providing the number of terms used is beyond the settling time of the true process, the FIR models have no bias, but exhibit large variance components when fitted by various methods. On the other hand, low order transfer function models will inevitably exhibit some small amount of bias but have a much smaller variance component. From Figure 2.5 it can be seen that the MSE_impulse obtained by fitting the transfer function models in Equation (2. 39) reach a lower bound for both the y_1-u_1 and y_1-u_2 models while the FIR models exhibit no such lower bound.

In this section, it has been demonstrated that much larger data sets are required to obtain a given precision when fitting non-parsimonious FIR models than when fitting parsimonious transfer function models. However, the eventual accuracy of the transfer function models is limited by any bias resulting from using an inadequate model structure to represent the complex process dynamics.

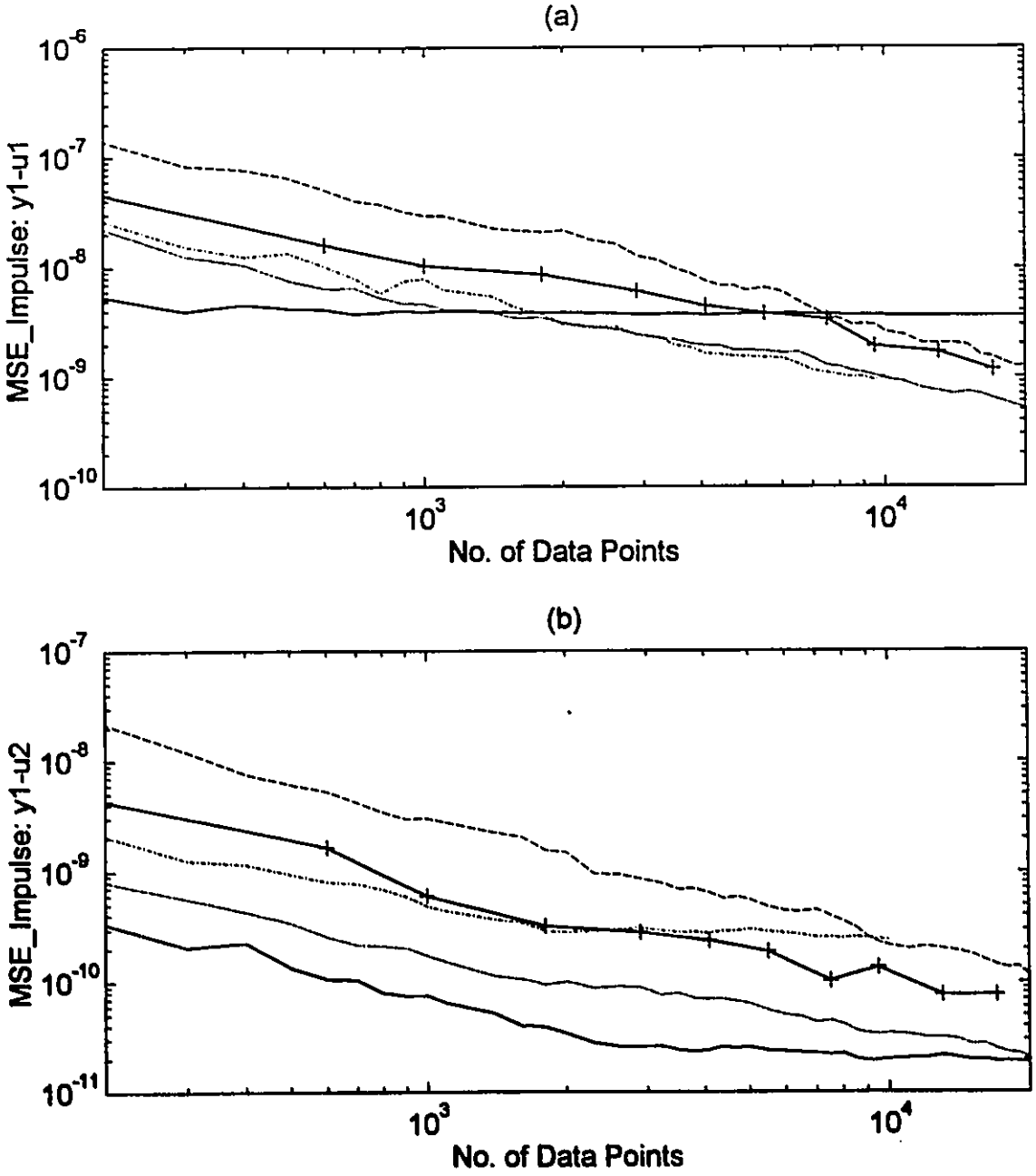

Figure 2.5: Mean squared error of deviations from true impulse weights versus the number of data points in the training data set.
Legends: solid line - parsimonious TF model; dashdot line - ARX model; dotted line: RRDlin; dashed line - OLS; and solid line with plus sign - PLS.

2.6.3 Steady State Robust Stability Analysis

A criterion that can be used to test the stability robustness of any control system using an identified model, provided that the process is open-loop stable, is the steady state robust stability criterion (Garcia and Morari, 1985):

$$\mathrm{Re}\left(\lambda_i\left(G_P(1)\hat{G}(1)^{-1}\right)\right) > 0, \ \forall i \tag{2.43}$$

where $G_P(1)$ and $\hat{G}(1)$ refer to the steady state gains of the true process and estimated process models, respectively, and the λ_i is the i^{th} eigenvalue of $G_P(1)\hat{G}(1)^{-1}$. This robust stability condition is based solely on estimating the process steady state gains correctly. If the real part of one or more of the eigenvalues is negative, then the closed loop system can not be stabilised using any controller designed with the estimated process model. This is equivalent to the well-known uncontrollable situation where the process and its estimated model have opposite signs in their steady state gains in a single-input single-output case and thus the resulting controller will become unstable due to positive feedback.

The estimated models from various identification techniques were analysed using the above robust stability criterion. The results are shown in Table 2.3. Both transfer function and ARX modelling techniques give stable steady state process gains for all 500 simulations. For FIR modelling using OLS and RRDlin, five estimated models for each regression method would yield unstable model predictive controllers. All five models yielding unstable controllers were estimated from data sets for the same simulations for both regression methods. Forty seven (47) models for validated PLS would yield unstable controllers. This is due to the fact that PLS eliminated too much information on the process steady state gains by stopping at fewer latent vectors. The number of latent vectors in PLS models were selected by validation on a testing data set. The results for PLS models with 10 latent vectors are similar to those of OLS and RRDlin.

Modelling Technique	Regression Method	No. of Simulations Failing Steady State Robust Stability Criterion $\exists i$: $\mathrm{Re}\left(\lambda_i\left(G(1)\hat{G}(1)^{-1}\right)\right) < 0$
Parsimonious TF	Nonlinear Opt.	-
ARX	OLS	-
FIR with Disturbance Model	OLS	5
	RRDlin	5
	PLS	47

Table 2.3: Steady state robust stability criterion results

2.6.4 Frequency Domain Analysis

In this section, we examine the precision and accuracy of the estimated dynamic responses in the frequency domain using all the identification methods; and we also examine a measure of the stability robustness of the models identified by the various methods when used in feedback controllers.

2.6.4.1 Joint Confidence Regions on the Nyquist Plot

In this section, the accuracy and precision of the various identification methods are compared by computing approximate joint confidence regions for the frequency response of the system at various frequencies. From the 500 simulations performed using each method, 500 joint estimates of the real and imaginary components of the process frequency response $\hat{G}(iw)$ are obtained. These joint estimates appear to be scattered randomly in approximately elliptical regions in the complex plane. Therefore, we approximate the sampling distribution of the frequency response estimates at each frequency by a bivariate normal distribution. For a bivariate normal variable, $\mathbf{Z}$=[Real($\hat{G}(iw)$) Imag($\hat{G}(iw)$)], the statistic $(\mathbf{Z}_i - \overline{\mathbf{Z}})^T \mathbf{S}^{-1} (\mathbf{Z}_i - \overline{\mathbf{Z}})$ will follow a central beta distribution with p/2 and (k-p-1)/2 degrees of freedom (Wierda, 1994):

$$(\mathbf{Z}_i - \overline{\mathbf{Z}})^T \mathbf{S}^{-1} (\mathbf{Z}_i - \overline{\mathbf{Z}}) \sim \frac{(k-1)^2}{k} B\left(\frac{p}{2}, \frac{k-p-1}{2}\right) \quad (2.44)$$

where $\overline{\mathbf{Z}}$ is the sample mean of k observations $\mathbf{Z}_i$ and S is their sample covariance matrix. Using this distribution the accuracy of each method in estimating the frequency response at any given frequency can be summarised in terms of a 100(1-α)% confidence region computed using the 500 replicated results at each frequency.

The ellipses for 98% approximate joint confidence regions on the Nyquist plots for the extractive distillation column example were computed at the following four normalised frequencies: w=[0.006 0.06 0.18 0.6]/Ts rad/time unit (here, Ts is the sampling time). The first three frequencies correspond to the process at steady state, the open loop process time constant (based on the bode plot of the larger singular value) and a selected closed loop process time constant, respectively. The fourth frequency is selected to demonstrate the effects of regression methods on the estimated process model at a very high frequency. The ellipses for 98% approximate joint confidence regions on the Nyquist plots for the estimated process models for y_1 and y_2 are shown in Figures 2.6 and 2.7, respectively. The parsimonious transfer function models provide the smallest confidence regions compared to FIR and ARX models at all frequencies. The ARX model provides the second smallest confidence regions. For the y_1-u_1 and y_2-u_2 frequency responses, the ellipses for the ARX models are not centred about the true Nyquist points implying some bias in the structures used. Among the FIR models, the models estimated using PLS with a small number of latent vectors (obtained from prediction validation) have the largest confidence regions as shown in Figure 2.7. For the first output (y_1), the confidence regions for the FIR models estimated using PLS are very large compared to the confidence regions for other modelling techniques and thus are not shown in Figure 2.6. The confidence regions for RRDlin are slightly better than those of OLS at lower frequencies but much better at higher frequency.

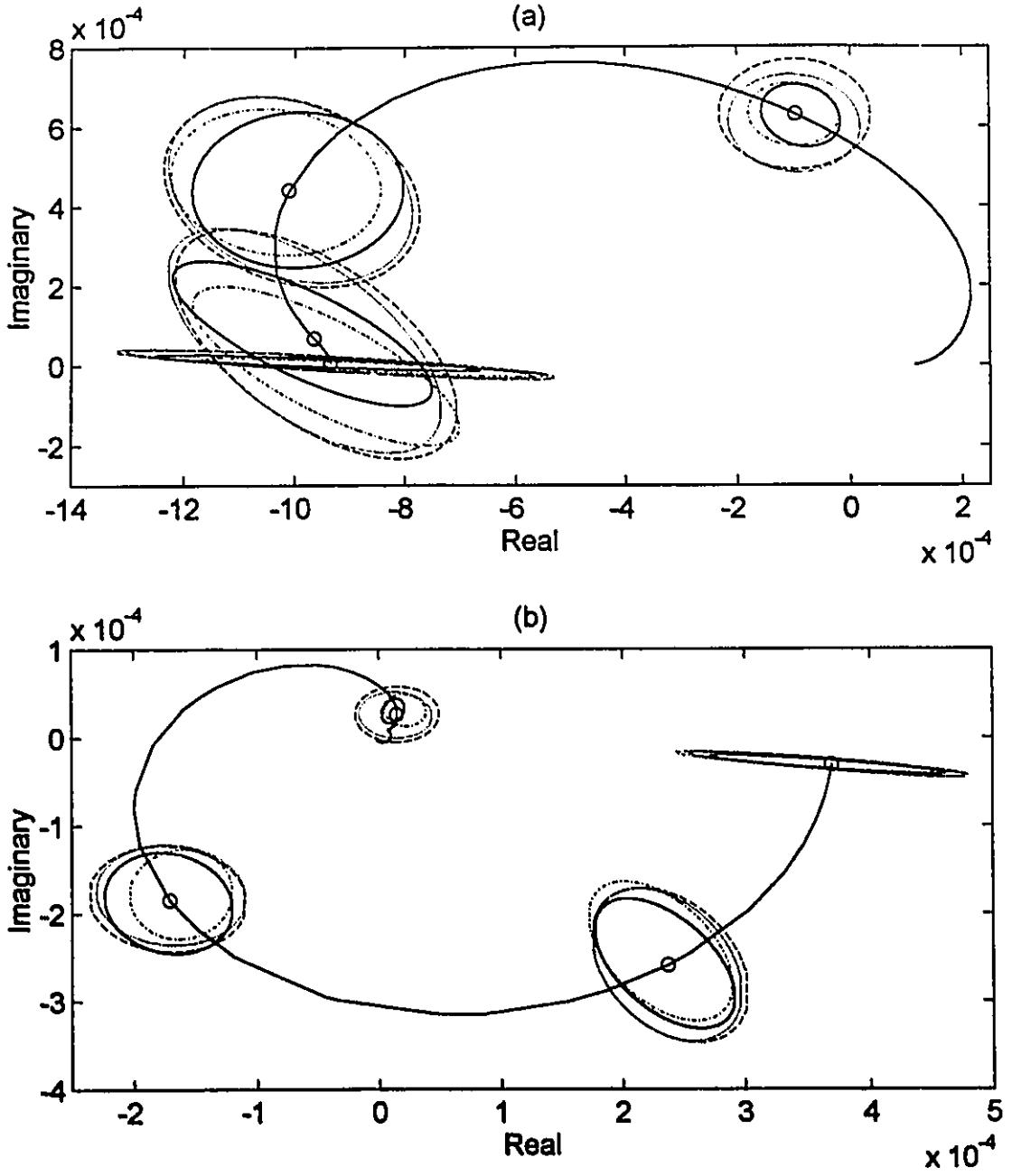

Figure 2.6: Approximate ellipses for 98% joint confidence regions on the Nyquist plot for (a) y_1-u_1 and (b) y_1-u_2.
Legends: solid line - parsimonious TF model; dashdot line - ARX model; dotted line: RRDlin; dashed line - OLS.

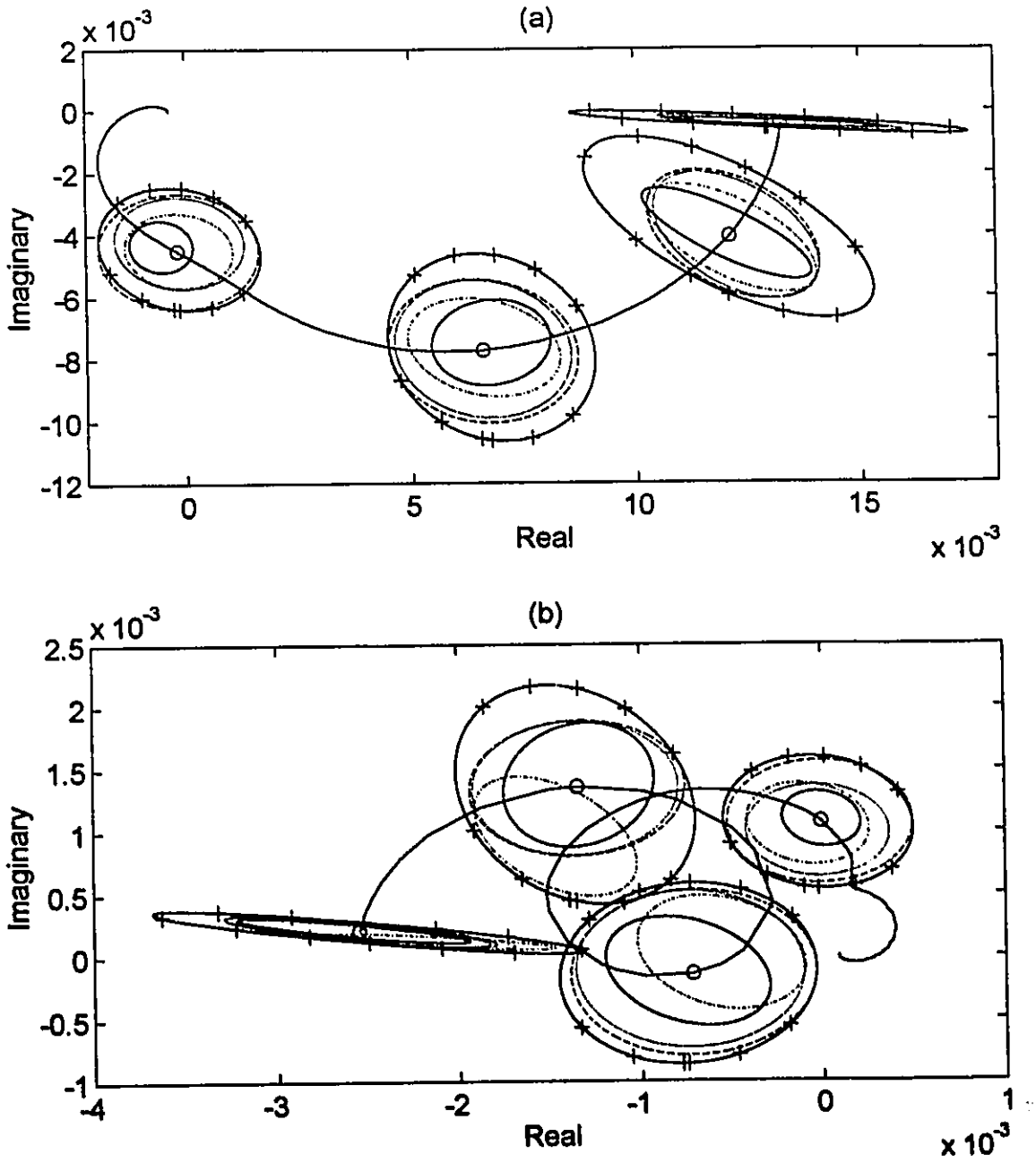

Figure 2.7: Approximate ellipses for 98% joint confidence regions on the Nyquist plot for (a) y_2-u_1 and (b) y_2-u_2.
Legends: solid line - parsimonious TF model; dashdot line - ARX model; dotted line: RRDlin; dashed line - OLS; and solid line with plus sign - PLS.

2.6.4.2 Stability Robustness Analysis

The objective of this section is to compare a measure of robustness of a model-based control system that would result from using the models identified by the various approaches. The measure is very general and independent of the specific controller design.

For the internal model control (IMC) structure, a sufficient condition for robust stability given by the small gain theorem (Morari and Zafiriou, 1989) is

$$\left\|(G_P(iw)-\hat{G}(iw))G_C(iw)\right\|_2 < 1 \quad 0<w<\frac{\pi}{T_s} \tag{2. 45}$$

where G_P and $\hat{G}$ are the true and estimated process models, respectively, and G_C is the internal model controller (i.e., $G_C = F\hat{G}_-^{-1}$). $\|\cdot\|_2$ is the 2-norm of a matrix and is the largest singular value. If the process/model uncertainty is defined by the output multiplicative uncertainty as

$$e_M(iw) = \left(G_P(iw)-\hat{G}(iw)\right)\left(\hat{G}(iw)\right)^{-1} \tag{2. 46}$$

then Equation (2. 45) can be rewritten as

$$\left\|G_C(iw)\hat{G}(iw)\right\|_2 < \frac{1}{\left\|e_M(iw)\right\|_2} \quad 0<w<\frac{\pi}{T_s} \tag{2. 47}$$

Equation (2. 47) implies that the gain of the nominal closed loop transfer function ($G_c(iw)\hat{G}(iw)$) is restricted by the possible process/model uncertainty at any given frequency w. The small gain theorem is very conservative and it is only a sufficient condition for robust stability. However, it provides a reasonable basis for comparing the expected robustness of controllers that will result from using various identified models. A structured singular value (μ) analysis gave almost identical results.

For all the regression methods investigated in this study, the 90% bounds on the inverse of multiplicative uncertainty (i.e., $1/\left\|e_M(iw)\right\|_2$) for the normalised frequency range between [1e-3 1] are shown in Figure 2.8. The 90% bound on the gain of $1/\left\|e_M(iw)\right\|_2$ at

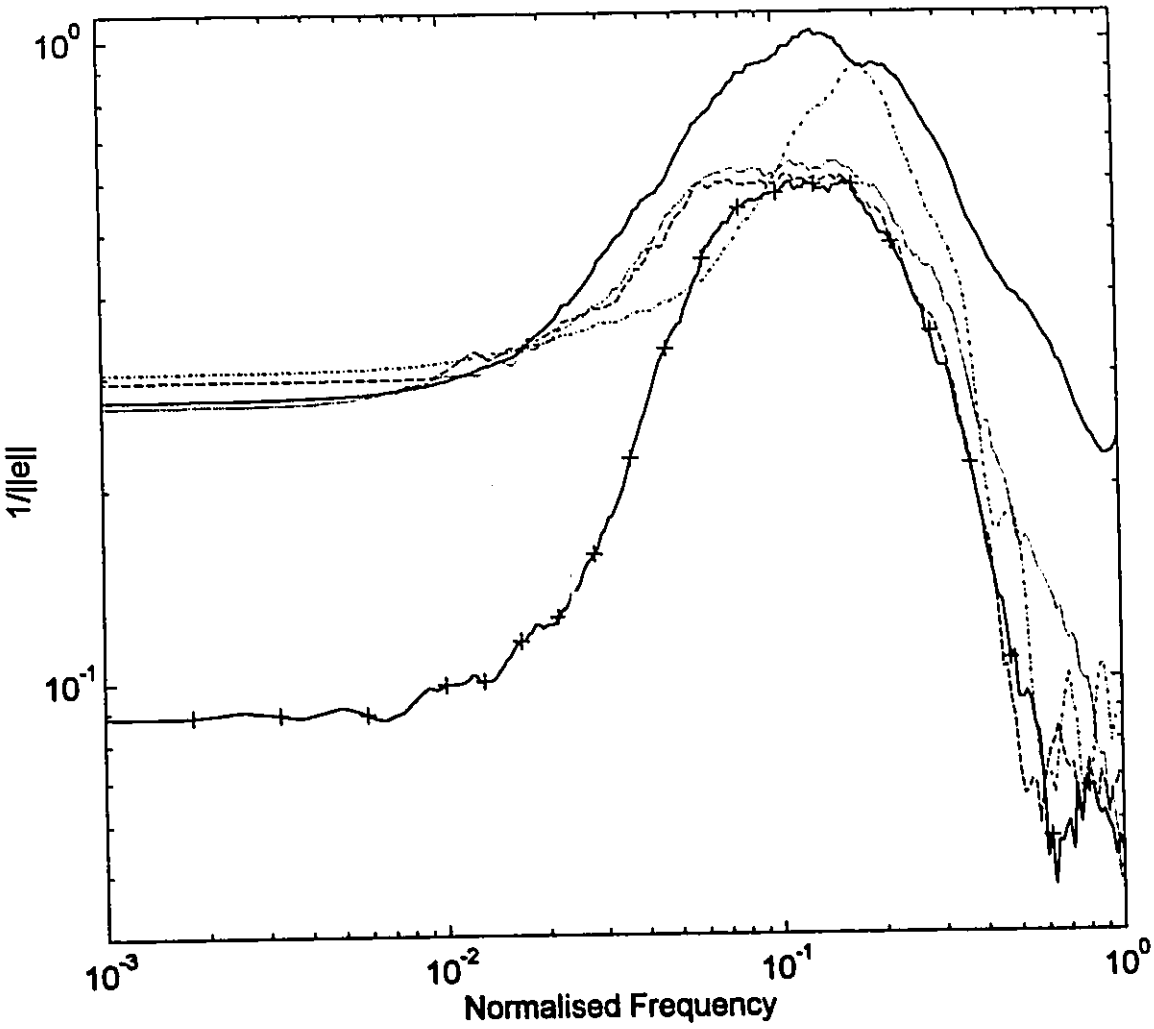

Figure 2.8: Approximate 90% bound on maximum allowable controller gain (Small gain theorem).
Legends: solid line - parsimonious TF model; dashdot line - ARX model; dotted line: RRDlin; dashed line - OLS. and solid line with plus sign - PLS.

each frequency w is computed as follows. The gains of $1/\|e_M(iw)\|_2$ at each frequency for all 500 simulations are computed and sorted in the order of smallest to the largest. The first fifty smallest gains at each frequency are discarded and the fifty first smallest gain at each frequency is taken as the 90% lower bound on the multiplicative uncertainty.

With the exception of FIR models estimated with PLS, all other models provide a similar bound on stability at lower frequencies (i.e. steady state). However, the parsimonious transfer function model clearly has the largest gain margin beyond frequency of 0.02. The FIR models provide better robust stability than the ARX model in the frequency range between 0.02 and 0.1 but give poorer robust stability beyond the frequency of 0.1. RRDlin provides similar robust stability as OLS at a lower frequency range between 10^{-3} to 0.1 but much better robust stability at frequencies greater than 0.1.

2.6.5 Performance of DMC Controllers using the Identified Models

In the last section, we looked at a very general but conservative overall measure of robustness to compare the quality of the identified models. In this section, we examine the actual performance of the models in a specific controller design - dynamic matrix control (DMC) (Cutler and Ramaker, 1979).

In DMC, for a 2-by-2 process, the manipulated variable moves $(\nabla \mathbf{u}_k)$ are computed by solving the following optimisation problem:

$$\min_{\nabla \mathbf{u}_k} \left\{ \sum_{i=1}^{2} \sum_{k=1}^{P} \left(\hat{y}_{i,t+k} - y^{sp}_{i,t+k} \right)^2 Q_1(i,k) + \sum_{j=1}^{2} \sum_{k=1}^{M} \nabla u^2_{j,t+k-1} Q_2(j,k) \right\} \tag{2.48}$$

where

$$\begin{bmatrix} \hat{y}_{1,t+1} \\ \vdots \\ \hat{y}_{1,t+P} \\ \hat{y}_{2,t+1} \\ \vdots \\ \hat{y}_{2,t+P} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \nabla u_{1,t} \\ \vdots \\ \nabla u_{1,t+M-1} \\ \nabla u_{2,t} \\ \vdots \\ \nabla u_{2,t+M-1} \end{bmatrix} + \begin{bmatrix} d_{1,t+1} \\ \vdots \\ d_{1,t+P} \\ d_{1,t+1} \\ \vdots \\ d_{2,t+P} \end{bmatrix} \tag{2.49}$$

The variables $\hat{y}_{i,t+k}$ and $y^{sp}_{i,t+k}$ are the future prediction and set-point of i^{th} process output variable at lag t+k, respectively, and $d_{i,t+k}$ is the future disturbance to the i^{th} output variable at lag t+k. The variable $\nabla u_{j,t}$ is the control move for j^{th} process input variable at lag t. The variables P and M are the output prediction and control horizons, respectively. The $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are the weighting matrices for the process output and input variables, respectively and **A** is the dynamic matrix of the system containing the estimated process step weights.

$$\mathbf{A}_{ij} = \begin{bmatrix} \hat{S}_{ij,1} & 0 & \cdots & 0 \\ \hat{S}_{ij,2} & \hat{S}_{ij,1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \hat{S}_{ij,P} & \hat{S}_{ij,P-1} & \cdots & \hat{S}_{P-M+1} \end{bmatrix} \tag{2. 50}$$

The $\hat{S}_{ij,k}$ is the k^{th} coefficient of the step response model relating the j^{th} input variable to i^{th} output variable. Note that even though M future control moves are computed in the above optimisation problem, only the current control move is implemented and the optimisation problem is solved again with the new disturbance measurement.

A dynamic matrix controller (DMC) is designed using the step response models obtained from various identification techniques. The step response weights can be computed from the impulse response weights using Equation (2. 42). The same DMC tuning parameters were used for the controller design for each method. The output prediction and control horizons are selected to be ∞ and 10, respectively. The weighting matrices for the output and input variables are

$$\begin{matrix} & y_1 & y_2 \end{matrix} \qquad \begin{matrix} & u_1 & u_2 \end{matrix} \tag{2. 51}$$

$$\mathbf{Q}_1 = \begin{bmatrix} 1 & \\ & 10 \end{bmatrix} \qquad \mathbf{Q}_2 = \begin{bmatrix} 0.1 & \\ & 0.003 \end{bmatrix}$$

Control simulations are carried out for a step change of 5 units in the feed flow rate. The performance of the controllers is compared in terms of integral of squared error (ISE) for both output variables.

The distributions of ISE for both output variables for all 500 simulations and for each regression method are shown on the histogram plots in Figure 2.9. Furthermore, the number of control simulations which provided unstable controllers for the weighting matrices given in Equation (2. 51) are also shown as a bar at ∞ on the plots. As is evident from these plots, the parsimonious transfer function modelling provides the best control performance results and yields no unstable controller simulation. The FIR modelling with RRDlin gives the second best results with only 9 unstable simulations. One hundred thirty seven (137) simulations were found to be unstable for ARX models however the stable simulations provided reasonable control performance. The FIR modelling using OLS and PLS provided the worst control performance results. Larger number of unstable simulations for ARX and FIR models estimated using OLS are the direct result of high frequency changes in the estimated step weights due to ill-conditioning of $\mathbf{X}^T\mathbf{X}$. For FIR modelling using RRDlin, we have used prior process knowledge during the model estimation to smooth the impulse weights and this smoothing of the impulse/step weights leads to improved control performance.

Most of the unstable control simulations (which meet the steady state robust stability criterion in Equation (2. 43)) can be stabilised by detuning the controller by selecting a weighting matrix, $\mathbf{Q}_2$, with larger magnitudes. However, this will require longer times for the process to return to steady state.

2.7 Conclusions

In this chapter we have compared various approaches to identifying non-parsimonious FIR models. Comparisons have been made on the basis of closeness of fit to the true process, robust stability provided by the resulting model, and the control performance obtained.

The major conclusion by all assessments is that obtaining FIR models by first identifying low order parameter transfer function models was much superior to any of the

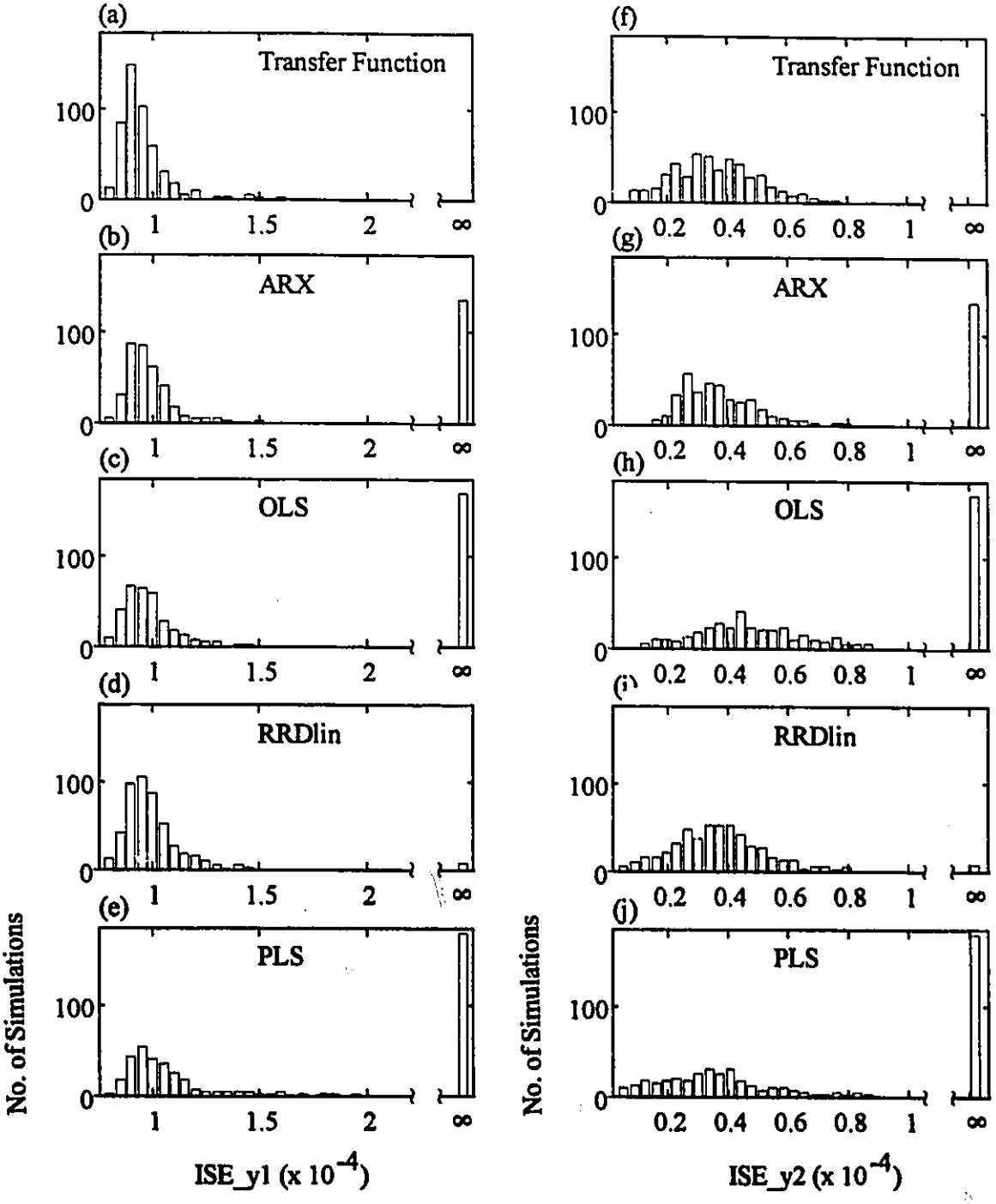

Figure 2.9: Performance of DMC controllers designed using the identified models
Histograms: (a) - (e): distributions of integral of squared error (ISE) for the first output variable; (f) - (j): distributions of integral of squared error (ISE) for the second output variable.

methods which directly identified the FIR coefficients. The main reason for this is that the parsimonious nature of the model leads to less overfitting of the data and induces, through its structure, a smooth behaviour of the estimated FIR coefficients. In particular, the latter feature leads to controllers which are more robust in terms of performance and stability. Therefore, although it is recognised that more effort is generally involved in identifying the structure of parsimonious transfer function models, it is recommended that this be done whenever the expected structure of the process is expected to be reasonably simple.

If it is desired to fit the non-parsimonious models directly, then several estimation methods were investigated: OLS, regularised least squares method (RRDlin) and PLS. Among these the performance of RRDlin was uniformly superior to all other regression methods by all assessment methods. RRDlin offered improvements over other regression methods mainly through introducing into its objective function the prior knowledge by penalising changes in the impulse weights (i.e., first derivatives) with an increased weighting over lags. This proved to be a meaningful way of regularising the LS solution.

PLS methods can provide good identification results, but the number of latent vectors used must in general be substantially larger than that suggested by cross-validation or validation against a test set. The latter criteria are only based on obtaining good predictions of Y which depend largely on identifying the dominant singular values / directions of the process. The small singular values become very important in the controller where the model is inverted.

3. Multi-Output Process Identification

3.1 Introduction

Multi-Input Multi-Output (MIMO) models are needed for the model based control of multivariate processes. To obtain such models, it has been common practice to identify a Multi-Input Single-Output (MISO) model for each output separately and then combine the individual models into a final MIMO model. If models for all outputs are independently parameterised then this approach is optimal. However, if there are common or correlated parameters among models for different output variables and/or correlated noise, then one should perform identification on all the outputs simultaneously. By identifying models for all outputs simultaneously, one should obtain better MIMO models in the sense that the model parameter estimates will be closer to their true values. One might also anticipate that by fitting all models simultaneously, the estimated models for all outputs will be more consistent with each other and hence lead to more robust MIMO control.

Furthermore, if one has additional measured variables that are not of direct interest but which are highly correlated with the primary measured output variables of interest, then by including them in the multivariate output vector and performing identification on all outputs simultaneously, one may obtain better and more robust MIMO models for primary outputs. Modelling all outputs simultaneously will utilise the correlation structure among the variables to estimate the process model parameters more accurately. Furthermore, the measurements for the primary variables may have low signal to noise ratios and modelling variables with low signal to noise content separately against all inputs may lead to poor process steady state gains and dynamics. However, the redundant or secondary variables may have higher signal to noise ratios and modelling these output

variables together may help to identify high noise content variables more accurately due to their correlation with secondary process output variables with lower noise content. This may be more helpful in the case of very ill-conditioned systems.

To illustrate the potential advantage of multi-output identification, consider the simple example of a process where composition is being measured with two different sensors, namely a chromatograph and an infrared sensor. By taking the weighted average of these two measurements the signal to noise ratio will be improved and then identifying the models between the weighted average composition and the process inputs should provide a better model. This would be equivalent to identifying the models for these two composition measurements simultaneously. In general duplicate measurements of the same property are not made. However, measurements are usually available on several different responses, many of them highly correlated with one another. Highly correlated output measurement would have a similar synergistic effect to multiple measurements of the same response. Models for all the responses could be improved by multi-output identification. Several example where the multi-output identification may be beneficial are:

(i) Quality control where many highly correlated quality variables are being measured. For example, for a tire yarn process, some of the typical yarn properties measured are the load at 3% extension, load at 9% extension, the percent extension for 10 pounds load and the percent extension at which the maximum load occurs, etc. As is evident these properties are very highly correlated and each contains some of the same information. This correlation structure among these properties would be expected to hold throughout changes in the manipulated variables. Furthermore, since all these properties are being measured on the same sample, the measurement noises and the disturbances in the various properties will be cross-correlated.

(ii) Spatial control in a paper machine. The moisture content along the cross direction (CD) of the paper has a certain profile and the profile dynamics are similar in the machine direction (MD) with respect to the process input variable, i.e., steam drier

pressure. Therefore, changes or upsets in the process will not affect each moisture content measurement in an independent manner and it is reasonable to expect that these changes or upsets may cause the entire moisture content profile to move up or down in a certain way.

(iii) Multicomponent distillation column → A chromatograph provides measurements on component compositions. However, there are some tray temperatures which are very highly correlated to these compositions and can be used during the multivariate identification to obtain better process models for the compositions. Some of the time constants for both temperature and composition variables may be similar (flow stream passing through same part of the process) thereby allowing a common parameterisation for parts of the models.

3.2 Theory For Multi-Output Identification

The theory for multi-response estimation is briefly discussed here. Further details are contained in Box and Draper (1965) and Box et al. (1973).

Suppose, the mathematical model for i^{th} ($i=1,\ldots,n_y$) output variable can be written as

$$y_{i,t} = \eta(\mathbf{u}_t, \beta) + \varepsilon_{i,t} \qquad (3.1)$$

$$E(\varepsilon_{i,t_1}, \varepsilon_{j,t_2}) = \sigma_{ij} \quad t_1 = t_2$$

$$= 0 \quad t_1 \neq t_2$$

where there are $t=1,\ldots,n$ observations on each process variable. The variables $\mathbf{u}_t$ and β refer to vectors of manipulated variables $(u_1, \cdots, u_{n_u})$ and unknown model parameters. ε_i are the normally distributed white noise sequences over time associated with i^{th} output variable. The variance-covariance matrix of $\mathbf{Y}$ is given by

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n_y} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n_y} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n_y 1} & \sigma_{n_y 2} & \cdots & \sigma_{n_y n_y} \end{bmatrix} \tag{3. 2}$$

where $\sigma_{ij} = \sigma_{ji}$. Let

$$\Lambda = \Sigma^{-1} = \begin{bmatrix} \sigma^{11} & \sigma^{12} & \cdots & \sigma^{1n_y} \\ \sigma^{21} & \sigma^{22} & \cdots & \sigma^{2n_y} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{n_y 1} & \sigma^{n_y 2} & \cdots & \sigma^{n_y n_y} \end{bmatrix} \tag{3. 3}$$

Now, if the variance-covariance matrix, Σ, is known, then the maximum likelihood estimation of Equation (3. 1) yields,

$$\min_{\beta} \quad \sum_{i=1}^{n_y} \sum_{j=1}^{n_y} \sigma^{ij} \sum_{t=1}^{n} (y_{i,t} - \eta_{i,t})(y_{j,t} - \eta_{j,t}) \tag{3. 4}$$

3.2.1 Special Cases:

(i) If there are no cross-correlation among the errors for the output variables, $\sigma_{ij}=0$, $\forall i,j$, then the maximum likelihood estimation reduces to

$$\min_{\beta} \quad \sum_{i=1}^{n_y} \frac{1}{\sigma_{ii}} \sum_{t=1}^{n} (y_{i,t} - \eta_{i,t})^2 \tag{3. 5}$$

This is same as weighted least squares.

(ii) If β further contains no common parameters, then one can show that the above is equivalent to minimising the sum of squares for each response separately,

$$\min_{\beta_i} \quad \sum_{t=1}^{n} (y_{i,t} - \eta_{i,t})^2 \tag{3. 6}$$

where $i=1,\ldots,n_y$. This is single output estimation at a time. Therefore, a separate analysis of each response is justified as long as the errors $\varepsilon_{i,t}$ and $\varepsilon_{j,t}$ are uncorrelated for $i \neq j$ and there are no common parameters among the output variables.

Therefore, to summarise, multi-output identification is called for cases where

(i) there are common parameters among the models for different output variables; and

(ii) there are cross-correlated noise models ($D_{i,t}$ and $D_{j,t}$ are cross correlated).

In a case where all models are independently parameterised, there are obviously no common parameters, and the only incentive for multi-output identification is if D_t's are cross-correlated. However, this advantage may be minor unless the number of observations is small.

3.2.2 Multi-Output Identification in Case of Unknown Variance-Covariance Matrix

In most cases, the variance-covariance matrix, Σ, of $\mathbf{Y}$ is seldom known. Therefore, the Bayesian a posteriori estimates, β, are obtained by minimising the determinant of $S(\beta)$ (Box and Draper, 1965),

$$\min_{\beta} |\mathbf{S}(\beta)| \tag{3.7}$$

where $\mathbf{S}(\beta)$ is given by

$$\mathbf{S}(\beta) = \tag{3.8}$$

$$\begin{bmatrix} \sum(y_{1,t}-\eta_{1,t})^2 & \sum(y_{1,t}-\eta_{1,t})(y_{2,t}-\eta_{2,t}) & \cdots & \sum(y_{1,t}-\eta_{1,t})(y_{n_y,t}-\eta_{n_y,t}) \\ \sum(y_{2,t}-\eta_{2,t})(y_{1,t}-\eta_{1,t}) & \sum(y_{2,t}-\eta_{2,t})^2 & \cdots & \sum(y_{2,t}-\eta_{2,t})(y_{n_y,t}-\eta_{n_y,t}) \\ \vdots & \vdots & \ddots & \vdots \\ \sum(y_{n_y,t}-\eta_{n_y,t})(y_{1,t}-\eta_{1,t}) & \sum(y_{n_y,t}-\eta_{n_y,t})(y_{2,t}-\eta_{2,t}) & \cdots & \sum(y_{n_y,t}-\eta_{n_y,t})^2 \end{bmatrix}$$

3.3 Multivariate Methods for Multi-Output Identification

As discussed in the previous chapter, there are many choices for the model structures for process identification from the input-output data. These range from a parsimonious rational transfer function model to a semi-parsimonious ARX model to a

non-parsimonious FIR model. In this chapter, the FIR and the parsimonious transfer function models are used for multi-output identification.

3.3.1 Parsimonious Transfer Function Models

In this case, one needs some knowledge as to which process parameters (i.e., time constants) are common to transfer function models for all process output variables. If one has this knowledge a *priori*, then these process parameters can be constrained to be the same in these transfer functions and one can use nonlinear optimisation to estimate the parameters by minimising the determinant of the covariance matrix as shown in Equation (3. 8).

3.3.2 Non-Parsimonious Model Structures: ARX and FIR models

A multivariate multiple linear regression model can be written as follows:

$$\mathbf{Y} = \mathbf{XB} + \mathbf{E} \tag{3. 9}$$

For process identification, the **Y** matrix consists of process output variables at time t. **X** contains lagged process inputs in case of FIR models and lagged inputs and lagged outputs in the case of ARX models. **B** contains the estimated FIR or ARX model coefficients. **E** is a matrix of disturbances or errors. For estimation of FIR models in presence of autocorrelated disturbance, one may need to use generalised least squares (Clarke, 1967) to identify the disturbance models as well.

The following multi-output regression methods can be used for identification when employing ARX and FIR model structures:

i. Two Block Partial Least Squares (PLS2).

ii. Canonical Correlation Regression (CCR)

iii. Reduced Rank Regression (RRR)

iv. Principal Component Analysis on the output variables and then regressing the few dominant principal components on the lagged process input variables (FIR models only).

All of these multi-output regression methods fall into the class of statistical projection methods in which the **X** and **Y** matrices are projected onto lower dimensional spaces defined by latent variables. This dimension reduction is possible because of the high correlation among variables in **X** and **Y**. Models having many fewer parameters than are present in Equation (3. 9) are built between the latent variables in the Y-space and those in the X-space. These lower dimension models can then be re-expressed in terms of the much less parsimonious original model Equation (3. 9). However, since all the parameters in β are related to the much smaller number of parameters relating the latent variables spaces, it is obvious that there is effectively a large number of common parameters when estimating β by these projection methods. As a result one should benefit by performing multi-output identification whenever the projection space has much smaller dimension than that of Y.

3.3.3 Multivariate Regression Methods

A brief introduction to the multivariate regression methods used in this research is given here. Further details can be found in Burnham et al. (1996).

3.3.3.1 Principal Component Analysis on Output Space (Y)

If the underlying dimension of the output space is less than the number of output variables being measured, then one can perform principal component analysis (PCA) on the output data to determine the reduced output space. PCA, first described by Pearson (1901), is a method of explaining the variance of a data matrix, **Y**, in terms of few latent vectors or principal components. The latent vectors are linear combinations of the original variables, $\mathbf{t}=\mathbf{Yp}$, and are uncorrelated (or orthogonal to each other). The first latent vector, $\mathbf{t}_1$, explains the greater amount of variability in **Y** and the second latent vector, $\mathbf{t}_2$, explains the second greatest amount of variability subject to the condition that it be uncorrelated to the first latent vector and so on. Therefore, the **Y** matrix is decomposed as

$$\mathbf{Y} = \sum_{i=1}^{a} \mathbf{t}_i \mathbf{p}_i + \mathbf{K} = \mathbf{T}_a \mathbf{P}_a^T + \mathbf{K} = \mathbf{Y}\mathbf{P}_a \mathbf{P}_a^T + \mathbf{K} \tag{3. 10}$$

where $\mathbf{p}_i$ and $\mathbf{t}_i$ are the loading vector and column score vector of i^{th} principal component, respectively and $\mathbf{K}$ is the residual matrix. One can compute as many principal components as the rank of $\mathbf{Y}$ however, in the case of correlated measured variables, the first few latent vectors explain most of the variability in $\mathbf{Y}$. The number of principal components, a, to be retained in the model can be determined by the cross-validation technique (Wold, 1978).

In PCA, the objective function to be maximised is

$$\begin{aligned} &\max \quad \mathbf{p}_i^T \mathbf{Y}^T \mathbf{Y} \mathbf{p}_i \\ &\text{subject to } \mathbf{p}_i^T \mathbf{p}_i = 1 \\ &\qquad \mathbf{p}_i^T \mathbf{p}_j = 0 \quad i \neq j \end{aligned} \tag{3. 11}$$

For the above objective function to be at its maximum, the loading vectors, $\mathbf{p}$, are the eigenvectors associated with the eigenvalues of $\mathbf{Y}^T\mathbf{Y}$. Once, the statistically significant number of principal component have been determined, the regression coefficients can be estimated by regressing the dominant principal components on the inputs:

$$\mathbf{B}_{PCA} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\left(\mathbf{Y}\mathbf{P}_a\mathbf{P}_a^T\right) \tag{3. 12}$$

3.3.3.2 Canonical Correlation Regression

In canonical correlation regression (CCR), the objective is to find linear combinations of $\mathbf{X}$, $\mathbf{X}\mathbf{f}_i$, and of $\mathbf{Y}$, $\mathbf{Y}\mathbf{g}_i$, that are most highly correlated. Furthermore, the subsequent linear combinations or canonical variates must be chosen with an additional constraint that they be orthogonal to the previous ones. Therefore, the objective function is:

$$\begin{aligned} &\max_{\mathbf{f}_i, \mathbf{g}_i} \quad \mathbf{f}_i^T \mathbf{X}^T \mathbf{Y} \mathbf{g}_i \\ &\text{subject to } \mathbf{f}_i^T \mathbf{X}^T \mathbf{X} \mathbf{f}_i = 1, \quad \mathbf{g}_i^T \mathbf{Y}^T \mathbf{Y} \mathbf{g}_i = 1 \\ &\qquad \mathbf{f}_i^T \mathbf{X}^T \mathbf{X} \mathbf{f}_j = 0 \quad \mathbf{g}_i^T \mathbf{Y}^T \mathbf{Y} \mathbf{g}_j = 0 \end{aligned} \tag{3. 13}$$

The solutions for $\mathbf{f}_i$ and $\mathbf{g}_i$ are the eigenvectors associated with the eigenvalues of $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{X}$ and $\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$, respectively. The eigenvalues range between 0 and 1 and are the squared correlation coefficients between $\mathbf{X}\mathbf{f}_i$ and $\mathbf{Y}\mathbf{g}_i$. The number of statistically significant canonical variates signifies the underlying dimensionality of the process output space and the canonical variates whose eigenvalues include zero in confidence intervals are white noise. The CCR regression coefficients are given by the following equation:

$$\mathbf{B}_{CCR} = \left(\mathbf{F}_a\mathbf{F}_a^T\right)\mathbf{X}^T\mathbf{Y} \tag{3. 14}$$

The maximum number of canonical variates that can be computed is the minimum of the number of variables in **X** or **Y** blocks. Furthermore, if one computes all canonical variates, then regression coefficients estimates will be same as OLS estimates. The number of statistically significant canonical variates, a, can be selected either by using the Bartlett criterion (Bartlett, 1947) or by validation on a testing data set. One can also use the generalised cross-validation based canonical correlation regression (GCV-CCR) method of Breiman and Friedman (1996) where a shrinkage, which is computed based on the ratio of the number of regression parameters to the sample size (i.e., signal to noise ratio), is applied to each canonical variate. The canonical variates with smaller eigenvalues represent lower signal to noise ratio and thus more shrinkage is applied to the high noise contaminated components to lessen their effect on the regression coefficients.

If the rank of the regression coefficients matrix, **B**, in Equation (3. 9) is known (i.e., the number of independent parameters to be estimated in **B** is known), then performing canonical analysis on the data and selecting as many canonical variates as the rank of **B** is equivalent to solving for the determinant of the matrix given in Equation (3. 8). The proof is given in Tso, 1981.

3.3.3.3 Reduced Rank Regression

Reduced rank regression (RRR) can be derived from two different techniques: redundancy analysis and the best rank a approximation to a matrix (Burnham et al., 1996). In this section, derivation for RRR is presented using the concept of best rank approximation a to a matrix.

The multivariate linear regression model is given in Equation (3. 9). If the rank, a, of the regression coefficients is known a priori, then the OLS regression coefficients matrix solution can be constrained to be of at most rank a. The objective function then becomes,

$$\min_{\mathbf{B}_a} \|\mathbf{Y}-\mathbf{X}\mathbf{B}_a\|^2 \text{ subject to } \text{rank}(\mathbf{B}_a) \le a \tag{3. 15}$$

The sum of squares of the residual in the above equation can be decomposed into the following two parts:

$$\|\mathbf{Y}-\mathbf{X}\mathbf{B}_a\|^2 = \|\mathbf{Y}-\hat{\mathbf{Y}}_{OLS}\|^2 + \|\hat{\mathbf{Y}}_{OLS}-\mathbf{X}\mathbf{B}_a\|^2 \tag{3. 16}$$

The first term of the decomposition contains OLS solution residuals and is thus constant. Minimisation of the above equation therefore depends on the minimisation of second term on the right hand side as $\mathbf{B}_a$ is allowed to vary subject to having at most rank a. This clearly shows that solution to $\mathbf{B}_a$ depends on the best rank a approximation to the OLS predictions, $\hat{\mathbf{Y}}_{OLS}$. The RRR estimates are obtained by first performing PCA on $\hat{\mathbf{Y}}_{OLS}$.

$$\hat{\mathbf{Y}}_{OLS} = \mathbf{T}\mathbf{J}^T + \mathbf{H} = \hat{\mathbf{Y}}_{OLS}\mathbf{J}_a\mathbf{J}_a^T + \mathbf{H} \tag{3. 17}$$

and then regressing the statistically significant principal components of $\hat{\mathbf{Y}}_{OLS}$ on **X**-block:

$$\mathbf{B}_{RRR} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\left(\mathbf{Y}\mathbf{J}_a\mathbf{J}_a^T\right) \tag{3. 18}$$

The differences among PCA, CCR and RRR are summarised in Table 3. 1. The major difference among these regression methods is the choice of matrix used to find the underlying dimensionality of the output space. In PCA, the reduced space is found using actual measurements of **Y** whereas, in RRR, this is done using OLS estimates of **Y**, $\hat{\mathbf{Y}}_{OLS}$.

In CCR, the canonical variates are computed using $\hat{\mathbf{Y}}_{OLS}$ but are normalised with respect to **Y**.

Method	PCA on Y	RRR	CCR
Matrix used to compute latent variables	$\mathbf{Y}^T\mathbf{Y}$	$\hat{\mathbf{Y}}_{OLS}^T\hat{\mathbf{Y}}_{OLS}$	$(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ $=(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{X}\mathbf{B}_{OLS}$ $=(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\hat{\mathbf{Y}}_{OLS}=(\mathbf{Y}^T\mathbf{Y})^{-1}\hat{\mathbf{Y}}_{OLS}^T\hat{\mathbf{Y}}_{OLS}$

Table 3. 1: Comparison of matrices used to compute the latent variables for various multivariate regression methods.

3.3.3.4 Partial Least Squares

Two block partial least squares (PLS2), where all outputs are modelled simultaneously, is also used for multi-output identification. An introduction to PLS is given in chapter 2.

3.4 Process Example #1: Quality Control

The first process example is taken to resemble a quality control process. The following model is used to generate the process data:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \frac{0.2B^5}{1-0.8B}\begin{bmatrix} 1.0 & 0.7 \\ 0.7 & -1.0 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} \qquad (3.19)$$

The additional output variables were computed as linear combinations of the true values (y^*) of the above two process variables:

$$\begin{bmatrix} y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} 1.0 & 1.0 \\ 1.0 & 0.2 \\ 0.2 & 1.0 \end{bmatrix}\begin{bmatrix} y_1^* \\ y_2^* \end{bmatrix} + \begin{bmatrix} \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \end{bmatrix} \qquad (3.20)$$

The rank of the true, noise-free output space is 2. Obviously the five output variables are related by a model with common parameters. White noise with a signal to noise ratio of 2.25 to 1 based on the variance (i.e., $\sigma_y^2 / \sigma_e^2 = 2.25$) was added to all output variables. Finite impulse response models relating the process inputs to process outputs were estimated using various multi-output regression methods listed in section 3.3.3. The finite impulse response model for the multi-output estimation can be expressed as follows :

$$\mathbf{Y}_t = \mathbf{X}_t\mathbf{B} + \mathbf{E}_t \qquad (3.21)$$

where $\mathbf{Y}_t = \left[y_t^1, y_t^2, \ldots, y_t^5\right]$, $\mathbf{X}_t = \left[u_{t-1}^1, u_{t-2}^1, \ldots, u_{t-30}^1, \ldots, u_{t-1}^2, u_{t-1}^2, \ldots, u_{t-30}^2\right]$ and $\mathbf{E}_t = [e_t^1, e_t^2, \ldots, e_t^p]$. $\mathbf{B}$ is a matrix of regression coefficients estimates with i^{th} column corresponding to the impulse response weights for the i^{th} output variable, y_t^i. $\mathbf{E}_t$ contains the residuals for each output variable at time t, assumed to be white noise in this study. Thirty impulse weights were computed for each input. One could use either generalised least squares (Clarke, 1967) or an ARX model structure for process identification if the $\mathbf{E}_t$ are autocorrelated. Univariate regression methods such as OLS and PLS1, which model one output at a time, were also used to estimate the FIR models. This is done in order to compare the performance of multivariate regression methods to that of univariate methods.

Four hundred and thirty data points were generated for both training and testing data sets. However, thirty data points were used for data shifting for FIR models, thus, leaving 400 data points for estimation. The data were generated by making PRBS changes, with magnitudes between 1 and -1 and switching interval of 5 sampling periods, to the process inputs. All output variables were scaled to unit variance prior to model estimation.

The following criteria were used to compare MISO and MIMO modelling using various regression methods:

(i) providing the best predictions;

(ii) providing the impulse and step weights closest to the true process; and

(iii) giving a model which leads to control systems with the best robust stability.

The following results are based on 500 Monte Carlo simulations. A different seed was used to generate the process input-output data for each simulation.

3.4.1 Best Model in Terms of Predictions

The estimated models from various regression methods are compared in terms of percent sum of squares (% SS) explained of the first two primary outputs for training and testing data sets. All FIR models are estimated using training data sets with 400 data points. The number of canonical variates in CCR and RRR models and the number of latent vectors in PLS and PCA models are selected by validating the models on a testing data set. The number of canonical variates or latent vectors were selected such that they provided the minimum predicted residuals sum of squares (PRESS) on the testing data set. The shrunk CCR method of Breiman and Friedman (1996) based on generalised cross-validation (GCV) was also investigated for multi-output estimation of FIR models. It shrank the three canonical variates with smaller eigenvalues to zero and thus giving results very similar to those of CCR. Therefore, the results for GCV-CCR are not presented here.

The results for prediction ability of both MISO (OLS and PLS1) and MIMO (PLS2, CCR, RRR and PCA) identification regression methods are provided in Table 3. 2. All estimated models are compared in terms of the percent sum of squares (% SS) explained of the first two primary output variables for both the training and testing sets. The % SS explained of the output is calculated as follows:

$$\% \text{ SS explained} = \left(1 - \frac{\sum_{t=1}^{n} \left(y_{i,t} - \hat{y}_{i,t} \right)^2}{\sum_{t=1}^{n} y_{i,t}^2} \right) \qquad (3.22)$$

where $y_{i,t}$ is the observation of i^{th} output variable at time t; $\hat{y}_{i,t}$ is the prediction of the i^{th} output variable and n is the number of data points in the testing or training data sets. The

theoretical % SS explained of both outputs for both training and testing data sets is also given in Table 3. 2. It is the % SS explained by the true process models and can be computed by replacing $\hat{y}$ by the noise-free y in the above equation.

	% Sum of Squares (SS) Explained of the Output Variables			
Regression	Training Data Set		Testing Data Set	
Method	y_1	y_2	y_1	y_2
Theoretical	69.48	69.55	69.54	69.46
OLS	74.05	74.08	64.04	64.21
PCA	71.63	71.67	67.00	67.03
CCR	71.63	71.66	66.97	66.99
RRR	71.65	71.68	56.99	67.02
PLS2	70.20	70.30	68.24	68.20
PLS1	71.01	71.10	67.79	67.77

Table 3. 2: Average results for % sum of squares (SS) explained of the primary output variables for the training and testing data sets by FIR models estimated using various MISO and MIMO regression methods (Number of data points in the training data set = 400).

It is evident from Table 3. 2 that all MIMO and MISO methods provide higher % SS explained than the theoretical % SS explainable for the training data set. This means that all methods are overfitting to some degree. OLS seems to give the most overfitting. MIMO methods such CCR, RRR and PCA provide less overfitting on the training data set and better predictions on the test data set than OLS. This implies that multi-output methods are taking advantage of the relevant information present in the redundant output variables to identify better models for the primary output variables. The number of canonical variates or principal components selected based on validation on a testing data set by these methods were two. This is same as the true rank of the output space. Two block PLS (PLS2), which models all five output variables simultaneously, provides the

best results in terms of the prediction ability. However, PLS1, which models one output variable at a time, also provides better results than multi-output methods such as CCR, RRR and PCA. The improvement in PLS1 results is due to better handling of the ill-conditioning in the $\mathbf{X}^T\mathbf{X}$ matrix. For PLS2, an additional improvement in results comes from the inclusion of the additional variables in the output vector. However, in this example, PLS2 provides very little improvement over PLS1. These findings are consistent with those of Brooks and Stone (1994) who, to their disappointment, found that PLS2 did only slightly better than its single output counterpart PLS1 in a steady state modelling study.

3.4.2 Closeness of the Fit to the True Model

Another comparison criterion used is the closeness of the fit to the true model. This is measured by amount of departure from the true impulse and step weights of the process example. The sum of the squared deviations from the true impulse response weights (MSE_Impulse) and the sum of the squared deviations from the true step weights (MSE_Step) are used to quantify the closeness of the fit to the true model. These quantities are defined in Equations (2.40) and (2.41). The MSE_Impulse primarily measures the closeness of the estimated process dynamics to the true process dynamics whereas the MSE_Step primarily measures the closeness of accuracy of the process steady state gain. The quantities MSE_Impulse and MSE_Step for first output are provided in Table 3. 3. The results for the second output are similar to the results for first output and are thus not shown here.

Again, OLS, which models one output at a time, provides the worst results. The multi-output methods such as CCR, RRR and PCA provide better results than OLS but there seems to be very little difference among the results for these multi-output methods. They all provide nearly identical results. PLS2 and PLS1 provide the best results and there is very little difference between the results for PLS1 and PLS2. As discussed in the previous section, the improvement in PLS results comes from the fact that it smoothes the

FIR coefficients by overcoming ill-conditioning of the input data by selecting fewer latent vectors.

Regression Method	MSE_Impulse (Deviations from the True Impulse Weights)		MSE_Step (Deviations from the True Step Weights)	
	y_1-u_1	y_1-u_2	y_1-u_1	y_1-u_2
OLS	0.2947	0.2784	0.2010	0.1954
PCA	0.1317	0.1288	0.1061	0.1111
CCR	0.1338	0.1298	0.1109	0.1126
RRR	0.1319	0.1287	0.1094	0.1111
PLS2	0.0153	0.0095	0.0826	0.0808
PLS1	0.0181	0.0120	0.0923	0.0943

Table 3. 3: Average results for deviations from the true impulse and step response models given by FIR models estimated using various MISO and MIMO regression methods (Number of data points in the training data set = 400).

3.4.3 Frequency Domain Analysis

The objective of this section is twofold: (i) to examine precision and accuracy of estimated FIR models in the frequency domain using all the identification methods; and (ii) to examine a measure of the stability robustness of the models identified by the various methods when used in feedback controllers. The ellipses for 98% approximate joint confidence regions for the real and imaginary components of the estimated process model $\hat{G}(iw)$ are plotted on the Nyquist diagram at selected frequencies. The 90% bound on the multiplicative or relative process/model mismatch is also plotted as function of frequency.

3.4.3.1 Joint Confidence Regions on the Nyquist Plot

In this section, we examine the accuracy and precision of the various MISO and MIMO identification methods by computing approximate joint confidence regions for the

frequency response of the system at various frequencies. The results are shown only for the FIR models between y_1 and u_1 and the results for other models are similar to the ones shown here. Further details regarding the calculation of these confidence regions can be found in chapter 2.

The ellipses for approximate joint confidence regions on the Nyquist plot were computed at the following frequencies: 0.01, 0.2231 and 0.4 rad/time unit. These frequencies correspond to the process steady state, open loop process time constant (τ_p=4.184) and a selected closed loop process time constant (τ_c=2.5), respectively.

The ellipses for the approximate 98% joint confidence regions on the Nyquist plot for each regression method are shown in Figure 3.1. As is evident from the plot, the multivariate methods such as CCR, RRR and PCA provide the smallest joint confidence regions at all three selected frequencies. Smaller confidence regions mean that there is less uncertainty in the process models. The ellipses for joint confidence regions for these three multivariate regression methods are identical and are almost superimposed on each other. OLS provides the largest joint confidence regions. Comparing the multivariate regression methods' joint confidence regions to univariate regression method confidence regions, the joint confidence regions for the multivariate regression methods are about half the area of those for OLS. This definitely means that these multivariate methods are benefiting from the information available in the redundant output variables and thus providing more accurate FIR models.

PLS2 does provide better results than OLS and PLS1 however the results are not as good as for CCR, RRR and PCA. Even though smoothing of impulse weights by PLS methods does lead to better prediction ability and smaller mean square error for impulse and step weights, it does not translate into smaller confidence regions in the frequency domain. The areas for ellipses for confidence regions for both PLS1 and OLS are comparable.

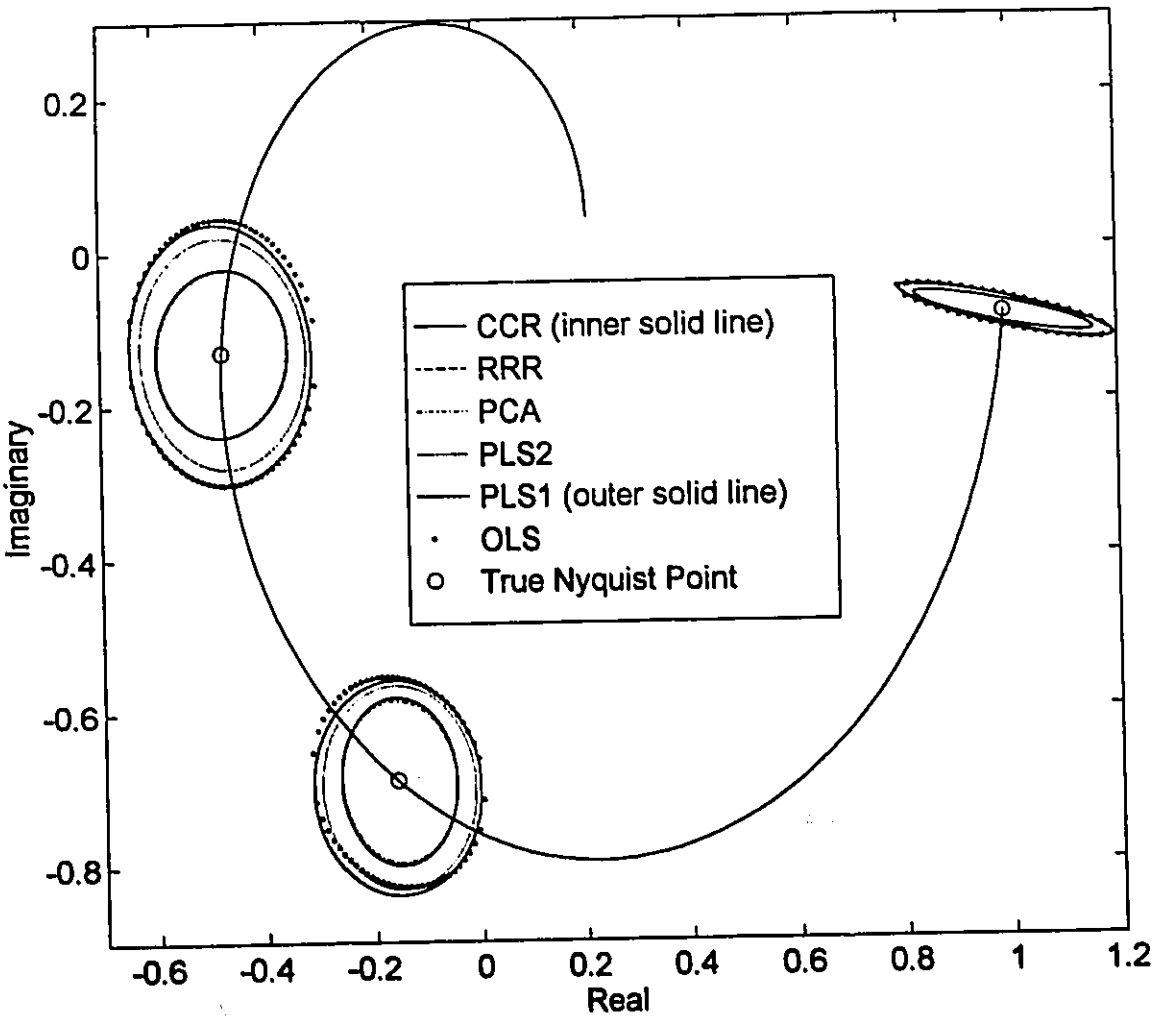

Figure 3.1: Approximate ellipses for the 98% joint confidence regions on the Nyquist plot for the models estimated for y_1-u_1 with a training data sets of 400 data points.

3.4.3.2 Stability Robustness Analysis

The objective of this section is to compare a measure of robustness of a model-based control system that would result from using the models identified by the various approaches. The measure is very general and independent of the specific controller design.

For all the regression methods investigated in this study, the 90% bounds on the inverse of multiplicative uncertainty (i.e., $1/\|e_M(iw)\|_2$) for the frequency range between [0.01 2]/T_s are shown in Figure 3.2. The small gain theorem is generally very conservative and is only a sufficient condition for robust stability. It implies that the system is stable as long as this condition is met. It does not imply that the system will become unstable even if this condition is exceeded. However, these small gain stability bounds should provide a useful measure of the relative robustness of the control system based on the models identified by the various methods.

The multivariate regression methods, CCR, RRR and PCA, provide the highest bounds and hence the best implied robustness. PLS2 provides slightly better robust stability than single output methods such as OLS and PLS1 but is still not as good as CCR, RRR and PCA. PLS focuses on singular directions in which the covariance is large and it drops weaker singular directions at certain frequencies because there is relatively little covariance in these singular directions (Wise and Ricker, 1993). Therefore, it gives the best predictions and the smallest MSE deviations from the true impulse weights but gives poor robust stability compared to other multivariate regression methods.

3.4.4 Multi-Output Estimation with Less Data

In most cases, one might not be able to conduct experiments to collect such a large amount of data or use such large input changes due to product degradation or market demands. In this section, the effect of fewer number of data points on the multi-output estimation is investigated. One hundred fifty (150) data points instead of 400 were used in the training data set for multi-output estimation.

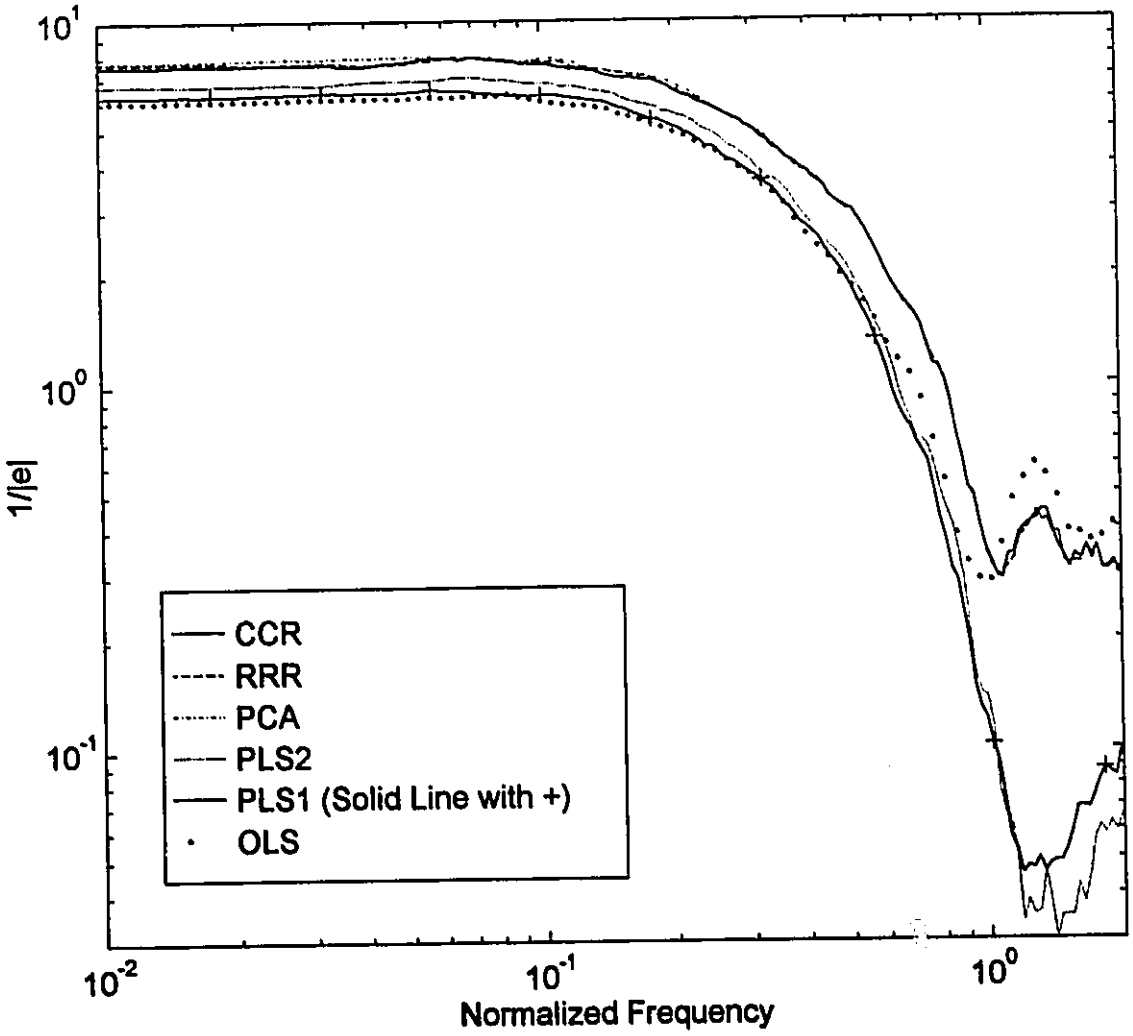

Figure 3.2: Approximate 90% bound on maximum allowable controller gain (Small gain theorem) for the process models estimated with training data sets with 400 data points.

The results for the prediction ability of the identified models and closeness of the fit to the true model are given in Table 3. 4 and Table 3. 5, respectively. The ellipses for the joint 98% approximate confidence regions on the Nyquist plot are shown in Figure 3.3 and the robust stability results, the 90% bound on the inverse of multiplicative uncertainty (i.e., $1/\|e_M(iw)\|_2$), are shown in Figure 3.4. With fewer data points, the differences between OLS and the multivariate methods such CCR, RRR and PCA become greater based on all comparison criteria. In terms of prediction ability of the identified models, the multivariate methods provide much better predictions than OLS. Both univariate and multivariate PLS methods, PLS1 and PLS2, provide better results than CCR, RRR and PCA in terms of prediction ability and closeness of the fit to the true models, however, they provide poor results in terms of frequency domain criteria. Furthermore, PLS2 provides marginal improvement over PLS1. As discussed in section 3.4.1, the major benefit of both univariate and multivariate PLS methods arises from their ability to overcome ill-conditioning in the input data and there is only marginal improvement for multi-output estimation by using PLS2.

Among CCR, RRR and PCA methods, the results for RRR and PCA are slightly better than that of CCR. It is probably due to the fact that in case of RRR, one tries to find linear combinations in X which try to explain the variance of outputs whereas, in the case of CCR, one finds linear combinations in X and Y which are strongly correlated.

3.4.5 Multi-Output Estimation with Different Signal to Noise Ratios for Output Variables

In the previous sections, all measured output variables had the same signal to noise ratio. However, this is seldom the case. There are different measurement noise variances associated with each sensor. Some sensors such as temperatures or an expensive chromatograph can provide more precise measurements whereas some sensors such as an inexpensive chromatograph provide only crude measurements. In this section, it is investigated whether more precise measurements for one of the variables can lead to better

Regression Method	% Sum of Squares (SS) Explained of the Output Variables			
	Training Data Set		Testing Data Set	
	y_1	y_2	y_1	y_2
Theoretical	68.84	68.66	68.38	68.73
OLS	81.34	81.10	46.52	47.30
PCA	74.68	74.47	57.91	58.54
CCR	74.46	74.19	57.54	58.14
RRR	74.71	74.50	57.90	58.53
PLS2	71.38	71.34	64.68	65.05
PLS1	73.21	72.98	63.34	63.76

Table 3. 4: Average results for % sum of squares (SS) explained of the primary output variables for the training and testing data sets by FIR models estimated using various MISO and MIMO regression methods (Number of data points in the training data set = 150).

Regression Method	MSE_Impulse (Deviations from the True Impulse Weights)		MSE_Step (Deviations from the True Step Weights)	
	y_1-u_1	y_1-u_2	y_1-u_1	y_1-u_2
OLS	1.1188	1.1105	0.8203	0.8465
PCA	0.5147	0.4998	0.4169	0.4341
CCR	0.5291	0.5149	0.4352	0.4532
RRR	0.5139	0.4996	0.4167	0.4337
PLS2	0.0235	0.0168	0.2823	0.2846
PLS1	0.0304	0.0229	0.3268	0.3929

Table 3. 5: Average results for deviations from the true impulse and step response models given by FIR models estimated using various MISO and MIMO regression methods (Number of data points in the training data set =150).

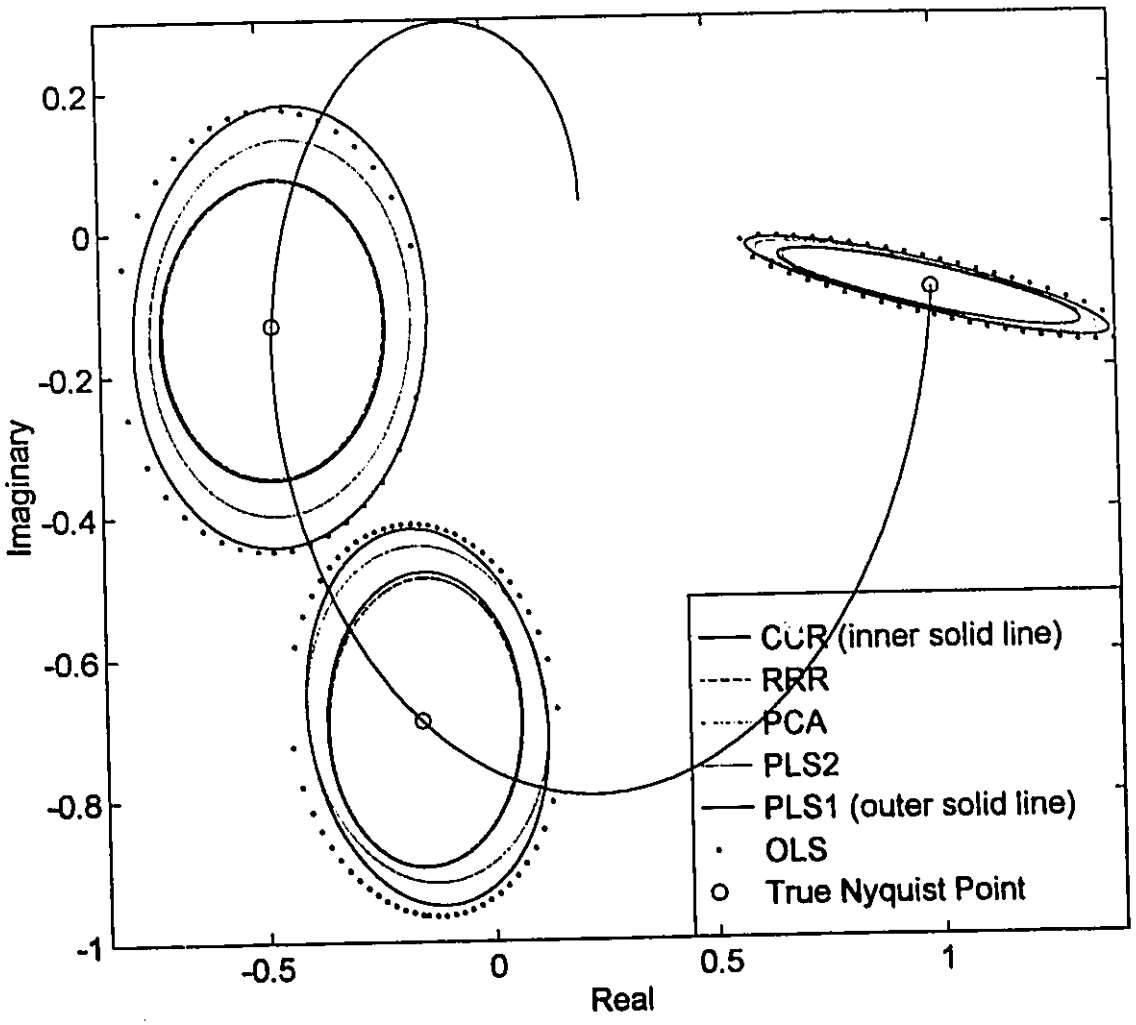

Figure 3.3: Approximate ellipses for the 98% joint confidence regions on the Nyquist plot for the models estimated for y_1-u_1 with a training data sets of 150 data points.

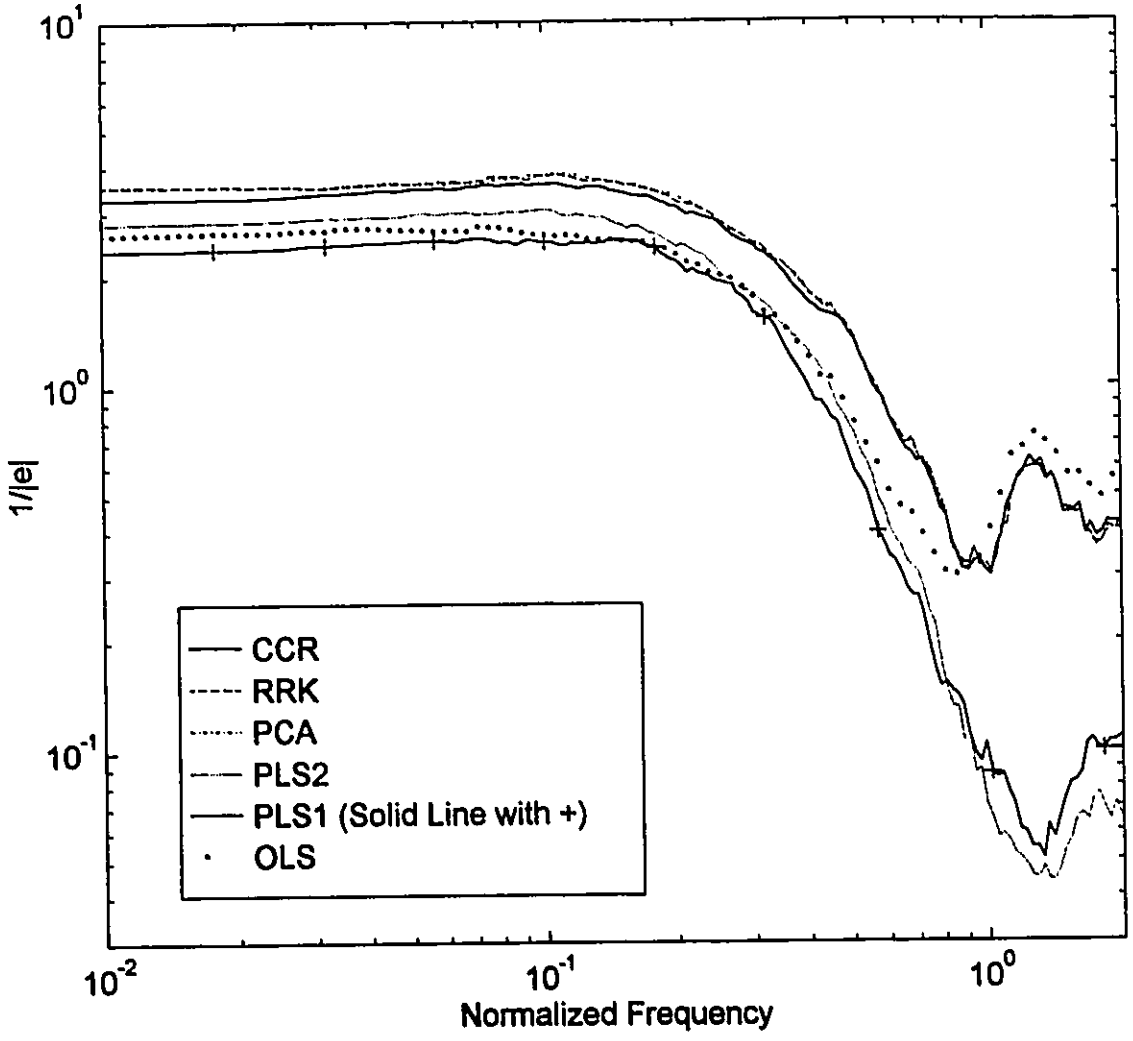

Figure 3.4: Approximate 90% bound on maximum allowable controller gain (Small gain theorem) for the process models estimated with training data sets with 150 data points.

and more robust models for other output variables.

In the previous simulations, white noise with the same signal to noise ratio $\left(\sigma_y^2 / \sigma_\varepsilon^2 = 2.25\right)$ was added to all output variables. In this section, the signal to noise ratio for the third output variable, y_3, is increased to 100:1. One hundred fifty data points were used in the estimation of finite impulse response models using various multivariate regression methods. Since the first two primary variables still have the same signal to noise ratio, therefore, the results for the univariate regression methods are same as the previous results given in sections 3.4.1-3.4.2 .

The results for the prediction ability of the identified models and closeness of the fit to the true model are given in Table 3. 6 and Table 3. 7, respectively. The ellipses for the joint 98% approximate confidence regions on the Nyquist plot are shown in Figure 3.5 and the 90% bound on the inverse of multiplicative uncertainty (i.e., $1/\left\|e_M(iw)\right\|_2$) are shown in Figure 3.6. Of all the methods investigated, CCR provides the most improvement over the results when all output variables have same signal to noise ratio. This is due to the fact that the variance covariance matrix of Y is explicitly built into the objective function for canonical correlation regression. Therefore, the observations for the third output variables with the highest signal to noise ratio are weighted much more heavily than the observations for the remaining outputs.

There is hardly any improvement in the results for PLS2. The results are in fact the same as those with output variables having the same signal to noise ratio.

3.4.6 Summary

All multi-output identification methods are better than single-output OLS on the basis of all comparison criteria. PLS2 gives the best predictions and smallest MSE deviations from the true models. However, other multi-output identification methods such as CCR, RRR and PCA on Y regression provide significantly better robust controller stability than PLS2 which still gives better robust stability than the single output identification methods.

Regression Method	% Sum of Squares (SS) Explained of the Output Variables			
	Training Data Set		Testing Data Set	
	y_1	y_2	y_1	y_2
Theoretical	68.84	68.66	68.38	68.73
OLS	81.34	81.10	46.52	47.30
PCA	75.15	74.91	59.00	59.59
CCR	72.93	72.73	60.76	61.14
RRR	75.15	74.92	59.23	59.80
PLS2	71.44	71.30	64.81	65.20
PLS1	73.21	72.98	63.34	63.76

Table 3. 6: Average results for % sum of squares (SS) explained of the primary output variables for the training and testing data sets by FIR models estimated using various MISO and MIMO regression methods with the third output being more precise (Number of data points in the training data set = 150).

Regression Method	MSE_Impulse (Deviations from the True Impulse Weights)		MSE_Step (Deviations from the True Step Weights)	
	y_1-u_1	y_1-u_2	y_1-u_1	y_1-u_2
OLS	1.1188	1.1105	0.8203	0.8465
PCA	0.4540	0.4468	0.4102	0.4140
CCR	0.3717	0.3697	0.3290	0.3597
RRR	0.4445	0.4378	0.3893	0.3935
PLS2	0.0226	0.0166	0.2756	0.2828
PLS1	0.0304	0.0229	0.3268	0.3929

Table 3. 7: Average results for deviations from the true impulse and step response models given by FIR models estimated using various MISO and MIMO regression methods with the third output being more precise (Number of data points in the Training data set = 150).

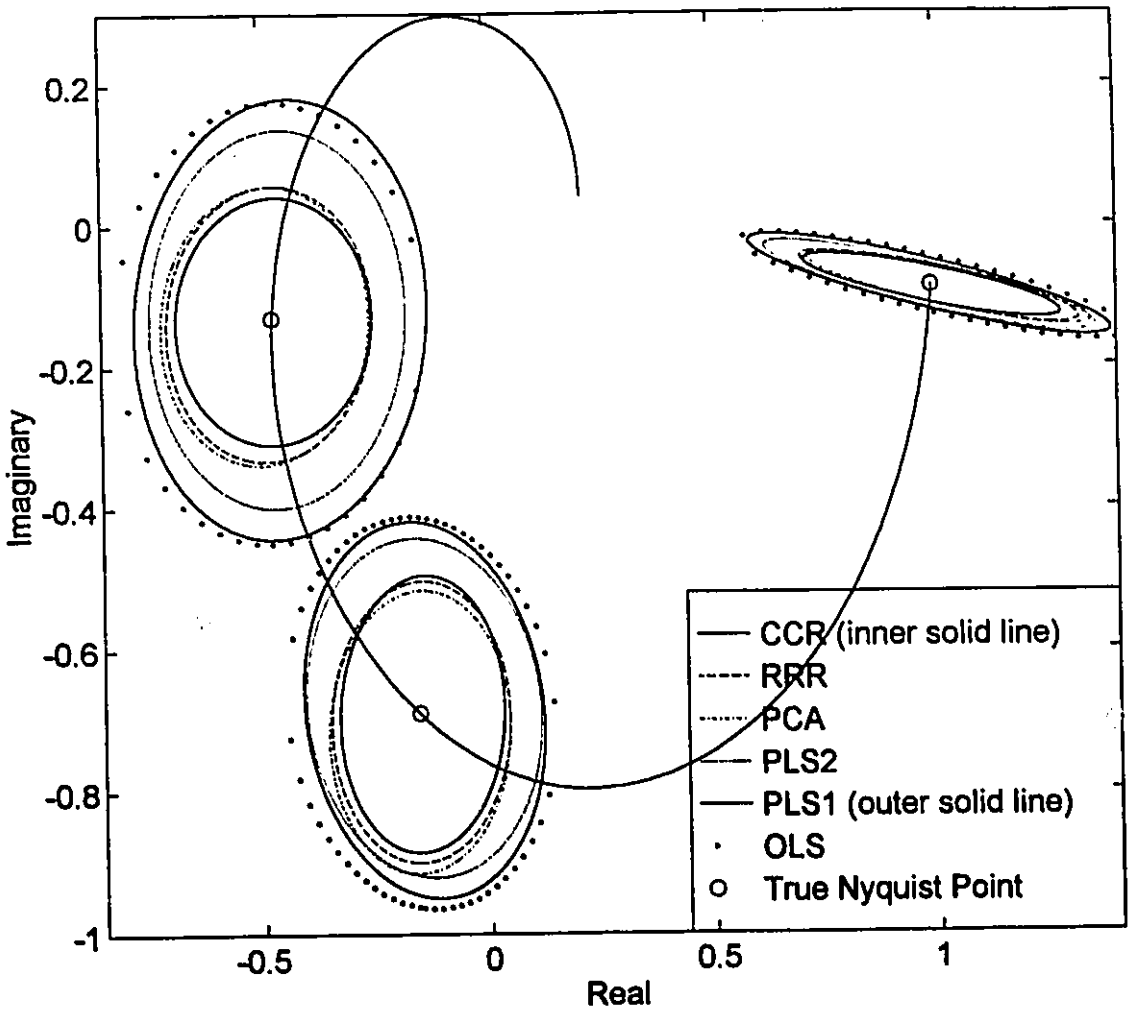

Figure 3.5: Approximate ellipses for the 98% joint confidence regions on the Nyquist plot for the models estimated for y_1-u_1 with a training data sets of 150 data points and the third output (y_3) having a better signal to noise ratio.

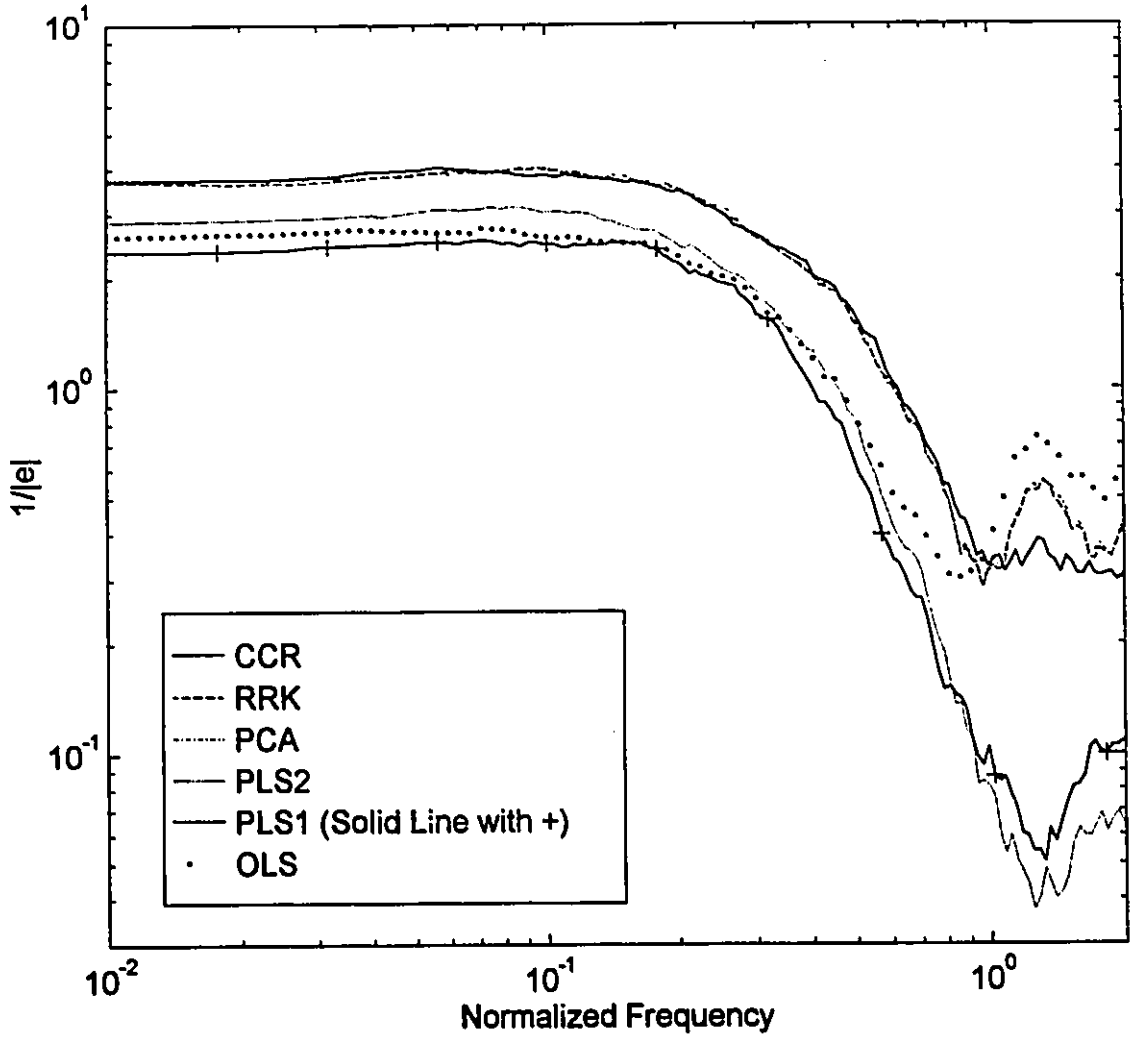

Figure 3.6: Approximate 90% bound on maximum allowable controller gain (Small gain theorem) for the process models estimated with training data sets with 150 data points and the third output (y_3) having better signal to noise ratio.

The reason for the flip-flop behaviour among the multi-output methods of PLS2 versus CCR, RRR and PCA regression, lies in the different fundamental basis behind these methods. PLS2 focuses on maximising the covariance among the X and Y spaces and in this study the number of latent variables retained was based on obtaining the best predictions on a test data set. It, therefore, focuses on high variance directions in the X-space and drops weaker singular directions in the data which have little impact on the prediction variance. However, in process control, the identified models are used not only to predict the process output, but are also inverted to calculate the control action. In this inversion step the weaker singular values and singular directions will dominate and if these are poorly estimated stability robustness will suffer. Since the other multi-output regression methods are based on maximising the correlation rather than the covariance they will not disregard the weak singular directions to the same extent. They will therefore provide better model inverse in the controller and hence better stability robustness.

3.5 Process Example #2: Methanol-Acetone-Water Extractive Distillation Column

The objective of this process example is to illustrate the potential of multi-output identification for processes where parts of the process dynamics between various output variables are similar thereby allowing a common parameterisation for parts of the transfer function models. The model parameters common to different transfer function models are constrained to be the same and the remaining model parameters are allowed to change freely to model process dynamics which are not common among output variables.

The process example used is a methanol-acetone-water (MAW) extractive distillation column simulation. A description of the distillation column is given in chapter 2 and a schematic of the extractive distillation column is shown in Figure 2.2. The output variables for the column are the acetone composition in the top product and methanol

composition on a water free basis (MWF) in the bottoms product. The manipulated variables are the steam temperature to the reboiler and the solvent flow rate. The disturbance is the feed flow rate. Finite impulse response models relating the process output variables to process input and disturbance variables were obtained by making pulse changes to both process inputs and the disturbance. Additional measured variables that are used in the multi-output identification are the trays 1 and 11 temperature measurements. The impulse response models for the tray temperatures are also available. For multi-output identification analysis, these linear models will be used to generate the process data for the extractive distillation column. The condition number of the steady state gain matrix for composition variables is 73.

In this process example, the tray temperatures are used as the secondary process variables only to illustrate the concept of using related variables during the identification. They are considered to be representative of a set of more precise measurements which are available only during the identification study to improve the model parameter estimates for the composition variables.

The impulse response model for both composition and both temperature variables are shown in Figure 3.7. As is evident from these plots, the impulse responses for both the top composition and temperature and for the bottom composition and temperature are very similar. This implies that some of the time constants must be common to the models for both composition and temperature. The data for the output variables from one of the 500 sets generated for process identification are plotted in Figure 3.8. The data have been autoscaled or normalised to unit variance so that the temperature and composition variables, which are measured in different units, can be plotted on the same plot. The top acetone composition and tray 11 temperature are plotted together in Figure 3.8a whereas the bottom MWF composition and tray 1 temperature are shown in Figure 3.8b. As can be seen from these plots, the composition and temperature data for both top and bottom parts of the column look very similar. The composition data are more noisy due to high

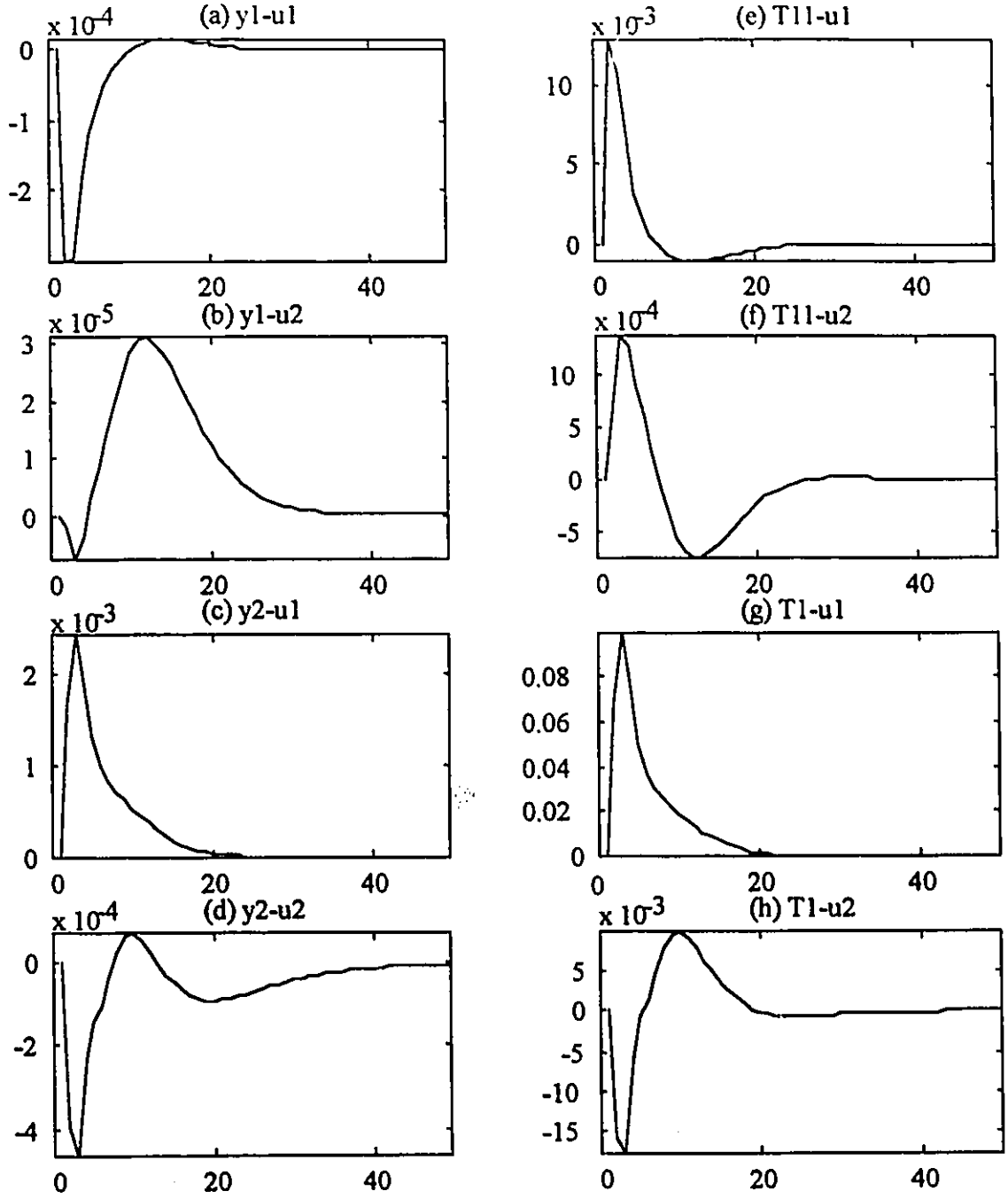

Figure 3.7: MAW process impulse response coefficients for the composition and temperature variables.

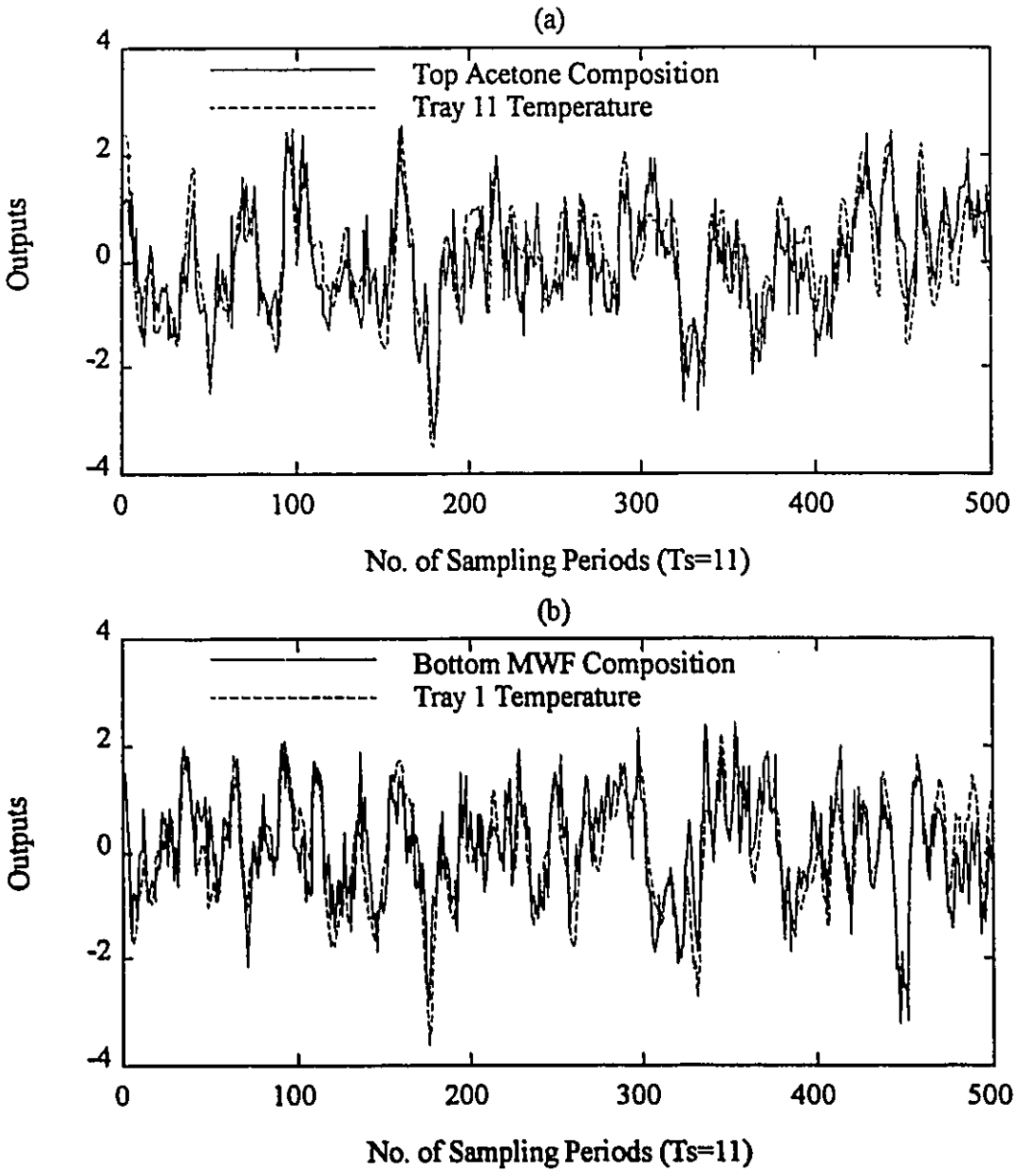

Figure 3.8: Plots of the data for the process output variables.

measurement noise added to them. The trends in the data for both temperature and composition are identical. Therefore, careful examination of the data is one way of determining the variables which have common parameters among them. Furthermore, one can initially identify separate FIR or transfer function models for each output variable and then look for any parameters or trends, which might be similar, in these independently estimated models. Another way of knowing common parameters among process models for different output variables is prior process knowledge.

3.5.1 Process Identification

The following transfer function models in the Laplace domain were identified using nonlinear optimisation:

$$y_1(s) = \frac{K_1(\tau_3 s+1)}{(\tau_1 s^2+\tau_2 s+1)} u_1(s) + \frac{K_2(\tau_7 s+1)}{(\tau_4 s^2+\tau_5 s+1)(\tau_6 s+1)} u_2(s) + \text{Noise Model} \quad (3.23)$$

$$T_{11}(s) = \frac{K_3(\tau_8 s+1)}{(\tau_1 s^2+\tau_2 s+1)} u_1(s) + \frac{K_4(\tau_{10} s+1)}{(\tau_4 s^2+\tau_5 s+1)(\tau_9 s+1)} u_2(s) + \text{Noise Model} \quad (3.24)$$

$$y_2(s) = \frac{K_5}{(\tau_{11} s^2+\tau_{12} s+1)} u_1(s) + \left[\frac{K_6(\tau_{15} s+1)}{(\tau_{13} s+1)(\tau_{14} s+1)} + \frac{K_7(\tau_{19} s+1)e^{-6s}}{(\tau_{16} s+1)(\tau_{17} s+1)(\tau_{18} s+1)}\right] u_2(s) + \text{Noise Model} \quad (3.25)$$

$$T_1(s) = \frac{K_8}{(\tau_{11} s^2+\tau_{12} s+1)} u_1(s) + \left[\frac{K_9(\tau_{21} s+1)}{(\tau_{13} s+1)(\tau_{20} s+1)} + \frac{K_{10}(\tau_{23} s+1)e^{-5s}}{(\tau_{16} s+1)(\tau_{17} s+1)(\tau_{22} s+1)}\right] u_2(s) + \text{Noise Model} \quad (3.26)$$

For MISO identification of composition variables, the model for each composition variable was identified separately. For MIMO identification, the models for the top acetone composition and tray 11 temperature were identified together and the models for the bottom MWF composition and tray 1 temperature were identified together. The time constants, which are assumed to be common to the transfer function models for both

composition and temperature variables, are labelled with same numbers. Since, there are no common parameters for models between top and bottom composition variables or tray 1 and tray 11 temperature variables, therefore, models for all four output variables need not be identified together . A separate disturbance model was identified independently for each output variable.

The sampling period for the composition measurements is 11 minutes. Even though the temperatures can be measured more frequently (i.e., every second), the temperature measurement at every 11^{th} minute is used for estimation so that the measurements for temperatures and compositions are synchronised. Note that temperature variables are used here only for illustration of the ideas. These variables could have been some other measurements which are very costly and collected only once during the identification study. All composition and temperature variables were scaled to unit variance prior to model estimation.

The process models were estimated using a training data set consisting of 500 data points. The data were generated by exciting the process inputs with pseudo random binary sequence (PRBS) changes with a switching interval of 5 sampling periods. The magnitudes for the PRBS changes were 1 and 4.5 for the steam temperature to the reboiler and the solvent flow rate, respectively. The PRBS magnitudes for both inputs were selected such that they contributed equally to output variables' variances. The approximate percentages of disturbance and measurement noise contributions toward the total variances of the output variables are given in Table 3. 8. The temperatures can be measured more precisely and thus have the least amount of measurement noise.

3.5.2 Results

The results in the following sections are based on 500 Monte Carlo simulations and are presented only for the primary output variables, the top acetone composition and the bottom MWF composition.

Output Variable	Percentage of Approximate Disturbance and Measurement Noise Contribution Toward Total Output Variance
Top Acetone Composition (y_1)	48%
Bottom MWF Composition (y_2)	54%
Tray 11 Temperature (T_{11})	31%
Tray 1 Temperature (T_1)	43%

Table 3. 8: Percentage of approximate disturbance and measurement noise contribution to the overall output variables' variances.

3.5.2.1 C oseness of the Fit to the True model

The results for MSE_Impulse and MSE_Step are shown in Table 3. 9 and Table 3. 10, respectively. MIMO identification provides lower MSE_Impulse for all four transfer function models. However, the results for MSE_Step are mixed. MISO identification provides lower MSE_Step quantities for the transfer function models relating the bottom MWF composition to the process inputs. This is probably due to the bias in the composition models caused by the multi-output identification as will be discussed in the next section.

Identification Method	MSE_Impulse (Deviations From the True Impulse Weights)			
	y_1-u_1 ($1x10^{-9}$)	y_1-u_2 ($1x10^{-10}$)	y_2-u_1 ($1x10^{-7}$)	y_2-u_2 ($1x10^{-8}$)
MISO	8.70	7.08	8.90	6.23
MIMO	7.68	5.99	6.61	5.34

Table 3. 9: Average results for deviations from the true impulse response models given by the estimated models using MISO and MIMO transfer function model identification.

Identification Method	MSE_Step (Deviations From the True Step Weights)			
	y_1-u_1 ($1x10^{-6}$)	y_1-u_2 ($1x10^{-7}$)	y_2-u_1 ($1x10^{-4}$)	y_2-u_2 ($1x10^{-6}$)
MISO	1.99	2.79	1.72	7.59
MIMO	1.79	2.12	2.06	7.81

Table 3. 10: Average results for deviations from the true step response models given by the estimated models using MISO and MIMO transfer function model identification.

3.5.2.2 Joint Confidence Regions on the Nyquist Plot

The ellipses for the 98% approximate joint confidence regions on the Nyquist plots for the models relating y_1 and y_2 to process inputs are shown in Figures 3.9 and 3.10, respectively. The solid line represents the ellipses for MISO identification technique and the dashed line represents the ellipses for MIMO identification technique. The approximate joint confidence regions are computed at the following four normalised frequencies: w=[0.006 0.06 0.18 0.6]/Ts rad/time unit. The first three frequencies correspond to the process at steady state, open loop process time constant (based on the bode plot of the larger singular value) and a selected closed loop process time constant, respectively. The fourth frequency is selected to demonstrate the effects of regression methods on the estimated process models at a higher frequency. As is evident from these plots, MIMO identification provides the smallest confidence regions for all four Nyquist plots. However, for the transfer function models relating the output variables to the first input, the confidence regions for MIMO identification are not found to be centred about the true Nyquist point. This is due to some bias present in the assumed transfer function model structures. This bias becomes more evident at higher frequencies

3.5.2.3 Steady State Robust Stability Analysis

Another criterion used to test the robustness of the models is the steady state robust stability (Garcia and Morari, 1985) criterion. The condition for this robust stability

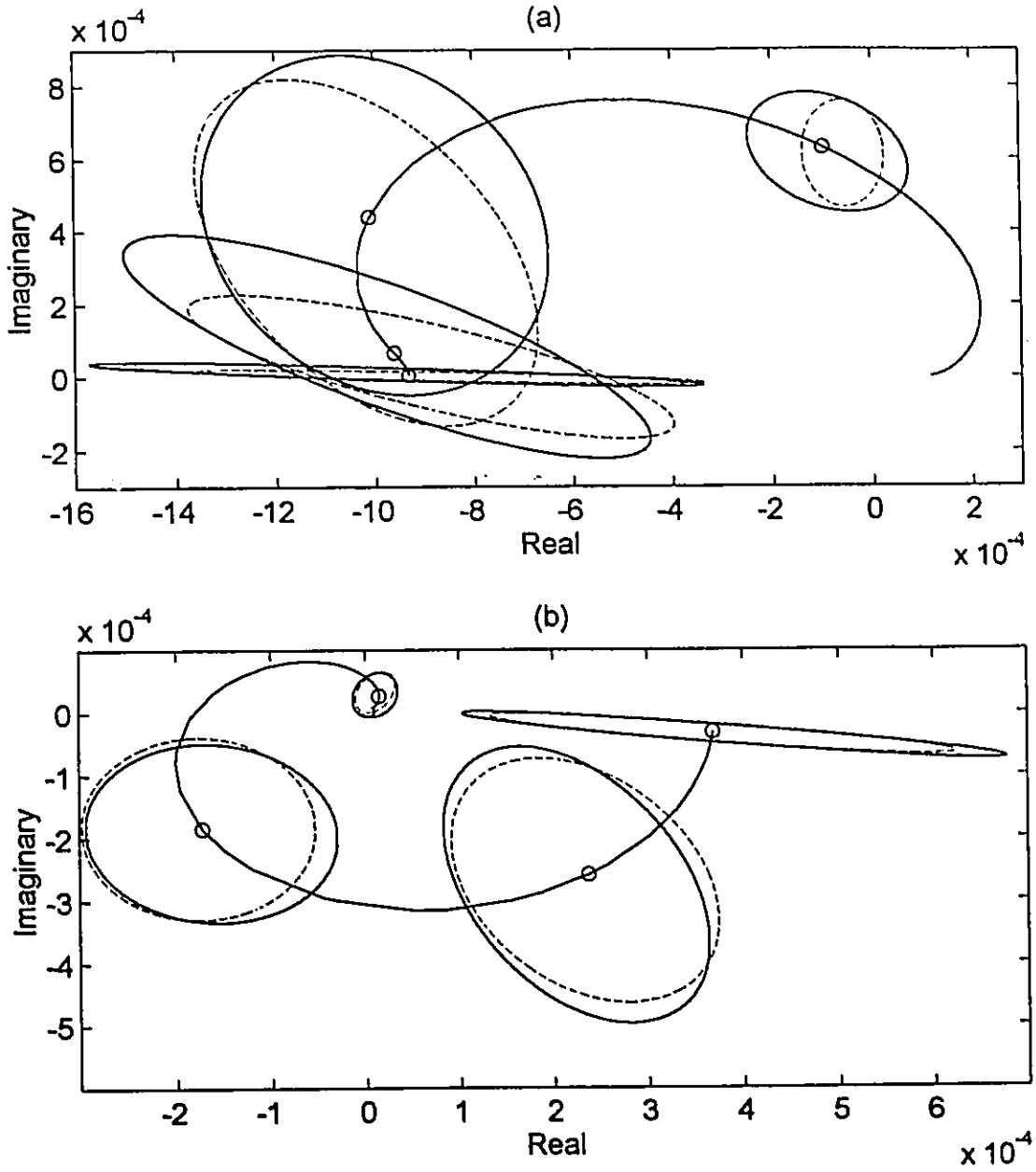

Figure 3.9: Approximate ellipses for the 98% joint confidence regions on the Nyquist plot for (a) y_1-u_1 and (b) y_1-u_2. Legends: solid line - transfer function models estimated using MISO identification technique; dashed line - transfer function models estimated using MIMO identification technique.

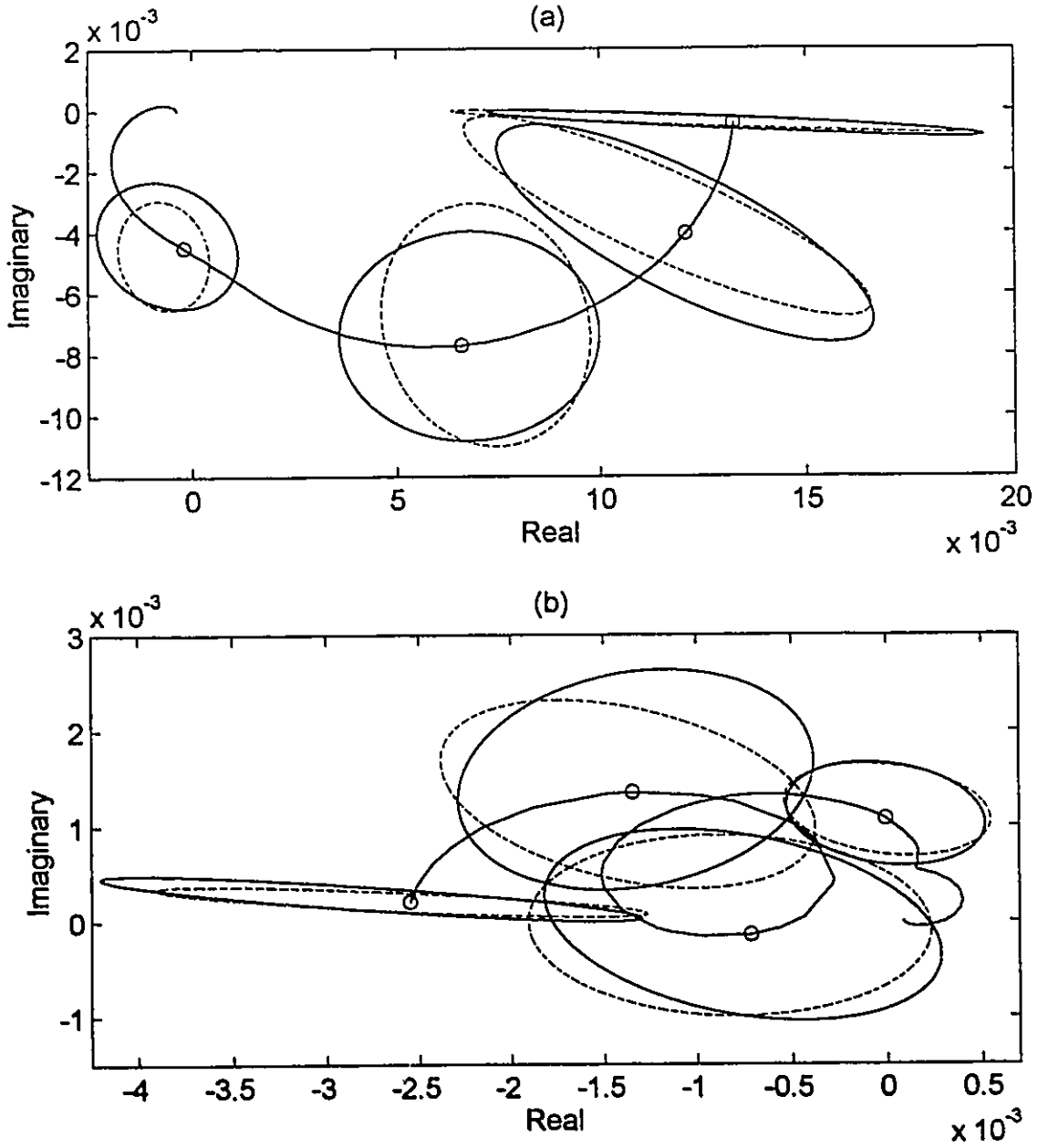

Figure 3.10: Approximate ellipses for the 98% joint confidence regions on the Nyquist plot for (a) y_2-u_1 and (b) y_2-u_2. Legends: solid line - transfer function models estimated using MISO identification technique; dashed line - transfer function models estimated using MIMO identification technique.

is based on only estimating the process steady-state gains correctly. If the real part of one or more of the eigenvalues of $G_p(1)\hat{G}(1)^{-1}$ is negative (see section 2.6.3 of chapter 2 for more details), then the closed loop system can not be stabilised using a multipredictive controller designed with the estimated process models.

The models from both MISO and MIMO identification were analysed using the above robust stability criterion. Fifty six (56) estimated models for MISO identification and twenty nine (29) models for MIMO identification would yield unstable model predictive controllers. Twenty four (24) of the models yielding unstable controllers were estimated from same data sets for both MISO and MIMO identification.

3.5.2.4 Frequency Domain Robust Stability Analysis

The estimated process models for MISO and MIMO identification were also analysed for frequency domain robust stability using the small gain theorem and structured singular value (μ) analysis. These criteria are very conservative but provide a reasonable basis for comparing the expected robustness of controllers that will result from using estimated models from MISO and MIMO identification techniques. The 80% bounds on the inverse of the multiplicative uncertainty (i.e., $1/\|e(iw)\|_2$) computed using both small gain theorem and the μ analysis for the normalised frequency range between 1e-3 and 1 are shown in Figure 3.11a and 3.11b, respectively. These results represent an upper bound on the allowable controller gain at each frequency. The solid line represents the results for MISO identification and the dashed line represents the results for MIMO identification. As seen from these plots, the MIMO identification provides better robust stability than MISO identification over the whole frequency range based on both the small gain theorem and the μ analysis.

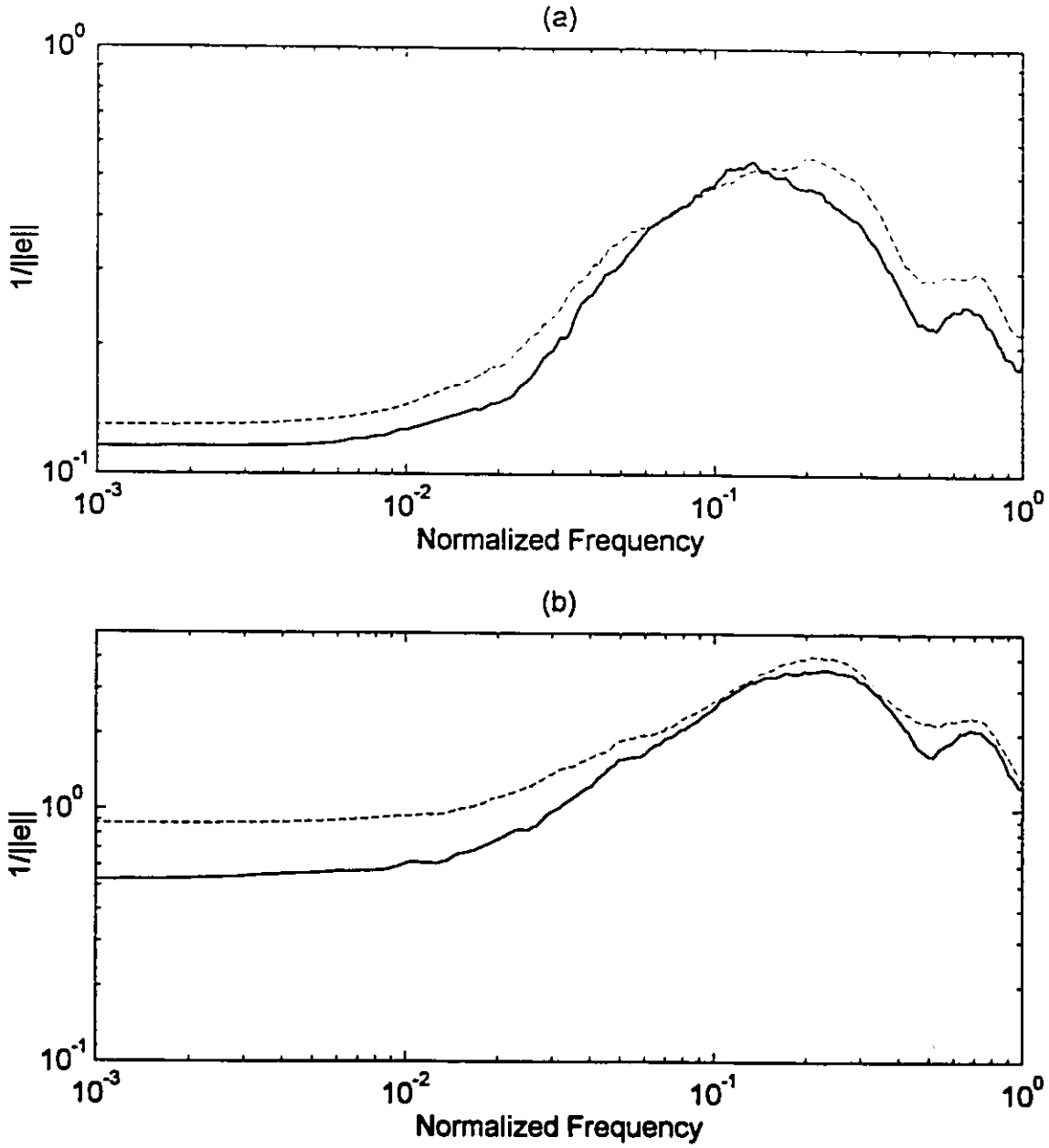

Figure 3.11: Approximate 80% bound on the maximum allowable controller gain. (a) small gain theorem and (b) structured singular value (μ) analysis. Legends: solid line - transfer function models estimated using MISO identification technique; dashed line - transfer function models estimated using MIMO identification technique.

3.5.2.5 Performance of DMC Controllers Using the Identified Models

In this section, we examine the actual performance of the models in a specific controller design - DMC (Cutler and Ramaker, 1979). A dynamic matrix controller (DMC) is designed using the step weights obtained from the identified transfer function models from MISO and MIMO identification techniques. The same DMC tuning parameters were used for controller design for each identification method. The output prediction and control horizons are selected to be 10 and ∞, respectively. The output penalty matrix, $\mathbf{Q}_1$, and the input penalty matrix, $\mathbf{Q}_2$, are

$$\begin{matrix} & y_1 & y_2 \\ \mathbf{Q}_1 = & \begin{bmatrix} 1 & \\ & 10 \end{bmatrix} \end{matrix} \quad \begin{matrix} & u_1 & u_2 \\ \mathbf{Q}_2 = & \begin{bmatrix} 0.1 & \\ & 0.003 \end{bmatrix} \end{matrix} \tag{3.27}$$

Control simulations are carried out for a step change of 5 units to the feed flow rate. The performance of the controllers designed with the models estimated using MISO and MIMO identification techniques is compared in terms of integral of squared error (ISE) for both output variables and integral of squared change in the manipulated variables (ISΔU) for both input variables. The average results for the control simulations are shown in Table 3. 11. The controllers designed from the estimated models for seventy two (72) simulations for MISO and thirty six (36) simulations for MIMO identification techniques were found to be unstable and thus were not included in the computations for the average results. The estimated models for 56 simulations for MISO and 29 simulations for MIMO identification techniques did not meet the steady state robust stability criterion (see section 5.1.3) and thus were unstable. The remaining unstable controllers can be stabilised by imposing larger penalties on the control moves.

The DMC controllers designed using the estimated models from the MIMO identification technique perform better than the controllers designed using the estimated models from the MISO identification technique. Furthermore, the difference becomes much greater when the penalty matrix on the control moves is relaxed.

Identification Method	No. of Unstable Simulations	ISE		ISΔU	
		y_1 ($1x10^{-4}$)	y_2 ($1x10^{-5}$)	u_1	u_2
MISO	72	1.27	4.44	0.60	10.35
MIMO	36	1.11	3.47	0.58	8.08

Table 3. 11: Average closed loop results with DMC controller designed using estimated process models from MISO and MIMO identification techniques. DMC parameters are M=10, P=∞, Q_1=[1 10]; Q_2=[0.1 0.003].

3.5.3 Summary

In this process example, the primary variables of interest (compositions) and the secondary variables (temperatures) have common parameters among their transfer function models. The transfer function models were identified for composition variables using both MISO and MIMO identification techniques. In MISO identification, the transfer function models for each composition variable was identified separately. In MIMO identification, the transfer function models for composition and temperature variables were identified together and some of the time constants in the transfer function models for composition and temperature were constrained to be same.

The MIMO identification provided better results than MISO identification based on all comparison criteria. However, slightly inadequate model structures resulting from the constraints of the common parameterisation among the output variables can lead to some bias in the models identified by MIMO methods.

3.6 Conclusions

In this chapter, the potential of multi-output identification methods for multivariate processes is investigated via simulations on two process examples: a quality control example and an extractive distillation column. Comparisons were made to traditional

single-output identification on the basis of prediction ability of the estimated model, closeness of the fit to the true process model, robust stability provided by the resulting model, and the control performance obtained.

The multi-output identification methods provided better results compared to those of single-output identification methods based on essentially all comparison criteria. However, the benefits for using multi-output identification are most obvious when there are limited amounts of data and when the secondary variables have better signal to noise ratios. The differences between multi-output identification and single-output identification method disappear with larger data sets and better signal to noise ratios for all process variables.

4. Improved PLS Algorithms

4.1 Introduction

A PLS model can be computed using either a classical NIPALS algorithm (Wold, 1982) or a kernel algorithm (Lindgren et al., 1993). These are sequential algorithms where one latent vector is computed at a time and then the **X** and **Y** matrices are deflated to compute the next latent vector.

A typical PLS algorithm is as follows:

(i) Mean-center and scale **X** and **Y** matrices

(ii) Compute the following quantities: $\mathbf{w}_a$, $\mathbf{t}_a$, $\mathbf{q}_a$, $\mathbf{u}_a$ and $\mathbf{p}_a$ using either the NIPALS algorithm or a kernel algorithm. The subscript a refers to the a^{th} latent vector.

(iii) Deflate **X** and **Y** matrices by subtracting the computed latent vectors from them:

$$\mathbf{X}_{a+1} = \mathbf{X}_a - \mathbf{t}_a\mathbf{p}_a^T \tag{4. 1}$$

$$\mathbf{Y}_{a+1} = \mathbf{Y}_a - \mathbf{t}_a\mathbf{q}_a^T \tag{4. 2}$$

(iv) Go to step (ii) to compute the next latent vector.

In PLS, generally both the **X** and **Y** matrices are deflated in step (iii) after each latent vector computation. However, Hoskuldsson (1988) has shown that deflating of **Y** matrix is optional. Lindgren et al. (1993) took advantage of this fact to make the kernel algorithm faster by deflating the **X** matrix only. In this chapter, it is shown that only one of either the **X** or the **Y** matrix needs to be deflated. This result then leads to two new and very fast PLS kernel algorithms.

4.2 Proof that Only One of X or Y Needs to be Deflated

To prove this result one must show that

$$\mathbf{X}_{a+1}^T\mathbf{Y}_{a+1} = \mathbf{X}_a^T\mathbf{Y}_{a+1} = \mathbf{X}_{a+1}^T\mathbf{Y}_a \tag{4. 3}$$

In PLS, X and Y are deflated as shown in Equations (4. 1) and (4. 2). Using Equations (4. 1) and (4. 2), $\mathbf{X}_{a+1}^T\mathbf{Y}_{a+1}^T$ can be expanded as shown below:

$$\begin{aligned}\mathbf{X}_{a+1}^T\mathbf{Y}_{a+1} &= (\mathbf{X}_a - \mathbf{t}_a\mathbf{p}_a^T)^T(\mathbf{Y}_a - \mathbf{t}_a\mathbf{q}_a^T) \\ &= \mathbf{X}_a^T\mathbf{Y}_a - \mathbf{X}_a^T\mathbf{t}_a\mathbf{q}_a^T - \mathbf{p}_a\mathbf{t}_a^T\mathbf{Y}_a + \mathbf{p}_a\mathbf{t}_a^T\mathbf{t}_a\mathbf{q}_a^T \\ &= \mathbf{X}_a^T\mathbf{Y}_a - \mathbf{X}_a^T\mathbf{t}_a\mathbf{q}_a^T - \mathbf{p}_a\mathbf{t}_a^T\mathbf{Y}_a + \mathbf{p}_a\mathbf{q}_a^T\mathbf{t}_a^T\mathbf{t}_a\end{aligned} \tag{4. 4}$$

Note that $\mathbf{t}_a^T\mathbf{t}_a$ is a scalar and, therefore, $\mathbf{p}_a(\mathbf{t}_a^T\mathbf{t}_a)\mathbf{q}_a^T = \mathbf{p}_a\mathbf{q}_a^T(\mathbf{t}_a^T\mathbf{t}_a)$.

The quantities $\mathbf{q}_a$ and $\mathbf{p}_a$ in PLS are computed as follows:

$$\mathbf{q}_a = \frac{\mathbf{Y}_a^T\mathbf{t}_a}{\mathbf{t}_a^T\mathbf{t}_a} \tag{4. 5}$$

and

$$\mathbf{p}_a = \frac{\mathbf{X}_a^T\mathbf{t}_a}{\mathbf{t}_a^T\mathbf{t}_a} \tag{4. 6}$$

Rearranging Equations (4. 5) and (4. 6),

$$\mathbf{t}_a^T\mathbf{Y}_a = \mathbf{q}_a^T(\mathbf{t}_a^T\mathbf{t}_a) \tag{4. 7}$$

and

$$\mathbf{X}_a^T\mathbf{t}_a = (\mathbf{t}_a^T\mathbf{t}_a)\mathbf{p}_a \tag{4. 8}$$

I. Proof that deflation of Y is optional or that $\mathbf{X}_{a+1}^T\mathbf{Y}_{a+1} = \mathbf{X}_{a+1}^T\mathbf{Y}_a$.

Substituting Equation (4. 8) into Equation (4. 4),

$$\begin{aligned} \mathbf{X}_{a+1}^T\mathbf{Y}_{a+1} &= \mathbf{X}_a^T\mathbf{Y}_a - \mathbf{p}_a\mathbf{q}_a^T(\mathbf{t}_a^T\mathbf{t}_a) - \mathbf{p}_a\mathbf{t}_a^T\mathbf{Y}_a + \mathbf{p}_a\mathbf{q}_a^T(\mathbf{t}_a^T\mathbf{t}_a) \\ &= \mathbf{X}_a^T\mathbf{Y}_a - \mathbf{p}_a\mathbf{t}_a^T\mathbf{Y}_a = (\mathbf{X}_a^T - \mathbf{p}_a\mathbf{t}_a^T)\mathbf{Y}_a = (\mathbf{X}_a - \mathbf{t}_a\mathbf{p}_a^T)^T\mathbf{Y}_a \\ &= \mathbf{X}_{a+1}^T\mathbf{Y}_a \end{aligned} \tag{4. 9}$$

II. Proof that deflation of **X** is optional or that $\mathbf{X}_{a+1}^T\mathbf{Y}_{a+1} = \mathbf{X}_a^T\mathbf{Y}_{a+1}$.

Substituting Equation (4. 7) into Equation (4. 4),

$$\begin{aligned} \mathbf{X}_{a+1}^T\mathbf{Y}_{a+1} &= \mathbf{X}_a^T\mathbf{Y}_a - \mathbf{X}_a^T\mathbf{t}_a\mathbf{q}_a^T - \mathbf{p}_a\mathbf{q}_a^T(\mathbf{t}_a^T\mathbf{t}_a) + \mathbf{p}_a\mathbf{q}_a^T(\mathbf{t}_a^T\mathbf{t}_a) \\ &= \mathbf{X}_a^T\mathbf{Y}_a - \mathbf{X}_a^T\mathbf{t}_a\mathbf{q}_a^T \\ &= \mathbf{X}_a^T(\mathbf{Y}_a - \mathbf{t}_a\mathbf{q}_a^T) \\ &= \mathbf{X}_a^T\mathbf{Y}_{a+1} \end{aligned} \tag{4. 10}$$

4.3 NIPALS Algorithm

In the NIPALS algorithm, the computational effort for deflating **X** and **Y** is minimal compared to the iterative process of computing the latent vectors; the exception being when there is only a single response variable (**y**). In this latter case, only two iterations are needed for the NIPALS algorithm to converge for each latent vector, and therefore a significant proportion of the computational effort is spent on deflating **X** and **y**. Since only the (N × 1) **y** vector needs to be deflated after each latent vector computation, the speed of the NIPALS algorithm is substantially improved. The MATLAB® code for the modified NIPALS algorithm is given in the appendix.

4.4 Kernel Algorithm

The kernel algorithm was proposed by Lindgren et al. (1993) and is given below. In the remainder of the chapter, it is assumed that columns of **X** and **Y** matrices are mean-centered and scaled appropriately prior to PLS model estimation using a kernel algorithm.

1. Compute the covariance matrices $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$. The kernel matrix $\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}$ can be computed by multiplication of $\mathbf{X}^T\mathbf{Y}$ with $\left(\mathbf{X}^T\mathbf{Y}\right)^T$.

2. The PLS weight vector $\mathbf{w}_a$ is computed as the eigenvector corresponding to the largest eigenvalue of $\left(\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}\right)_a$ using either the power method or other approaches (SVD, etc.).

$$\mathbf{w}_a \propto \left(\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}\right)_a \mathbf{w}_a \tag{4. 11}$$

3. The PLS loading vectors $\mathbf{p}_a$ and $\mathbf{q}_a$ are computed as given below

$$\mathbf{p}_a^T = \frac{\mathbf{w}_a^T\left(\mathbf{X}^T\mathbf{X}\right)_a}{\mathbf{w}_a^T\left(\mathbf{X}^T\mathbf{X}\right)_a \mathbf{w}_a} \tag{4. 12}$$

$$\mathbf{q}_a^T = \frac{\mathbf{w}_a^T\left(\mathbf{X}^T\mathbf{Y}\right)_a}{\mathbf{w}_a^T\left(\mathbf{X}^T\mathbf{X}\right)_a \mathbf{w}_a} \tag{4. 13}$$

4. After each latent vector computation, the covariance matrices, $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$, can be updated as

$$\left(\mathbf{X}^T\mathbf{X}\right)_{a+1} = \left(\mathbf{I} - \mathbf{w}_a\mathbf{p}_a^T\right)^T\left(\mathbf{X}^T\mathbf{X}\right)_a\left(\mathbf{I} - \mathbf{w}_a\mathbf{p}_a^T\right) \tag{4. 14}$$

$$\left(\mathbf{X}^T\mathbf{Y}\right)_{a+1} = \left(\mathbf{I} - \mathbf{w}_a\mathbf{p}_a^T\right)^T\left(\mathbf{X}^T\mathbf{Y}\right)_a \tag{4. 15}$$

Note that only the **X** matrix is being deflated in the above equations.

4.4.1 Modification to Kernel Algorithm

De Jong and Ter Braak (1994) proposed a modification to step four of the kernel algorithm of Lindgren et al. (1993) whereby the deflation of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$, by expansion of Equations (4. 14) and (4. 15), is simplified to

$$\left(\mathbf{X}^T\mathbf{X}\right)_{a+1} = \left(\mathbf{X}^T\mathbf{X}\right)_a - \mathbf{p}_a\mathbf{p}_a^T(\mathbf{t}_a^T\mathbf{t}_a) \tag{4. 16}$$

$$\left(\mathbf{X}^T\mathbf{Y}\right)_{a+1} = \left(\mathbf{X}^T\mathbf{Y}\right)_a - \mathbf{p}_a\mathbf{q}_a^T(\mathbf{t}_a^T\mathbf{t}_a) \tag{4. 17}$$

This reduces the computational effort during the deflation step by avoiding the multiplication of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ by $\left(\mathbf{I}-\mathbf{w}_a\mathbf{p}_a^T\right)$. This modified kernel algorithm was proven to be much faster than the original kernel algorithm.

4.4.2 A Further Modification to Kernel Algorithm

As shown earlier, only one of either the **X** or the **Y** matrix needs to be deflated. Therefore, in the kernel algorithm, one does not need to deflate **X**. The only deflation necessary is the deflation of **Y** in $\mathbf{X}^T\mathbf{Y}$.

$$\left(\mathbf{X}^T\mathbf{Y}\right)_{a+1} = \left(\mathbf{X}^T\mathbf{Y}\right)_a - \mathbf{X}_a^T\mathbf{t}_a\mathbf{q}_a^T \tag{4. 18}$$

Substituting Equation (4. 8) into the above equation:

$$\left(\mathbf{X}^T\mathbf{Y}\right)_{a+1} = \left(\mathbf{X}^T\mathbf{Y}\right)_a - \mathbf{p}_a\mathbf{q}_a^T(\mathbf{t}_a^T\mathbf{t}_a) \tag{4. 19}$$

Note that the above equation is same as Equation (4. 17). If **X** is not being deflated, then Equations (4. 12) and (4. 13) in the third step in the kernel algorithm need to be modified so that loading vectors $\mathbf{p}_a$ and $\mathbf{q}_a$ can be computed using the non-deflated or original **X** matrix. The score vectors, **T**, can be directly computed from the original **X** by the following equation:

$$\mathbf{T} = \mathbf{XR} \tag{4. 20}$$
$$\left[\mathbf{t}_1\ \mathbf{t}_2\ \ldots\ \mathbf{t}_A\right] = \mathbf{X}\left[\mathbf{r}_1\ \mathbf{r}_2\ \ldots\ \mathbf{r}_A\right]$$

where **R** is given by

$$\mathbf{R} = \mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1} \tag{4. 21}$$

The columns of **R** can be easily computed sequentially. A sequential relationship to compute **R** is derived below:

$$\begin{aligned} \mathbf{t}_1 &= \mathbf{X}_1\mathbf{w}_1 = \mathbf{X}\mathbf{w}_1 \\ \mathbf{t}_2 &= \mathbf{X}_2\mathbf{w}_2 = \mathbf{X}\left(\mathbf{I}-\mathbf{w}_1\mathbf{p}_1^T\right)\mathbf{w}_2 \\ &\vdots \\ \mathbf{t}_A &= \mathbf{X}\left(\mathbf{I}-\mathbf{w}_1\mathbf{p}_1^T\right)\left(\mathbf{I}-\mathbf{w}_2\mathbf{p}_2^T\right)\cdots\left(\mathbf{I}-\mathbf{w}_{A-1}\mathbf{p}_{A-1}^T\right)\mathbf{w}_A \end{aligned} \tag{4. 22}$$

Therefore,

$$\begin{aligned} \mathbf{r}_1 &= \mathbf{w}_1 \\ \mathbf{r}_2 &= \left(\mathbf{I}-\mathbf{w}_1\mathbf{p}_1^T\right)\mathbf{w}_2 \\ &\vdots \\ \mathbf{r}_A &= \left(\mathbf{I}-\mathbf{w}_1\mathbf{p}_1^T\right)\left(\mathbf{I}-\mathbf{w}_2\mathbf{p}_2^T\right)\cdots\left(\mathbf{I}-\mathbf{w}_{A-1}\mathbf{p}_{A-1}^T\right)\mathbf{w}_A \end{aligned} \tag{4. 23}$$

The above relationship can be represented as

$$\mathbf{r}_a = \mathbf{B}_a\mathbf{w}_a \tag{4. 24}$$

where **B** is computed recursively using the following equation (Hoskuldsson, 1992).

$$\mathbf{B}_{a+1} = \mathbf{B}_a\left(\mathbf{I}-\mathbf{w}_a\mathbf{p}_a^T\right) = \mathbf{B}_a - \mathbf{B}_a\mathbf{w}_a\mathbf{p}_a^T = \mathbf{B}_a - \mathbf{r}_a\mathbf{p}_a^T \tag{4. 25}$$

and

$$\mathbf{B}_1 = \mathbf{I} \tag{4. 26}$$

Computing $\mathbf{r}_a$ using the above equations involves maintaining a K by K matrix, $\mathbf{B}_a$, and multiplication of this matrix with $\mathbf{w}_a$ in Equation (4. 24) which could be very computationally intense with larger number of X-variables. This can be avoided by computing $\mathbf{r}_a$ using the following recursive relationship:

$$\begin{aligned} \mathbf{r}_1 &= \mathbf{w}_1 \\ \mathbf{r}_a &= \mathbf{w}_a - \mathbf{p}_1^T\mathbf{w}_a\mathbf{r}_1 - \mathbf{p}_2^T\mathbf{w}_a\mathbf{r}_2 - \cdots - \mathbf{p}_{a-1}^T\mathbf{w}_a\mathbf{r}_{a-1} \quad a > 1 \end{aligned} \tag{4. 27}$$

The major rate determining step in the kernel algorithm in terms of computational effort is the construction of matrices $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$. In this modified kernel algorithm, construction of $\mathbf{X}^T\mathbf{X}$ is not necessary. If the number of data points in **X** are much greater than the number of variables in **X** (N >> K), then it might be convenient to compute $\mathbf{X}^T\mathbf{X}$ since storing $\mathbf{X}^T\mathbf{X}$ would require less memory space than storing **X**. However, if one is

not restricted by the computer memory, then not computing $\mathbf{X}^T\mathbf{X}$ and directly using $\mathbf{X}$ in the kernel algorithm to compute $\mathbf{p}$ and $\mathbf{q}$ would require less computational effort.

In this chapter, two new modified kernel algorithms are developed. In the first algorithm, algorithm #1, the covariance matrix $\mathbf{X}^T\mathbf{X}$ is not computed and $\mathbf{X}$ is used directly in computations for $\mathbf{p}_a$ and $\mathbf{q}_a$ as shown in step 4a (Equations (4. 31)-(4. 33)). In the second modified algorithm, algorithm #2, the covariance matrix $\mathbf{X}^T\mathbf{X}$ is computed once and then used subsequently in the computations for $\mathbf{p}_a$ and $\mathbf{q}_a$ as shown in step 4b (Equations (4. 34)-(4. 35)). The new kernel algorithms are presented below. Algorithms 1 and 2 differ only in steps 1 and 4.

1. Compute the covariance matrix $\mathbf{X}^T\mathbf{Y}$. Computation of $\mathbf{X}^T\mathbf{X}$ is optional.
2. If there are fewer Y-variables, then compute $\mathbf{q}_a$ as the eigenvector corresponding to the largest eigenvalue of $\left(\mathbf{Y}^T\mathbf{X}\mathbf{X}^T\mathbf{Y}\right)_a$ and $\mathbf{w}_a$ can be computed from the following relationship

$$\mathbf{w}_a = \left(\mathbf{X}^T\mathbf{Y}\right)_a \mathbf{q}_a \tag{4. 28}$$

$$\mathbf{w}_a = \frac{\mathbf{w}_a}{|\mathbf{w}_a|} \tag{4. 29}$$

If there are fewer X-variables, then compute $\mathbf{w}_a$ as the eigenvector corresponding to the largest eigenvalue of $\left(\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}\right)_a$ as shown in Equation (4. 11).

3. Compute $\mathbf{r}_a$.

$$\mathbf{r}_1 = \mathbf{w}_1 \tag{4. 30}$$
$$\mathbf{r}_a = \mathbf{w}_a - \mathbf{p}_1^T\mathbf{w}_a\mathbf{r}_1 - \mathbf{p}_2^T\mathbf{w}_a\mathbf{r}_2 - \cdots - \mathbf{p}_{a-1}^T\mathbf{w}_a\mathbf{r}_{a-1} \text{ for } a > 1$$

4. Compute the loading vectors $\mathbf{p}_a$ and $\mathbf{q}_a$ according to either of the following modified algorithms.

(a). Modified Algorithm #1: The covariance matrix $\mathbf{X}^T\mathbf{X}$ is not constructed, but rather $\mathbf{X}$ is used directly in the computations as shown below:

$$\mathbf{t}_a = \mathbf{X}\mathbf{r}_a \tag{4. 31}$$

$$\mathbf{p}_a^T = \frac{\mathbf{t}_a^T \mathbf{X}}{\mathbf{t}_a^T \mathbf{t}_a} \tag{4. 32}$$

$$\mathbf{q}_a^T = \frac{\mathbf{r}_a^T \left(\mathbf{X}^T \mathbf{Y}\right)_a}{\mathbf{t}_a^T \mathbf{t}_a} \tag{4. 33}$$

(b). Modified Algorithm #2: The covariance matrix $\mathbf{X}^T\mathbf{X}$ is computed only once with the original $\mathbf{X}$ data. Then in all dimensions:

$$\mathbf{p}_a^T = \frac{\mathbf{r}_a^T \left(\mathbf{X}^T \mathbf{X}\right)}{\mathbf{r}_a^T \left(\mathbf{X}^T \mathbf{X}\right) \mathbf{r}_a} \tag{4. 34}$$

$$\mathbf{q}_a^T = \frac{\mathbf{r}_a^T \left(\mathbf{X}^T \mathbf{Y}\right)_a}{\mathbf{r}_a^T \left(\mathbf{X}^T \mathbf{X}\right) \mathbf{r}_a} \tag{4. 35}$$

5. Update the covariance matrix, $\mathbf{X}^T\mathbf{Y}$.

$$\left(\mathbf{X}^T \mathbf{Y}\right)_{a+1} = \left(\mathbf{X}^T \mathbf{Y}\right)_a - \mathbf{p}_a \mathbf{q}_a^T (\mathbf{t}_a^T \mathbf{t}_a) \tag{4. 36}$$

6. Store $\mathbf{w}_a$, $\mathbf{p}_a$, $\mathbf{q}_a$, and $\mathbf{r}_a$ in $\mathbf{W}$, $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$.

$$\begin{aligned} \mathbf{W} &= \left[\mathbf{w}_1\ \mathbf{w}_2 \cdots \mathbf{w}_a\right] \\ \mathbf{P} &= \left[\mathbf{p}_1\ \mathbf{p}_2 \cdots \mathbf{p}_a\right] \\ \mathbf{Q} &= \left[\mathbf{q}_1\ \mathbf{q}_2 \cdots \mathbf{q}_a\right] \\ \mathbf{R} &= \left[\mathbf{r}_1\ \mathbf{r}_2 \cdots \mathbf{r}_a\right] \end{aligned} \tag{4. 37}$$

7. Goto step 2 for the next latent vector computation.

8. When done computing latent vectors, the regression coefficients for the PLS model are given by

$$\mathbf{B}_{PLS} = \mathbf{R}\mathbf{Q}^T \tag{4. 38}$$

The MATLAB® code for both modified algorithms is given in the appendix at the end of this chapter.

4.5 Comparison of Kernel Algorithms

Here, the new kernel algorithms are compared to De Jong & Ter Braak's kernel algorithm in terms of speed. All kernel algorithms provided the same PLS models as the NIPALS algorithm. Since the De Jong & Ter Braak's kernel algorithm is proven to be faster than the original kernel algorithm of Lindgren et al. (1993), the original algorithm is not included in this comparison study. The comparison criterion used is the floating points operations (flops) used in MATLAB® to compute the PLS model. This criterion is the same as that of De Jong and Ter Braak (1994) when they compared their kernel algorithm to the original kernel algorithm.

The algorithms were tested for speed to variations in the following four design variables: (i) the number of data points (N); (ii) number of X-variables (K); (iii) number of Y-variables (M); and (iv) number of latent vectors (A) used in the PLS model. The low and high levels of these designed variables are given in Table 4.1. The 2^4 factorial design augmented with a center point and shown in Table 4.2 was performed.

The performance results of the three kernel algorithms are plotted in Figure 4.1. In Figure 4.1a, the total number of flops consumed are plotted against the run number of the factorial design whereas in Figure 4.1b, the relative flops (i.e., total number of flops for given method / total number of flops for the De Jong & Ter Braak method) are shown. The total flops consumed are the combined flops used for constructing the covariance matrices and computing the latent vectors. As is evident from Figure 4.1a, algorithm #1, where $\mathbf{X}^T\mathbf{X}$ is not computed, is much faster than algorithm #2 and the De Jong & Ter Braak algorithm. In Figure 4.1b, it is seen to consume as little as one fifth of the flops as that for the De Jong & Ter Braak algorithm as the number of X-variables increases. There is no apparent difference between algorithm #2 and the De Jong & Ter Braak algorithm since majority of the flops consumed in these algorithms are for construction of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$.

Design Variables	Low	Center	High
Number of Data Points (N)	500	750	1000
Number of X-Variables (K)	10	20	30
Number of Y-Variables (M)	1	3	5
Number of Latent Vectors (A)	3	4	5

Table 4.1: Design variables for speed comparison of various kernel algorithms

Run Number	Number of data points (N)	Number of X-Variables (K)	Number of Y-Variables (M)	Number of Latent Vectors (A)
1	500	10	5	3
2	500	10	5	5
3	500	10	1	3
4	500	10	1	5
5	500	30	5	3
6	500	30	5	5
7	500	30	1	3
8	500	30	1	5
9	750	20	3	4
10	1000	10	5	3
11	1000	10	5	5
12	1000	10	1	3
13	1000	10	1	5
14	1000	30	5	3
15	1000	30	5	5
16	1000	30	1	3
17	1000	30	1	5

Table 4.2: 2^4 factorial design for comparison of various kernel algorithms

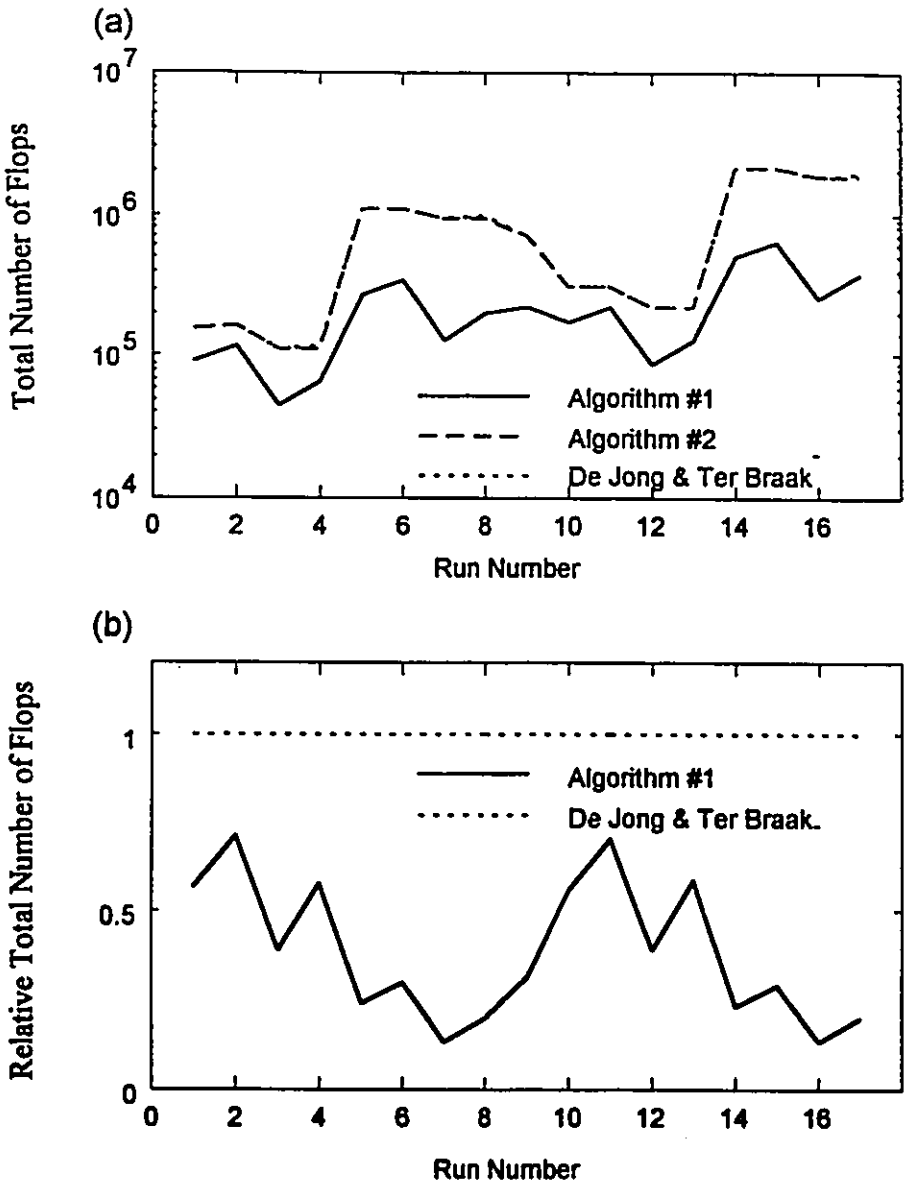

Figure 4.1: Comparison of various kernel algorithms: (a) total number of flops and (b) relative total number of flops. (Note: The lines for algorithm #2 and De Jong & Ter Braak algorithm are superimposed on each other).

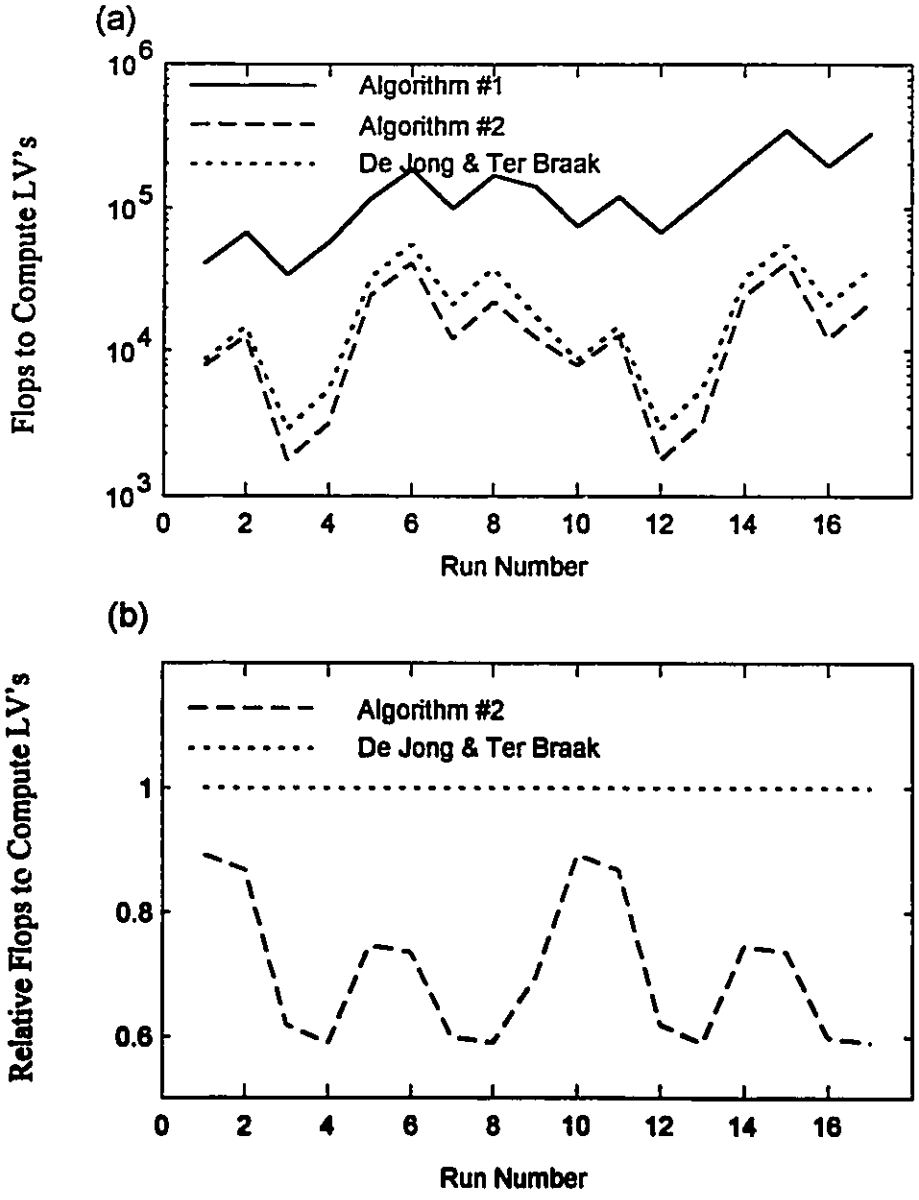

Figure 4.2: Comparison of various kernel algorithms: (a) flops required to compute latent vectors and (b) relative flops required to compute latent vectors.

In Figure 4.2, the flops consumed for computing only the latent vectors of PLS model (after the covariance matrices have been calculated) are plotted against the run number of the factorial design for each of three kernel algorithms. In this case, algorithm #2 provides the best results. Algorithm #1 requires the highest number of flops because it uses $\mathbf{X}$ (which is larger in dimension than $\mathbf{X}^T\mathbf{X}$) directly in the computations for $\mathbf{p}_a$ and $\mathbf{q}_a$. The relative flops for algorithm #2 and the De Jong & Ter Braak algorithm are shown in Figure 4.2b. As is evident, algorithm #2 is much faster than De Jong & Ter Braak algorithm as the number of X-variables increase.

4.6 Discussion

In this section, we discuss the modified algorithms that would be advantageous to use under different circumstances.

4.6.1 Missing Data

Rannar et al. (1995) discussed how to handle missing data with the original kernel algorithm of Lindgren et al. (1993). They proposed a method along the lines of the EM algorithm where the missing data is replaced with some initial values (i.e., column or row mean, etc.). A PLS model is then calculated and the missing data is replaced by their PLS estimates. This procedure is repeated until some convergence criterion is satisfied. The problem they found with this procedure was that after replacing the missing data with PLS estimates, the new covariance matrices need to be recomputed. To overcome this problem, Rannar et al. (1995) proposed a so-called reduced EM algorithm where the replacement of the missing data is accomplished by updating but not recomputing the covariance matrices. This gives a faster but less precise algorithm. As discussed earlier, the construction of $\mathbf{X}^T\mathbf{X}$ is the most computationally intense task in the kernel algorithm. Since the modified algorithm #1 does not require the computation of $\mathbf{X}^T\mathbf{X}$, it can be used

with the full EM algorithm for handling missing data without sacrificing the precision and speed, and is therefore recommended when one has missing data.

4.6.2 Cross-Validation

If one is not sure as to how many latent vectors would be sufficient for a PLS model, then one can use the cross-validation (CV) technique to select the optimal number of latent vectors. In this situation, it might be beneficial to invest in a one time construction of $\mathbf{X}^T\mathbf{X}$ rather than using $\mathbf{X}$ matrix directly in the computations for $\mathbf{p}_a$ and $\mathbf{q}_a$ since these latter vectors would have to be recomputed many times in any CV method. Therefore, modified algorithm #2, which is faster for computing latent vectors once $\mathbf{X}^T\mathbf{X}$ is available, is recommended for cross-validation.

4.7 Conclusions

In PLS, generally both the $\mathbf{X}$ and $\mathbf{Y}$ matrices are deflated after each latent vector computation. In this chapter, a proof is given that only one of either the $\mathbf{X}$ or the $\mathbf{Y}$ matrix in PLS algorithms needs to be deflated. Using this proof, the original kernel algorithm of Lindgren et al. (1993) is modified to develop two new faster and more economical algorithms. In the first algorithm, the covariance matrix $\mathbf{X}^T\mathbf{X}$ is not computed and $\mathbf{X}$ is used directly in computations for $\mathbf{p}$ and $\mathbf{q}$. In the second algorithm, $\mathbf{X}^T\mathbf{X}$ is computed once and then used in all subsequent computations for $\mathbf{p}$ and $\mathbf{q}$. The performances of these new algorithms are compared to that of De Jong and Ter Braak's algorithm in terms of speed and the new algorithms are shown to be much faster. Each of these new algorithms is shown to have certain advantages when performing cross-validation, or treating missing data. One of the algorithms also provides a fast kernel algorithm for updating PLS models in a recursive manner and for exponentially discounting past data. This is further discussed in the next chapter.

4.8 Nomenclature

The following notation was used in this chapter. Upper case bold variables are matrices and lower case bold variables are column vectors.

$\mathbf{X}$	Predictor variables matrix (N × K)
$\mathbf{Y}$	Response variables matrix (N × M)
$\mathbf{B}_{PLS}$	PLS regression coefficients matrix (K × M)
$\mathbf{W}$	PLS weights matrix for $\mathbf{X}$ (K × A)
$\mathbf{P}$	PLS loadings matrix for $\mathbf{X}$ (K × A)
$\mathbf{Q}$	PLS loadings matrix for $\mathbf{Y}$ (M × A)
$\mathbf{R}$	PLS weights matrix to compute scores $\mathbf{T}$ directly from original $\mathbf{X}$ (K×A)
$\mathbf{T}$	PLS scores matrix of $\mathbf{X}$ (N×A)
$\mathbf{w}_a$	a column vector of $\mathbf{W}$
$\mathbf{p}_a$	a column vector of $\mathbf{P}$
$\mathbf{q}_a$	a column vector of $\mathbf{Q}$
$\mathbf{r}_a$	a column vector of $\mathbf{R}$
$\mathbf{t}_a$	a column vector $\mathbf{T}$
K	Number of X-variables
M	Number of Y-variables
N	Number of objects
A	Number of components in PLS model
a	Integer counter for the latent variable dimension

4.9 Appendix

Modified NIPALS Algorithm

```
function [W,P,Q,R,beta]=nipals_y(X,Y,A)      % A = number of PLS latent vectors to be computed
                                             % beta=PLS regression coefficients.
convcrit=1e-10;                              % convergence criteria for the NIPALS algorithm.
for i=1:A,
  u=Y(:,find(std(Y)==max(std(Y))));          % assigning the y-variables with the largest variance
                                             % to u_initial.
  diff=1;
  while (diff > convcrit),
    u0=u;
    w=X'*u/(u'*u);                           % compute w
    w=w/(sqrt(sum(w.^2)));                   % normalizing w to unity.
    r=w;                                     % compute r so that the score vector, t, can be
    if i>1,                                  % computed directly from the non-deflated X (or the
      for j=1:i-1,                           % original X).
        r=r-(P(:,j)'*w)*R(:,j);
      end
    end
    t=X*r;                                   % compute t
    q=Y'*t/ (t'*t);                          % compute q
    u=Y*q/(q'*q);                            % compute u
    diff=(sqrt(sum((u0).^2)))/(sqrt(sum(u.^2)));
  end
  p=X'*t/(t'*t);                             % compute X-loadings
  W=[W w];                                   % storing loadings and weights
  R=[R r];
  P=[P p];
  Q=[Q q];
  T=[T t];
  U=[U u];

  Y=Y-t*q';                                  % only Y block is deflated.
end
beta=R*Q';                                   % compute the regression coefficients.
end
```

Modified Algorithm #1

```
XY=X'*Y;                                     % Compute X^TY
for i=1:A,                                   % A - number of PLS components to computed
        [C,D]=eig(XY'*XY);                   % Compute eigenvectors of Y^TXX^TY
        q=C(:,find(diag(D)==max(diag(D))));  % find the eigenvector corresponding to the largest eigenvalue
        w=(XY*q);                            % compute X-weights
```

```
        w=w/sqrt(w'*w);                 % normalize w to unity
        r=w;                            % loop to compute r_i
           for j=1:i-1,
              r=r-(P(:,j)'*w)*R(:,j);
        end
        t=X*r;                          % compute score vector
        tt=(t'*t);                      % compute t^T t
        p=(X'*t)/tt;                    % X-loadings
        q=(r'*XY)'/tt;                  % Y-loadings
        XY=XY-(p*q')*tt;                % X^T Y deflation
        W=[W w];                        % store  loadings and weights
        P=[P p];
        Q=[Q q];
        R=[R r];
end
beta=R*Q';                              % compute the regression coefficients
```

Modified Algorithm #2

```
XY=X'*Y;                                % Compute the covariance
XX=X'*X;                                % matrices
for i=1:A,                              % A - number of PLS components to be computed
   [C,D]=eig(XY'*XY);                   % Compute eigenvectors of Y^T XX^T X
   q=C(:,find(diag(D)==max(diag(D))));  % find the eigenvector corresponding to the largest
                                        eigenvalue
   w=(XY*q);                            % compute X-weights
   w=w/sqrt(w'*w);                      % normalize w to unity
    r=w;                                % loop to compute r_i
    for j=1:i-1,
        r=r-(P(:,j)'*w)*R(:,j);
  end
   tt=(r'*XX*r);                        % compute t^T t
   p=(r'*XX)'/tt;                       % X-loadings
   q=(r'*XY)'/tt;                       % Y-loadings
   XY=XY-(p*q')*tt;                     % X^T Y deflation
   W=[W w];                             % storing loadings and weights
   P=[P p];
  Q=[Q q];
  R=[R r];
end
beta=R*Q';                              % compute the regression coefficients
```

5. Recursive Exponentially Weighted PLS and Its Applications

5.1 Introduction

Very often new data are being collected on a regular basis and it is desirable to recursively update the regression model using each new multivariate object as it becomes available. Furthermore, the process may be slowly changing with time and one would like to weight the recent data more heavily and discount past data in an exponentially weighted manner. Such situations arise frequently in the area of adaptive process control and calibration updating.

Recursive least squares (RLS) is the most commonly used method for recursive on-line estimation of model parameters. In most cases, the process variables are highly correlated and the correlation structure among process variables can lead to estimation difficulties with the recursive least squares algorithm. For instance, during adaptive control, the process variables are not only autocorrelated due to feedback but they are also cross-correlated. Lately the partial least squares (PLS) regression method has been used to overcome difficulties encountered with the existing correlation structure in process data. A recursive PLS algorithm along the lines of recursive least squares algorithm can also be employed to estimate on-line models for time-varying processes.

In this chapter, a faster kernel algorithm for exponentially weighted updating of a PLS regression model is developed by combining the improved kernel algorithm #2 from chapter 4 with the recursive updating of the covariance matrices $\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)_t$ and $\left(\mathbf{X}^{\mathrm{T}}\mathbf{Y}\right)_t$. This newly developed recursive PLS algorithm is then applied to adaptive control of a simulated multivariable continuous stirred tank reactor and updating of a multi-output

prediction model for an industrial mineral flotation circuit. Furthermore, the performance of the recursive PLS algorithm is compared to that of the recursive least squares algorithm.

5.2 Recursive Algorithms

As the new data become available, the old data can be exponentially discounted by updating the covariance matrices as shown below:

$$\left(\mathbf{X}^T\mathbf{X}\right)_t = \lambda_t\left(\mathbf{X}^T\mathbf{X}\right)_{t-1} + \mathbf{x}_t^T\mathbf{x}_t \tag{5. 1}$$

$$\left(\mathbf{X}^T\mathbf{Y}\right)_t = \lambda_t\left(\mathbf{X}^T\mathbf{Y}\right)_{t-1} + \mathbf{x}_t^T\mathbf{y}_t \tag{5. 2}$$

Here, $\mathbf{x}_t$ and $\mathbf{y}_t$ are the new (p x 1) and (m x 1) predictor and response vectors observed at time t, and $\left(\mathbf{X}^T\mathbf{X}\right)_t$ and $\left(\mathbf{X}^T\mathbf{Y}\right)_t$ are the updated covariance matrices at time t. At each new sampling period, the previous data in the covariance matrices are being exponentially discounted with a forgetting factor $\lambda_t (0 < \lambda_t \leq 1)$ and the new data are being added. For λ_t=1 no discounting of the past data is done.

5.2.1 Recursive Least Squares

Once the covariance matrices in Equations (5. 1) and (5. 2) have been updated, the new model parameters can be computed using the OLS solution:

$$\mathbf{b}_t = \left(\mathbf{X}^T\mathbf{X}\right)_t^{-1}\left(\mathbf{X}^T\mathbf{Y}\right)_t \tag{5. 3}$$

The above equation can be rearranged to update the model parameter estimates recursively as each data vector becomes available (Astrom and Wittenmark, 1989).

$$\mathbf{b}_t = \mathbf{b}_{t-1} + \left(\mathbf{X}^T\mathbf{X}\right)_t^{-1}\mathbf{x}_t^T\left[\mathbf{y}_t - \mathbf{x}_t\mathbf{b}_{t-1}\right] \tag{5. 4}$$

To avoid inverting $\left(\mathbf{X}^T\mathbf{X}\right)_t$, which can be very computationally intense, at each step, it is convenient to update, $\mathbf{L}_t$, the inverse of $\left(\mathbf{X}^T\mathbf{X}\right)_t$.

$$\mathbf{L}_t = \left(\mathbf{X}^T\mathbf{X}\right)_t^{-1} = \left[\lambda_t\left(\mathbf{X}^T\mathbf{X}\right)_{t-1} + \mathbf{x}_t^T\mathbf{x}_t\right]^{-1} \tag{5. 5}$$

By applying the matrix inversion lemma, the above equation can be updated at each sampling interval as

$$\mathbf{L}_t = \frac{\mathbf{L}_{t-1} - \dfrac{\mathbf{L}_{t-1}\mathbf{x}_t^T\mathbf{x}_t\mathbf{L}_{t-1}}{\lambda_t + \mathbf{x}_t\mathbf{L}_{t-1}\mathbf{x}_t^T}}{\lambda_t} \tag{5. 6}$$

where $\mathbf{L}_{t-1}$ is the inverse of $\left(\mathbf{X}^T\mathbf{X}\right)_{t-1}$.

The complete recursive least squares algorithm is summarised below:

Estimates: $$\mathbf{b}_t = \mathbf{b}_{t-1} + \mathbf{K}_t\mathbf{e}_t \tag{5. 7}$$

Error: $$\mathbf{e}_t = \mathbf{y}_t - \hat{\mathbf{y}}_t = \mathbf{y}_t - \mathbf{x}_t\mathbf{b}_{t-1} \tag{5. 8}$$

Gain: $$\mathbf{K}_t = \mathbf{L}_t\mathbf{x}_t^T = \frac{\mathbf{L}_{t-1}\mathbf{x}_t^T}{\lambda_t + \mathbf{x}_t\mathbf{L}_{t-1}\mathbf{x}_t^T} \tag{5. 9}$$

Updating: $$\mathbf{L}_t = \frac{\left(\mathbf{I} - \mathbf{K}_t\mathbf{x}_t\right)\mathbf{L}_{t-1}}{\lambda_t} \tag{5. 10}$$

The main problem with updating $\mathbf{L}_t$ is that it can become non-symmetric and indefinite due to round-off errors and this can lead to numerical instability. Therefore, it may be useful to represent $\mathbf{L}_t$ in a factorized form to keep it better conditioned,

$$\mathbf{L}_t = \mathbf{M}_t\mathbf{M}_t^T \tag{5. 11}$$

where $\mathbf{M}_t$ is a lower triangular matrix. In this work, the algorithm of Morf and Kailath (1975) based on the Householder transformation is used to update $\mathbf{M}_t$ rather than $\mathbf{L}_t$ and then $\mathbf{L}_t$ and $\mathbf{K}_t$ are computed from $\mathbf{M}_t$. Further details pertaining to the algorithm can be found in Ljung (1987).

5.2.2 Recursive PLS

A fast kernel algorithm for recursive exponentially weighted updating of a PLS regression model can be obtained by combining the improved kernel algorithm #2 from chapter 4 with the covariance updating Equations (5. 1) and (5. 2). Since the covariance

matrices in Equations (5. 1) and (5. 2) can be updated with very little computational effort, the modified kernel algorithm #2 will be extremely fast in these applications.

5.2.2.1 Literature Review

Wold (1994) recently published exponentially weighted algorithms for PCA and PLS. His algorithms are not recursive in that the entire expanded data set (**X**, **Y**) is used to build the model every time a new object becomes available, rather than just recursively updating the previous covariance matrices with new objects as in Equations (5. 1) and (5. 2). Wold's use of the NIPALS algorithm at each stage also makes his approach slower than the fast kernel algorithms proposed here. If one wishes to obtain exponentially weighted updated estimates of individual scores and loading vectors, then some form of stabilisation method similar to those proposed by Wold (1994) would have to be incorporated into our algorithm as well in order to prevent unwarranted rotations in the latent vector space. However, such rotations are of no concern if one is only interested in an updated regression model.

Helland et al. (1991) presented a recursive but not exponentially weighted algorithm for PLS. In their algorithm, the old data in **X** and **Y** is captured by their loading matrices (i.e., $\mathbf{P}^T$ and $\mathbf{Q}^T$; the score vectors **t** and **u** are scaled to unit variance) and new data are added to these loading matrices. Although, this algorithm keeps the size of the matrices which are used for PLS model calculations constant, it is still very slow compared to the new recursive exponentially weighted PLS kernel algorithm proposed here. Furthermore, during the data updating step, not all latent vectors are retained in $\mathbf{P}^T$. This can lead to loss of information and deterioration of the parameter estimates.

5.2.2.2 Mean-Centring and Scaling of the Variables

In ordinary PLS, the X and Y-variables are mean-centred and scaled prior to regression model building. Each variable is mean-centred using the mean computed from the entire column. In a time-varying process, the mean levels of the variables may be

changing with time. Thus, instead of mean-centring the data, one can augment $\mathbf{x}_t$ with a unity element to account for the constant term or the intercept. For scaling of the variables, one can either use prior process knowledge to select scaling factors for all variables and then scale the variables with these constant scaling factors. For a time-varying process, the standard deviation of each process variable can also be updated at each sampling interval. No extra effort or memory is required to store the standard deviations as the standard deviations can be computed using the already available information from the updated $\left(\mathbf{X}^T\mathbf{X}\right)_t$ at each sampling interval. Suppose, the $\mathbf{x}_t$ vector consists of the following variables:

$$\mathbf{x}_t = \begin{bmatrix} x_{1,t} & x_{2,t} & \cdots & x_{p,t} & 1 \end{bmatrix} \tag{5. 12}$$

Therefore, the elements of the covariance matrix, $\left(\mathbf{X}^T\mathbf{X}\right)_t$, will be as follows:

$$\left(\mathbf{X}^T\mathbf{X}\right)_t = \lambda_t\left(\mathbf{X}^T\mathbf{X}\right)_{t-1} + \mathbf{x}_t^T\mathbf{x}_t \tag{5. 13}$$

$$= \begin{bmatrix} \lambda_t\left(\Sigma x_1^2\right)_{t-1} + x_{1,t}^2 & \lambda_t\left(\Sigma x_1 x_2\right)_{t-1} + x_{1,t}x_{2,t} & \cdots & \lambda_t\left(\Sigma x_1\right)_{t-1} + x_{1,t} \\ \lambda_t\left(\Sigma x_2 x_1\right)_{t-1} + x_{2,t}x_{1,t} & \lambda_t\left(\Sigma x_2^2\right)_{t-1} + x_{2,t}^2 & \cdots & \lambda_t\left(\Sigma x_2\right)_{t-1} + x_{2,t} \\ \vdots & \vdots & \ddots & \\ \lambda_t\left(\Sigma x_1\right)_{t-1} + x_{1,t} & \lambda_t\left(\Sigma x_2\right)_{t-1} + x_{2,t} & \cdots & \lambda_t\left(\Sigma 1\right)_{t-1} + 1 \end{bmatrix}$$

where the (i,j) element of $\left(\mathbf{X}^T\mathbf{X}\right)_t$ can be expanded as

$$\begin{aligned} \left(\Sigma x_i x_j\right)_t &= \lambda_t\left(\Sigma x_i x_j\right)_{t-1} + x_{i,t}x_{j,t} = \lambda_t\left[\lambda_{t-1}\left(\Sigma x_i x_j\right)_{t-2} + x_{i,t-1}x_{j,t-1}\right] + x_{i,t}x_{j,t} \\ &= \lambda_{t-1}\left[\lambda_{t-2}\left(\Sigma x_i x_j\right)_{t-3} + x_{i,t-2}x_{j,t-2}\right] + \lambda_t x_{i,t-1}x_{j,t-1} + x_{i,t}x_{j,t} \\ &= x_{i,t}x_{j,t} + \lambda_t x_{i,t-1}x_{j,t-1} + \lambda_t\lambda_{t-1}x_{i,t-2}x_{j,t-2} + \lambda_t\lambda_{t-1}\lambda_{t-2}x_{i,t-3}x_{j,t-3} + \cdots \\ &= x_{i,t}x_{j,t} + \sum_{k=1}^{\infty} x_{i,t-k}x_{j,t-k} \prod_{l=0}^{k-1} \lambda_{t-l} \end{aligned} \tag{5. 14}$$

For a constant forgetting factor, $\lambda_t = \lambda$, the effective memory length of the data can be computed as (Astrom and Wittenmark, 1989),

$$N = \frac{1}{1-\lambda} \tag{5. 15}$$

For a variable forgetting factor, this would not give a good estimate of the effective memory length of the data. However, the information regarding the effective memory length can be obtained from $\left(\mathbf{X}^T\mathbf{X}\right)_t$. The last diagonal element of $\left(\mathbf{X}^T\mathbf{X}\right)_t$ represents the effective memory length of the data.

$$N_t = \lambda_t(\Sigma 1)_{t-1} + 1 = \lambda_t N_{t-1} + 1 \tag{5. 16}$$

The mean of each variable can be computed using the last column or the row, $(\Sigma x_i)_t = \lambda_t(\Sigma x_i)_{t-1} + x_{i,t}$, and the current effective memory length of the data.

$$\bar{x}_{i,t} = \frac{(\Sigma x_i)_t}{N_t} \tag{5. 17}$$

The updated variance for each variable can then be computed using the following formula

$$\mathrm{var}(x_i)_t = \frac{\left(\sum(x_i - \bar{x}_i)^2\right)_t}{N_t - 1} = \frac{\left(\Sigma x_i^2\right)_t - N_t \bar{x}_{i,t}^2}{N_t - 1} \tag{5. 18}$$

The sum of squares of each variable, $\left(\Sigma x_i^2\right)_t$, is available from the covariance matrix and the mean can be computed using Equation (5. 17).

The standard deviations for the output variables can also be updated in a same manner. However, the quantities such as $\left(\Sigma y_i^2\right)_t$ and $(\Sigma y_i)_t$ need to be stored and updated at each sampling interval.

5.2.3 Variable Forgetting Factor

The discounting of the old data is necessary to account for the time-varying nature of a process. As the new data become available, the old data are continuously discounted. The discounting of the old data with a constant forgetting factor works well if there is persistent excitation in the process. However, a problem arises with a constant forgetting factor if there is no information in the new data and the old data are still being discounted. Under these circumstances, the covariance matrix will lose the essential process information and become extremely ill-conditioned, and the precision of the resulting

parameter estimates will be poor. To avoid this problem, one needs a variable forgetting factor which discounts the old data only when there is information in the new data and retains the old data when there is no information in the new data.

In this work, a variable forgetting factor is computed at each sampling interval using the algorithm of Fortescue et al. (1981). In their algorithm, the amount of discounting at each instance depends on the new information in the latest data vector and is computed such that the estimation is always based on same amount of information. The variable forgetting factor is calculated as

$$\lambda_t = 1 - \frac{\left[1 - \mathbf{x}_t \left(\mathbf{X}^T\mathbf{X}\right)_t^{-1} \mathbf{x}_t^T\right] e_t^2}{\Sigma_o} = 1 - \frac{\left[1 - \mathbf{x}_t \mathbf{K}_t\right] e_t^2}{\Sigma_o} \qquad (5.19)$$

$$\lambda_t = \lambda_{min} \text{ if } \lambda_t < \lambda_{min}$$

where

$$\Sigma_o = \sigma_o^2 N_o \qquad (5.20)$$

The variable σ_o^2 is the expected measurement noise variance of the output variable. This can be based on the process knowledge. N_o is the asymptotic memory length and will control the speed of the adaptation.

The variable forgetting factor equation can be modified for use in the recursive PLS algorithm. The error term, e_t, can be computed using the PLS regression estimates. For a larger number of X-variables, inverting $\left(\mathbf{X}^T\mathbf{X}\right)_t$ at each sampling interval can be very computationally intense. However, the $\left(\mathbf{X}^T\mathbf{X}\right)_t$ can be replaced by its estimate from the PLS model.

$$\left(\mathbf{X}^T\mathbf{X}\right) \approx \left(\hat{\mathbf{X}}^T\hat{\mathbf{X}}\right) \qquad (5.21)$$

where $\hat{\mathbf{X}} = \mathbf{T}\mathbf{P}^T$. Now, the inverse of $\hat{\mathbf{X}}^T\hat{\mathbf{X}}$ can be easily computed using the information available from the PLS model.

$$\left(\hat{\mathbf{X}}^T\hat{\mathbf{X}}\right)^{-1} = \left(\mathbf{P}\mathbf{T}^T\mathbf{T}\mathbf{P}^T\right)^{-1} = \left(\mathbf{P}^T\right)^{-1}\left(\mathbf{T}^T\mathbf{T}\right)^{-1}\mathbf{P}^{-1} \qquad (5.22)$$

The $\mathbf{T}^T\mathbf{T}$ is a diagonal matrix and its inverse can be easily computed by inverting the diagonal elements. Since $\mathbf{P}$ is usually not a square matrix, its inverse can be computed using the generalized inverse or the pseudoinverse.

$$\mathbf{P}^{-1} = \left(\mathbf{P}^T\mathbf{P}\right)^{-1}\mathbf{P}^T \tag{5. 23}$$

Therefore,

$$\left(\hat{\mathbf{X}}^T\hat{\mathbf{X}}\right)^{-1} = \mathbf{P}\left(\mathbf{P}^T\mathbf{P}\right)^{-1}\left(\mathbf{T}^T\mathbf{T}\right)^{-1}\left(\mathbf{P}^T\mathbf{P}\right)^{-1}\mathbf{P}^T \tag{5. 24}$$

The $\mathbf{P}^T\mathbf{P}$ is a tridiagonal matrix and usually of much smaller dimension than $\mathbf{X}^T\mathbf{X}$. Thus, inverting $\mathbf{P}^T\mathbf{P}$ would require less computational effort.

In certain cases, the number of latent vectors computed in a PLS model might not be sufficient to model most of the variance in $\mathbf{X}$ and $\hat{\mathbf{X}}^T\hat{\mathbf{X}}$ might not be a good approximation to $\mathbf{X}^T\mathbf{X}$. In such situations, one can always compute few additional latent vectors than that required for the PLS regression model.

5.3 Applications of Exponentially Weighted Recursive PLS Algorithm

The exponentially weighted recursive PLS algorithm is applied to two process examples. The first application is adaptive control of a simulated multivariable nonlinear continuous stirred tank reactor. The second application is the updating of a prediction model for 10 response variables for an industrial mineral flotation circuit. In both cases, the performance of the recursive PLS algorithm is compared to that of the recursive least squares algorithm.

5.3.1 Adaptive Control of a Multivariable Nonlinear CSTR

The physical system studied here is a continuous stirred tank reactor (CSTR) with an irreversible exothermic first order reaction

$$\mathbf{A} \rightarrow \mathbf{B}$$

A pure stream of species A enters the constant volume reactor and a well mixed stream of species A and B exits the reactor. A dynamic simulation for the system is obtained by writing ordinary differential equations for the material and energy balances.

Reactant Mass Balance

$$V\frac{dC_A}{dt} = F_I C_{A_I} - F_O C_A - Vk_o \exp\left(\frac{-E_A}{RT}\right)C_A \tag{5.25}$$

Product Mass Balance

$$V\frac{dC_B}{dt} = -F_O C_B + Vk_o \exp\left(\frac{-E_A}{RT}\right)C_A \tag{5.26}$$

Reactor Energy Balance

$$V\rho_s C_{P_s}\frac{dT}{dt} = \rho_s C_{P_s} F_I T_I - \rho_s C_{P_s} F_O T + V(-\Delta H_R)k_o \exp\left(\frac{-E_A}{RT}\right)C_A + UA_H(T_{CJ} - T) \tag{5.27}$$

Cooling Jacket Energy Balance

$$V_{CJ}\rho_W C_{P_w}\frac{dT_{CJ}}{dt} = \dot{m}C_{P_w}(T_I - T_{CJ}) + UA_H(T - T_{CJ}) \tag{5.28}$$

Model parameters are defined in the nomenclature section and the specific parameters used for the CSTR are given in Table 5.1. The process is sampled every 20 seconds. The objective of the controller is to keep the reactor conversion (y_1) and the reactor temperature (y_2) at their desired settings amid coming disturbances by manipulating the inlet feed flow rate of species A ($u_1=F_I$) and the cooling water flow rate ($u_2=\dot{m}$). Both output variables have dead-times of 2 sampling periods with respect to the process inputs due to transportation lag. The conversion variable has an additional dead-time of 3 sampling periods due to analytical analysis time. The random measurement noise with variances of 4×10^{-6} and 0.002 was added to both conversion and temperature, respectively.

The process is controlled using a dynamic matrix controller (DMC) (Cutler and Ramaker, 1979). At each sampling interval, the DMC controllers are designed using the step weights obtained from three different identified models: (i) a state space process

model by linearising the true nonlinear CSTR mechanistic model; (ii) ARX models identified using the recursive PLS algorithm; and (iii) ARX models identified using a recursive least squares algorithm. The performances of these various controllers are compared on the basis of integral of squared errors (ISE) for both output variables and integral of squared change in the manipulated variables (ISVu).

Parameter	Value
A_H	5.0 m^2
$C_{A,F}$	866.0 kg/m^3
$C_{p,s}$	1.791 J/kg · K
$C_{p,w}$	4.181 J/kg · K
E_A	60000 J/mol · K
F_I	0.02 m^3/s
F_o	0.02 m^3/s
ΔH_R	-140.0 J/kg
k_o	4×10^8 s^{-1}
R	8.314 J/mol · K
T_I	290.0 K
T_F	293.0 K
U	30.0 W/m^2 · K
V	1.0 m^3
V_{CJ}	0.20 m^3
ρ_s	866.0 kg/m^3
ρ_w	998.0 kg/m^3

Table 5.1: Specific parameters used for the CSTR

5.3.1.1 Locally Linearised Mechanistic Model

A local state space model of the CSTR is obtained by linearising the nonlinear Equations (5. 25) to (5. 28) at each sampling interval.

$$\dot{\mathbf{x}} = \mathbf{A}_t\mathbf{x} + \mathbf{B}_t\mathbf{u} \qquad (5.29)$$
$$\mathbf{y} = \mathbf{C}_t\mathbf{x}$$

All the parameters in the nonlinear mechanistic models are assumed to be known. The step weights needed to design DMC are then obtained from this state space model. This controller is used as a benchmark to compare the performances of the controllers designed using the models identified from the recursive algorithms.

5.3.1.2 Identification for the Recursive Algorithms

The following ARX model was identified at each sampling interval using the recursive algorithms (Clarke et al., 1987):

$$A\left(z^{-1}\right)y_t = B_1\left(z^{-1}\right)u_{1,t-t_{d_1}} + B_2\left(z^{-1}\right)u_{2,t-t_{d_2}} + \frac{1}{\nabla}a_t \qquad (5.30)$$

The variable y_t is the output variable at time t and $u_{1,t}$ and $u_{2,t}$ are the first and second process input variables at time t, respectively. The variable z^{-1} is the backward shift operator (i.e., $z^{-1}y_t = y_{t-1}$) and ∇ is the difference operator ($\nabla = 1 - z^{-1}$). The variable t_{d_i} is the existing dead-time between the i^{th} input, $u_{i,t}$, and the output, y_t. $A(z^{-1})$ and $B_i(z^{-1})$ are polynomials in z^{-1}.

$$A\left(z^{-1}\right) = \left(1 + a_1z^{-1} + a_2z^{-2} + \cdots + a_{na}z^{-na}\right) \qquad (5.31)$$

$$B_i\left(z^{-1}\right) = \left(b_{i,1}z^{-1} + b_{i,2}z^{-2} + \cdots + b_{i,nb_i}z^{-nb_i}\right) \qquad (5.32)$$

The disturbance affecting the process is non-stationary and thus the data are differenced once to remove the non-stationary effect. The identified model now becomes

$$A\left(z^{-1}\right)\nabla y_t = B_1\left(z^{-1}\right)\nabla u_{1,t-t_{d_1}} + B_2\left(z^{-1}\right)\nabla u_{2,t-t_{d_2}} + a_t \qquad (5.33)$$

The ARX models for the conversion and temperature variables are identified separately. Six lags of the output variable (na=6) and six lags of each input variable (nb_i=6) are

sufficient to model the process adequately over the entire range of the disturbance. Therefore, for regression, the $\mathbf{x}_t$ vector consists of the following variables:

$$\mathbf{x}_t = \left[\nabla y_{t-1}, \cdots, \nabla y_{t-6}, \nabla u_{1,t-t_{d_1}-1}, \cdots, \nabla u_{1,t-t_{d_1}-6}, \nabla u_{2,t-t_{d_2}-1}, \cdots, \nabla u_{1,t-t_{d_2}-6}\right] \qquad (5.34)$$

There are total of 18 parameters to be estimated in Equation (5. 33). During the on-line identification, the dead-times existing between the process inputs and output are assumed to be known and the data are shifted to remove the dead-times. Differencing the data once removes the mean and thus augmenting $\mathbf{x}_t$ with a unity element as in Equation (5. 12) is not necessary. For the recursive PLS algorithm, the elements of $\left(\mathbf{X}^T\mathbf{X}\right)_t$ and $\left(\mathbf{X}^T\mathbf{Y}\right)_t$ were scaled using the updated standard deviations from Equation (5. 18) at each sampling interval. The PLS regression model was computed with four latent vectors. The old data were discounted using a variable forgetting factor computed using the algorithm of Fortescue et al. (1981). The measurement variances, σ_o^2, selected for conversion and temperature were 8×10^{-6} and 0.001, respectively. They are different from the variances of the measurement noise added to the output variables because the data have been differenced. The asymptotic memory length, N_o, for the data was taken to be 200. The forgetting factor for the recursive PLS algorithm was computed by inverting $\left(\mathbf{X}^T\mathbf{X}\right)_t$ in Equation (5. 19). A lower bound of 0.95 was placed on the forgetting factor.

5.3.1.3 Dynamic Matrix Controller

In DMC, the control moves are computed using the following equation.

$$\nabla \mathbf{u} = \left(\mathbf{A}^T\mathbf{Q}_1^T\mathbf{Q}_1\mathbf{A} + \mathbf{Q}_2^T\mathbf{Q}_2\right)^{-1}\mathbf{A}^T\mathbf{Q}_1^T\mathbf{Q}_1\mathbf{E} \qquad (5.35)$$

where **A** is the dynamic matrix containing the estimated process step weights. The matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are the weighting matrices for the process output and input variables, respectively. **E** is a vector of predicted deviations from the set-point assuming no new control actions. The control moves, $\nabla \mathbf{u}$, are penalised by selecting an appropriate penalty matrix, $\mathbf{Q}_2$. A fixed input penalty matrix poses a problem during adaptive control. The

elements of the dynamic matrix are changing and thus the penalty matrix also needs to be updated to get same amount of constraining of the input variables at each sampling interval. The need for time-varying penalty matrix was solved by applying principal component analysis to compute the control moves (Maurath et al., 1988). In this work, the control moves are computed by inverting the three largest singular values of $\mathbf{Q}_1\mathbf{A}$ at each sampling interval. These singular values account for roughly about 80% of the sum of squares of $\mathbf{Q}_1\mathbf{A}$.

The same DMC tuning parameters are used for the controller design for each method. The output prediction and control horizons are selected to be 25 and 5, respectively. The weighting matrix for the output variables is

$$\begin{matrix} & y_1 & y_2 \end{matrix} \qquad (5.36)$$

$$Q_1 = \begin{bmatrix} 10 & \\ & 1 \end{bmatrix}$$

5.3.1.4 Adaptive Control Simulation

The control simulation was carried out for a random walk variation in k_o, the Arrhenius rate constant. The range of variation of k_o used in the simulation run changes both the process gains and dynamics (i.e., the residence time of the reactor) by a factor of 5. Therefore, a fixed controller was not able to reject the disturbance very effectively.

For the recursive algorithms, the process is initially controlled by a fixed DMC during the first two hundred sampling periods and small pseudo random binary sequence (PRBS) signals are added to both manipulated variables. This generates data to identify initial models for both recursive least squares and PLS algorithms and thereafter the models are updated at each sampling interval.

The performance of various adaptive controllers is evaluated by computing the integral of squared errors (ISE) for both output variables and the integral of squared change in the manipulated variables (ISVu). The results are given in Table 5.2. The plots of the closed loop response for the three adaptive controllers are shown in Figure 5.1 and

5.2. Figure 5.1 shows the response of the output variables whereas Figure 5.2 shows the response of the manipulated variables.

Identification Method	ISE Conversion (y_1)	ISE Temperature (y_2)	IS∇u_1 Feed Flow Rate ($\times 10^{-4}$)	IS∇u_2 Cooling Water Flow rate ($\times 10^{-4}$)
Locally Linearised Mechanistic Model	0.3662	68.85	0.2367	0.2103
Recursive Least Squares	1.1950	116.37	0.3284	0.5586
Recursive PLS	0.3461	68.56	0.2956	0.5408

Table 5.2: CSTR adaptive control results

As expected the locally linearised model obtained from the true mechanistic process model provides the best control performance. The recursive least squares algorithm gives the worst performance. This is due to two reasons: (i) the input and the output variables are autocorrelated and; (ii) both input variables are also cross-correlated because they move together under feedback. The autocorrelation and cross-correlation among various variables makes the $\left(\mathbf{X}^T\mathbf{X}\right)_t$ matrix very ill-conditioned and the recursive least squares algorithm 'blew' up quite a few times during the simulation. Therefore, the DMC designed with the step weights from the recursive least squares algorithm provided poorer performance during these instances (seen as large spikes in the output variables). During these 'blow' ups, the manipulated variables move stochastically. This breaks up the correlation structure due to feedback in the data and the recursive least square algorithm uses this data to retrain itself. The recursive PLS algorithm provides good

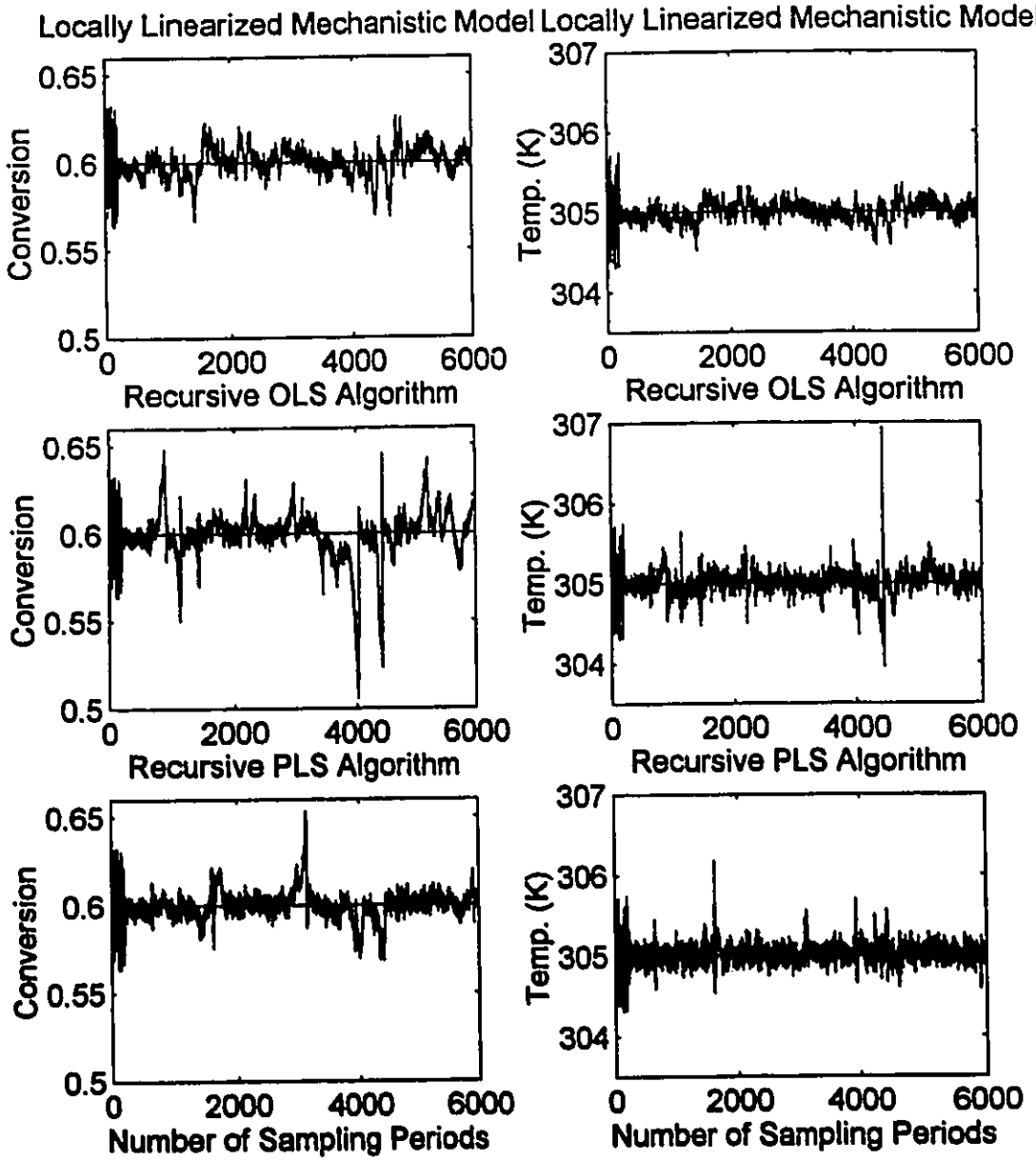

Figure 5.1: The closed loop response of the output variables under various adaptive controllers.

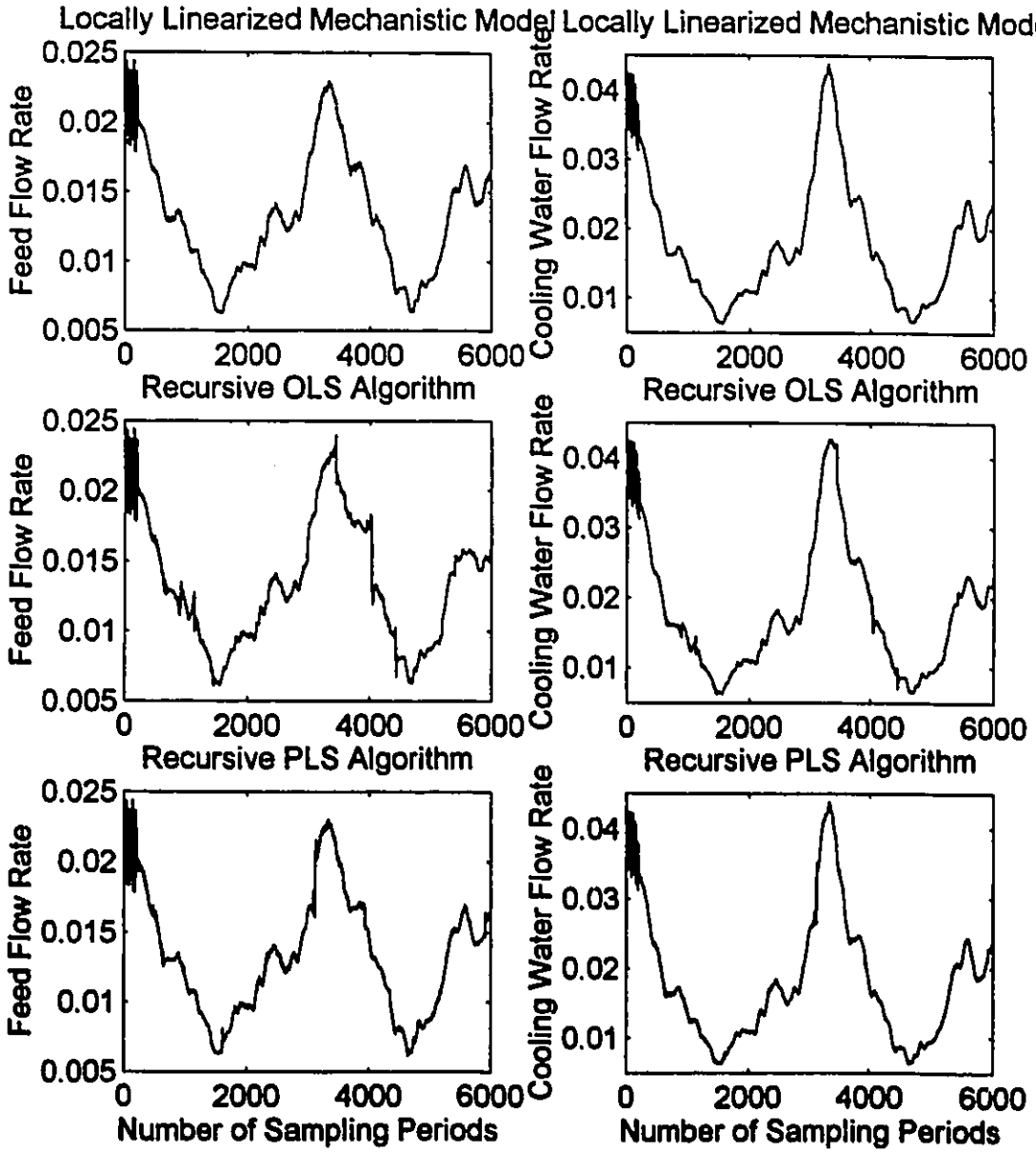

Figure 5.2: The closed loop response of the manipulated variables (Feed flow rate (m^3/s) and cooling water flow rate (kg/s)) under various adaptive controllers.

control; however, it also experienced a few 'blow' ups. This is probably due to the fact that there might be insufficient process information in the covariance matrices. In such cases, the process needs to be excited by adding external dither signals to the manipulated variables. The ISE quantities computed for the output variables are slightly better than those of the locally linearised mechanistic model. However, this comes at the cost of larger control moves as is evident by larger ISVu quantities compared to those of the linearised mechanistic model. The reason for the larger control action is that PLS sometimes underestimates the process gains. This leads to more aggressive controller. The process gain underestimation is probably due to the fixed number of latent vectors used to compute the PLS regression model during the entire simulation. The number of latent vectors used for the regression model could be updated as well. This can be done off-line by analysing the available process data.

5.3.2 Updating of a Prediction Model for an Industrial Mineral Flotation Circuit

The second process example selected to illustrate the potential of a multi-output recursive PLS algorithm is a mining process located in New Brunswick, Canada. The mine processes a very fine grained complex sulphide ore containing mainly marmatite, galena and chalcopyrite and produces lead, copper, zinc and lead-zinc concentrates. Here only the rougher-scavenger part of the flotation circuit for the copper and lead recovery is considered. A schematic of the flotation circuit is given in Figure 6.1. Further details pertaining to the process and multivariate analysis of the process data can be found in Hodouin et al. (1993).

The process input and output variables are listed in Tables 5.3 and 5.4, respectively. There are 13 process input variables and 10 output variables. The objective is develop a recursive multivariate regression model to predict the future process output variables. All ten output variables are modelled together using the two block PLS (PLS2). Historical data for a period of 348 hours were used for recursive model building. Data

VARIABLE NO.	X-VARIABLE
1	Fresh ore feed rate
2	Slurry percentage of solids in the GC product or the FC feed
3	% of particle finer than 37 μm in the GC product
4	Pb content in FC feed
5	Cu content in FC feed
6	Pb flow rate in FC feed
7	Cu flow rate in FC feed
8	Soda ash flow rate to GC
9	pH in FC feed
10	air flow rate to RS
11	Xanthate flow rate to RS
12	Xanthate per ton of Pb and Cu to FC
13	Promoter Flow rate/Xanthate Flow rate

Table 5.3: Rougher-Scavenger Input Variables.

Variable No.	Y-Variable
1	RS tail flow rate
2	RS tail Pb content
3	RS tail Pb flow rate
4	Rougher concentrate Pb content
5	Rougher concentrate Cu content
6	RS concentrate flow rate
7	RS concentrate Pb content
8	RS concentrate Cu content
9	RS Pb recovery
10	RS Pb Floatability

Table 5.4: Rougher-Scavenger Output Variables.

represents hourly averages. The data for four of the output variables are shown as solid lines in Figure 5.3.

Initially, an ordinary PLS model was developed using the entire data set. Five latent vectors, as determined by the cross-validation method, were found to explain about 56% of the overall sum of square of the output variables. Therefore, for the recursive PLS algorithm, the regression model was computed at each sampling interval with five latent vectors. The number of latent vectors to be computed during the recursive updating can be determined by the cross validation technique by analysing the available data off-line. Furthermore, this can be repeated regularly off-line to determine if the number of PLS latent vectors computed to model the time-varying data also need to be updated.

In this example, the process data were scaled using the standard deviations computed using the entire data set. The scaling factors can be updated by computing the standard deviation of each variable from the available data occasionally off-line or they can be updated at each sampling interval as discussed in section 5.2.2.2. The data were not mean-centred as $\mathbf{x}_t$ was augmented with a unity element to account for the constant term.

The regression models were computed using the recursive PLS algorithms with a fixed forgetting factor of 1 (λ_t=1) and a variable forgetting factor and the recursive least squares algorithm with a variable forgetting factor. For λ_t=1, no discounting of the old data is done and the entire data up to time t are used to develop the regression model. The variable forgetting factors were computed using the algorithm of Fortescue et al. (1981). This algorithm is only applicable to univariate output. It was extended to multivariate outputs by taking the weighted average of the prediction errors for all output variables and then using this weighted average error in the algorithm. The measurement noise variances, σ_o^2, of all the output variables were taken to be same for both recursive PLS and least squares algorithms. The asymptotic memory lengths were 20 and 48 for the recursive PLS and least squares algorithms, respectively. They are optimal values for each method to provide the best one step ahead predictions. The forgetting factor for the recursive PLS algorithm was computed using Equation (5. 24) with 5 latent vectors. A

lower bound of 0.85 was placed on the forgetting factor for both recursive PLS and recursive least squares algorithms.

The initial models were computed with 15 data points. Then, a new data vector, $\mathbf{x}_t$, was taken and the covariance matrices were updated as in Equations (5. 1) and (5. 2) and a new PLS regression model was recomputed. The recursive least squares estimates were computed by updating $\mathbf{M}_t$ rather than $\mathbf{L}_t$ as discussed in section 5.2.1. The performance of the various recursive algorithms was evaluated based on one step ahead predictions. The percent sum of squares (% SS) explained of each variable was computed using the following equation:

$$\% \text{ SS Explained } = \left(1 - \frac{\sum_{t=16}^{348} \left(y_{i,t} - \hat{y}_{i,t} \right)^2}{\sum_{t=16}^{348} y_{i,t}^2} \right) \times 100 \qquad (5.37)$$

The variable $y_{i,t}$ is the measured i^{th} output variable at time t and $\hat{y}_{i,t}$ is the prediction of i^{th} output at time t (i.e., $\hat{\mathbf{y}}_t = \mathbf{x}_t \mathbf{b}_{t-1}$). The results for the prediction ability of each recursive algorithm are given in Table 5.5. The recursive PLS with a variable forgetting factor provides the highest % SS explained for all output variables. The recursive PLS with a fixed $\lambda_t = 1$ gives the worst results. This means that the process is truly time-varying. The results for the recursive least squares algorithm are computed with and without the residuals for 170^{th} observation. The process variables are seen to change abruptly at the 170^{th} observation (see Figure 5.3) and recursive least squares provides very poor predictions at this instance. This leads to either very low or negative % SS explained quantities for the recursive least squares. The adjusted % SS explained (with the errors for the 170^{th} observation set to zero) by the recursive least squares algorithm is better than the recursive PLS with a constant forgetting factor of 1 but as not as good as the recursive PLS with a variable forgetting factor. This is due to the fact that the process input variables are highly correlated and, furthermore, a shorter data window is used to construct $\left(\mathbf{X}^T\mathbf{X}\right)_t$. This makes the $\left(\mathbf{X}^T\mathbf{X}\right)_t$ quite ill-conditioned.

Variable No.	% Sum of Squared Explained			
	Recursive PLS with Adaptive λ_t	Recursive PLS with a Fixed $\lambda_t=1$	Recursive OLS with Adaptive λ_t	Recursive OLS with Adaptive λ_t (Adjusted)[1]
Y1	73.3	51.6	57.8	74.2
Y2	78.8	49.8	74.7	76.3
Y3	80.7	48.0	78.2	78.3
Y4	34.8	24.0	-72.0	14.8
Y5	38.1	23.3	-901.7	25.9
Y6	57.3	16.7	11.2	45.6
Y7	21.7	4.9	-47.1	9.1
Y8	90.3	77.3	89.9	90.0
Y9	84.1	57.7	81.5	81.8
Y10	72.2	34.6	48.3	61.9
average	63.1	38.8	-	55.8

Table 5.5: % sum of squares explained of the output variables by various recursive algorithms.

The predictions of the recursive algorithms are shown in Figure 5.3. The predictions are shown only for four of the output variables. The data have been scaled. The solid line represents the true process data. The dashed, dashdot and the dotted lines represent the predictions for recursive PLS with a variable forgetting factor, recursive PLS with a fixed $\lambda_t=1$ and the recursive least squares algorithm with a variable forgetting factor, respectively.

As is evident from the plots, the recursive PLS algorithm with a variable forgetting factor provides the best predictions. The recursive PLS algorithm with a fixed $\lambda_t=1$ is

[1] Sum of squares explained for the adjusted recursive OLS algorithm is computed with the residuals for the 170th object set to 0.

very slow to adapt. For some shorter period lengths, it does not adapt at all and provides the worst results. This is due to the fact that old data is not being discounted and all the data (new and old) are weighted equally. The recursive least squares algorithm provides good predictions most of the times however it is seen to 'blow up', as evident by the large spikes on the plots, at certain instances. This is due to fact that the $\left(\mathbf{X}^T\mathbf{X}\right)_t$ becomes extremely ill-conditioned and susceptible to numerical and computational difficulties.

5.4 Conclusions

In this chapter, an algorithm for updating PLS models in a recursive manner with exponential discounting of past data is presented. This algorithm is developed by combining the improved and faster kernel algorithm #2 from chapter 4 with the recursive updating of the covariance matrices $\left(\mathbf{X}^T\mathbf{X}\right)_t$ and $\left(\mathbf{X}^T\mathbf{Y}\right)_t$. A technique for determining the updated variances of process variables for autoscaling purposes is also presented. The potential of the this newly developed recursive PLS algorithm is illustrated with two examples: (i) an adaptive control application to a simulated multivariable continuous stirred tank reactor; and (ii) updating of a multi-output prediction model for an industrial mineral flotation circuit. The performance of the recursive PLS algorithm was much better than that of the recursive least squares algorithm.

One advantage of the recursive PLS algorithm is that it does not suffer from the problems associated with correlated process variables and a shorter data window as illustrated in the industrial mineral flotation circuit example. Furthermore, during adaptive control, it provided satisfactory control when the recursive least squares algorithm experienced difficulties (i.e., blew up) due to ill-conditioned $\left(\mathbf{X}^T\mathbf{X}\right)_t$.

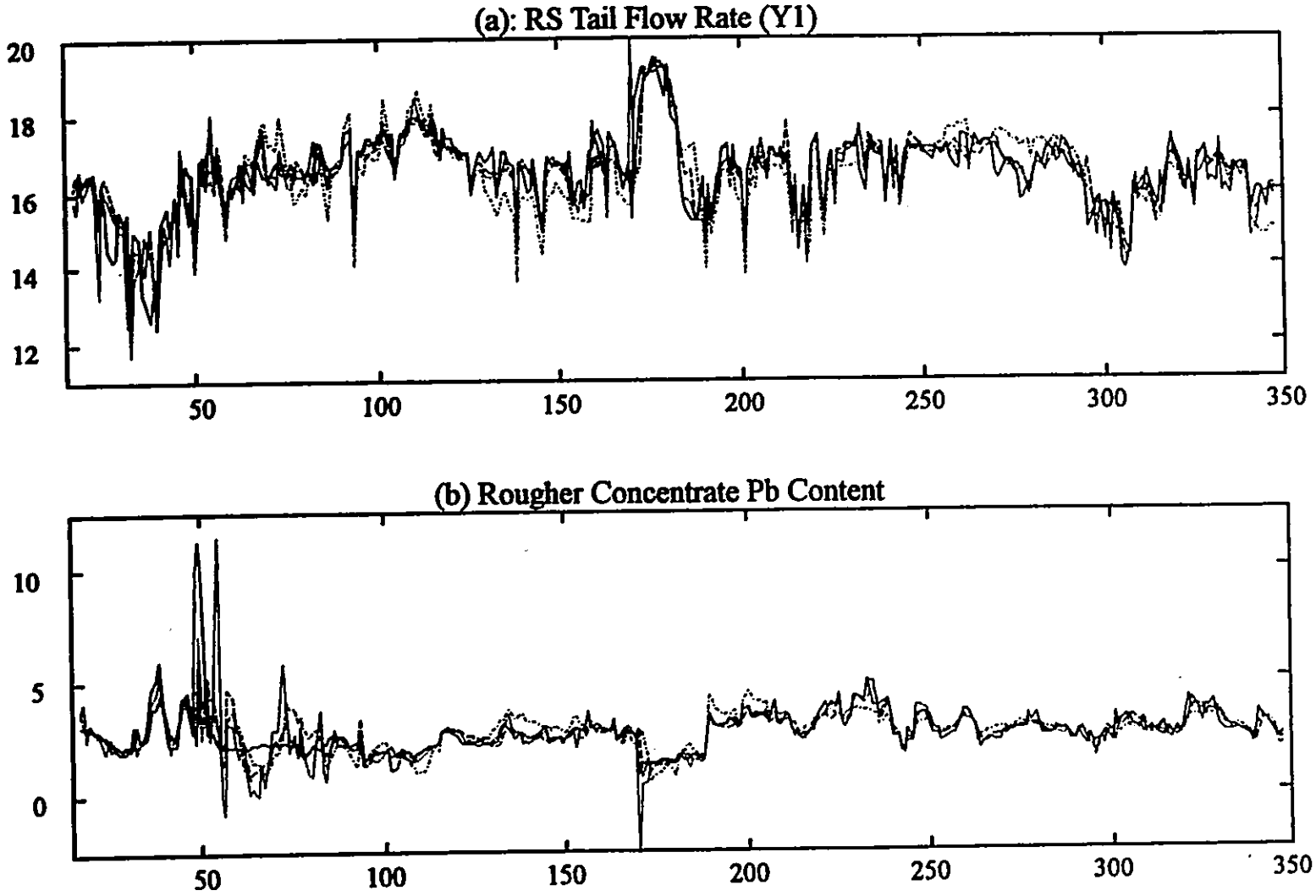

Figure 5.3a-b: The process output variables and their predictions from various recursive algorithms. Legends: solid line - the true process data; dashed line - recursive PLS with a variables forgetting factor; dashdot line - recursive PLS with a fixed forgetting factor ($\lambda_t = 1$); and the dotted line - recursive OLS with variable forgetting factor.

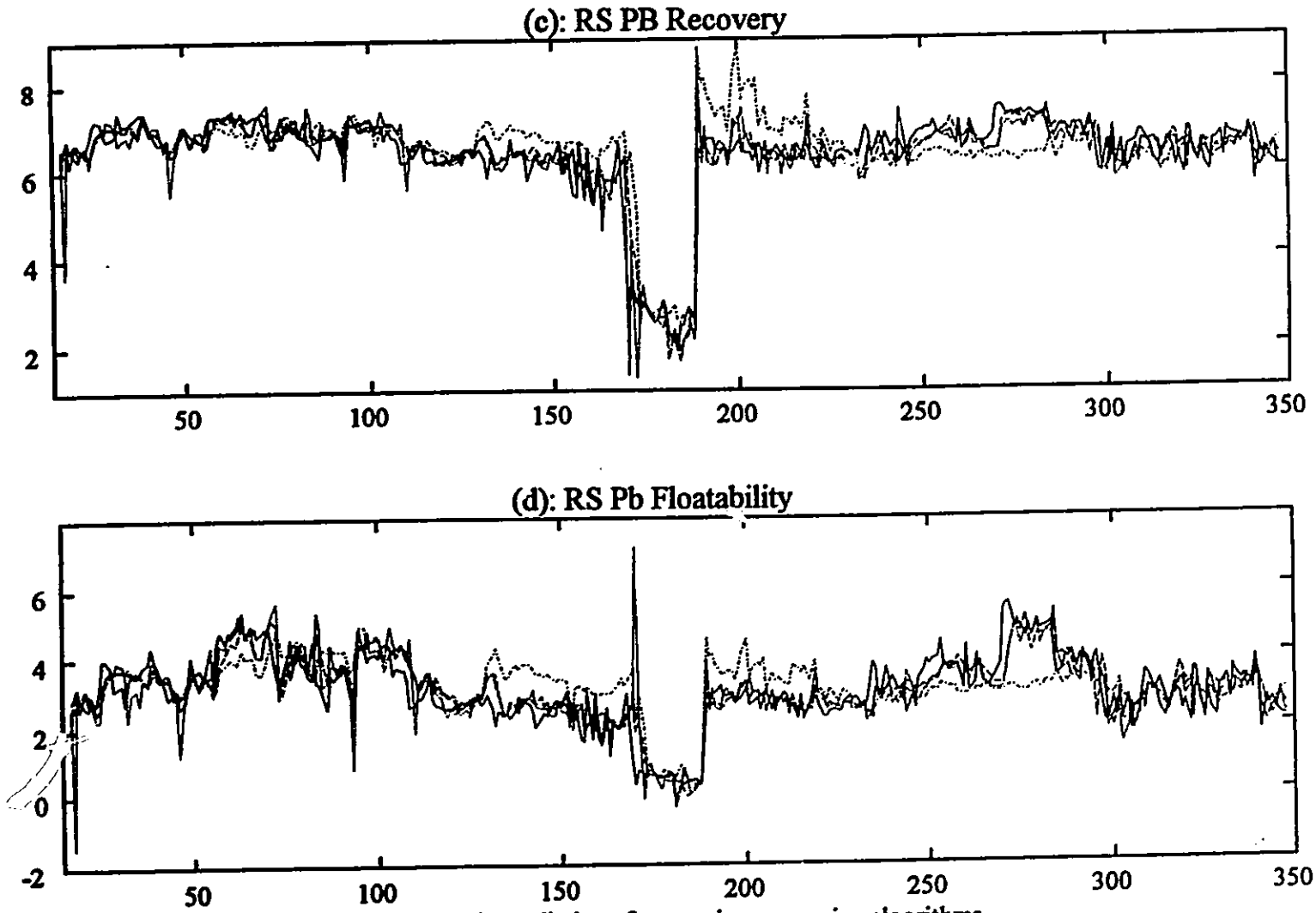

Figure 5.3c-d: The process output variables and their predictions from various recursive algorithms.
Legends: solid line - the true process data; dashed line - recursive PLS with a variables forgetting factor; dashdot line - recursive PLS with a fixed forgetting factor ($\lambda_t = 1$); and the dotted line - recursive OLS with variable forgetting factor.

5.5 Nomenclature

A_H	heat Transfer area, m^2
C_A	concentration of species A, kg/m^3
C_{AF}	initial concentration of species A in inlet stream, kg/m^3
C_B	concentration of species B, kg/m^3
$C_{p,S}$	specific heat capacity of species A and B, J/kg.K
$C_{p,W}$	specific heat capacity of water, J/kg.K
E_A	activation energy, J/mol
F_I	inlet feed flow rate, m^3/s
F_o	outlet flow rate, m^3/s
ΔH_R	heat of reaction, J/kg
k_o	Arrhenius rate constant, s^{-1}
$\dot{m}$	cooling water flow rate, kg/s
R	gas law constant, J/mol.K
t	time, s
T	reactor temperature, K
T_{CJ}	exit cooling jacket temperature, K
T_f	temperature of the inlet stream, K
T_I	entering cooling water temperature, K
U	overall heat transfer coefficient, $W/m^2.K$
V	volume of CSTR, m^3
V_{CJ}	volume of the cooling jacket, m^3
ρ_s	density of species, kg/m^3
ρ_w	density of water, kg/m^3

6. Multivariate Design of Experiments in Latent Variables

6.1 Introduction

It has always been desirable for both economical and environmental reasons to operate any process at its optimal conditions to produce the best product quality or to maximise production (e.g., maximum yield in a chemical reaction, etc.). The best approach to arrive at optimal conditions would be to develop a mechanistic model for the process from fundamental laws of mass and energy conservation, etc. However, these models are very tedious and time-consuming to develop (i.e., involving sometimes thousands of equations) and, for certain processes, it is practically impossible to develop such models due to lack of theoretical understanding. This has led to experimental techniques such as evolutionary operation (EVOP) (Box, 1957 and Box and Draper, 1969) and response surface methodology (RSM) (Box and Wilson, 1951) where the process is optimised experimentally by varying the manipulative process variables in a sequence of designed experiments.

In industry, there have been many successful applications of EVOP (Hunter and Kittrell, 1966) and RSM (Hill and Hunter, 1964) for the optimisation of processes. The disadvantage of these techniques is that they can handle only a small number of variables (usually 2 to 7) at a time. Two or three variables are usually varied using a factorial design and the process is moved in the steepest ascent/descent direction for optimisation. One runs into problems with these techniques when faced with processes with hundreds of potential manipulative variables. The design of experiments also becomes difficult due to interaction or dependencies among process variables due to physical relationships or operating constraints. A typical example of this would be a temperature profile in a chemical reactor. The temperatures can not be controlled independently but will move in the same direction due to a physical relationship. Furthermore, since most plants are

operated by human operators who can only think in two or three dimensions, then, during control, the process manipulated variables are only moved in very few dimensions (i.e., some of the variables are moved proportionally to others). Problems arising from dependencies or correlations among process variables can be treated well with partial least squares (PLS) regression. In this chapter, the objective is to develop a design methodology similar to RSM/EVOP for use in high dimensional systems. A variation of the PLS algorithm will be used as a tool to select meaningful groupings of variables (latent variables) so that the process designed experiments can be performed in these few groupings of variables rather than in a large number of individual manipulative variables. Therefore, the approach is to use the PLS regression method to analyse the historical plant data in order to find the important variables or groupings of variables (in the latent variables). Then the designed experiments can be performed in these few latent variables rather than in the original individual manipulative variables.

PLS will model the covariance structure which exists among the process manipulative variables and the output variables in the plant during the data collection process. Since the initial data are collected under the normal operating conditions, this method will find the linear combinations of the manipulated variables which have been moved together during normal operations either due to interaction/dependencies among process variables from physical relationships, operating constraints or operation policies; and which are correlated with the process output variables. Therefore, the first stage of experimental design in latent variables would be to start with reduced dimensions obtained from the historical data analysis. The experiments designed in this reduced space will be consistent with the past operation of the plant and will select variable groupings already seen to be correlated with the outputs. It may be easier to convince the operators to carry out the experimental design in these latent variables which are consistent with the past operation of the plant. In some cases, a process variable might not have been moved at all and thus would not show up in the analysis of the historical plant data. Therefore, process knowledge from a process engineer or a plant operator, where available, can be utilised to

incorporate other variables into experimental design or to break up certain combinations of process variables appearing in the original latent variables.

In ordinary PLS, the latent variables are a linear combination of all process variables which have been included in the X space (i.e., manipulative variables and non-manipulative variables such as disturbance and intermediate variables) and do not have any physical interpretation or meaning. Therefore, the design experiments can not be performed in this reduced space. However, if the latent vectors are selected in such a way that they are linear combinations of only certain similar process variables, then the latent variables can be given some kind of physical interpretation or meaning. This is referred to as the selective PLS. One example of selective PLS would be a latent variable being a linear combination of alike manipulated variables (i.e., solution flows). Now the EVOP or RSM designed experiments can be performed in this reduced space.

The ideas outlined in this chapter are in their preliminary development stage. A project was undertaken at the Eastman Kodak Company to investigate the ideas discussed in the succeeding sections. The data from a photographic paper coating machine was analysed and a set of factorial experimental designs, with input from the process engineer, was proposed. Due to certain circumstances, the experiments could not be carried out.

Several of these preliminary ideas on selective PLS and design of experiments have been outlined in Kettaneh-Wold et al., 1993.

6.2 Literature Review

There has been very little work relating to experimental design in latent variables reported in the literature. Wold et al. (1986) described the use of experimental design in latent variables. Their work relates to changing the position of an amino acid in a peptide compound. However, changing the position of the amino acid influences many physical and chemical properties of the peptide. These properties have been adequately explained by three latent variables and these latent variables are used as design variables.

6.3 Selective PLS

Two types of selective PLS algorithms are investigated as means of providing reduced subsets of variables for experimental designs.

1. Non-exclusive selective PLS - Process variables may appear in more than one latent variable. In this case, the columns of the loading matrix **W** may or may not be orthogonal.
2. Exclusive selective PLS - Process variables appear only in one of the latent variables. In this case, the columns of the loading matrix **W** will be orthogonal. This is due to the fact that process variables are included only in one of the latent vectors and their loadings will be zero in other latent vectors.

Both exclusive and non-exclusive algorithms retain the remaining PLS properties (2, 3 and 4) outlined in section 2.3.3 of chapter 2. This is because of the way residual matrices for **X** and **Y** are computed.

Frank (1987) has also presented an intermediate least squares (ILS) regression method by combining PLS and stepwise regression. The procedure for ILS consists of computing an ordinary PLS latent vector and then retaining the input with the largest absolute loading in the latent vector.

The algorithms for both exclusive and non-exclusive selective PLS are given in the appendix at the end of this chapter.

6.3.1 Procedure for Selective PLS

The procedure for selective PLS is as follows. Information regarding which variables to include in selective PLS can come either from the process experts (i.e., process engineers or operators) or from the careful analysis of the historical data collected from the process. It is always advantageous and desirable to analyse historical plant data to gain better understanding of the underlying process. Analysis of the historical data provides us with information regarding plant operations. Therefore, first of all, ordinary

PLS analysis is performed on the historical plant data using cross-validation techniques. This will provide us with the number of statistically significant latent vectors in the regression model and the percentage of sum of squares (% SS) explained of the output space, Y-block. The % SS explained by ordinary PLS will be used as a bench-mark for the comparison of the data analysis using selective PLS techniques. This will indicate if the important variables chosen in selective PLS are able to explain almost the same amount of SS as the ordinary PLS. This will verify the importance of those variables and ensure that no major groupings of variables have been rejected.

6.3.1.1 Non-Exclusive PLS

Once the ordinary PLS analysis has been carried out, then the selective PLS algorithms are started. First, the non-exclusive algorithm is used. The purpose of the non-exclusive selective PLS is to find a smaller number of variables which help to explain approximately the same amount of SS as the ordinary PLS with all variables. In other words, non-exclusive PLS is being used to prune the inputs. No distinction is made among the types of the variables to be included in the latent variables, that is the disturbance, intermediate and the manipulated variables can be incorporated in the same latent variable. The selection of the variables to be included in the non-exclusive selective PLS latent variable is based on the magnitude of their X-weights or **w** from the ordinary PLS analysis. The magnitude of the X-weights indicates the contribution of each variable in X-block toward defining a latent variable. Ordinary PLS analysis is performed on **X** and **Y** blocks to compute one latent vector. Inputs having weights above a certain cut-off or threshold in the first latent vector of ordinary PLS are included in the selective PLS latent vector. The value for cut-off can be selected using various approaches: (i) the process knowledge of the user; (ii) trial and error (try many arbitrary levels for cut-off until satisfied); (iii) the Pareto principle (cut-off $=1/\sqrt{K}$, where K is the number of X-variables); and (iv) cross-validation (Lindgren et al. ,1994). Once the value for cut-off has been decided, then the weights for the variables with ordinary PLS weights less then the

cut-off are set to zero and the weights for the variables with the ordinary PLS loadings greater than the cut-off are maintained. Then **w** is normalised to unity and the remaining vectors such as **t**, **p**, **q** ,**u** and the residual matrices for **X** and **Y** are computed in a same manner as in ordinary PLS. The X-loadings, **p**, versus X-weights, **w**, plot can be analysed to determine the correlations among X-block variables. The **p** vector contains the coefficients resulting from the regression of **t** on **X**. Therefore, the variables with large p-loadings will have high correlations to variables selected in the selective PLS latent vector. The same information can also be obtained by looking at the plot of % SS explained of X-block by each latent vector. High % SS explained of X-block variables which had not been selected indicates that these variables are highly correlated to some of the selected variables. However, when the residual matrices are computed for selective PLS, the correlated information content will also be subtracted from the nonselected variables.

Once the residual matrices for **X** and **Y** have been computed, the procedure described above is repeated to compute up to the desired number of selective latent vectors. One can either select the same number of latent vectors as in ordinary PLS analysis of the data or continue selecting latent vectors until the % SS explained by this selective PLS is same as that by the ordinary PLS. The non-exclusive selective PLS procedure can always be repeated many times with different values of cut-off until one has a minimal number of inputs which can explain nearly the same % SS as the ordinary PLS with all inputs.

6.3.1.2 Exclusive PLS

Once the importance of the variables from the non-exclusive PLS has been determined, the exclusive selective PLS procedure is started. In exclusive selective PLS, variables appear in only one of the latent vectors. If the intermediate or disturbance variables are found to have large X-weights, they can be selected in the exclusive selective PLS but are grouped separately from the manipulative variables. This is due to fact that

these variables can not be manipulated directly. The designed experiments are then performed on the latent variables from the exclusive PLS.

In the exclusive selective PLS, two ordinary PLS latent vectors of X and Y blocks are computed. The X-weights for both latent vectors are plotted against each other to find clusters or groups of variables. If some inputs are clustered together in a region, then these variables are highly correlated and it would be better to include them in the same latent variable. The variables or group of variables with large weights are selected in a latent vector. The exclusive selective PLS X-weights can then be computed using only these selected variables in X-block. The weights for the remaining variables are set to zero. Then the remaining vectors such as **t**, **p**, **q** and **u** are computed in a same manner as the ordinary PLS. The p-loadings versus the w-loadings plot can also be analysed to see if other variables have correlations to selected variables. If some variables are shown to have strong correlation (large p-loadings), then the exclusive selective PLS analysis can be repeated with these variables included in selective PLS X-block. Other plots such as % SS explained of Y-block can be examined to see the effect of these selected variables on the output space.

For the computation of **w** for the next selective PLS latent vector, inputs selected in previous latent vectors are set to zero. Since these inputs are not to reappear in any of the subsequent latent vector X-weights, their influence on the computation of the subsequent ordinary PLS latent vectors is neutralised by zeroing these inputs. This is due to the fact that weights from the ordinary PLS will be analysed to determine the next group of important variables to be included in the next selective latent vector. However, when p-loadings, residual matrices and % SS explained of X-block are calculated, the values for these inputs are brought back in.

6.4 Design of Experiments in Latent Variables

Once the selective PLS analysis is complete, then the design experiments can be performed in the latent variables consisting only of manipulative variables. Depending on the number of exclusive selective PLS latent variables, a full or fractional factorial design can be selected. Designing experiments in latent variables can give us a lot of degrees of freedom. For instance, to have a latent variable at its upper and lower levels of experimental design, the inputs, which form that latent variable, can be moved in many directions to attain the upper and lower levels. However, this will require the break-up of the existing correlation structure among manipulative variables. On the other hand, the existing correlation structure can be retained during the experimental design to arrive at the best conditions possible with this correlation structure. The amount of variation in the manipulative variables can be computed using the relationship,

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\mathrm{T}} \qquad (6.1)$$

When no further improvement in the process is possible by designing experiments in these latent variables, the correlation structure among variables may then be broken to achieve extra improvement. If the correlation structure is broken in the early stages of experimental design, there will be more independent manipulated process variables. Therefore, the number of latent variables computed from the experimental design data will increase.

Process knowledge will be required to determine the maximum allowable variation in each manipulated variable without affecting the plant operation. Inputs having little variations in the data set but which are considered important can also be included in the design experiments. However, this should all be based on the process knowledge of the experts.

Once the first set of designed experiment has been conducted, the new data is analysed and the process is moved in the steepest ascent/descent for optimisation. This procedure is carried out as long as there is continuous improvement in the process.

For this methodology, only steady state processes are considered where the product quality sampling interval is larger than the time constant of the process. Even if there is feedback control, it is hoped that the process would have returned to steady state before the next product quality measurements are taken.

Selective PLS algorithms have been applied to analyse the historical data sets from two industrial processes: the Noranda mineral flotation circuit and the Eastman Kodak photographic paper coating process. The results for the mineral flotation circuit are provided here.

6.5 Industrial Mineral Flotation Circuit Example

The process selected to illustrate the potential of selective PLS is a mining process located in New Brunswick, Canada. The mine processes a very fine grained complex sulphide ore containing mainly marmatite, galena and chalcopyrite and produces lead, copper, zinc and lead-zinc concentrates. Here only the flotation circuit for the copper and lead recovery is considered. The flotation circuit consists of an aerator for particle conditioning, rougher and scavenger, two cleaning stages and a regrinding circuit. The schematic of the flotation circuit is shown in Figure 6.1. Further details can be found in Hodouin et al. (1993).

The process input and output variables are listed in Tables 6.1 and 6.2, respectively. The manipulated and disturbance variables in Table 6.1 are marked accordingly. Most of the variables are measured whereas some of the variables such as metal flow rates are computed from mass balances. Of the seven output variables, only the concentrate flow rate and the Pb and Cu grades are measured and the remaining variables are computed from mass balances. Historical data for a period of 308 hours were analysed. Data represents hourly averages.

The objective of the flotation circuit is to maximise the metal grades and recoveries. However, the grade and the recovery has an inverse relationship as shown in

NUMBER	VARIABLE	Type of Variable
1	Fresh ore feed rate	Manipulated
2	Pb content in feed	Disturbance
3	Pb flowrate in feed	Disturbance
4	Cu content in feed	Disturbance
5	Cu flowrate in feed	Disturbance
6	Zn content in feed	Disturbance
7	Zn flowrate in feed	Disturbance
8	% fines particles	Manipulated
9	Soda ash flowrate in grinding	Manipulated
10	pH of feed	Intermediate
11	pH to RS circuit	Manipulated
12	pH to cleaner 1	Manipulated
13	Xanthate flow to RS	Manipulated
14	Xanthate to cleaner	Manipulated
15	Xanthate per ton of Pb & Cu to FC1	Manipulated
16	Xanthate per ton of Pb & Cu to FC2	Manipulated
17	Promoter flow per xanthate flow	Manipulated
18	Air flow to aerator	Manipulated
19	Air flow to cleaner 1	Manipulated
20	Air flow to cleaner 2	Manipulated
21	% solids in feed	Manipulated
22	Xanthate residual	Manipulated
23	Xanthate residual per feed rate	Manipulated

Table 6.1: Mining process input variables

NUMBER	VARIABLE
1.	Concentrate flowrate
2.	Concentrate Pb grade
3.	Concentrate Pb flow
4.	Pb recovery
5.	Concentrate Cu grade
6.	Concentrate Cu flow
7.	Cu recovery

Table 6.2: Mining process output variables

Figure 6.2. Therefore, one is interested in operating the mine as close to the theoretical upper bound of the grade-recovery relationship as possible. There are several variables that can be manipulated to the optimise the process. Most of the manipulated process variables such as the ore feed rate (variable #1) and the xanthate flow variables (variables 12-17) can be manipulated directly whereas some variables can be controlled indirectly by manipulating other variables in the grinding circuit.

First of all, an ordinary PLS analysis was carried out on the data set. This reveals if there is any information in the X-variables relating to the process output variables. The results are provided in Table 6.3. Five PLS components, as determined by cross-validation, were required to explain most of the variation in Y-block. Metal grades and flow rates are well explained whereas the metal recovery variables are explained around approximately 65%.

Now, a non-exclusive selective PLS variation is applied to the data set. The objective of the non-exclusive selective PLS is to find smaller number of variables which can explain the same amount of sum of squares of the output variables as the ordinary PLS with all variables. The cut-off value, selected based on heuristics, for the PLS weights for all selective latent vectors was 0.25. The results of the first non-exclusive selective PLS

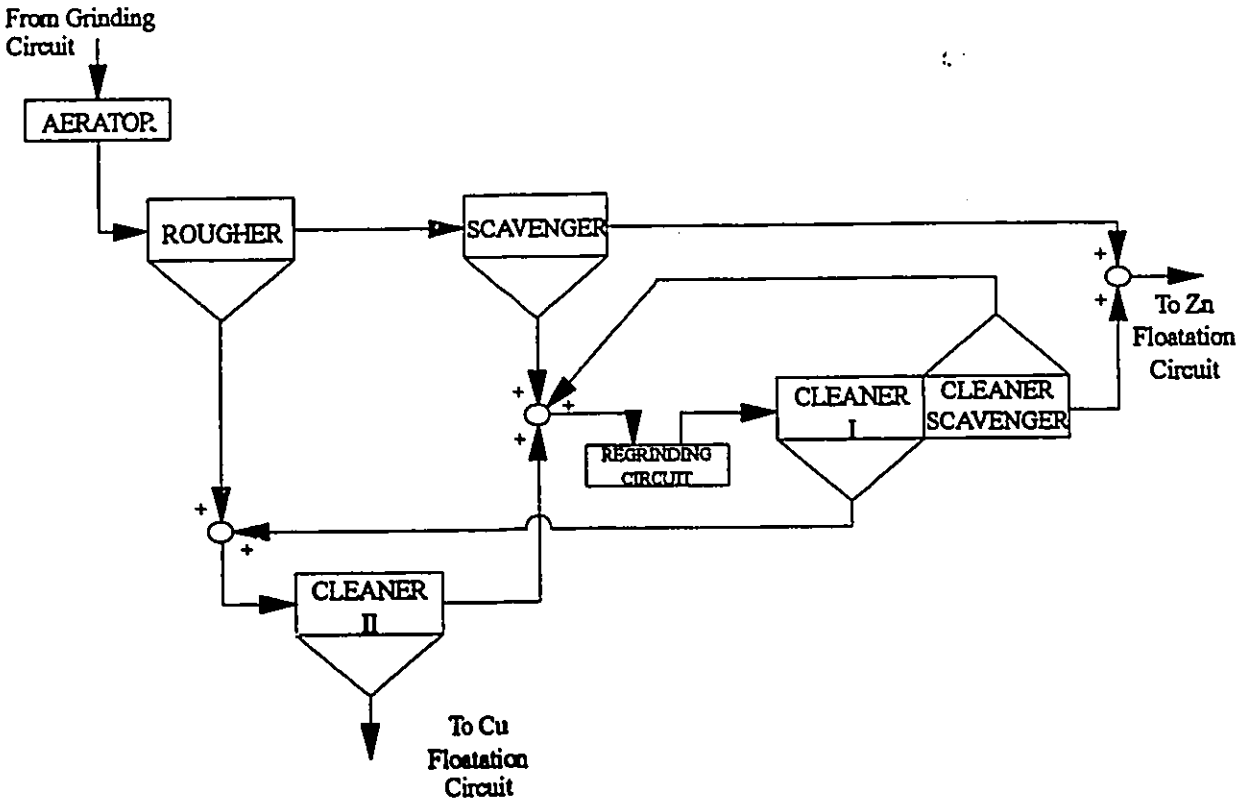

Figure 6.1: A schematic of the flotation circuit.

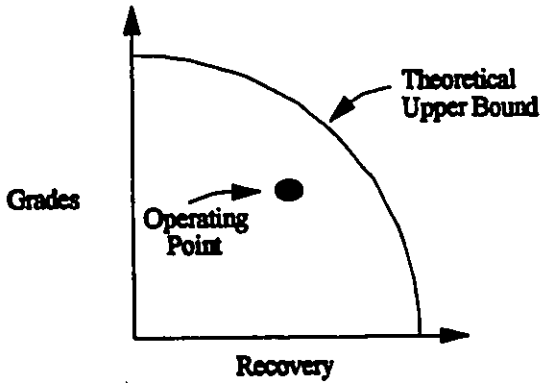

Figure 6.2: Process recovery-grade relationship

	% Sum of Squares (SS) Explained						
Component Number	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
1	47.6	9.2	43.3	14.2	72.6	43.6	19.0
2	72.9	12.2	86.0	38.9	72.8	64.3	54.7
3	88.6	89.8	86.3	54.6	72.8	72.0	58.9
4	88.6	91.2	86.4	60.8	84.9	90.4	60.1
5	91.2	95.8	87.3	60.6	87.7	90.8	65.7

Table 6.3: Ordinary PLS results

latent vector are shown in Figure 6.3. The X-weights for all variables from the first ordinary PLS latent vector along with the cut-off value shown as the dashed line are displayed in Figure 6.3a. Most of the disturbance variables (i.e., metal contents and flow rates, variables #2 to #7 with the exception of variable #6) and the Xanthate flow (variable #13) to rougher and scavenger section of the flotation circuit are seen to have large loadings. The reason for the Xanthate flow variable having large loading along with the metal content and flow rate variables is that the Xanthate flow to rougher and scavenger is probably manipulated relative to the metal contents to achieve higher recovery. By examining the X-weights (w_i) versus p-loadings (p_i) plot, variable #6 (Zn content in the feed) is seen to have a large correlation (i.e., large p-loading) to other metal content variables. However, due to larger cut-off value, this variable was not chosen in the selective PLS latent vector but this analysis can always be repeated with slightly lower cut-off value to include variable #6 in the selective PLS latent vector. The % SS explained of Y-block by this selective PLS latent vector is shown in Figure 6.3d. This selective latent vector basically concentrates on explaining the metal flow rates in the Y-block. The variables selected in subsequent selective latent vectors are shown in Figure 6.4. Some of the variables (i.e., metal contents and flow rates) are seen to appear in more than one latent vector. Note that variables can appear in more than one latent vector in this non-exclusive version of the selective PLS algorithm. The first few selected PLS latent vectors

are dominated by the metal content and flow rate variables, however, as their influence is removed by computing the residual matrices, other variables are seen to have larger loadings in the last three latent vectors. Some of manipulated variables which are seen to have stronger influence on the output variables are the air flow rates to cleaner 1 (#19) and cleaner 2 (#20) and Xanthate variables (#22-23).

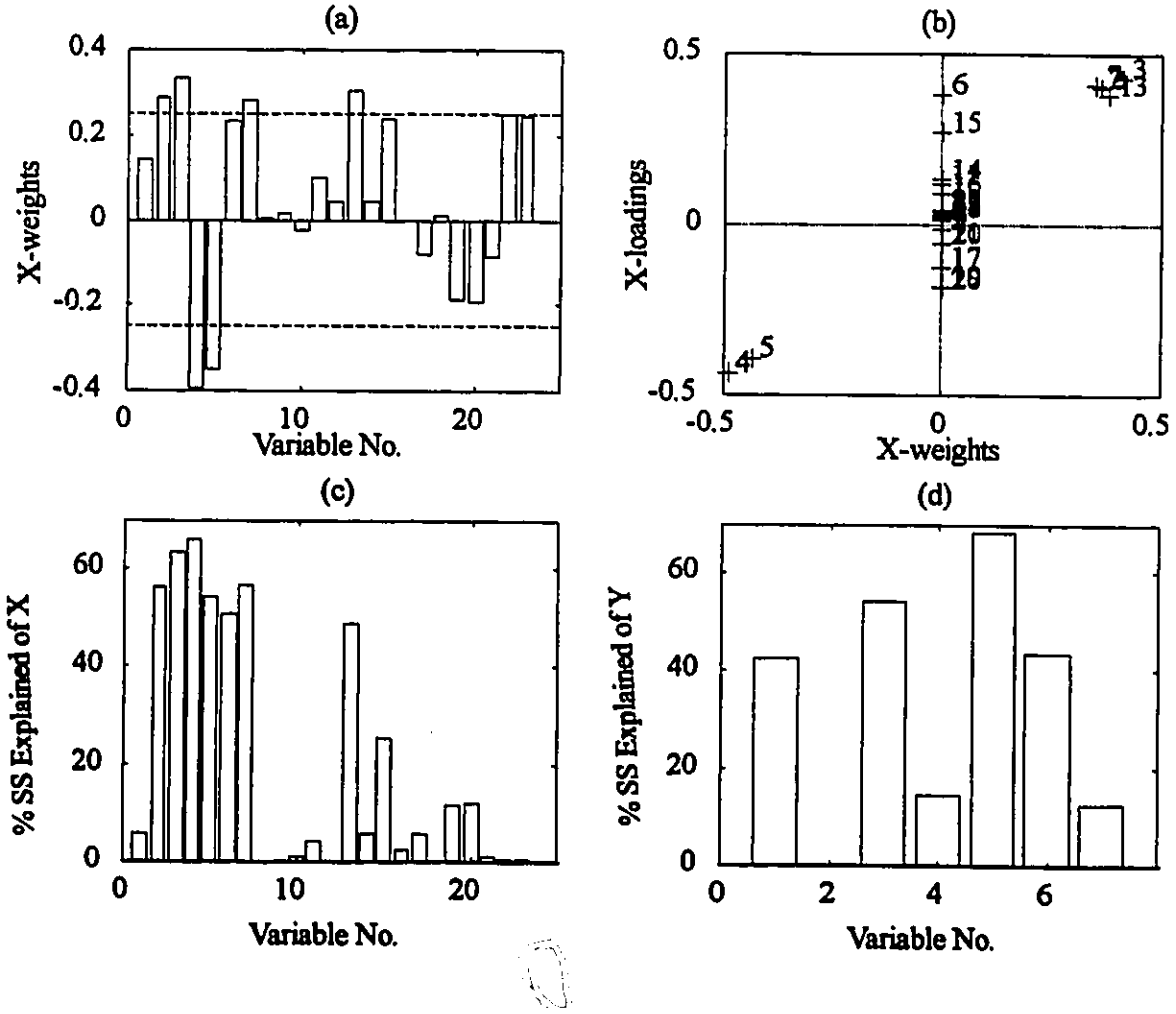

Figure 6.3: Results for the first dimension of the non-exclusive selective PLS.

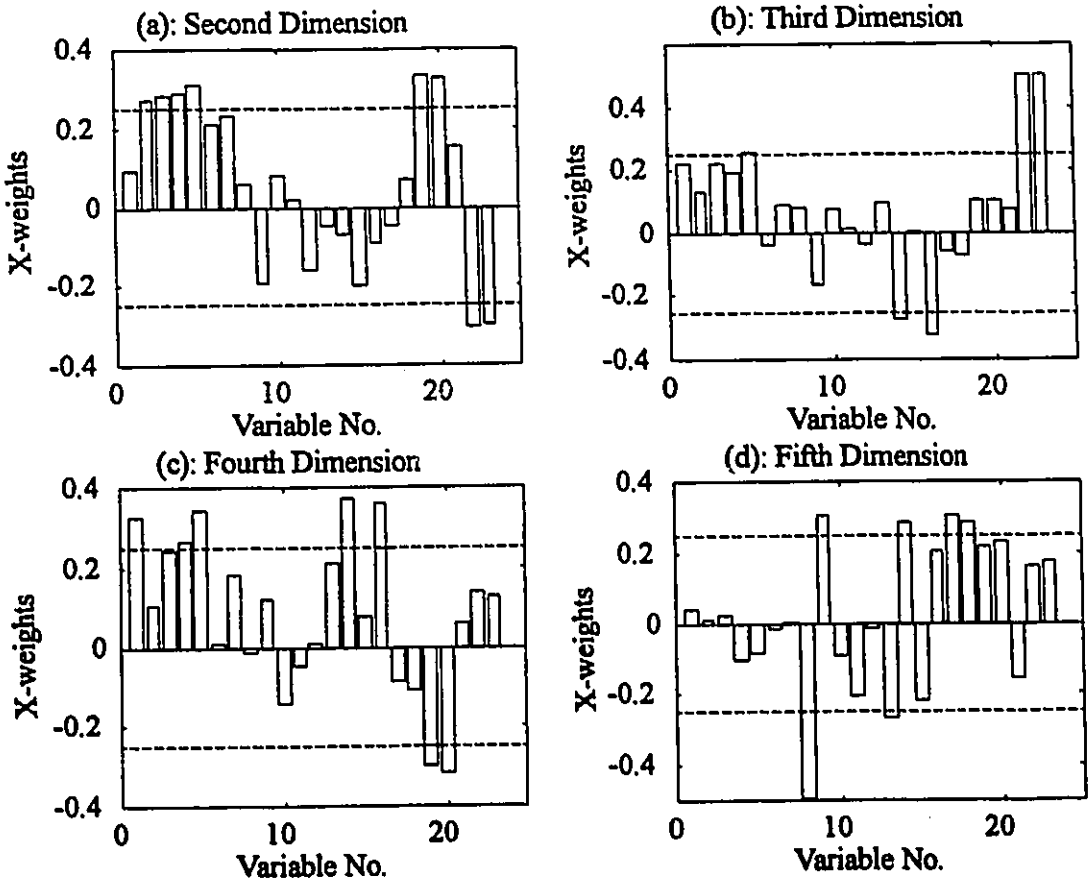

Figure 6.4: Loading plots for dimensions two through five of non-exclusive selective PLS.

Altogether, the non-exclusive PLS algorithm selected 18 of the 23 process variables. The % SS explained of the Y-block by ordinary and non-exclusive selective PLS algorithms is shown in Figure 6.5. The solid line represents the % SS explained by the non-exclusive PLS and the dashed line represents the % SS explained by ordinary PLS. As is evident, the non-exclusive PLS version provides slightly improved fit for four of the output variables and slightly worse for three of them.

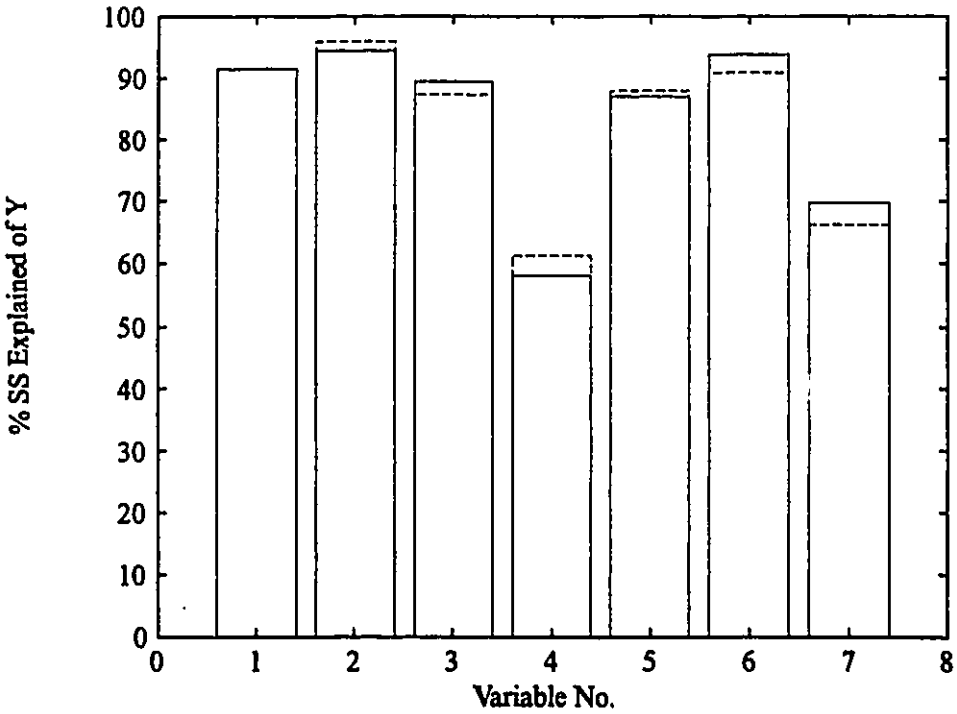

Figure 6.5: Percentage sum of squares explained of Y-block after five dimensions.
Legends: solid line: non-exclusive PLS; dashed line: ordinary PLS.

After performing the non-exclusive PLS analysis, the next step is to use the available process knowledge to interpret the non-exclusive PLS results. At this stage, one needs to consult with a process engineer or an operator and ask the following questions:

(i) Why are the variables selected by non-exclusive PLS important?

(ii) Why are the variables grouped together?

(iii) Is it due to real process dependencies or due only to the mode of operation?

(iv) Which variables should be grouped together?

The understanding gained from the careful analysis of the non-exclusive PLS results is essential for selection of exclusive PLS latent vectors. In exclusive PLS, the objective is to separate the dominant process variables into logical groupings that can be used for subsequent designed experiments for process optimisation. This stage should be performed along with the input from a process engineer or an operator. Several exclusive

PLS models can built, refining the decisions on the groupings, until a consensus has been established among the process experts.

Figure 6.6 shows the X-weights for all variables for the exclusive selective PLS analysis. Since the metal contents and flow rates in the feed were seen to have large effect on the output variables but can not be manipulated physically, therefore, these variables (#2-#7) were grouped together in the first latent vector of the exclusive version of the selective PLS algorithm. Variables #22 and #23 were found to have large loadings in the ordinary PLS analysis of the residual matrices after the first latent vector effect has been removed from X and Y-blocks. Variables # 22 and #23 are the Xanthate residual and Xanthate residual per feed rate. Xanthate residual is the excess Xanthate added to the flotation circuit over that required based on the metal flow rates. This helps to explain the metal grades in the outlet flow. In the third latent vector, the air flow rates to cleaner 1 and cleaner 2 (variables #19 and #20) are chosen. Due to the mode of operation, these two variables have always been moved together. This is evident from the fact that these two variables always appear in a group when analysing the ordinary PLS X-loadings plot of two different latent vectors against each other. The Xanthate flows to cleaner and flotation circuit 2 (variables #14 and #16) are selected in fourth latent vector whereas, in the fifth latent vector, fresh ore feed rate is picked.

The % SS explained of Y-block by ordinary and exclusive selective version of the PLS algorithms is shown in Figure 6.7. The exclusive selective PLS results are comparable to the ordinary PLS for almost all outputs with the exceptions of copper flow rate and recovery. However, the objective here is not to get a good fit for the model but to be able to find possible groupings of variables which were manipulated together due to operating procedures or physical dependencies.

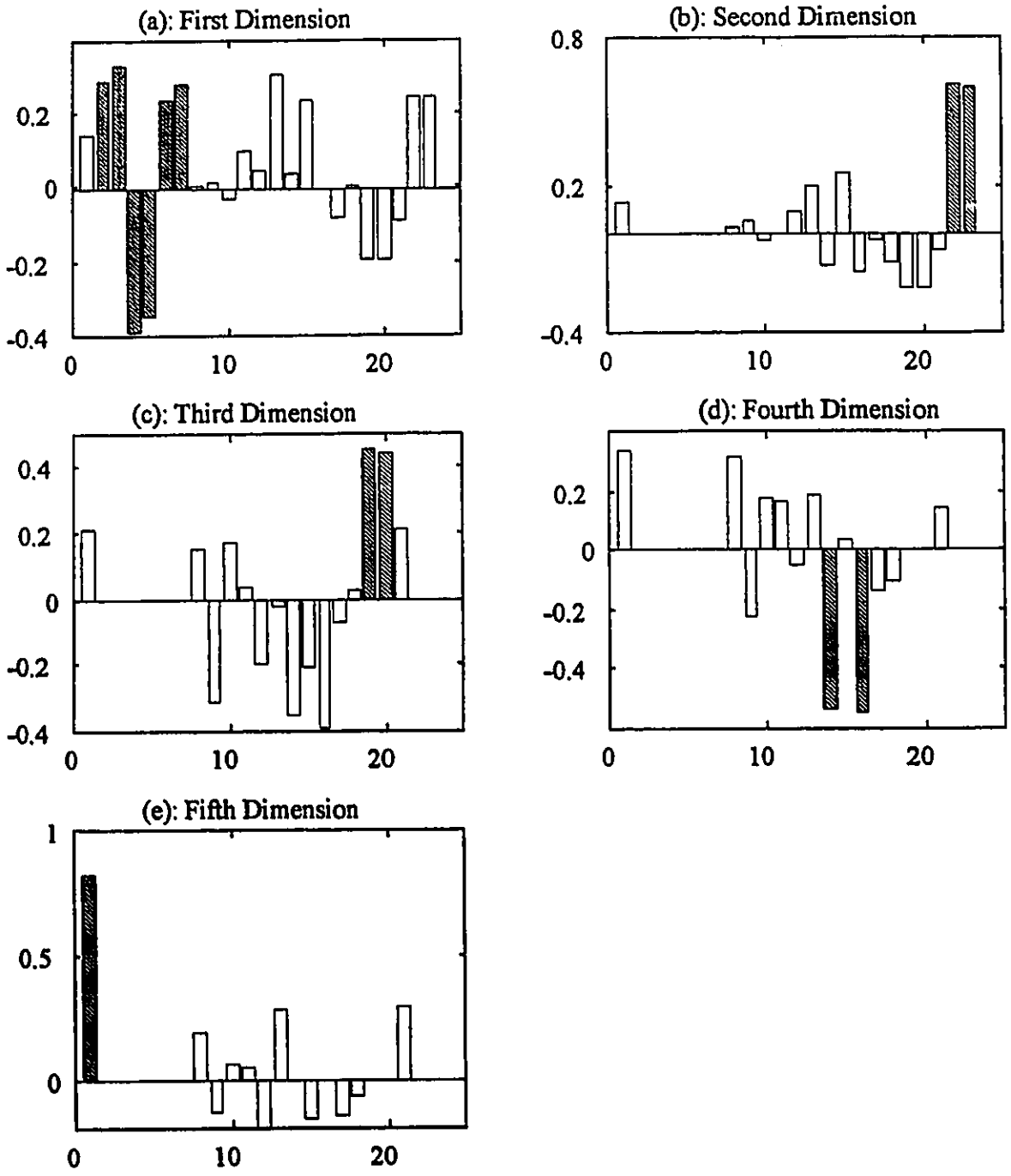

Figure 6.6: Loadings plots for the first five dimensions of the exclusive selective PLS.

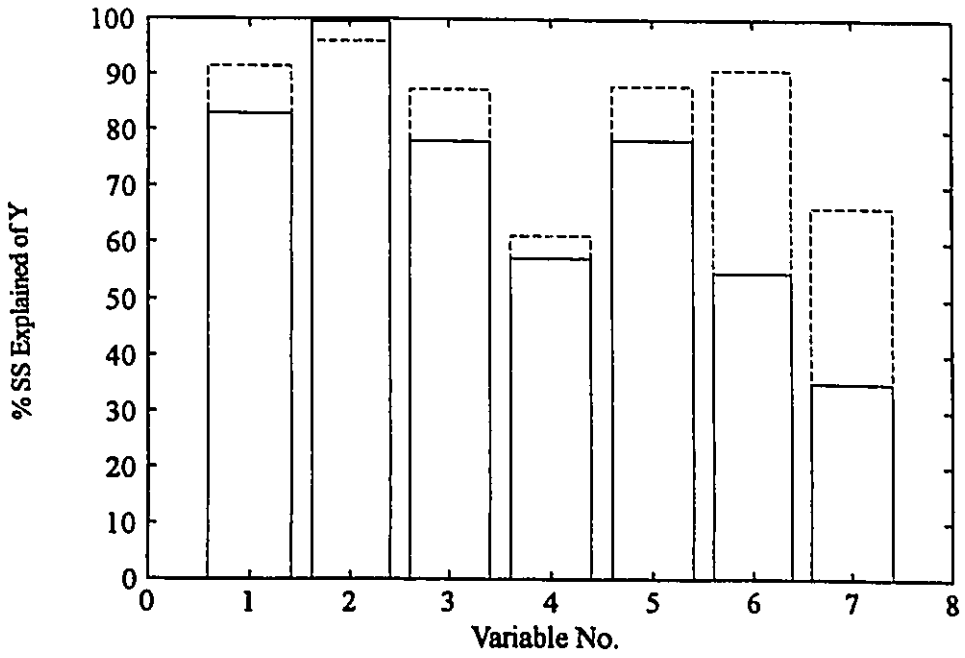

Figure 6.7: Percentage sum of squares explained of Y-block after five dimensions. Legends: solid line: exclusive PLS; dashed line: ordinary PLS.

Now that the groupings of alike variables have been selected, the design experiments can be conducted in these latent vectors. Based on the historical data analysis, there are four groupings in which the experiments can be performed:

1. Variables #22 and #23. Only one variable needs to be changed and the second variable is just the result of that change.
2. Variables #19 and #20. These variables have been increased or decreased together. It would be of interest to know their effect when these variables are decoupled.
3. Variables #14 and #16.
4. Variable #1(fresh ore feed rate).

Some of the variables such as promoter flow per Xanthate flow is found to have negligible effect on the outputs. However, as discussed earlier, the reason might be that this variable has not been varied enough during the process operations. If considered important, this

variable can also be included as a grouping of its own in the designed experiments. Either a complete factorial or fractional factorial experimental design can be carried out to investigate the effects of these groupings of variables on the output variables.

6.6 Conclusions and Future Work

As discussed earlier, evolutionary operation (EVOP) and response surface methodology (RSM) methods have been successfully applied in many instances. In this chapter, an extension of the concepts of EVOP and RSM to large multivariable systems is being proposed. A variation of the PLS algorithm, called selective PLS, can be used as a tool to obtain meaningful groupings of variables in which the EVOP/RSM experiments can be performed. It is a new methodology which could be used to select orthogonal variables consisting of linear combinations of alike manipulated variables by analysing the historical plant data. Since most of the process variables are moved in very few directions, therefore, selective PLS could be applied to find these underlying directions. Furthermore, a sequential design methodology in these groupings of variables is proposed.

The mineral flotation circuit data were only analysed to illustrate the potential and usefulness of the selective PLS to find latent variables in which the design experiments could be performed. Further work needs to be carried out to explore and investigate the potential of performing EVOP/RSM in these few latent variables for optimisation of chemical processes.

The basic ideas outlined in this chapter were to be tested on a real plant at the Eastman Kodak Company. The plant historical data were collected and analysed with the help of the process engineer to find groupings of variables for the process experimental design. Furthermore, a set of factorial designs was proposed. Unfortunately, the proposed experiments could not be carried out due to circumstances beyond everyone's control.

The usefulness of this new methodology can not be easily investigated on a process simulation. The difficulty with process simulation is one's inability to realistically simulate the plant under normal operating conditions.

6.7 Appendix

6.7.1 Non-Exclusive Selective PLS Algorithm

For each component, a=1,...,A

(a) 1. Compute an ordinary PLS latent vector using either the NIPALS algorithm or the kernel algorithm.

$$\mathbf{w}_a \propto \left(\mathbf{X}_a^T\mathbf{Y}_a\mathbf{Y}_a^T\mathbf{X}_a\right)\mathbf{w}_a$$

2. plot $\mathbf{w}_a$.

(b) 1. Select X-variables with $\left|w_a^k\right|$ > cut-off. [k is the index for the X-variables].

if $\left|w_a^k\right|$ < cut-off, $\left|w_a^k\right|$=0.

2. Normalise $\mathbf{w}_a$ and compute $\mathbf{t}_a$ and $\mathbf{p}_a$ and $\mathbf{q}_a$.

$$\mathbf{w}_a = \frac{\mathbf{w}_a}{\sqrt{(\mathbf{w}_a^T\mathbf{w}_a)}}$$

$$\mathbf{t}_a=\mathbf{X}_a\mathbf{w}_a$$

$$\mathbf{p}_a = \frac{\mathbf{X}_a^T\mathbf{t}_a}{\mathbf{t}_a^T\mathbf{t}_a}$$

$$\mathbf{q}_a = \frac{\mathbf{Y}_a^T\mathbf{t}_a}{\mathbf{t}_a^T\mathbf{t}_a}$$

3. Deflate **X** and **Y**.

$$\mathbf{X}_{a+1} = \mathbf{X}_a - \mathbf{t}_a\mathbf{p}_a^T$$

$$\mathbf{Y}_{a+1} = \mathbf{Y}_a - \mathbf{t}_a\mathbf{q}_a^T$$

(c) Goto step (a) to compute the next latent vector.

6.7.2 Exclusive Selective PLS Algorithm

For each component, a=1,...,A

(a) 1. Compute two ordinary PLS latent vectors of $\mathbf{X}_a$ and $\mathbf{Y}_a$ using either the NIPALS algorithm or the kernel algorithm.

(i) $\mathbf{X}_{temp}=\mathbf{X}_a$ and $\mathbf{Y}_{temp}=\mathbf{Y}_a$.

$$\mathbf{w}_1^{temp} \propto \left(\mathbf{X}_{temp}^T \mathbf{Y}_{temp} \mathbf{Y}_{temp}^T \mathbf{X}_{temp}\right)\mathbf{w}_1^{temp}$$

(ii) Compute the remaining loading vectors and then deflate $\mathbf{X}_{temp}$ and $\mathbf{Y}_{temp}$.

(iii) $$\mathbf{w}_2^{temp} \propto \left(\mathbf{X}_{temp}^T \mathbf{Y}_{temp} \mathbf{Y}_{temp}^T \mathbf{X}_{temp}\right)\mathbf{w}_2^{temp}$$

2. plot $\mathbf{w}_1^{temp}$ and $\mathbf{w}_2^{temp}$. Look for clusters or groupings of variables.

(b) 1. Select X-variables. The X-variables can be selected by examining the $\mathbf{w}_1^{temp}$ versus $\mathbf{w}_2^{temp}$ plot or from process knowledge.

2. Compute $\mathbf{w}_a$ based on the selected variables and compute $\mathbf{t}_a$ and $\mathbf{p}_a$ and $\mathbf{q}_a$.

$$\mathbf{w}_a \propto \left(\mathbf{X}_{a,selected}^T \mathbf{Y}_a \mathbf{Y}_a^T \mathbf{X}_{a,selected}\right)\mathbf{w}_a$$

$$\mathbf{t}_a=\mathbf{X}_a\mathbf{w}_a$$

$$\mathbf{p}_a = \frac{\mathbf{X}_a^T \mathbf{t}_a}{\mathbf{t}_a^T \mathbf{t}_a}$$

$$\mathbf{q}_a = \frac{\mathbf{Y}_a^T \mathbf{t}_a}{\mathbf{t}_a^T \mathbf{t}_a}$$

4. Deflate **X** and **Y**.

$$\mathbf{X}_{a+1} = \mathbf{X}_a - \mathbf{t}_a \mathbf{p}_a^T$$

$$\mathbf{Y}_{a+1} = \mathbf{Y}_a - \mathbf{t}_a \mathbf{q}_a^T$$

5. Zero out the columns for the variables which are selected in step (1).

(c) Goto step (a) to compute the next latent vector.

7. Summary and Conclusions

In this thesis, various multivariate statistical regression methods were investigated for estimating process models from the process input-output data. These identified models were to be used for designing model based controllers and experimental optimisation of multivariate processes. The major conclusions and the main contributions of this thesis are outlined in this chapter.

In chapter 2 of this thesis, various approaches to identifying non-parsimonious finite impulse response (FIR) models were compared on the basis of closeness of fit to the true process, robust stability provided by the resulting model, and the control performance obtained. The non-parsimonious FIR models can be estimated directly from the process input-output data using various regression methods such as ordinary least squares, regularised least squares and partial least squares. Alternatively, a low order parsimonious rational transfer function model can be fitted to the process input-output data by prediction error methods and then the FIR weights can be obtained from it.

Directly identifying FIR models have certain advantages such as that no structural decisions need to be made and that they can model any complex linear system. However, as was shown in chapter 2, along with these advantages come substantial disadvantages such as poor robustness and performance of the resulting controllers designed using the directly identified FIR. models. The major conclusion by all assessments is that obtaining FIR models by first identifying low order transfer function models by prediction error methods was much superior to any of the methods which directly identified the FIR models. This is due to the fact that the parsimonious nature of the model leads to less overfitting of the data and induces, through its structure, a smooth behaviour on the FIR weights obtained from the estimated models. In particular, the latter feature leads to controllers which are more robust in terms of performance stability. Although, it is

recognised that more effort is generally involved in identifying the structure of parsimonious transfer function models, is generally involved in identifying the structure of parsimonious transfer function models, it is recommended that this be done whenever the structure of the process is expected to be reasonably simple.

If it is desired to fit the non-parsimonious FIR models directly, then one should try to incorporate the available prior process knowledge into a regression method's objective function, if possible, to improve its estimates. The regularised least squares method where the changes in the impulse weights were penalised by an increased weighting over lags provided uniformly superior performance compared to other regression methods such as ordinary least squares and partial least squares by all assessment methods. The PLS regression method provided excellent predictions but poor robust stability and control performance. The number of latent vectors in the PLS model were selected by validation against a data set. This criterion is only based on obtaining good predictions of the output space which depends largely on identifying the dominant directions in the data. Therefore, a substantially larger number of latent vectors than that suggested by the cross-validation or validation against a test data set are required to provide satisfactory results.

In chapter 3, the potential of multi-output identification for multivariate processes was investigated via simulations on two process examples: a quality control example and an extractive distillation column. If there are common or correlated parameters among models for different output variables and/or correlated noise, then modelling of all the outputs together could lead to better and more robust models. The identification of both the parsimonious transfer function models using multivariate prediction error methods, and of non-parsimonious FIR models using multivariate statistical regression methods such as two-block partial least squares (PLS2), canonical correlation regression (CCR) and reduced rank regression (RRR) were considered. The multi-output identification results were compared to traditional single-output identification from several points of view: (i) best predictions; (ii) closeness of the model to the true process; (iii) the precision of the identified models in frequency domain; (iv) stability robustness of the resulting

model based control system; and (v) the multivariable control performance. The multi-output identification methods provided better results compared to those of the single-output identification methods based on essentially all comparison criteria. The benefits for using multi-output identification are most obvious when there are limited amounts of data and when the secondary variables have better signal to noise ratios. The differences between multi-output identification and single-output identification methods disappear with larger data sets and better signal to noise ratios for all process variables. Furthermore, slightly inadequate model structures resulting from the constraints of the common parameterisation among the output transfer function models led to some bias in the models identified by multi-output prediction error methods.

In chapter 4, an improvement to the PLS algorithm was made. In PLS, generally both the **X** and **Y** matrices are deflated after each latent vector computation. It was shown that only one of either the **X** or the **Y** matrix needs to be deflated during the sequential process of computing latent vectors. This result then led to two very fast PLS kernel algorithms. In the first algorithm, the covariance matrix, $\mathbf{X}^T\mathbf{X}$, is not computed and **X** is used directly in the computations for the loading vectors. This kernel algorithm is advantageous in a situation where the construction of $\mathbf{X}^T\mathbf{X}$ is the major rate determining step in terms of the computational effort. In the second algorithm, the covariance matrix $\mathbf{X}^T\mathbf{X}$ is computed once and then used subsequently in the loading vectors computations. This new algorithm can be used in situations where there are more observations than the number of X-variables. In such a situation, it is convenient to compute $\mathbf{X}^T\mathbf{X}$ once. The performance of these new algorithms were compared to that of De Jong and Ter Braak's modified kernel algorithm in terms of the speed and the new algorithms were shown to be much faster. These algorithms required as little as one-fifth of the flops (floating points operations) of the De Jong and Ter Braak algorithm. Furthermore, each of these algorithms was shown to have certain advantages when performing cross-validation and treating missing data.

In chapter 5, a new and fast recursive, exponentially weighted PLS algorithm which provided greatly improved parameter estimates in most process situations was presented. This new recursive algorithm for updating of the PLS regression model was developed by combining the improved PLS kernel algorithm developed in chapter 4 with the recursive updating of the covariance matrices $\left(\mathbf{X}^T\mathbf{X}\right)_t$ and $\left(\mathbf{X}^T\mathbf{Y}\right)_t$. A technique for updating the standard deviation of each process variable at each sampling period for autoscaling of the data was also presented. The recursive PLS algorithm was then applied to adaptive control of a simulated 2 by 2 multivariable continuous stirred tank reactor (CSTR) and updating of a multi-output prediction model for an industrial mineral flotation circuit. The recursive PLS algorithm provided much better performance than the recursive least squares algorithm. The main advantage of the recursive PLS algorithm is that it did not suffer from the problems associated with correlated variables and short data windows. During the adaptive control simulations of the multivariable CSTR, it provided satisfactory control when the recursive least squares algorithm 'blew' up a few times due to the ill-conditioned covariance matrix $\left(\mathbf{X}^T\mathbf{X}\right)_t$. For the industrial soft sensor application, the new algorithm provided much improved estimates of all ten response variables whereas the recursive least squares algorithm provided very poor estimates at certain instances.

In chapter 6, a design methodology similar to the evolutionary operation (EVOP) and the response surface methodology (RSM) for optimisation of high dimensional system was proposed. Two variations of the PLS algorithm, called non-exclusive and exclusive selective PLS, were developed. They can be used as a tool to obtain meaningful groupings of the process variables in which the EVOP/RSM experiments can be performed. It is a new methodology which can be used to select orthogonal variables consisting of linear combinations of alike manipulated variables by analysing the historical plant data. During normal plant operations, most of the process variables are moved in very few directions and the selective PLS could be applied to find these underlying

directions. The experiments designed in this reduced space will be consistent with the past operation of the plant and the selective PLS will select variable groupings already seen to be correlated with the process outputs. The mineral flotation circuit data were analysed to illustrate the potential and usefulness of the selective PLS to find latent variables in which the experiments could be performed.

The basic ideas of multivariate design of experiments in latent variables outlined in chapter 6 were to be illustrated using historical data from an industrial photographic paper coating machine. With the help from the process engineers, groupings of variables for the process experimental design were found and a set of factorial designs was proposed. Although these designs were not actually applied to the industrial process they appear to hold promise. Further progress in the area will depend upon working closely with industry in an actual application.

8. References

Astrom, K.J. and Wittenmark, B., "Adaptive Control", Addison-Wesley Publishing Company, 1989.

Bartlett, M.S., "Multivariate Analysis", *J. Roy. Statist. Soc.* **B-9**, 176-197, 1947.

Box, G.E.P., "Evolutionary Operation: A Method for Increasing Industrial Productivity", *Applied Statistics* **6**, 81-101, 1957.

Box, G.E.P. and Draper, N.P., "Evolutionary Operation", John Wiley & Sons, N.Y., 1969.

Box, G.E.P. and Draper, N.R., "The Bayesian Estimation of Common Parameters from Several Responses", *Biometrica* **52** (3/4), 355-365, 1965.

Box, G.E.P., Hunter, W.G., MacGregor, J.F. and Erjavec, J., "Some Problems Associated with the Analysis of Multiresponse Data", *Technometrics* **15**(1), 33-51, 1973.

Box, G.E.P. and Jenkins, G.W., "Time Series Analysis: Forecasting and Control", Holden Day, San Francisco, 1976.

Box, G.E.P., and Wilson, K.B., "On the Experimental Attainment of Optimum Conditions", *J. Roy. Statist. Soc.* **B-13**, 1-45, 1951.

Breiman, L. and Friedman J.H., "Predicting Multivariate Responses in Multiple Linear Regression", accepted for publication, *J. Roy. Statist. Soc.*, 1996.

Brooks, R. and Stone, M., "Joint Continuum Regression for Multiple Predictands", *J. Am. Statist. Assoc.* **89**, 1374-1377, 1994.

Burnham, A.J., Viveros-Aguilera, R. and MacGregor, J.F., "Frameworks for Latent Variable Multivariate Regression", *J. Chemometrics* **10**, 31-45, 1996.

Clarke, D.W., "Generalized Least Squares Estimation of Parameters of a Dynamic Model", First IFAC Symposium on Identification in Automatic Control Systems, Prague, 1967.

Clarke, D.W., Mohtadi, C. and Tuffs, R.S., "Generalized Prediction Control - Part I. The Basic Algorithm", *Automatica* **23**, 137-148, 1987.

Chin, A., "A Fundamental Tray-by-Tray Model for Methanol-Acetone-Water Extractive Distillation Column Simulation", Department of Chemical Engineering, 1989.

Cutler, C.R. and Ramaker, B.L., "Dynamic Matrix Control - A Computer Control Algorithm", AIChE National Meeting, Houston, TX, 1979.

De Jong, S. and Ter Braak, C. J. F., "Comments on the PLS Kernel Algorithm", *J. Chemometrics* **8**, 169-174 (1994).

Fortescue, T.R., Kershenbaum, L.S. and Ydstie, B.E., "Implementation of Self-tuning Regulators with Variable Forgetting Factors", *Automatica* 17 (6), 831-835, 1981.

Frank, I.E., "Intermediate Least Squares Regression Methods", *Chemometrics Intell. Lab. Syst.* **1**, 233-242, 1987.

Garcia, C.E. and Morari, M., "Internal Model Control. 2. Design Procedure for Multivariable Systems", *Ind. Eng. Chem. Fundam.* **24**, 472-484, 1985.

Geladi, P. and Kowalski, B., "Partial Least Squares Regression: A Tutorial", *Anal. Chim. Acta* **185**, 1-17, 1986.

Goodwin, G.C., Gevers, M., and Ninness, B., "Quantifying the Error in Estimated Transfer Functions with Application to Model Order Selection", *IEEE Trans. Autom. Contr.* **37** (7), 913-927, 1992.

Helland, K., Berntsen, H.E., Borgen, O.S., and Martens, H., "Recursive Algorithm for Partial Least Squares Regression", *Chemometrics Intell. Lab. Syst.* **14**, 129-137, 1991.

Hill, W.G. and Hunter, W.J., "A Review of Response Surface Methodology: A Literature Survey", *Technometrics* **8**(4), 571-590, 1964.

Hodouin, D., MacGregor, J.F., Hou, M. and Franklin, M., "Multivariate Statistical Analysis of Mineral Processing Plant Data", *CIM Bulletin* **86** (975), 23-34, 1993.

Hoerl, A.E. and Kennard, R.W., "Ridge Regression: Biased Estimation for Nonorthogonal Problems", *Technometrics* **12** (1), 55-67, 1970a.

Hoerl, A.E. and Kennard, R.W., "Ridge Regression: Applications to Nonorthogonal Problems", *Technometrics* **12** (1), 69-82, 1970b.

Hoskuldsson, A., "PLS Regression Methods", *J. Chemometrics* **2**, 211-228, 1988.

Hoskuldsson, A., "The H-Principle in Modelling with Applications to Chemometrics", *Chemometrics Intell. Lab. Syst.* **14**, 139-153, 1992.

Hunter, W.G. and Kittrell, J.R., "Evolutionary Operation: A Review", *Technometrics* **8** (3), 389-397, 1966.

Kattaneh-Wold, N., MacGregor, J.F., Dayal, B. and Wold, S., "Multivariate Design of Process Experiments (M-DOPE)", *Chemometrics Intell. Lab. Syst.* **23**, 39-50, 1994.

Kozub, D., Personal Communication, 1994.

Lindgren, F., Geladi, P. and Wold, S., "The Kernel Algorithm for PLS", *J. Chemometrics* **7**, 45-59, 1993.

Lindgren, F., Geladi, P., Rannar, S. and Wold, S., "Interactive Variable Selection (IVS) for PLS Part I: Theory and Algorithms", *J. Chemometrics* **8**, 349-363, 1994.

Ljung, L., "System Identification: Theory for the User", Prentice-Hall, Englewood Cliffs, N.J., 1987.

MacGregor, J.F., Kourti, T. and Kresta, J.V., "Multivariate Identification: A Study of Several Methods", IFAC International Symposium, ADCHEM '91 Proceedings, Pergamon Press, 369-375, Tolouse, France, Oct. 1991.

Maurath, P.R., Laub, A.J., Seborg, D.E. and Mellichamp, D.A., "Predictive Controller Design by Principal Component Analysis", *Ind. Eng. Chem. Res.* **27**, 1204-1212, 1988.

Morari, M. and Zafiriou, E., "Robust Process Control", Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

Morf, M. and Kailath, T., "Square-Root Algorithms for Least Squares Estimation", *IEEE Trans. Autom. Contr.* **AC-20** (4), 487-497, 1975.

Myers, R.H., "Classical and Modern Regression with Applications", PWS-Kent Publishing Company, Boston, 1990.

Pearson, K., "On Lines and Planes of Closest Fit to Systems of Points in Space", *Philos. Mag.* **2**, 559-572, 1901.

Rannar, S., Geladi, P., Lindgren, F. and Wold, S., "A PLS Kernel Algorithm for Data Sets with Many Variables and Few Objects. Part II: Cross-Validation, Missing Data and Examples", *J. Chemometrics* **9**, 459-470, 1995.

Ricker, N.L., "The Use of Biased Least Squares Estimators for Parameters in Discrete-Time Pulse-Response Methods", *Ind. Eng. Chem. Res.* **27**, 343-350, 1988.

Söderström, T. and Stoica, P., "System Identification", Prentice-Hall International, U.K., 1989.

Tso, M. K.-S., "Reduced Rank Regression and Canonical Analysis", *J. Roy. Statist. Soc.* **43**(2), 183-189, 1981.

Wierda, S.J., "Multivariate Statistical Process Control - Recent Results and Directions for Future Research", *Statistica Neerlandica* **48** (2), 147-168, 1994.

Wise , B.M. and Ricker, N.L., "Identification of Finite Impulse Response Models by Principal Components Regression: Frequency-Response Properties", *Process Control & Quality* **4**, 77-86, 1992.

Wise, B.M. and Ricker, N.L., "Identification of Finite Impulse Response Models with Continuum Regression", *J. Chemometrics* **7**, 1-14, 1993.

Wold, S., "Exponentially Weighted Moving Principal Component Analysis and Projections to Latent Structures", *Chemometrics Intell. Lab. Syst.* **23**, 149-161, 1994.

Wold, S., "Cross-Validatory Estimation of the Numbers of Components in Factor and Principal Component Models", *Technometrics* **20**, 397, 1978.

Wold, H., "Soft Modelling. The Basic Design and Some Extensions", in "Systems Under Indirect Observation", K. Joreskog and H. Wold, Ed., North Holland, Amsterdam, 1982.

Wold, S., Sjostrom, M., Carlson, R., Torbjorn, L., Hellberg, S., Skagerberg, B., Wikstrom, C. and Ohman, J., "Multivariate Design", *Anal. Chim. Acta* **191**, 17-32 (1986).