Subspace Identification Methods for Process Dynamic Modeling

BY

Ruijie Shi, M. Eng.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

McMaster University

Subspace Identification Methods for Process Dynamic Modeling

DOCTOR OF PHILOSOPHY (2001) McMaster University

(Chemical Engineering) Hamilton, Ontario

Title: Subspace Identification Methods for Process Dynamic Modeling

Author: Ruijie Shi

B.Eng. (Tsinghua University)

M. Eng. (Tsinghua University)

Supervisor: Professor John F. MacGregor

Number of Pages: xiv, 229

Abstract

Subspace Identification Methods (SIMs) are a class of new identification methods that have drawn considerable interest in recent years. The key idea of these methods is to estimate the process states or the extended observability matrix directly from the process input and output data. The best-known SIMs are Canonical Variate Analysis (CVA), Numerical Subspace State-Space System Identification (N4SID) and Multivariable Output Error State sPace (MOESP). This thesis focuses on both fundamental research and application study of SIMs.

The first part of the fundamental research involves the analysis of SIM algorithms from a statistical estimation viewpoint. For this purpose, a multi-step state-space model is set up first to reveal the relationships between the process states and the process data sets. Based on this model, SIM algorithms are analyzed to reveal their basic principles and bias issues. Several new SIM algorithms are proposed and shown to have similar performance as the existing algorithms.

Relationships between SIMs and Latent Variable Methods (LVMs) for identification are then explored. It is shown that N4SID can be derived from Reduced-Rank Analysis (RRA) just as CVA is developed based on Canonical Correlation Analysis (CCA). Insights from this relationship lead to a variety of approaches to improve the performance of N4SID. The similarities and differences between SIMs and LVMs are investigated, with emphases on their causality, data collection and applications. For estimating the states, CCA and RRA are shown to be more efficient than Principal Component Analysis (PCA) and Partial Least Squares (PLS).

A general statistical framework is proposed to unify SIM algorithms. The framework breaks all SIMs down into three common steps: 1) use of a linear regression method to estimate the predictable subspace, 2) use of a latent variable method to estimate a minimal set of the state variables, and 3) then fitting the estimated process states to the state-space model. Combining the approaches in the first two steps leads to a

whole set of new SIM algorithms. Simulation studies show that these new SIM algorithms have similar performance as the existing SIMs. This framework reveals the nature of the computation steps in SIM algorithms and the fundamental ideas behind SIM algorithms. It also discloses the relationships among different SIM algorithms.

The applicability of SIMs for closed-loop data is investigated. The original N4SID algorithm and the CVA algorithm based on regressing out the effects of future inputs are shown to give biased results. In general, whether a subspace identification algorithm is applicable for closed-loop data depends on how the effects of future inputs are treated in estimating the predictable subspace (step one of the proposed framework). Based on this analysis, several new N4SID and CVA algorithms are proposed for closed-loop data.

Practical issues arising from applications of SIMs are also discussed. SIMs are shown to be able to handle the delays, common dynamics, non-stationary and co-integrating disturbances in the process. The advantages, as well as solutions for possible problems, are also presented. Some general guidelines are provided for applications of SIMs.

Acknowledgment

I would like to express my greatest gratitude to my supervisor, Dr. John MacGregor, from the bottom of my heart. His continuous guidance and support have made this work possible. I got many heuristic ideas from discussions with Dr. MacGregor in my study at McMaster. It has been a great pleasure to work with him for the last few years. My gratitude also goes to the members of my supervisory committee, Dr. Paul Taylor and Dr. Gary Bone, for their helpful comments and suggestions. I appreciate the help from Dr. Thomas Marlin.

I would like to thank my colleagues in the Penthouse: Ali, Manish, San, Seongkyu, Carl, Tracy, Chunliang, Tong, Jean Francois, Ken, Jose, Jason, Song, Laura, Christiane, Angela, Steve, Francois and Athanassios, I am glad to have discussions with them and get help from them in both academic and non-academic aspects. I would like to thank Dora, Barb, Yaqiu, Sal, Niki, Honglu, Yongsong, Adam, Chris, Youngmin, Kevin, Pierre, Danielle, Li, Yingjun and many other friends for their help during my stay at McMaster.

I want to specially thank my mother and father for their sacrifices and encouragements. I credit my parents with where I am. I also want to thank my wife Hong, for her supports and efforts for the family.

Abbreviations and Acronyms

AIC	Akaike Information Criterion
ARX	Auto Regressive with eXogenous variables model
ARMAX	Auto Regressive Moving Average eXogenous variables model
CCA/CCR	Canonical Correlation Analysis/Regression
CCC	Canonical Correlation Coefficient
CSTR	Continuous Stirred Tank Reactor
CV	Canonical Variate
CVA	Canonical Variate Analysis
FDD	Fault Detection and Diagnosis
FIR	Finite Impulse Response
GSVD	Generalized Singular Value Decomposition
IV/IVM	Instrumental Variable/Instrumental Variable Method
LS	Least Square regression
LV	Latent Variable
LVM/LVR	Latent Variable Method/Regression
MIMO	Multiple-Input Multiple-Output system
MISO	Multiple-Input Single-Output system
ML	Maximum Likelihood function method
MOESP	Multivariable Output Error State sPace
MPC	Model Predictive Control
N4SID	Numerical Subspace State-Space System Identification
PC	Principal Component
PCA/PCR	Principal Component Analysis/Regression
PEM	Prediction Error method
PLS	Projection to Latent Structure/Partial Least Square
PRBS	Pseudo-random Binary Signal
QR	QR factorization
RRA/RRR	Reduced Rank Analysis/Regression
SNR	Signal to Noise Ratio
SIM	Subspace Identification Method
SIMO	Single-Input Multiple-Output system
SISO	Single-Input Single-Output system
SV	Singular Value
SVD	Singular Value Decomposition

Nomenclature

$\boldsymbol{A, B, C, D}$	System matrices in the state-space model
$\boldsymbol{B}$	Linear regression coefficient matrix
$C(z)$	Controller Transfer function
$\boldsymbol{D}_f, \boldsymbol{D}_p$	Future and past data matrix of d_k
$G(z)$	Dynamic Transfer function
$H(z)$	Disturbance model
$\boldsymbol{H}_f$	Toeplitz matrix containing f steps of impulse weights (refer to page 8)
$\boldsymbol{H}_{f\_e}$	Estimated $\boldsymbol{H}_f$ matrix
$\boldsymbol{H}_{f,s}$	Toeplitz matrix for the disturbance (refer to page 8)
$\boldsymbol{P}_{IO}$	Past input and output data matrix, i.e., $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$
$\boldsymbol{T}$	Latent variable matrix
$\boldsymbol{B}$	Linear regression coefficient matrix
$\boldsymbol{U}_p$	Past input data matrix (refer to page 7)
$\boldsymbol{U}_f$	Future input data matrix (refer to page 7)
$\boldsymbol{V}_f, \boldsymbol{V}_p$	Future and past data matrix of v_k
$\boldsymbol{W}_f, \boldsymbol{W}_p$	Future and past data matrix of w_k
$\boldsymbol{W}$	Coefficient matrix in LVMs
$\boldsymbol{X}_k$	State sequence
$\boldsymbol{X, Y, Z}$	General data matrices
$\boldsymbol{Y}_p$	Past output data matrix (refer to page 7)
$\boldsymbol{Y}_f$	Future output data matrix (refer to page 7)
$\boldsymbol{\Gamma}_f$	Extended observability matrix (f steps)
$\boldsymbol{\Omega}_f$	Extended controllability matrices (f steps)
d_k	Dither signals to a closed-loop system
f	Number of steps in future horizon
l	Number of output variables (dimension of y_k)
m	Number of input variables (dimension of u_k)
n	System order (dimension of x_k)
N	Number of collected data points
p	Number of steps in past horizon
T_s	Switching time period of a PRBS (clock period)
u_k	Inputs to a system
y_k	Outputs from a system
x_k	States of a system

w_k	Noise on system states x_k
v_k	Measurement noise on the outputs y_k

Subscript

p	Past time
f	Future time
s	Stochastic subsystem
_ro	Regression out (U_f)

Superscript

s	of stochastic subsystem
d	of deterministic subsystem
c	of controller
T	Transpose of a matrix
+	pseudoinverse of a matrix

Operator (MATLAB convention)

1:a	1st to a-th row or column vectors of a matrix
/	Projection
Y/X	Results of orthogonal projection of Y onto X
$Y/_Z X$	Results of oblique projection of Y along Z onto X
$[X; Y]$	Data matrix X stack on data matrix Y

Table of Contents

List of Figures

List of Tables

1 Introduction

This thesis will present fundamental and application research of subspace identification methods (SIMs) for process dynamic models. The research focuses on the basic principles, new algorithms, relationships among these methods and between other methods, as well as the issues arising from practical applications. This chapter introduces system identification and SIMs briefly, and presents the thesis objectives and outline.

1.1 Introduction to System Identification and SIMs

Process dynamic models are built to capture the transient behaviour of processes and have an important role in a wide range of engineering applications, such as process analysis, control system design, optimization and process monitoring. For the success of these applications, it usually requires great efforts to identify a process dynamic model. In practice with advanced control, system identification is the single most time-consuming task, and 80-90% of the implementation is for an adequate dynamic model (Andersen, et al., 1991).

There are two basic approaches for building the process dynamic model: the fundamental principle method (mechanistic or white-box models) and the system identification method (black-box models). The former approach needs a clear understanding of the nature of the process, therefore limiting its potential for practical applications. The system identification approach is to model dynamic systems from experimental data (Söderström and Stöica, 1989). The dynamic model employed in system identification is usually expressed in the discrete form of the transfer function or the state-space representation.

The typical system identification procedure is a set of iterative steps, including obtaining prior knowledge, design of experiment, data collection, choice of model

structure, parameter estimation and model validation. Choice of model structure and estimation of model parameters are the most mathematically involved steps in system identification, and these two steps are the major research subjects of system identification methods.

There are various methods to determine the model structure in traditional system identification methods, such as AIC (Akaike Information Criterion), *F*-test, and rank of the Hankel matrix. Prior knowledge of the process is often employed in model structure determination. Several methods have been widely used for parameter estimation, such as Prediction Error Method (PEM), Maximum Likelihood Method (ML), Instrumental Variable Method (IVM) and Two-step Method (e.g., COR-LS) (Ljung, 1999; Söderström and Stöica, 1989; Box and Jenkins, 1970).

PEM and ML are the best-known methods for parameter estimation. PEM determines the model parameters by minimizing the predicted error, while ML estimates parameters based on maximizing the likelihood function. In general, both methods result in a nonlinear optimization problem, which leads to iterative non-linear search procedures and may turn out local minima. In some special cases, PEM or ML may retrograde to simple Least Square (LS) regression, which guarantees the convergence and unique solution. In the case of collinear variables or close to collinear, Latent Variable Regression (LVR, refer to section 4.1) can be employed to overcome the ill-conditioning problem encountered in LS regression. In IVM, instrumental variables are constructed to be independent of the noise. This method allows one to avoid simultaneous identification of a noise model. The process and noise models can be estimated sequentially and iteratively. These traditional methods are useful for SISO systems; however, they become very involved when applied to MIMO systems.

As a class of system identification methods, subspace identification methods (SIMs) emerged in the 1990s and have drawn considerable interest. The key idea of SIMs is to estimate the process states or the extended observability matrix directly from the input and output data. The best-known methods are Canonical Variate Analysis (CVA, Larimore, 1990), Numerical Subspace State-Space System IDentification (N4SID, Van

Overschee and De Moor, 1994) and Multivariable Output Error State sPace (MOESP, Verhaegen and Dewilde, 1992). A brief overview of these methods will be presented in the next chapter.

In SIMs, the state-space model is employed, and the model structure is therefore determined by the system order, i.e. the number of states used in the model. The parameters in the state-space model are generally estimated by LS regression. Though the basic idea of SIMs is different from traditional system identification methods, many computation steps in SIMs are based on traditional system identification methods. Yet, SIMs show many advantages, especially for MIMO systems and in the aspect of computation.

1.2 Research Objectives and Thesis Outline

This thesis focuses on providing a deep and comprehensive understand of SIMs and solutions for application problems. The objectives of the thesis research mainly concentrate on developing the fundamental theory, exploring the relationships among SIMs and with other methods, proposing new algorithms as well as exploring issues related to practical applications.

The next chapter will be an overview of SIMs. The basic concepts, notations and developments of SIMs will be briefly introduced, and then a brief overview will be given for each of the major methods for a basic understanding of SIMs. Finally, the general characteristics of SIMs will be discussed in comparison with the traditional system identification methods.

Chapter 3 will analyze the major SIMs, provide fundamental proofs on the biasness issue, and propose some novel algorithms. In this chapter, a multi-step state-space model will be proposed first. Based on this model, each method will be analyzed to reveal its basic principles and nature of computational procedures. Several SIM

algorithms will be proposed. The fundamental analysis gains insights into SIMs, and provides heuristic ideas for the next two chapters.

In Chapter 4, the relationships between SIMs and LVMs will be explored. These two categories of methods have been studied separately in different fields; here their connections are disclosed. The similarities among and the differences between the resultant models from these methods will be explored to show their relationships clearly.

Chapter 5 will focus on the relationships among SIMs by unifying these methods in a general framework. This framework will catch the common ideas behind the distinct computation procedures and interpretations of SIMs and provide a comprehensive understanding of the nature of SIMs. This framework will also show clearly the connections and differences between SIMs and provide inspirations for new SIM algorithms.

In Chapter 6, the applicability of SIMs for the closed-loop data will be investigated in order to clarify disputes over this issue. The difference between closed-loop data and open loop data will be discussed first, and then SIM algorithms will be scrutinized for the closed-loop data with a focus on the bias issue. Several novel SIM algorithms will be proposed, particularly for closed-loop identification.

Chapter 7 will address some practical issues and general guidelines for application of SIMs. The applicability of SIMs will be analyzed for the presence of delays, common dynamics, non-stationary and co-integral disturbances in the process. This chapter will also explore for the possible solutions for these practical issues. The general guidelines will show the situations and applications where SIMs can show their potential advantages and avoid their limitations.

The last chapter will summarize the general conclusions of this research, and provide some comments on open issues for future research.

In general, Chapters 3 to 5 of this thesis focus on the fundamental research of SIMs, and some research results have been published in journals or at conferences. Chapters 6 and 7 focus on practical applications of SIMs and the results will be published in the near future.

2 Overview of Subspace Identification Methods

This chapter provides an overview of SIMs. The first section will cover the background as well as a literature overview, including notations and basic concepts. The subsequent three sections will provide a brief overview of CVA, N4SID and MOESP respectively. The last section will summarize the characteristics of SIMs.

2.1 Background and Literature Overview

Assume a linear deterministic-stochastic combined system of order n has m input variables and l output variables. The system can be represented in the following state-space model form:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{2.1.1}$$

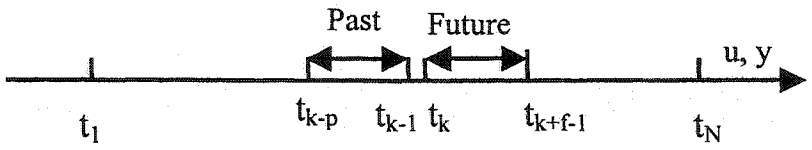
$$y_k = Cx_k + Du_k + Nw_k + v_k \tag{2.1.2}$$

$$E\left\{\begin{pmatrix} w_k \\ v_k \end{pmatrix} \begin{pmatrix} w_l^T & v_l^T \end{pmatrix}\right\} = \begin{pmatrix} Q & 0 \\ 0 & R \end{pmatrix}\delta_{kl} \geq 0$$

where outputs $y_k \in \mathfrak{R}^{l\times 1}$, inputs $u_k \in \mathfrak{R}^{m\times 1}$ and state variables $x_k \in \mathfrak{R}^{n\times 1}$. The process state noise $w_k \in \mathfrak{R}^{n\times 1}$ and measurement noise $v_k \in \mathfrak{R}^{l\times 1}$ are of corresponding dimensions and un-correlated with each other. Coefficient matrix N here shows the direct action of w_k on the outputs. The transfer function for the dynamic part of the system is $C(Iz-A)^{-1}B+D$. The impulse response weight is D at time k=0, and $CA^{i-1}B$ at k=i (i>0). These impulse response weights are also called Markov parameters.

A dynamic process is different from a static process in that the past inputs affect the future outputs, and there exists correlation between the past outputs and the future outputs. The past inputs and outputs are the necessary information to predict the future outputs. For an arbitrary time point deemed as current time k, all the past p steps of the input variables form a past input vector $\underline{\mathbf{u}}_p \in \mathfrak{R}^{mp\times 1}$. The current and all the future f-1 steps

of the input variables form a future input vector $\underline{\mathbf{u}}_f \in \Re^{mf\times 1}$. Similar symbols can be used to denote the lagged output variables (most SIM algorithms assume $p=f=i$ for convenience, but this is not necessary):



$$\underline{\mathbf{u}}_p = \begin{pmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-p+1} \\ u_{k-p} \end{pmatrix} \quad \underline{\mathbf{y}}_p = \begin{pmatrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-p+1} \\ y_{k-p} \end{pmatrix} \quad \underline{\mathbf{u}}_f = \begin{pmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ \vdots \\ u_{k+f-1} \end{pmatrix} \quad \underline{\mathbf{y}}_f = \begin{pmatrix} y_k \\ y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+f-1} \end{pmatrix} \tag{2.1.3}$$

These data vectors associate with one particular current time point k. Considering the end effect of past and future horizon for the collected data set, there are possible $N_e=N-f-p+1$ current time points (i.e., $k=p+1, p+2, \ldots, N-f+1$, N is the total number of data points collected). The collections of all corresponding vectors form the past/future input/output data matrices are: (e.g., $U_p \in \Re^{mp\times Ne}$ is the collection of all the possible $\underline{u}_p$)

$$Y_p = \begin{bmatrix} y_p & y_{p+1} & \cdots & y_{k-1} & \cdots & y_{N-f} \\ y_{p-1} & y_p & \cdots & y_{k-2} & \cdots & y_{N-f-1} \\ y_{p-2} & y_{p-1} & \cdots & y_{k-3} & \cdots & y_{N-f-2} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ y_1 & y_2 & \cdots & y_{k-p} & \cdots & y_{N-f-p+1} \end{bmatrix} \quad U_p = \begin{bmatrix} u_p & u_{p+1} & \cdots & u_{k-1} & \cdots & u_{N-f} \\ u_{p-1} & u_p & \cdots & u_{k-2} & \cdots & u_{N-f-1} \\ u_{p-2} & u_{p-1} & \cdots & u_{k-3} & \cdots & u_{N-f-2} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ u_1 & u_2 & \cdots & u_{k-p} & \cdots & u_{N-f-p+1} \end{bmatrix}$$

$$Y_f = \begin{bmatrix} y_{p+1} & y_{p+2} & \cdots & y_k & \cdots & y_{N-f+1} \\ y_{p+2} & y_{p+3} & \cdots & y_{k+1} & \cdots & y_{N-f+2} \\ y_{p+3} & y_{p+4} & \cdots & y_{k+2} & \cdots & y_{N-f+3} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ y_{p+f} & y_{p+f+1} & \cdots & y_{k+f-1} & \cdots & y_{N-f} \end{bmatrix} \quad U_f = \begin{bmatrix} u_{p+1} & u_{p+2} & \cdots & u_k & \cdots & u_{N-f+1} \\ u_{p+2} & u_{p+3} & \cdots & u_{k+1} & \cdots & u_{N-f+2} \\ u_{p+3} & u_{p+4} & \cdots & u_{k+2} & \cdots & u_{N-f+3} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ u_{p+f} & u_{p+f+1} & \cdots & u_{k+f-1} & \cdots & u_{N-f} \end{bmatrix}$$

(2.1.4)

Similarly, the state sequence X_k denotes the collection of all the possible "current" state vectors x_k, *i.e.*, $X_k=\{x_{p+1}, x_{p+2}, \ldots, x_k, \ldots x_{N-f+1}\}$. X_k^d and X_k^s are for the deterministic and stochastic state sequence respectively. Similar notation can be used for the stochastic signals w_k and v_k.

The state-space model (2.1.1) and (2.1.2) clearly shows the linear relationships between the input variables and the output variables. The same model form can be used for description of SISO (single input-single output) processes and MIMO (multiple input-multiple output) processes. The extended observability matrix Γ_i and extended controllability matrix Ω_i ($i>n$) are defined as: (both Γ_i and Ω_i are of rank n)

$$\Gamma_i = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{i-1} \end{pmatrix} \qquad \Omega_i = [B \;\; AB \;\; \ldots\ldots \; A^{i-2}B \;\; A^{i-1}B] \tag{2.1.5}$$

Hankel matrix of dimension m by n is defined by the system impulse response weights (Markov parameters) as follows:

$$H_{m,n} = \begin{pmatrix} CB & CAB & CA^3B & \cdots & CA^{n-1}B \\ CAB & CA^2B & CA^4B & \cdots & CA^nB \\ CA^2B & CA^3B & CA^5B & \cdots & CA^{n+1}B \\ \vdots & \vdots & \vdots & \cdots & \cdots \\ CA^{m-1}B & CA^iB & CA^{i+1}B & \cdots & CA^{m+n-2}B \end{pmatrix} \tag{2.1.6}$$

It is easy to verify that this Hankel matrix is a production of the extended observability matrix Γ_m and extended controllability matrix Ω_n, i.e.,

$$H_{m,n}=\Gamma_m\Omega_n \tag{2.1.7}$$

The so-called Trapezoid matrices for deterministic and stochastic subsystem are defined respectively as

$$H_i = \begin{pmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & 0 \\ CA^{i-2}B & CA^{i-3}B & CA^{i-4}B & \cdots & D \end{pmatrix} \quad H_{s,i} = \begin{pmatrix} N & 0 & 0 & \cdots & 0 \\ C & N & 0 & \cdots & 0 \\ CA & C & N & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & 0 \\ CA^{i-2} & CA^{i-3} & CA^{i-4} & \cdots & N \end{pmatrix} \quad (2.1.8)$$

or

$$H_i = \begin{pmatrix} w_0 & 0 & 0 & \cdots & 0 \\ w_1 & w_0 & 0 & \cdots & 0 \\ w_2 & w_1 & w_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & 0 \\ w_{i-1} & w_{i-2} & w_{i-3} & \cdots & w_0 \end{pmatrix} \quad (2.1.9)$$

Here w_0, w_1, w_2, ... are the impulse response of the dynamic system. $\boldsymbol{H}_i$ and $\boldsymbol{H}_{s,i}$ indicate the effects of future inputs and future noise on the future outputs respectively. All the above defined data matrices or special matrices are employed extensively in later analysis of SIM algorithms.

The Least Squares (LS) regression is also frequently used in SIMs. When variables in $\underline{y}$ depend linearly on a set of variables in $\underline{x}$ with independent noise $\underline{e}$ as:

$$\underline{y}=\beta\underline{x}+\underline{e} \quad (2.1.10)$$

The coefficient β between $\underline{y}$ and $\underline{x}$ can be estimated by LS regression based on the observations (recorded in columns of matrix $\boldsymbol{Y}$ and $\boldsymbol{X}$ respectively):

$$\hat{\beta} = YX^T\left(XX^T\right)^{-1} \quad (2.1.11)$$

The least square result is known as the best linear unbiased estimation (BLUE). The covariance matrix $\boldsymbol{XX}^{\mathrm{T}}$ must be of full rank in LS. The LS procedure may face the ill-conditioning problem if the variables in $\boldsymbol{X}$ are highly correlated or close to collinear (matrix $\boldsymbol{XX}^{\mathrm{T}}$ is close to singular). Under such a situation, the estimated parameters will have high variances. The part of $\boldsymbol{Y}$ that can be explained by $\boldsymbol{X}$ (i.e., $\hat{\beta}X$) is also called the projection of $\boldsymbol{Y}$ onto the $\boldsymbol{X}$ space, noted as simply as $\boldsymbol{Y/X}$. Besides the above approach of matrix inverse, the computation can also be realized by QR decomposition or Singular Value Decomposition (SVD).

If the above noise $\underline{e}$ is correlated with $\underline{x}$, the estimated coefficient and the projection result from LS are theoretically biased. If measurement errors occur with the variables in $\underline{x}$ (called error-in-variable), the estimated coefficient and the projection result by LS are also biased in theory.

Besides LS, other regression methods exist, e.g., PLS (partial Least Squares), RRR (Reduced Rank Regression) and CCR (Canonical Correlation Regression). They have different objectives from LS regression in building the relationship between Y and X variables. See Section 4.1 for details.

AIC (Akaike Information Criterion) is another concept often used in system identification. For single output case, AIC is defined as:

$$AIC(n)=N\log(\sigma^2)+2M_n \tag{2.1.12}$$

where σ^2 is the residual variance, N is the number of data points used, and M_n is the number of parameters used in n-th order model. For multiple output case,

$$AIC(n)=N\log(\det(\Sigma))+2M_n \tag{2.1.13}$$

where Σ is the residual covariance matrix. AIC is an optimal estimation of the Kullback discrimination information for a large data set. It is a combined index showing a trade-off between the model accuracy and the model complexity. Minimizing AIC can avoid underfitting or overfitting, and can achieve optimal decision on model order determination (Shibata, 1981). For small data sets, better order determination can be achieved by a modified AIC (Hurvich and Tsai, 1989):

$$AIC(n)=N\log(\det(\Sigma))+2fM_n \qquad f=N/(N-(M_n/n+(n+1)/2)) \tag{2.1.14}$$

In general, SIMs can be classified into two major categories: realization-based (basic) SIMs and direct SIMs (Viberg, 1995). Realization-based SIMs can be traced back to the 1960s. Ho and Kalman (1966) constructed the Hankel matrix based on the explicitly estimated impulse weights, then they performed an SVD on this Hankel matrix to get the extended observability matrix Γ, which can be used to estimate system matrices. More research has been carried out in recent years to extend this approach

(Ljung, 1991). Because of the difficulty in obtaining accurate impulse weights, this approach is not very attractive to researchers and practitioners.

Direct SIMs estimate the state sequence or extended observability matrix directly from the input-output data sets instead of impulse responses. The best-known methods in this category are CVA, MOESP and N4SID (refer to later sections for details). Van Overschee and De Moore (1995) proposed a unifying theorem of these three methods from the viewpoint of projection approximation. Some other methods are available, such as algorithms based on linear regression (Jasson and Wahlberg, 1992; Di Ruscio, 1995).

There are many different directions in the research of subspace identification. Some ongoing topics are: survey of SIMs (Viberg 1995); exploring new SIMs (e.g., Li and Qin, 2000); investing the relationship between the different algorithms; improving the accuracy of the identified model, especially for the case with high noise level; consistency properties of the methods (Deistler, et al., 1995. Bauer, et al., 1999); improving the computational performance (Cho and Kailath, 1995, Ewerbring, et al., 1990); comparative studies of algorithms (Favoreel, et al., 1999); extending the methods to unstable systems, Linear Time Variant (LTV) systems, nonlinear systems (Larimore and Baillieul, 1990) and systems with errors-in-variables (Chou and Verhaegen, 1997); applying the methods to the closed-loop situation (Verhaegen, 1993; Jha and Georgakis, 1996; Ljung and McKelvey, 1996); and fault detection (Wang, et al., 1997; Basseville, et al., 2000; Qin, 2000).

CVA (Larimore, 1990) performs a Canonical Correlation Analysis (CCA) on two data sets: a matrix of past input and output data and a matrix of future output data. Based on properties of a Markov process and the likelihood function, the dominant canonical variates of those two data sets are proven to be approximates of state variables (called "memory"). The system matrices are estimated by fitting the estimated states to the state-space model by LS regression. A brief overview of this method is available in the next section.

In N4SID (Van Overschee and De Moore, 1994), an oblique projection of future outputs along future inputs onto the past data is performed first, and then Singular Value Decomposition (SVD) on the projection result will give estimates of state variables. The system matrices are estimated by LS regression. A simple N4SID algorithm will be reviewed in section 2.3.

The original MOESP method (Verhaegen, 1992) is based on the extended observability matrix for the system matrices. It performs a QR factorization of the input and output data, then the extended observability matrix is estimated from a part of the coefficient matrix $\boldsymbol{R}$ by SVD. System matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ can be directly extracted from the estimated observability matrix; $\boldsymbol{B}$ and $\boldsymbol{D}$ are obtained by another LS regression. Section 2.4 will give a brief overview of this method.

2.2 Overview of CVA Method

CVA (Canonical Variate Analysis) was proposed by Larimore (1990), and a software package (Adaptix) based on this method is available for model identification. The key point in CVA is to estimate the state variables by Canonical Correlation Analysis (CCA). CCA was first developed by Hotelling (1936) as a statistical method to get the most correlated linear combinations (canonical variates) from two sets of variables (for more details, see Section 4.1).

A general deterministic-stochastic combination process can be described by the state-space model in the following form:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{2.2.1}$$

$$y_k = Cx_k + Du_k + Nw_k + v_k \tag{2.2.2}$$

Here w_k and v_k are uncorrelated stochastic signals. The term Nw_k in the output equation makes the model better to describe the colored noise.

In the first stage of CVA, a series of ARX(p) models with different orders (p=1, 2, …, L. L is the maximum lag steps) are obtained by fitting current outputs y_k against the past input and output data [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$]. p is the number of lag steps used in fitting the ARX model. The number of lag steps corresponding to the minimum AIC value is called OML (Optimal Memory length).

In the second stage, CCA is performed on two data sets: the OML steps of modified future outputs $\boldsymbol{Y}=\boldsymbol{Y}_f-\boldsymbol{Y}_f/\boldsymbol{U}_f$, i.e., regressing $\boldsymbol{U}_f$ out of $\boldsymbol{Y}_f$, and the OML steps of past input and output data ($\boldsymbol{X}$):

$$\boldsymbol{Y}=\mathbf{Y}_f \text{ or } \boldsymbol{Y}_f-\boldsymbol{Y}_f/\boldsymbol{U}_f \qquad\qquad \boldsymbol{X}=\boldsymbol{P}_{IO}=[\boldsymbol{Y}_p;\ \boldsymbol{U}_p] \tag{2.2.3}$$

The maximum possible number of canonical variates (CVs) from CCA is OML× l. For an n-th order system, the first n CVs are approximates of the system state variables (Larimore, 1997). In system identification, the system order n must be determined from the input and output data. Here AIC is employed again. Different numbers of CVs are used to get the one-step-ahead prediction error. The number of CVs corresponding to the minimum AIC is estimated as the system order (denoted as $\hat{n}$). So the state sequence is estimated as a linear combination of the past input and output data (J is the coefficient matrix for CVs from $\boldsymbol{P}_{IO}$, and $J_{\hat{n}}$ is the first $\hat{n}$ rows of J):

$$\hat{x}_k = m_k = J_{\hat{n}} P_{IO} = J_1 Y_p + J_2 U_p \tag{2.2.4}$$

In the third stage, the input data, output data and estimated states are fitted to the state-space model (2.2.1) and (2.2.2) to estimate system matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ by LS regression:

$$\begin{bmatrix} \hat{x}_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ u_k \end{bmatrix} + \begin{bmatrix} I_n & 0 \\ N & I_l \end{bmatrix} \begin{bmatrix} \hat{w}_k \\ \hat{v}_k \end{bmatrix} \tag{2.2.5}$$

Matrix N and estimates of w_k and v_k can be estimated based on the fitting residuals of the above equations.

CVA was originally developed by Larimore (1990), and interpreted in the maximum likelihood principle using conditional likelihood functions for a Markov process (here p is the likelihood function):

$p(\boldsymbol{Y}_f|(\boldsymbol{U}_f, \boldsymbol{U}_p, \boldsymbol{Y}_p)) = p((\boldsymbol{Y}_f|\boldsymbol{U}_f)|(\boldsymbol{U}_p, \boldsymbol{Y}_p)) = p(\boldsymbol{Y}_{f\_e}|\boldsymbol{P}_{IO}) = p(\boldsymbol{Y}_{f\_e}|\boldsymbol{X}_k)$

and $\boldsymbol{Y}_f|\boldsymbol{U}_f = \boldsymbol{Y}_{f\_e} = \boldsymbol{Y}_f - \boldsymbol{B}_1\boldsymbol{U}_f$, $\boldsymbol{B}_1 = \boldsymbol{Y}_f\boldsymbol{U}_f^T(\boldsymbol{U}_f\boldsymbol{U}_f^T)^{-1}$

maximizing the above likelihood function leads to that CVs from CCA are optimal estimates of process states in the sense of maximum likelihood. These CVs are linear combination of the past data $\boldsymbol{P}_{IO}=[\boldsymbol{Y}_p; \boldsymbol{U}_p]$, and have the minimum prediction error of $\boldsymbol{Y}_{f\_e}$ based on a special weighting matrix (inverse of the $\boldsymbol{Y}_{f\_e}$ covariance matrix).

In fact, the above procedures are correct only if the process inputs are not auto-correlated. If the inputs are auto-correlated or the data are collected under a closed-loop situation, regressing $\boldsymbol{U}_f$ out of $\boldsymbol{Y}_f$ in (2.2.3) will cause problem. Refer to Section 3.3 and Section 6.3 for a detailed discussion.

2.3 Overview of N4SID Method

Van Overschee and De Moore (1994) published N4SID method, and it has been developed as a MATLAB function in the System Identification Toolbox (Ljung, 1997). It employs oblique projection to get a special subspace, then uses SVD to determine the model order and estimate the state variables, and then estimates the system matrices by LS regression.

The model structure used in N4SID has the following form:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{2.3.1}$$

$$y_k = Cx_k + Du_k + v_k \tag{2.3.2}$$

$$E\left\{\begin{pmatrix} w_k \\ v_k \end{pmatrix} \begin{pmatrix} w_l^T & v_l^T \end{pmatrix}\right\} = \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix}\delta_{kl} \geq 0$$

and N4SID is based on the following assumptions:

- Both the noises v_k and w_k are of zero-mean, random Gaussian distributions
- Both noises are independent of the input u_k

- The input is persistently exciting greater than order $2i$ (i is the number of lagging steps for past and future horizons, i.e., $p=f=i$)
- The number of data points (N) tends to infinity.

A simple N4SID algorithm (called approximation algorithm) includes the following major procedures:

(1) Oblique projection

The first step of N4SID is to get the oblique projection of $\boldsymbol{Y}_\mathrm{f}$ along $\boldsymbol{U}_\mathrm{f}$ onto [$\boldsymbol{Y}_\mathrm{p}$; $\boldsymbol{U}_\mathrm{p}$], denoted as $Y_f \Big/ {}_{U_f}\begin{bmatrix} U_p \\ Y_p \end{bmatrix}$. This can be realized by regressing $\boldsymbol{Y}_\mathrm{f}$ onto [$\boldsymbol{Y}_\mathrm{p}$; $\boldsymbol{U}_\mathrm{p}$; $\boldsymbol{U}_\mathrm{f}$] first

$$Z_f = Y_f / \begin{bmatrix} U_p \\ U_f \\ Y_p \end{bmatrix} = L_1 Y_p + L_2 U_p + L_3 U_f \tag{2.3.4}$$

and then eliminating the effect of the future inputs $\boldsymbol{U}_\mathrm{f}$:

$$Y_f \Big/ {}_{U_f}\begin{bmatrix} U_p \\ Y_p \end{bmatrix} = Z_i - L_3 U_f = L_1 Y_p + L_2 U_p \approx \Gamma_i \hat{X}_i \tag{2.3.5}$$

The LS projection result $\boldsymbol{Z}_\mathrm{f}$ has the minimum total prediction error of $\boldsymbol{Y}_\mathrm{f}$ based on [$\boldsymbol{Y}_\mathrm{p}$; $\boldsymbol{U}_\mathrm{p}$; $\boldsymbol{U}_\mathrm{f}$]. The oblique projection result, $L_1 Y_p + L_2 U_p$, has the minimum total prediction error of $\boldsymbol{Y}_\mathrm{f}$ based on [$\boldsymbol{Y}_\mathrm{p}$; $\boldsymbol{U}_\mathrm{p}$] after considering the effects of $\boldsymbol{U}_\mathrm{f}$. This oblique projection is shown to be an approximate of $\Gamma_i \hat{X}_i$, where $\hat{X}_i$ is interpreted as the result of a series of non-steady state Kalman filter over the past i steps. The approximation comes from assuming $\boldsymbol{L}_3 \approx \boldsymbol{H}_\mathrm{i}$.

(2) SVD on subspace $L_1 Y_p + L_2 U_p$

The relationship of (2.3.5) shows that the rank of $L_1 Y_p + L_2 U_p$ should be the system order. Because of the estimation error, all the singular values of $L_1 Y_p + L_2 U_p$ are non-zero in practice, therefore the system order is determined by the number of dominant singular values (diagonal values of $\boldsymbol{S}_1$ below).

$$L_1 Y_p + L_2 U_p = USV^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \approx U_1 S_1^{1/2} \cdot S_1^{1/2} V_1^T = \Gamma_i \cdot \hat{X}_k \qquad (2.3.6)$$

where Γ_i is estimated as (different ways to partition S_1 lead to similar results):

$$\Gamma_i = U_1 S_1^{1/2}$$

and the states are estimated as:

$$\hat{X}_k = S_1^{1/2} V_1^T = \Gamma_i^+ (L_1 Y_p + L_2 U_p) = L_1^N Y_p + L_2^N U_p \qquad (2.3.7)$$

where Γ_i^+ is the pseudo-inverse of Γ_i.

(3) Estimate ***A***, ***B***, ***C***, ***D*** and w_k, v_k

The same LS regression and similar procedures are employed in N4SID as in CVA to estimate the system matrices and noise covariance matrix.

These are the main steps of N4SID method. The above estimated state sequence is based on the approximation of $L_3 \approx H_i$ in (2.3.5), and this algorithm is biased for general case. An unbiased N4SID algorithm is available by using unknown H_i instead of L_3 in (2.3.5) to estimate X_k. Similarly, X_{k+1} is estimated by taking time point k+1 as the current time point (i.e., past horizon of i+1 steps and future horizon of i-1 steps). Since both estimated states involve the unknown ***B*** and ***D*** in H_i, another LS regression is required to estimate ***B*** and ***D*** (refer to Van Overschee and De Moore, 1994, 1996).

Ljung and McKelvey (1996) explained N4SID from the viewpoint of infinite impulse response. It was pointed out that the subspace $L_1 Y_p + L_2 U_p$ is the multiple-step-ahead predictions of the future outputs. Based on this viewpoint, Ljung (1999) gave a more general explanation of the basic idea behind N4SID.

2.4 Overview of MOESP Method

MOESP (Multivariable Output-Error State Space) method was originally proposed by Verhaegen and Dewilde (1992). It employs a QR decomposition to factorize

the joint input and output data matrices into a triangular coefficient matrix $\boldsymbol{R}$ and an orthonormal signal matrix $\boldsymbol{Q}$. Γ_f is estimated from one part of $\boldsymbol{R}$ directly by SVD. System matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ are obtained from Γ_f, and $\boldsymbol{B}$ and $\boldsymbol{D}$ are obtained by another LS regression.

The model structure used in MOESP is:

$$\begin{aligned} x_{k+1} &= \mathrm{A}x_k + Bu_k \\ y_k &= Cx_k + Du_k + v_k \end{aligned} \tag{2.4.1}$$

Here noise v_k can be a colored noise, which is assumed to be independent of the input.

Future input data $\boldsymbol{U}_f$ and output data $\boldsymbol{Y}_f$ can be factorized in the QR form:

$$\begin{bmatrix} U_f \\ Y_f \end{bmatrix} = RQ = \begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \tag{2.4.2}$$

Verhaegen and Dewilde (1992) showed that Γ_f is a factor matrix of $\boldsymbol{R}_{22}$. The number of dominant singular values of $\boldsymbol{R}_{22}$ (dimension of $\boldsymbol{S}_1$ in (2.4.3)) will determine the system order, and the corresponding matrix $\boldsymbol{U}_1$ (left singular vectors) is taken as Γ_f:

$$R_{22} = USV^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \tag{2.4.3}$$

$$\Gamma_i = U_1$$

The system matrix $\boldsymbol{C}$ is taken directly from the first l rows of the estimated Γ_i (l is the number of outputs) and $\boldsymbol{A}$ is obtained based on the shift invariant property of matrix Γ_i: (using MATLAB notation)

$$\boldsymbol{C}=\Gamma_i(1{:}l, :)$$

$$\Gamma_{f-1} A = \overline{\Gamma}_f \quad \text{and} \quad A = \Gamma_{f-1}^{+} \overline{\Gamma}_f \tag{2.4.4}$$

Here $\overline{\Gamma}_f$ is matrix Γ_i but without the first l rows. System matrices $\boldsymbol{B}$ and $\boldsymbol{D}$ are determined by LS regression of equations involving $\boldsymbol{U}_2$, $\boldsymbol{R}_{21}$ and $\boldsymbol{R}_{11}$ (for details, refer to Verhaegen and Dewilde, 1992).

A more practical algorithm (PO-MOESP) is to perform QR decomposition on [$\boldsymbol{U}_f$; $\boldsymbol{U}_p$; $\boldsymbol{Y}_p$; $\boldsymbol{Y}_f$], and system matrices are extracted from the $\boldsymbol{R}$ matrix. In this algorithm, the

past input and output data sets are used as instrumental variables to reduce the effects of noise on the estimated parameters (Verhaegen, 1994).

2.5 Characteristics of SIMs

Compared with the traditional identification methods, SIMs show some distinguishable characteristics in both the principle and the computation:

- Based on the state-space representation, only the system order is necessary to be estimated in determination of the model structure. The system order is determined by the rank of a estimated subspace (or the number of dominant singular values) before parameter estimation; therefore no iteration is needed for model structure determination.
- These SIM algorithms do not involve nonlinear optimization in parameter estimation, thereby avoiding the problems associating with nonlinear optimization, such as divergence, local minima and iterative search. The computation in SIMs mainly includes LS regression and SVD, which are numerically reliable and with predictable computation load.
- With only increase of matrix dimension, SIM algorithms can be applied for MIMO systems as simple and elegant as for SISO system.
- The resultant models from SIMs are generally very close to the global optimal result (such as result from PEM), but they are not identical in general. This is due to the different objective functions and the reduction of the general nonlinear search problem to a series of linear regressions.

These characteristics give SIMs many advantages in system identification, especially for MIMO systems and on-line applications, such as model predictive control and adaptive control.

3 Analysis of SIMs

In this chapter, a multi-step state-space model will be proposed first as a systematic frame to reveal the relationships among the process data. Based on this model, each SIM will be analyzed with emphases on revealing the basic principles and the nature of computational procedures. Several new SIM algorithms will be proposed during the analysis. The fundamental analysis in this chapter gains insights into SIMs and provides heuristic ideas for the next two chapters.

3.1 Multi-step State-space Model

From the overview of SIMs in the last chapter, one may have noticed that each SIM is developed and interpreted with different principles. This makes SIMs hard to understand and cuts the links between these methods. In this section, a multi-step state-space model is proposed to disclose the relationships between the states and the past/future inputs and outputs. This model provides a systematic basis for the analysis of SIMs in later sections.

Use of past and future process inputs and outputs is necessary to capture the process dynamics. On one hand, as in a discrete transfer function, the process dynamics is shown as an expression of the current outputs in terms of the past inputs and outputs. Here the past inputs and outputs are the driving force of the process momentum (states). On the other hand, the process momentum (states) will evolve over a future horizon, and the effect will show in the future outputs. Therefore, the dynamic characteristics of the process are also shown in the future outputs. It is natural to use the past data and the future outputs in building the process dynamic model. Since the future outputs include the effects of the future inputs, it is also necessary to include the future inputs. Furthermore, the past inputs and outputs far away from current time point have little effect on the

current states, and the current states have little effect on the outputs far away in the future. From the signal-to-noise ratio (SNR) point of view, using a finite number of steps of past/future input and output variables is efficient to capture the process dynamics. This is the basis and motivation for the multi-step state-space model.

As we know, a process model can be expressed in the following state space representation:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{3.1.1}$$

$$y_k = Cx_k + Du_k + Nw_k + v_k \tag{3.1.2}$$

$$E\left\{\begin{pmatrix} w_k \\ v_k \end{pmatrix} \begin{pmatrix} w_l^T & v_l^T \end{pmatrix}\right\} = \begin{pmatrix} Q & 0 \\ 0 & R \end{pmatrix}\delta_{kl} \geq 0$$

Here outputs $y_k \in \Re^{l\times 1}$, inputs $u_k \in \Re^{m\times 1}$, state variables $x_k \in \Re^{n\times 1}$. Process state noise $w_k \in \Re^{n\times 1}$ and measurement noise $v_k \in \Re^{l\times 1}$ are random noise signals (stochastic driving signals). Coefficient matrix N shows the direct action of w_k on the outputs.

Take an arbitrary time point k as the "current" time point, and consider the past horizon of p steps and the future horizon of f steps. For the past horizon, k-p is the initial time point, and x_{k-p} is the initial state vector. If equation (3.1.1) is recursively used for the time points in the past horizon, it is easy to get the state vectors in the past horizon:

$$x_{k-p+i} = A^i x_{k-p} + \Omega_i \underline{\mathbf{u}}_p + \Omega_{s,i} \underline{w}_p \quad (i=1, 2, \ldots, p) \tag{3.1.3}$$

and the state vector for time point k is represented in the multiple step of the past data as:

$$x_k = A^p x_{k-p} + \Omega_p \underline{\mathbf{u}}_p + \Omega_{s,p} \underline{w}_p$$

If N data points are collected, the maximum number of possible "current" time points will be N_e=N-p-f+1. Using the data matrix notations defined in section 2.1, the multi-step state-space model for current state sequence X_k is:

$$X_k = A^p X_{k-p} + \Omega_p U_p + \Omega_{s,p} W_p \tag{3.1.4}$$

If all the state vectors in past horizon (i=1, 2, …, p-1 in (3.1.3)) are used for the past outputs based on (3.1.2), the multi-step state-space model for the past outputs is:

$$Y_p = \Gamma_p X_{k-p} + H_p U_p + H_{s,p} W_p + V_p \tag{3.1.5}$$

Equation (3.1.4) and (3.1.5) show how the current state sequence and past outputs relate to the initial state vector and the past inputs. From equation (3.1.5), the initial state sequence X_{k-p} can be expressed in terms of past inputs and outputs (Γ_p^+ is the pseudo-inverse of Γ_p):

$$X_{k-p} = \Gamma_p^+ Y_p - \Gamma_p^+ H_p U_p - \Gamma_p^+ H_{s,p} W_p - \Gamma_p^+ V_p$$

Substituting this expression into (3.1.4), the result shows the relationship between the current state sequence and the past input and output data:

$$X_k = A^p \Gamma_p^+ Y_p + \left(\Omega_p - A^p \Gamma_p^+ H_p\right) U_p + \left(\Omega_{s,p} - A^p \Gamma_p^+ H_{s,p}\right) W_p - A^p \Gamma_p^+ V_p \qquad (3.1.6)$$

That is, the current state sequence X_k is a linear combination of the past inputs, outputs, and the stochastic signals. If these coefficient matrices in (3.1.6) are known or estimated, the current state sequence can be estimated based on the past inputs and outputs.

Equations (3.1.4), (3.1.5) and (3.1.6) show the relationships among current states and the past data. On the other hand, similar relationships exist in the future horizon.

From state-space model (3.1.1) and (3.1.2), the following relationship between the current state sequence and the future outputs can be obtained (i.e., the multi-step state-space model for the future outputs):

$$Y_f = \Gamma_f X_k + H_f U_f + H_{s,f} W_f + V_f \qquad (3.1.7)$$

It clearly shows that the future outputs as a whole consist of three components: the effects of the current states (1st term on right-hand side of the equation), the effects of the future inputs (2nd term), and the effects of the future stochastic signals (3rd and 4th term). The first component is the free evolution of the process outputs (that is, the evolution that would occur without any future inputs or noise to the process). It is a product of the observability matrix and the process state sequence (key terms to be estimated in SIMs). This component includes both the deterministic state effects $\Gamma_f X_k^d$ and the stochastic state effects $\Gamma_f X_k^s$. Although the numerical values of both Γ_f and X_k depend on the choice of state basis, the value of $\Gamma_f X_k$ is independent of this choice and is an invariant nature of the process. The second component of the future outputs comes from the action of future

inputs, and it is determined by coefficient matrix H_f and the known data U_f. The third component comes from the future noise signals, and is un-correlated to the previous two components.

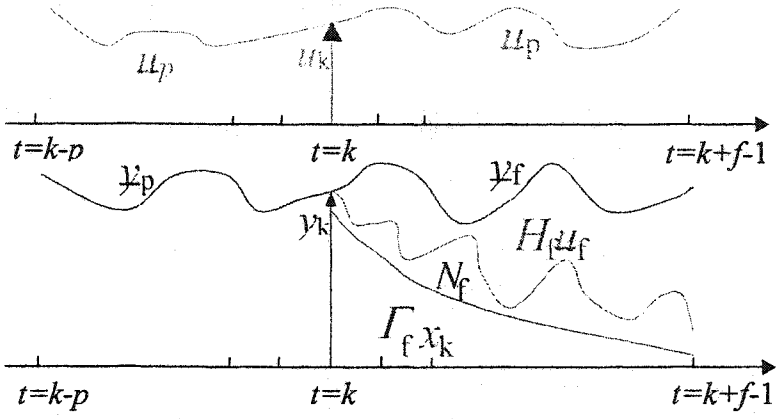


Figure 3.1 Three components of the future outputs

Process states are generically defined as "the minimum amount of information about the past history of a system which is required to predict the future motion" (Aström, 1970), the linear combination on the right-hand side of equation (3.1.6) summarizes the necessary information in the past history to predict the future outputs Y_f. This linear combination of the past inputs and outputs gives the best estimation of the current states in (3.1.7) in the sense of least squares or Kalman filter. If the future outputs Y_f in (3.1.7) is scrutinized from the viewpoint of prediction, the first component $\Gamma_f X_k$ can possibly be predicted based on the past data via the current states in (3.1.6). Equation (3.1.6) extracts the information in past data $[Y_p; U_p] \in \Re^{p(m+l)\times Ne}$ and concentrates it into the current state sequence X_k. In other words, the useful information in the data space of dimension $p(l+m)$ shrinks to a subspace of dimension n, which is then used to predict the subspace of $\Gamma_f X_k$. Therefore, $\Gamma_f X_k$ is called the predictable subspace of Y_f thereafter. The true predictable subspace is only of rank n (both Γ_f and X_k are of rank n), though the dimensions of the subspace may be much higher. In SIM algorithms, this subspace is the

basis to estimate the state sequence X_k or the observability matrix Γ_f. Based on (3.1.6), the true predictable subspace is:

$$\begin{aligned}\Gamma_f X_k = {} & \Gamma_f A^p \Gamma_p^+ Y_p + \Gamma_f \left(\Omega_p - A^p \Gamma_p^+ H_p\right) U_p \\ & + \Gamma_f \left(\Omega_{s,p} - A^p \Gamma_p^+ H_{s,p}\right) W_p - \Gamma_f A^p \Gamma_p^+ V_p\end{aligned} \tag{3.1.8}$$

This indicates that the predictable subspace can be estimated as a linear combination of the past input and output data if the coefficient matrices are known or estimated.

The second component $\boldsymbol{H_f U_f}$ in (3.1.7) is not predictable in the sense of causality. This component may be estimated partially based on the correlation between the future inputs $\boldsymbol{U_f}$ and the past inputs $\boldsymbol{U_p}$ if the inputs are auto-correlated. However, it comes from the correlation structure in the input signals, not a causal effect of the process inputs, therefore not the goal of system identification (more discussion on causal models and correlation models is available in the next chapter). Based on this reason, the second component should be eliminated away from $\boldsymbol{Y_f}$ in system identification to avoid confounding between the causal relationship and the correlation relationship. The third component in (3.1.7) involves future stochastic terms. These terms are unpredictable based on the past data. Note, these terms are only the unpredictable part of the future disturbances; the predictable part $\Gamma_f X_k^s$ is included in the first component. The later two components should be moved away from $\boldsymbol{Y_f}$ in estimation of the predictable subspace.

By substituting (3.1.6) into (3.1.7), a linear relationship between the future outputs, the past data and the future inputs is obtained:

$$\begin{aligned}Y_f = {} & \Gamma_f A^p \Gamma_p^+ Y_p + \Gamma_f \left(\Omega_p - A^p \Gamma_p^+ H_p\right) U_p + H_f U_f \\ & + \Gamma_f \left(\Omega_{s,p} - A^p \Gamma_p^+ H_{s,p}\right) W_p - \Gamma_f A^p \Gamma_p^+ V_p + H_{s,f} W_f + V_f\end{aligned} \tag{3.1.9}$$

All the past terms (term 1, 2, 4 and 5 on the right-hand sight) are for the predictable subspace $\Gamma_f X_k$. This relationship is very useful and gives the possibility of estimating the coefficient matrices by linear regression or other techniques. It is worth noticing that the true coefficient matrices for $\boldsymbol{Y_p}$ and $\boldsymbol{U_p}$ in (3.1.9) are the same as those in (3.1.8), and their estimated values can be used to estimate the predictable subspace.

The predictable subspace showing in (3.1.8) or (3.1.9) is essentially the multi-step ahead predictions based on the past inputs and outputs with no future inputs (in Ljung, 1999, the multi-step ahead predictions, called *k*-step-ahead predictions, include the future input effects). Multi-step ahead predictions have been widely used in prediction error method (PEM) for future forecasting. Some algorithms in the family of PEM also use the weighted multiple prediction errors for identification purposes (e.g., weighted *k*-step-ahead PE parametric ID, Kozub, 1994). Normally in PEM, the prediction errors are weighted by the inverse of their covariance matrix. In CVA methods, the prediction errors of the future outputs are weighted with the inverse of the covariance matrix of future outputs (Larimore, 1990). In N4ISD, only the total sum of the prediction errors is considered, that is, the weighting matrix is an identity matrix.

3.2 Analysis of N4SID

The original N4SID algorithm was proposed by Van Overschee and De Moor (1994) based on an analogy to a series of non-steady state Kalman filters. In that approach, the algorithm was solely expressed in a mathematical manner that the nature and the essential ideas behinds the computation could not be shown explicitly. Here, based on the multi-step state-space model, the N4SID algorithm and some novel variations are analyzed with emphases on the basic principles and the bias issues.

3.2.1 Analysis of N4SID

The first step of N4SID is to perform an oblique projection of $\boldsymbol{Y}_f$ along $\boldsymbol{U}_f$ onto $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$, i.e., the part corresponding to $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ in the projection of $\boldsymbol{Y}_f$ onto $[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{U}_f]$. In fact, the oblique projection result is an estimate of the predictable subspace $\Gamma_f X_k$, and this point as well as the bias becomes clear in analysis of N4SID based on the multi-step

state-space model. The future outputs are shown to be a linear combination of the past data and the future input data as:

$$\begin{aligned} Y_f &= \Gamma_f A^p \Gamma_p^{+} Y_p + \Gamma_f \left(\Omega_p - A^p \Gamma_p^{+} H_p\right) U_p + H_f U_f \\ &\quad + \Gamma_f \left(\Omega_{s,p} - A^p \Gamma_p^{+} H_{s,p}\right) W_p - \Gamma_f A^p \Gamma_p^{+} V_p + H_{s,f} W_f + V_f \end{aligned} \tag{3.1.9}$$

Here the effects of past data (including not only terms of $\boldsymbol{Y}_p$ and $\boldsymbol{U}_p$, but also terms of $\boldsymbol{W}_p$ and $\boldsymbol{V}_p$) as a whole represent the true value of the predictable subspace $\Gamma_f X_k$ (see (3.1.8)). All the past/future input and output data in (3.1.9) are known. The effects of stochastic signals are unknown and act as noise term for the linear regression. Projection of $\boldsymbol{Y}_f$ onto $[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{U}_f]$ by LS regression is based on (3.1.9), which can be expressed briefly:

$$Y_f = C_1 Y_p + C_2 U_p + C_3 U_f + E \tag{3.2.1}$$

where $\boldsymbol{C}_1$, $\boldsymbol{C}_2$ and $\boldsymbol{C}_3$ ($\boldsymbol{C}_3=\boldsymbol{H}_f$) are coefficient matrices for $\boldsymbol{Y}_p$, $\boldsymbol{U}_p$ and $\boldsymbol{U}_f$ in (3.1.9) respectively, and the stochastic effects $\boldsymbol{E}=\boldsymbol{E}_p+\boldsymbol{E}_f$ include the past stochastic effects $E_p = \Gamma_f(\Omega_{s,p} - A^p \Gamma_p^{+} H_{s,p}) W_p - \Gamma_f A^p \Gamma_p^{+} V_p$ and the future stochastic effects $E_f = H_{s,p} W_f + V_f$. This model is different from the standard LS regression showing in (2.1.10). Here the stochastic effects $\boldsymbol{E}=\boldsymbol{E}_p+\boldsymbol{E}_f$ are correlated with the regressor $[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{U}_f]$ because of the correlation between $\boldsymbol{E}_p$ and $\boldsymbol{Y}_p$ (both include the effects of $\boldsymbol{W}_p$ and $\boldsymbol{V}_p$, refer to (3.1.5)).

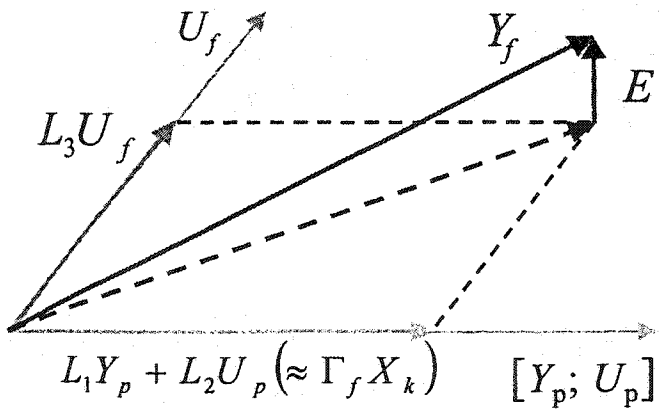


Figure 3.2 Geometric explanation of the projection in N4SID

The resultant coefficient matrices from LS regression in N4SID are:

$$\begin{bmatrix} L_1 & L_2 & L_3 \end{bmatrix} = Y_f X^T \left(XX^T\right)^{-1} \qquad X = \begin{bmatrix} Y_p; & U_p; & U_f \end{bmatrix} \tag{3.2.2}$$

and the detailed value for each coefficient matrix is: (see Appendix A3.1)

$$\begin{aligned} L_1 &= C_1 + EY_p^T \varphi^{-1} \\ L_2 &= C_2 - EY_p^T \varphi^{-1} R_p \\ L_3 &= C_3 - EY_p^T \varphi^{-1} R_f \end{aligned} \tag{3.2.3}$$

Here $\varphi = Y_p Y_p^T - Y_p U_{pf}^T \left(U_{pf} U_{pf}^T\right)^{-1} U_{pf} Y_p^T$, $\begin{bmatrix} R_p & R_f \end{bmatrix} = Y_p U_{pf}^T \left(U_{pf} U_{pf}^T\right)^{-1}$ and $U_{pf} = \begin{bmatrix} U_p; & U_f \end{bmatrix}$. In general, the correlation between $\boldsymbol{Y}_\mathrm{p}$ and $\boldsymbol{E}_\mathrm{p}$ leads to all the estimated coefficients in (3.2.3) biased from the true values. The second terms in (3.2.3) are bias. In the special case of a process with ARX model, future outputs can be expressed without involving $\boldsymbol{E}_\mathrm{p}$, therefore the estimated coefficients in (3.2.3) are unbiased (see Appendix A3.1).

Based on the LS regression result in (3.2.3), the projection of $\boldsymbol{Y}_\mathrm{f}$ onto $[\boldsymbol{Y}_\mathrm{p}; \boldsymbol{U}_\mathrm{p}; \boldsymbol{U}_\mathrm{f}]$ is: (noted as $\boldsymbol{Z}_\mathrm{f}$ in N4SID)

$$\hat{Y}_f = Z_f = L_1 Y_p + L_2 U_p + L_3 U_f \tag{3.2.4}$$

In the above projection result, the contribution from the past data sets (first two terms) is the result of oblique projection of $\boldsymbol{Y}_\mathrm{f}$ along $\boldsymbol{U}_\mathrm{f}$ onto $[\boldsymbol{Y}_\mathrm{p}; \boldsymbol{U}_\mathrm{p}]$:

$$\hat{Y}_{f|p} = L_1 Y_p + L_2 U_p \tag{3.2.5}$$

Incorporate the regression results in (3.2.3) into (3.2.5), and the result will be:

$$\begin{aligned} \hat{Y}_{f|p} &= L_1 Y_p + L_2 U_p \\ &= \left(C_1 + EY_p^T \varphi^{-1}\right) Y_p + \left(C_2 - EY_p^T \varphi^{-1} R_p\right) U_p \\ &= C_1 Y_p + C_2 U_p + EY_p^T \varphi^{-1} Y_p - EY_p^T \varphi^{-1} R_p U_p \\ &= \Gamma_f X_k - E_p + EY_p^T \varphi^{-1} \left(Y_p - R_p U_p\right) \end{aligned} \tag{3.2.6}$$

In general, this result is a close approximation of the predictable subspace $\Gamma_\mathrm{f} X_\mathrm{k}$. The first terms of $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ in (3.2.3) are the true coefficient matrices for $\boldsymbol{Y}_\mathrm{p}$ and $\boldsymbol{U}_\mathrm{p}$, which lead to an accurate estimate of the predictable subspace from the past data. However, the second

terms of L_1 and L_2 in (3.2.3) cause bias for the estimated predictable subspace. The expectation of the bias is:

$$\begin{aligned} Exp\left\{\hat{Y}_{f|p} - \Gamma_f X_k\right\} &= Exp\left\{-E_p + EY_p^T \varphi^{-1}\left(Y_p - R_p U_p\right)\right\} \\ &= Exp\left\{E_p Y_p^T \varphi^{-1}\left(Y_p - R_p U_p\right)\right\} \end{aligned} \tag{3.2.7}$$

If the true process is of ARX (or ARARX) model, the estimated coefficients in (3.2.6) are unbiased, and the estimated predictable subspace is therefore unbiased (E_p in (3.2.7) is null). That is, the oblique projection $L_1Y_p+L_2U_p$ in N4SID is an unbiased estimate of the predictable subspace. For a general process of Box-Jenkins model form (other than ARX model), the oblique projection $L_1Y_p+L_2U_p$ in N4SID is a biased estimate of the predictable subspace as shown in (3.2.7). Though the result is biased, the estimation error of E_p is partially canceled by the biased terms from Y_p and U_p. This bias in fact decreases the estimation error of the predictable subspace in the sense of variance.

The severity of the bias depends on correlation between Y_p and past stochastic signals as well as the condition number of data set [Y_p; U_p; U_f]. As is well known, a general Box-Jenkins model can be well approximated with a high order ARX model (a long past horizon), and the result is asymptotically unbiased if the number of steps in the past horizon tends to infinity. However, if the past horizon is too long, the co-linearity within [Y_p; U_p; U_f] becomes more severe (the condition number becomes larger and close to an ill-conditioning situation), and this will increase the variance of the results. The choice of the past horizon length becomes a trade-off between the bias and the variance, and AIC can be employed for determination of the optimal past horizon length. In most practical cases, the noise does not dominate the process, and the correlation between the past stochastic signals and the past outputs is small. Therefore, the oblique projection is a close estimate of the true predictable subspace.

In general, the coefficient matrix L_3 is a biased estimate of H_f but it is a close approximation of H_f. The resultant L_3 is a full matrix and is not consistent to the futures of H_f, such as block diagonal matrix, equal entries on the block diagonals as shown in (2.1.8). Y_f-L_3U_f is a biased estimate but a close approximate of the predictable subspace.

In the paper where N4SID was proposed (Van Overschee and De moor, 1994), the relationship between $L_1Y_p+L_2U_p$ and $\Gamma_f X_k$ was not revealed, and the bias issue was not clearly addressed. In fact, Z_f-H_fU_f is a biased estimate of the predictable subspace because of the biased terms in the estimated coefficient matrices in (3.2.3). The estimated state sequence $\Gamma_f^+(Z_f$-$H_fU_f)$ (analogous to results of a non-steady state Kalman filter over the past horizon) is a biased estimate of true state sequence for a general process due to the bias in the estimated initial states. The LS regression used to retrieve system matrices B and D in the unbiased N4SID algorithm is to correct the bias effects in the estimated states.

Another N4SID algorithm (called robust algorithm) was proposed by in Van Overschee and De Moor (1996). In this algorithm, the oblique projection result (3.2.5) is further projected onto the orthogonal space of U_f (regressing U_f out) before performing SVD for the estimate of Γ_f. Equation (3.2.6) indicates that regressing U_f out in fact deteriorates the SNR (the predictable subspace vs. the estimation error). Process states are still estimated by $\Gamma_f^+(Z_f$-$H_fU_f)$. System matrices B and D are parameterized in the state-space model, and estimated by a different LS regression, which might improve the final estimated model.

The true predictable subspace is a product of Γ_f and X_k, and its rank equals the system order n. Therefore, in N4SID, SVD (PCA) is applied on the estimated predictable subspace $L_1Y_p+L_2U_p$. The number of the dominant singular values is taken as the system order, and the left singular matrix and the right singular matrix are used to estimate Γ_f and X_k respectively (refer to Section 5.3 for more discussion on state estimation).

3.2.2 New N4SID Algorithms

N4SID-ARX algorithm

In the analysis of N4SID for closed-loop data, Ljung (1996, refer to Chapter 6) suggested a modified N4SID algorithm for closed-loop case in order to avoid the bias in

the result. In fact, this modified N4SID algorithm is also applicable to the open loop case. The major steps in this algorithm are as following (denoted as N4SID_ARX thereafter):

- Fit a high-order ARX model based on the current outputs y_k and past input and output data $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$, and form a one-step-ahead prediction model in ARX model form.
- Recursively use the one-step-ahead prediction model for the predictions of the multiple step outputs in the future horizon with future inputs to be null.
- Then, as in N4SID, use SVD to estimate the system order and the state sequence, and then fit the estimated states to the state-space model for system matrices.

Fitting y_k against $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ is a special case of the least square regression in (3.1.9) with only y_k considered, and the resultant ARX model is only related to the part of coefficient matrices in (3.2.3) corresponding to y_k. Since a long past horizon is usually used, the result is a high-order ARX model (a high-order vector ARX model for MIMO case). This ARX model is an unbiased estimate of the true process model (ARX model process) or a close approximation (Box-Jenkins model process) as discussed in Section 3.2.1. The one-step-ahead prediction by this ARX model gives a good prediction of the current outputs. The same ARX model is then used to predict the multi-step-ahead future outputs. Here in the prediction, the future input effects are eliminated by setting the future inputs to null (refer to (3.1.9)). In predicting the outputs at $k+i$ time point (i=1, 2, f-1), the outputs between $k+i$-1 and k used in ARX are substituted with the known predicted values. In this way, all the future outputs within the future horizon are predicted based on the same ARX model and the past data set (without any involvement of future data). The result of the multi-step ahead predictions is essentially an estimate of the predictable subspace.

This algorithm is also applicable for the process with instantaneous action (system matrix $\boldsymbol{D}$ is not null) in the process for open loop case. In this case, y_k should be fitted against $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ and u_k, and the coefficient for u_k is an estimate of $\boldsymbol{D}$. In the estimation of

the predictable subspace, the instantaneous action is also eliminated from the future outputs with zero-set future inputs.

N4SID-Hf algorithm

There is another approach to estimating the predictable subspace based on the fitted ARX model: the first f impulse weights $\hat{w}_0$, $\hat{w}_1$, $\hat{w}_2$, …, $\hat{w}_{f-1}$ can be calculated from the ARX model, and an estimated $\boldsymbol{H}_f$ matrix can be constructed based on these estimated impulse weights according to the form shown in (2.1.9) as:

$$H_{f\_e} = \begin{pmatrix} \hat{w}_0 & 0 & 0 & \cdots & 0 \\ \hat{w}_1 & \hat{w}_0 & 0 & \cdots & 0 \\ w_2 & \hat{w}_1 & \hat{w}_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & 0 \\ \hat{w}_{f-1} & \hat{w}_{f-2} & \hat{w}_{f-3} & \cdots & \hat{w}_0 \end{pmatrix}$$

remove the future input effects from the future outputs ($\boldsymbol{Y}_{f\_e}=\boldsymbol{Y}_f - \boldsymbol{H}_{f\_e}\boldsymbol{U}_f$); then project the remaining $\boldsymbol{Y}_{f\_e}$ onto the past data ($(\boldsymbol{Y}_f-\boldsymbol{H}_{f\_e}\boldsymbol{U}_f)/[\boldsymbol{Y}_p; \boldsymbol{U}_p]$) to obtain an estimated predictable subspace. This algorithm is denoted as N4SID_Hf thereafter.

This approach is easy to understand based on (3.1.7) and (3.1.9). Here $\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ is an estimate of the future input effects on the future outputs. After removing the estimated future input effects (2nd component in (3.1.7)), the remaining ($\boldsymbol{Y}_{f\_e}=\boldsymbol{Y}_f-\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$) is the predictable subspace plus the effects of the stochastic signals, and it is also an estimate of the predictable subspace. If the effects of future stochastic signals ($\boldsymbol{H}_{s,f}\boldsymbol{W}_f+\boldsymbol{V}_f$) are significant (e.g., a long future horizon used), projecting $\boldsymbol{Y}_{f\_e}$ to $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ will remove the effects of the future stochastic signals, which are uncorrelated with the past data, and a better estimate of the predictable subspace is obtained.

In general, the estimated predictable subspace from this approach is a close approximation of the true value, and is close to the result from N4SID_ARX algorithm. However, The results from these two methods are different. In N4SID_Hf algorithm, $\boldsymbol{Y}_f$ involves the effects of future inputs $\boldsymbol{H}_f\boldsymbol{U}_f$ and the future disturbances, and the future input effects $\boldsymbol{H}_f\boldsymbol{U}_f$ is removed in the procedure of $\boldsymbol{Y}_f-\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$. In N4SID_ARX algorithm, $\boldsymbol{U}_f$ is

not involved. In addition, the effects of the regression error are different in these two methods. Suppose the inputs are positive and the estimated impulse weights are greater than the true values, then N4SID_Hf algorithm will remove more than the true effects of the future inputs, and the final estimated future outputs will be smaller than the true values. However, N4SID_ARX algorithm will give larger estimated predictions of the future outputs than the true values. In practice, the errors on the estimated impulse weighs as well as the inputs are always positive and negative. For the estimated predictable subspace, both methods experience the cancellation of these errors, and the final results are very close to the true values. If the true ARX model is available, both methods will result in the true predictable subspace.

N4SID algorithms based on fitted FIR model

The key idea of these N4SID algorithms is to eliminate the future input effects based on a preliminary process model (estimated impulse weights). There are other approaches to estimate the impulse weights than ARX fitting, such as using the FIR model. The FIR model can be obtained by projecting the current outputs onto the past inputs (long past horizon, only inputs are included in the past data set in (3.1.9)). Both new N4SID algorithms are feasible based on the estimated impulse weights from the fitted FIR model.

Based on the fitted FIR model, both algorithms (projecting the estimated future outputs onto the past inputs in N4SID_Hf algorithm) can only extract the deterministic states and the dynamic model (disturbance model can be estimated thereafter based on the residuals). SVD (PCA) on the predicted future outputs is to reduce the model order, and this is analogous to Box-Jenkins methodology by fitting the data to a lower-order parsimonious model based on the general trend of FIR weights.

3.2.3 Simulation Studies for N4SID Algorithms

In this section, different algorithms are applied to simulation examples to show the principles and the bias issue of the N4SID method. For easier understanding, simple single-input single-output (SISO) processes with ARX and Output-Error (OE) models are chosen for simulation in this subsection. More complicated and practical examples are available in Section 3.5.

N4SID algorithms for an ARARX process

The first simulation example is an ARARX model process to show the basic ideas and the unbiasness of N4SID for ARX process (ARARX is a special ARX model with a disturbance denominator that includes the dynamic denominator):

$$y(k)=\frac{0.2z^{-1}}{1-0.8z^{-1}}u(k)+\frac{1}{(1-0.8z^{-1})(1-0.95z^{-1})}e(k) \tag{3.2.8}$$

In time domain, the following equation holds for the process:

$$y_k = 1.75y_{k-1} - 0.76y_{k-2} + 0.2u_{k-1} - 0.19u_{k-2} + e_k \tag{3.2.9}$$

The process has the following state-space representation in the innovation form:

$$x_{k+1} = \begin{bmatrix} 0.8 & 0 \\ 0 & 0.95 \end{bmatrix} x_k + \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} u_k + \begin{bmatrix} -4.2667 \\ 6.0167 \end{bmatrix} e_k$$

$$y_k = \begin{bmatrix} 1 & 1 \end{bmatrix} x_k + e_k \tag{3.2.10}$$

The process is a 2nd order system with poles at 0.8 and 0.95 respectively. In the simulation, the input u_k is a pseudorandom binary signal (PRBS) signal with magnitude of ±1 and switching time period of 5 (T_s=5). The variance of white noise e_k is adjusted to let the signal-to-noise ratio (SNR, in the sense of variance at the output) to be 1.0, and 1500 simulation data points are collected. The true values of the states and the predictable subspace ($\Gamma_f X_k$) in the simulation are also collected for comparison.

The original N4SID is applied for the simulation data. Both the past and future horizons are of 3 steps, and there are totally 1495 effective data points after considering

the edge effects of these time horizons. The coefficient matrices of LS regression of $\boldsymbol{Y}_f$ against $[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{U}_f]$ are shown as $\boldsymbol{L}_1$, $\boldsymbol{L}_2$ and $\boldsymbol{L}_3$ respectively (the true $\boldsymbol{H}_f$ matrix is also shown to compare with $\boldsymbol{L}_3$. See equation (3.2.3) for notation):

$$L_1 = \begin{bmatrix} 1.7016 & -0.6771 & -0.0381 \\ 2.2508 & -1.2524 & -0.0354 \\ 2.5824 & -1.5716 & -0.0781 \end{bmatrix} \quad L_2 = \begin{bmatrix} 0.1983 & -0.1792 & -0.0073 \\ 0.1583 & -0.3190 & -0.0049 \\ 0.1274 & -0.4086 & -0.0160 \end{bmatrix}$$

$$L_3 = \begin{bmatrix} -0.0022 & 0.0027 & -0.0006 \\ 0.1945 & 0.0026 & 0.0014 \\ 0.1535 & 0.2008 & 0.0023 \end{bmatrix} \quad H_f = \begin{bmatrix} 0 & 0 & 0 \\ 0.2 & 0 & 0 \\ 0.16 & 0.2 & 0 \end{bmatrix} \tag{3.2.11}$$

The three rows of above matrices correspond to predictions of y_k, y_{k+1}, y_{k+2} based on past data respectively.

y_k can be predicted based on the following ARX model (from fitting y_k to three steps of past inputs and outputs):

$$\hat{y}_k = 1.7011y_{k-1} - 0.6762y_{k-2} - 0.0385y_{k-3} + 0.1981u_{k-1} - 0.1791u_{k-2} - 0.0076u_{k-3} \tag{3.2.12}$$

The true prediction model of y_k can be obtained as (using (3.2.9) for y_{k-1} while keeping the coefficient for y_{k-3} in the right-hand side the same as that in (3.2.12)):

$$\hat{y}_k = 1.6993y_{k-1} - 0.6713y_{k-2} - 0.0385y_{k-3} + 0.2u_{k-1} - 0.1799u_{k-2} - 0.0096u_{k-3} \tag{3.2.13}$$

The regression result is very close to true model with minuscule estimation errors. In the original N4SID algorithm, the predictable subspace is estimated as $\boldsymbol{L}_1\boldsymbol{Y}_p+\boldsymbol{L}_2\boldsymbol{U}_p$.

In the N4SID_ARX algorithm, ARX model (3.2.12) is the prediction of y_k based on past data. When ARX model (3.2.12) is used for the next step, the result is:

$$\hat{y}_{k+1} = 1.7011y_k - 0.6762y_{k-1} - 0.0385y_{k-2} + 0.1981u_k - 0.1791u_{k-1} - 0.0076u_{k-2}$$

When y_k in the above equation is substituted with its estimate in (3.2.12) and u_k is set as 0 to remove the effect of the future input, the prediction of y_{k+1} based on past data is:

$$\hat{y}_{k+1|p} = 2.2175y_{k-1} - 1.1888y_{k-2} - 0.0655y_{k-3} + 0.1579u_{k-1} - 0.3122u_{k-2} - 0.0129u_{k-3} \tag{3.2.14}$$

Similarly, using (3.2.12) recursively, the prediction of y_{k+2} based the past data is:

$$\hat{y}_{k+2|p} = 2.58355y_{k-1} - 1.5650y_{k-2} - 0.0854y_{k-3} + 0.1270u_{k-1} - 0.4101u_{k-2} - 0.0169u_{k-3} \tag{3.2.15}$$

Equation (3.2.12) and the above two equations are the predictions of future outputs based on the past data, and these consist of the predictable subspace estimated by recursively using the one-step-ahead ARX model. Note, the coefficients in (3.2.14) and (3.2.15) are close to but do not equal those in the 2nd and 3rd row of L_1 and L_2 in (3.2.11) respectively.

The impulse weights of the ARX model shown in (3.2.12) can be easily calculated. The first 7 impulse weights are 0, 0.1981, 0.1579, 0.1270, 0.1017, 0.0811 and 0.0642 respectively. Based on these estimated impulse weights, the H_f matrix can be constructed as (refer to Section 2.1):

$$H_{f\_e} = \begin{bmatrix} 0 & 0 & 0 \\ 0.1981 & 0 & 0 \\ 0.1579 & 0.1981 & 0 \end{bmatrix} \tag{3.2.16}$$

It is very close to the true value of H_f shown in (3.2.11). Based on this estimated H_f, the predictable subspace is calculated as Y_f-$H_{f\_e}U_f$. It is the estimated predictable subspace by the N4SID_Hf algorithm.

In N4SID, SVD is performed on the estimated predictable subspace to determine the system order and to estimate the process states. The singular values of the estimated predictable subspace show the relative importance of the singular vectors (estimated states) in explanation of the variance of this subspace. The number of the significant singular values is an estimate of the system order. The singular values of the estimated predictable subspace from different algorithms are listed in Table 3.1. These algorithms all indicate that the process is of order 2, and first two singular vectors from the SVD are estimates of the process states.

Table 3.1 Singular values from N4SID algorithms (ARARX process)

Algorithm	1st singular value	2nd singular value	3rd singular value
Original N4SID	48.9291	4.6004	0.0515
N4SID_ARX	48.8985	4.6324	0.0245
N4SID_Hf	49.1927	4.7186	0.0210

The estimated predictable subspaces from different N4SID algorithms are tested for the biasness. Here Student's t-test is employed to test the bias in each variable of the estimated predictable subspace. The true predictable subspace is $\Gamma_f x_k$, which is known in the simulation study. For each variable of the estimated predictable subspace, the paired differences between the true values and the estimated values are in fact the estimation errors. The Student's t-statistic based on these paired differences can be used to test whether there is bias in the estimates values. For all three N4SID algorithms, the t-statistic values for the three variables in the estimated predictable subspace are listed in Table 3.2. All the absolute values of the t-statistic result are less than 1.96. This clearly indicates that the estimated predictable subspaces are unbiased (with 95% confidence level). Note, the estimated predictable subspace from N4SID_Hf algorithm includes the effect of the future stochastic signals. Here the effect of the future stochastic signals is removed for an accurate test.

Student's t-statistic can also be used to test the biasness on the estimated states. The estimated states and the true states are usually in different basis. The paired differences can be obtained as the residuals of projecting the estimated states to the true states. For the two estimated states from different N4SID algorithm, the Student's t-statistic values based on the paired differences are listed in the last two columns of Table 3.2. These values clearly indicate that the estimated states from these N4SID algorithms are unbiased.

Table 3.2 t-statistic values from N4SID algorithms (ARARX process)

Algorithm	Estimated $y_{k\|p}$	Estimated $y_{k+1\|p}$	Estimated $y_{k+2\|p}$	Estimated $x_{1,k}$	Estimated $x_{2,k}$
Original N4SID	0.7279	0.4722	0.4598	0.5161	-0.3265
N4SID_ARX	0.8704	0.6897	0.6799	0.5392	-0.3706
N4SID_Hf	0	-1.3193	-1.4129	-0.6517	0.7007

N4SID algorithms for an OE process

The second simulation example is a process with Output-Error (OE) model. It is used to show the bias in N4SID algorithms caused by using limited order ARX model to approximate a general dynamic-stochastic process. The simulation example has the following transfer function:

$$y_k = \frac{0.2z^{-1}}{1-0.8z^{-1}}u_k + e_k \tag{3.2.17}$$

The input used in simulation is a PRBS with magnitude of ±4 and T_s=5. The variance of noise e_k is adjusted to have SNR to be 1.0 (similar conditions as in 1st example). 1500 data points are collected

Similar to the first example, both the past and the future horizon consist of 3 steps. Projecting of $\boldsymbol{Y}_f$ onto [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{U}_f$] gives the coefficient matrices $\boldsymbol{L}_1$, $\boldsymbol{L}_2$ and $\boldsymbol{L}_3$ as:

$$L_1 = \begin{bmatrix} 0.1164 & 0.1041 & 0.1628 \\ 0.0853 & 0.1269 & 0.1053 \\ 0.0953 & 0.0561 & 0.0590 \end{bmatrix} \qquad L_2 = \begin{bmatrix} 0.2230 & 0.1081 & 0.1704 \\ 0.1309 & 0.1268 & 0.0933 \\ 0.1402 & 0.0031 & 0.1835 \end{bmatrix}$$

$$L_3 = \begin{bmatrix} 0.0259 & -0.0921 & -0.0595 \\ 0.2282 & 0.0170 & -0.0257 \\ 0.1304 & 0.2177 & 0.0284 \end{bmatrix} \qquad H_f = \begin{bmatrix} 0 & 0 & 0 \\ 0.2 & 0 & 0 \\ 0.16 & 0.2 & 0 \end{bmatrix} \tag{3.2.18}$$

There are obvious differences between coefficient matrix $\boldsymbol{L}_3$ and the true value of $\boldsymbol{H}_f$. The ARX model fit from the data is:

$$\hat{y}_k = 0.1175y_{k-1} + 0.1068y_{k-2} + 0.1651y_{k-3} + 0.2145u_{k-1} + 0.1082u_{k-2} + 0.1648u_{k-3} \tag{3.2.19}$$

and the first 7 impulse weights from this ARX model are 0, 0.2145, 0.1334, 0.2034, 0.0736, 0.0524, and 0.0476. The constructed $\boldsymbol{H}_f$ matrix based on the above estimated impulse weights is :

$$H_{f\_e} = \begin{bmatrix} 0 & 0 & 0 \\ 0.2145 & 0 & 0 \\ 0.1334 & 0.2145 & 0 \end{bmatrix} \tag{3.2.20}$$

Based on the above result, the predictable subspace and the process states can be estimated by the original N4SID, N4SID_ARX and N4SID_Hf algorithms. The results of

the singular values from SVD of the estimated predictable subspaces are listed in Table 3.3. All these algorithms can give a clear cut-off after the first singular value on the magnitude. The system is determined of order 1. The Student's *t*-statistic values based on the paired differences between the true and the estimated predictable subspace and states are listed in Table 3.4. These results clearly show that all three N4SID algorithms give biased estimates of the predictable subspace for this OE model process. Similar conclusions can be drawn from the *t*-test for the estimated state in this example.

Table 3.3 Singular values from N4SID algorithms (OE process)

Algorithm	1st singular value	2nd singular value	3rd singular value
Original N4SID	34.5304	2.0064	1.1180
N4SID_ARX	33.5558	3.9037	3.1026
N4SID_Hf	35.3802	0.7080	0.1538

Table 3.4 *t*-statistic values from N4SID algorithms (OE process)

Algorithm	Estimated $y_{k\|p}$	Estimated $y_{k+1\|p}$	Estimated $y_{k+2\|p}$	Estimated x_k
Original N4SID	-3.8038	-5.0553	-4.4511	-2.9308
N4SID_ARX	-4.0494	-4.6861	-4.2675	-2.6091
N4SID_Hf	-0.4656	-4.5558	3.1957	-2.5849

3.3 Analysis of CVA

CVA uses Canonical Correlation Analysis (CCA) to estimate the process states. In statistics, CCA finds the most correlated linear combinations from two sets of variables respectively, in other words: the most common variates between two data sets. From the prediction point of view, CCA finds a linear combination of one data set to explain the most percentage of variation in a direction of the other data set. Refer to Chapter 4 and Appendix A4.3 for more details and the computation approaches of CCA.

CVA was first proposed by Larimore (1990). It was illustrated and explained in terms of the maximum likelihood principle, but the ideas were not straightforward, and there was a minor mistreatment in the paper. Here, based on the multi-step state-space model, the correct procedures of the existing algorithm are summarized, and a new algorithm is proposed. Both algorithms will be analyzed with focuses on the principles and the biasness of the estimated state space. The mistakes in publications will also be discussed.

3.3.1 CVA Regression Out Algorithm

The primary feature of the CVA regression out algorithm is to perform CCA on the past data set ($\boldsymbol{Y}_p$ and $\boldsymbol{U}_p$) and the future outputs ($\boldsymbol{Y}_f$), both of which have been projected onto the orthogonal space of the future outputs (i.e., regressing the future inputs out of both the past data and future output data). The basic idea of this algorithm appeared in different papers (Larimore, 1990; Van Overschee and De Moor, 1995; Carrette, 2000). However, the correct and complete procedures of this algorithm have not been summarized or proved in any published paper. Here, the procedures of the algorithm are first summarized, and then a detailed analysis on the principles and a strict proof on the biasness issue will be presented.

The CVA regression out algorithm (referred as CVA_RO hereafter) includes the following procedures:

- Regress $\boldsymbol{U}_f$ out of the future outputs $\boldsymbol{Y}_f$ and the past input/output data $\boldsymbol{P}_{IO}=[\boldsymbol{Y}_p;\ \boldsymbol{U}_p]$, i.e., $\boldsymbol{Y}_{f\_ro}=\boldsymbol{Y}_f\boldsymbol{P}_{ufo}=\boldsymbol{Y}_f-\boldsymbol{B}_{ro\_Yf}\boldsymbol{U}_f$, $\boldsymbol{P}_{IO\_ro}=\boldsymbol{P}_{IO}\boldsymbol{P}_{ufo}=\boldsymbol{P}_{IO}-\boldsymbol{B}_{ro\_PIO}\boldsymbol{U}_f$. Here $\boldsymbol{P}_{ufo}=\boldsymbol{I}-\boldsymbol{U}_f^T(\boldsymbol{U}_f\boldsymbol{U}_f^T)^{-1}\boldsymbol{U}_f$ is the projection matrix onto the orthogonal space of $\boldsymbol{U}_f$, $\boldsymbol{B}_{ro\_Yf}=\boldsymbol{Y}_f\boldsymbol{U}_f^T(\boldsymbol{U}_f\boldsymbol{U}_f^T)^{-1}$ and $\boldsymbol{B}_{ro\_PIO}=\boldsymbol{P}_{IO}\boldsymbol{U}_f^T(\boldsymbol{U}_f\boldsymbol{U}_f^T)^{-1}$.
- Perform CCA on $\boldsymbol{Y}_{f\_ro}$ and $\boldsymbol{P}_{IO\_ro}$, denote the coefficient matrix for canonical variates from $\boldsymbol{P}_{IO\_ro}$ as $\boldsymbol{J}_1$. The system order can be determined by the number of the dominant canonical correlation coefficients (CCCs).

- Take estimated process states from $X_k = J_1 P_{IO}$ (note, not $J_1 P_{IO\_ro}$, the CVs of above CCA).
- Fit the estimated X_k to the state-space model.

Here P_{ufo} is the projection matrix for projection onto the orthogonal space of U_f. In this algorithm, CCA is performed on the modified past data and the modified future output data, and the coefficient matrix from the CCA is used for the original past data to estimate process states.

Based on the multi-step state-space model for Y_f in (3.1.7), it is clear that regressing U_f out of Y_f removes not only the future input effects $H_f U_f$, but also removes the projection of the predictable subspace $\Gamma_f X_k$ on the future inputs U_f, i.e., $\Gamma_f X_k / U_f$. This projection generally is not a null space because of the correlation between X_k and U_f. The correlation exists in two scenarios: U_f is determined by a feedback controller (state feedback controller or output feedback controller, i.e., the data is collected under closed-loop), and the inputs are auto-correlated signals (hence U_f is correlated to U_p, which make a contribution to X_k). Chapter 6 will discuss the closed-loop case, and this section considers only the open loop case.

Projecting both sides of (3.1.7) onto the orthogonal space of U_f (post multiplying P_{ufo}) gives the following relationship:

$$Y_{f\_ro} = \Gamma_f X_{k\_ro} + H_{s,f} W_f + V_f \tag{3.3.1}$$

here $X_{k\_ro}=X_k P_{ufo}$ is the result of regressing U_f out of X_k. The left-hand side is the result of regressing U_f out of Y_f. On the right-hand side, the effects of the future inputs $H_f U_f$ are totally removed, and $\Gamma_f X_{k\_ro}$ is the result of regressing U_f out of the predictable subspace. The stochastic term $H_{s,f} W_f + V_f$ are kept intact in the regressing out because of its no correlation with U_f. Equation (3.3.1) shows that Y_{f_ro} should lie in the space of X_{k_ro} in the sense of expectation. The subspace X_{k_ro} relates to the past data by projecting both sides of (3.1.6) onto the orthogonal space of U_f:

$$X_{k\_ro} = \left[A^p\Gamma_p^{+} \quad \Omega_p - A^p\Gamma_p^{+}H_p\right]\begin{bmatrix} Y_p \\ U_p \end{bmatrix} P_{ufo} + \left(\Omega_{s,p} - A^p\Gamma_p^{+}H_{s,p}\right)W_p - A^p\Gamma_p^{+}V_p \quad (3.3.2)$$
$$= \left[A^p\Gamma_p^{+} \quad \Omega_p - A^p\Gamma_p^{+}H_p\right]P_{IO\_ro} + \left(\Omega_{s,p} - A^p\Gamma_p^{+}H_{s,p}\right)W_p - A^p\Gamma_p^{+}V_p$$

Based on the above two equations, it is clear that $\boldsymbol{X}_{k\_ro}$ is the link between $\boldsymbol{Y}_{f\_ro}$ and $\boldsymbol{P}_{IO\_ro}$ and variables in $\boldsymbol{X}_{k\_ro}$ are the most common variates between the two data sets (stochastic variables are independent of each other and other variables). Therefore, the dominant CVs from CCA on $\boldsymbol{P}_{IO\_ro}$ and $\boldsymbol{Y}_{f\_ro}$ are estimates of $\boldsymbol{X}_{k\_ro}$. They are not the estimates for the process states $\boldsymbol{X}_k$. However, the coefficient matrix $\boldsymbol{J}_1$ between $\boldsymbol{P}_{IO\_ro}$ and $\boldsymbol{X}_{k\_ro}$ is the same as between $\boldsymbol{P}_{IO}$ and $\boldsymbol{X}_k$ (see (3.3.2) and (3.1.6)), and therefore the first few variables of $\boldsymbol{J}_1\boldsymbol{P}_{IO}$ are estimates of the process states $\boldsymbol{X}_k$. The system order, i.e., the number of variables should be taken from $\boldsymbol{J}_1\boldsymbol{P}_{IO}$, can be determined by the number of dominant canonical correlation coefficients (CCCs) of the above CCA, or by other criteria (such as AIC). These are the basic ideas behind the CVA_RO algorithm. See Appendix A3.2 for details and the conditions for biased or unbiased results. Note, if there is co-linearity between $\boldsymbol{X}_k$ and $\boldsymbol{U}_f$, information of one or more states is completely removed from $\boldsymbol{Y}_f$ and $\boldsymbol{P}_{IO}$; therefore, not all the states can be estimated by this algorithm. This corresponds to the rank deficiency of $\boldsymbol{D}_1$ in Appendix A3.2.

As for the biasness issue, if the true process is of ARX (or ARARX) model, CVA_RO gives an unbiased result for the process states and the process model. For other general processes, CVA_RO gives biased estimates of the process states and the process model.

CVA_RO algorithm is shown to use the same approach as the original N4SID algorithm to eliminate the future input effects from the $\boldsymbol{Y}_f$ (see Chapter 5). In fact, the CVA approaches described in Van Overschee and De Moor (1995) is a realization of the CVA_RO (in the form of square-root-factor for CCA computation, see Appendix A4.3). The analysis based on the multi-step state-space model shows the differences and the connections between the CVs and the process states. The essence of each step in CVA_RO algorithm becomes clearer and easier for understanding.

3.3.2 CVA Algorithm Based on Estimated H_f

A key issue in CVA is to remove the future input effects H_fU_f from Y_f. Here U_f is a known data set, and only the coefficient matrix H_f is unknown. If H_f is estimated, then future input effects can be estimated and removed from the future outputs. Based on this idea, a new CVA algorithm becomes possible: eliminate the effects of U_f from Y_f based on an estimated H_f, and then use CCA to estimate the state variables. The detail procedures in this algorithm are described below:

- Fit an ARX model by regressing the current output y_k against the past input and output data $P_{IO}=[Y_p; U_p]$
- Construct the estimated H_f matrix based on the estimated impulse weights from the above ARX model, denoted as H_{f_e}, and estimate the predictable subspace as $Y_{f\_e}=Y_f - H_{f\_e}U_f$
- Perform CCA on P_{IO} and Y_{f_e}, the number of dominant canonical correlation coefficients (CCCs) are taken as the estimated system order $\hat{n}$, and the first $\hat{n}$ CVs from P_{IO} are taken as the estimated state sequence X_k
- Fit the estimated state sequence X_k to the state-space model

This algorithm is based on an estimated coefficient matrix H_f, it is referred as CVA_Hf algorithm hereafter, or simply CVA algorithm if there is no confusion.

Matrix H_f shows the effects of the future inputs on the future outputs. From Section 2.1, it is clear that H_f is a lower triangle matrix with the first f step impulse weights (or weight blocks) on its lower diagonals. Fitting an ARX model is a way to obtain the estimated values of these impulse weights (fitting an FIR model is another possible approach), and then H_f can be constructed by setting these estimated impulse weights on the lower diagonals, denote this estimated H_f as H_{f_e}. The multi-step state-space model for future outputs (3.1.7) indicates that data set Y_f-$H_{f\_e}U_f$ is an estimate of the predictable subspace $\Gamma_f X_k$ plus future stochastic terms. Considering (3.1.6), it is clear

that data set Y_f-$H_{f\_e}U_f$ and the past data $[Y_p;\ U_p]$ have the common variate of X_k. This implies that CCA on these two data sets provides an approach to estimation of the state sequence. These are the fundamental ideas behind the new CVA algorithm. Refer to Appendix A3.3 for the details and the biasness properties of the algorithm.

This CVA algorithm gives an unbiased estimate of state sequence if the true process is of ARX (or ARARX) model form. For other general process models, CVA can only give asymptotical unbiased results with an infinite number of past steps. With a limited number of past steps, there is some bias on the estimated state sequence. This is determined by the first step of fitting the ARX model. If the true process is an ARX process, and the lag steps in the past horizon is long enough, the fitted ARX model is unbiased and the impulse weights from the ARX model are also unbiased. If the true process is a general Box-Jenkins model, the fitted ARX model with limited past steps will be an approximation of the true model, and the impulse weight estimates will be biased. When the number of past steps tends to infinity, the bias tends to disappear. With unbiased impulse weights, H_{f_e} and the estimated predictable subspace Y_f-$H_{f\_e}U_f$ will be unbiased.

In this algorithm, CCA is performed on $P_{IO}=[Y_p;\ U_p]$ and the estimated predictable subspace $Y_{f\_e}=Y_f$-$H_{f\_e}U_f$, which includes both the true predictable subspace $\Gamma_f X_k$ and the effects of future stochastic variables $H_{s,f}W_f+V_f$. From (3.1.8), it is clear that the first term $\Gamma_f X_k$ can be predicted by the past input and output data P_{IO}. However, the second term is uncorrelated with P_{IO} and cannot be predicted by P_{IO}. Since $\Gamma_f X_k$ is of rank n, there will be only n significant correlations between the two data sets. The first n canonical variates from P_{IO} are the best summarization of the past data P_{IO} for the prediction of $\Gamma_f X_k$. CCA gives the best prediction of Y_{f_e} in the sense of explaining the relative percentage of variation, not in the sense of the absolute magnitude of the variance as in N4SID. The resultant canonical variates are also independent of the scaling of the input and output variables. This is an advantage when using this method without

considering the variable unit or variable scaling; however, this also could be a disadvantage that no variable can have more influence on the result.

In the case of auto-correlated inputs, the removal of the effects of the future inputs is necessary prior to performing CCA between the future outputs and the past data. Without removal of $H_f U_f$, the effects of the current states and the effects of the future inputs on the future outputs will be confounded when performing CCA. The correlation between X_k and U_f may lead to more components in Y_f explained by P_{IO} than the predictable subspace $\Gamma_f X_k$. However, this is due to the correlation between $H_f U_f$ and P_{IO}, not due to the causal relationship in the process.

3.3.3 Discussion of CVA Algorithms

Though both CVA algorithms have the same bias properties, the CVA_Hf algorithm has a better SNR than CVA_RO in the computation. The true state information in Y_f is $\Gamma_f X_k$. In the case of auto-correlated inputs, CVA_RO algorithm loses part of the state information $\Gamma_f X_k / U_f$ in regressing U_f out of Y_f, and only $\Gamma_f X_k P_{ufo}$ is left in Y_{f_ro}; meanwhile, the effects of the future stochastic signals are kept intact in regressing U_f out of Y_f. This loss of state information will reduce the signal-to-noise ratio (SNR) in the Y_{f_ro}, and therefore result in a larger variance for the estimated coefficient matrix and the estimated states in the CVA_RO algorithm. The degradation in SNR is directly related to the severity of the auto-correlation of the process inputs. CVA_Hf does not suffer such a problem and has a better SNR in $Y_{f\_e} = Y_f - H_{f\_e} U_f$.

The estimated state sequences from the two CVA algorithms have different orthogonality properties. Estimated states from CVA_Hf are CVs from a CCA, and they are orthogonal to each other. The CVs from CVA_RO are $J_1 P_{IO\_ro}$, and they are orthogonal. However, the estimated states are $J_1 P_{IO}$, and they are not orthogonal to each other. Orthogonal state variables are preferable in the LS fitting for system matrices since they should lead to a better-conditioned set of normal equations.

If the input signals are not auto-correlated (e.g., white noise or PRBS with T_s=1), regressing $\boldsymbol{U}_f$ out of $\boldsymbol{Y}_f$ only removes the future input effects from $\boldsymbol{Y}_f$. CVA_RO will be equivalent to CVA_Hf though there might be a small numerical difference for data sets with finite length.

In some papers discussing the CVA method, there are still some mistakes in state estimation by CCA. One of such mistakes is to take the CVs from CCA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_f$) as estimated states (e.g., Lakshminarayanan, 1997), without removing the future input effects. This approach is analogous to estimating the predictable subspace and the states by projection of $\boldsymbol{Y}_f$ onto $\boldsymbol{P}_{IO}$. Only when the inputs are not auto-correlated (e.g., white noise signals), will the CVs be unbiased estimates of the process states. Because of the additional variations arising from $\boldsymbol{U}_f$, however, the state estimates will have larger variance than those from the CVA_Hf algorithm.

Another misleading approach is to take CVs from CCA ($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_{f\_ro}$) as estimated states (e.g., Larimore, 1990). Here the future input effects are eliminated by simply regressing $\boldsymbol{U}_f$ out of $\boldsymbol{Y}_f$. For general cases (auto-correlated inputs), as discussed above, this approach will not only remove the future input effects, but will also eliminate the projection of $\boldsymbol{\Gamma}_f\boldsymbol{X}_k$ onto $\boldsymbol{U}_f$. CVs from this CCA are only estimates of $\boldsymbol{X}_{k\_ro}$ rather than $\boldsymbol{X}_k$. This approach is valid only when the inputs are white noise signals.

The third mistake is to use CVs from CCA ($\boldsymbol{P}_{IO\_ro}$, $\boldsymbol{Y}_{f\_ro}$) as the estimated states in CVA_RO algorithm. In this case, the CVs are $\boldsymbol{J}_1\boldsymbol{P}_{IO\_ro}$, which is in fact an estimate of $\boldsymbol{X}_{k\_ro}$ as shown in Section 3.3.1. The difference between CVs and $\boldsymbol{X}_k$=$\boldsymbol{J}_1\boldsymbol{P}_{IO}$ is $\boldsymbol{J}_1\boldsymbol{B}_{ro\_PIO}\boldsymbol{U}_f$ (where $\boldsymbol{B}_{ro\_PIO}$=$\boldsymbol{P}_{IO}\boldsymbol{U}_f^T(\boldsymbol{U}_f\boldsymbol{U}_f^T)^{-1}$).

3.3.4 Simulation Studies for CVA Algorithms

The simulation studies in this subsection focus on illustration of the biasness issues of the CVA_RO and CVA_Hf algorithms discussed above. The simulation examples of the ARARX and OE process used previously in Section 3.2.3 (for the N4SID

algorithms) are employed again to show the results of these CVA algorithms under different conditions. Simulation examples will also demonstrate the errors in the misleading approaches discussed in the last subsection.

CVA algorithms for an ARARX process

Both CVA_Hf and CVA_RO algorithms are applied to the simulation data from the ARARX process showing in (3.2.8). The CVA_Hf algorithm is based on the same estimated predictable subspace $\boldsymbol{Y}_{f\_e} = \boldsymbol{Y}_f - \boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ as in Section 3.2.3 ($\boldsymbol{H}_{f\_e}$ is the same as in (3.2.16)). This estimated predictable subspace is already tested to be unbiased in Section 3.2.3. The estimated states come from the significant CVs from CCA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_{f\_e}$). In CVA_RO algorithm, the LS regression coefficient matrix of $\boldsymbol{Y}_f$ against $\boldsymbol{U}_f$ is:

$$B_{ro\_Yf} = \begin{bmatrix} -0.0656 & -0.0827 & -0.1042 \\ 0.1673 & -0.0389 & -0.0484 \\ 0.4440 & 0.5554 & -0.4465 \end{bmatrix} \tag{3.3.3}$$

then the result of regressing $\boldsymbol{U}_f$ out of $\boldsymbol{Y}_f$ is $\boldsymbol{Y}_{f\_ro} = \boldsymbol{Y}_f - \boldsymbol{B}_{ro\_Yf}\boldsymbol{U}_f$. This result is obviously different from $\boldsymbol{Y}_{f\_e}$ due to the large difference between this $\boldsymbol{H}_{f\_e}$ and the $\boldsymbol{H}_{f\_e}$ in (3.2.8). The result of $\boldsymbol{P}_{IO\_ro}$ can be obtained similarly. In the CVA_RO algorithm, the coefficient matrix for CVs from CCA($\boldsymbol{P}_{IO\_ro}$, $\boldsymbol{Y}_{f\_ro}$) is applied to $\boldsymbol{P}_{IO}$ for process state estimates. The computation and test results by these two algorithms are listed in Table 3.5.

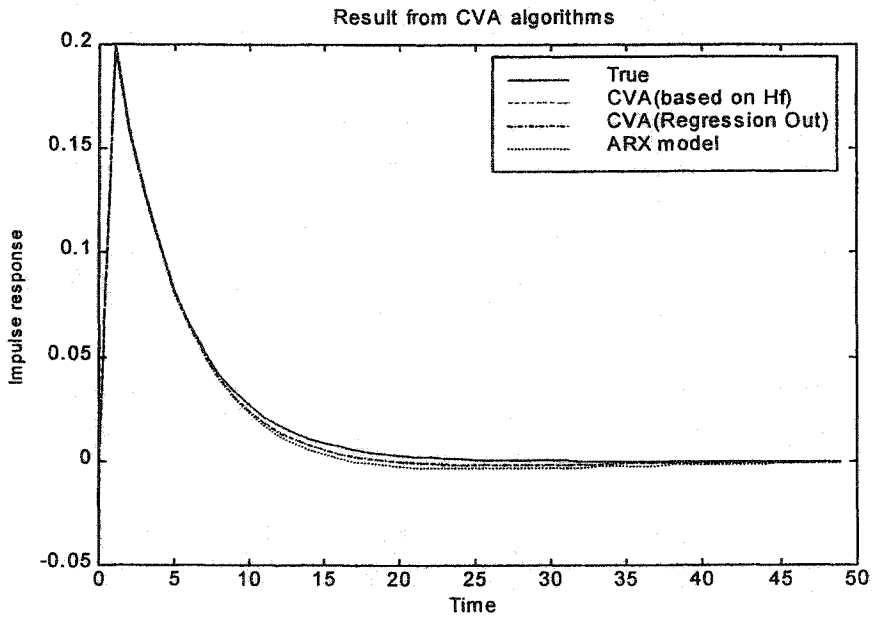
Table 3.5 Results from CVA algorithms (ARARX process)

	CCA	CCCs	Estimated order	t-statistic	Variance of state estimation error
CVA_Hf	CCA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_{f\_e}$)	0.9992,0.8399, 0.0666	2	0.4682 0.2930	7.2×10^{-6} 1.1×10^{-3}
CVA_RO	CCA($\boldsymbol{P}_{IO\_ro}$, $\boldsymbol{Y}_{f\_ro}$)	0.9990, 0.8269, 0.0635	2	0.4577 0.3142	8.4×10^{-6} 1.6×10^{-3}

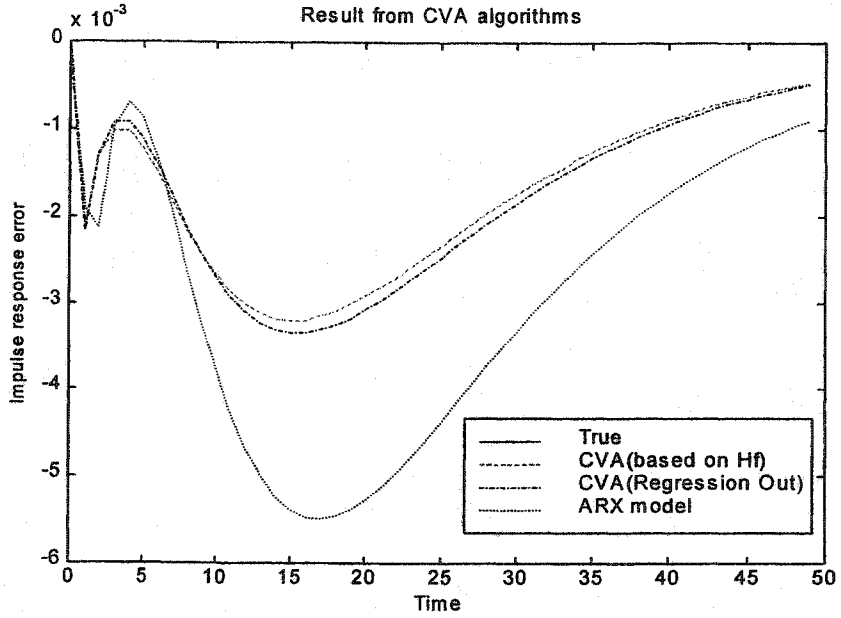
CCCs: Canonical Correlation Coefficients

It is clear that both CVA algorithms give clear cut-off on the magnitude of CCCs after the 2nd CV; therefore both algorithms correctly determined the system order (the same conclusion is drawn based on AIC). In the CVA_Hf algorithm, the first 2 CVs are

taken as the estimated states (unit variance, orthogonal to each other). In CVA_RO algorithm, the coefficients for the first 2 CVs are applied to P_{IO} to estimate process states. The variances of these two estimated states are 1.2544 and 1.1291 respectively, and their correlation coefficient is -0.1516 (not orthogonal to each other). In Table 3.5, Student's *t*-statistic values for the paired difference between the estimated states and the true states clearly indicate that both algorithms give unbiased estimates of process states for this ARARX process (α=0.05, refer to Section 3.2.3). The estimation errors on the estimated states are rather small, and the CVA_Hf algorithm gives a slightly better result than the CVA_RO algorithm. This conclusion is confirmed by the resultant impulse responses shown in Figure 3.3. Both CVA algorithms give better results than the ARX model.



(a) Impulse responses



(b) Error on the impulse responses

Figure 3.3 Impulse responses of models from CVA algorithms (ARARX process)

This simulation example is also used to test the three incorrect CCA state estimation approaches discussed at the last subsection: CVs from CCA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_f$), CCA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_{f\_ro}$) and CCA($\boldsymbol{P}_{IO\_ro}$, $\boldsymbol{Y}_{f\_ro}$). The results of the canonical correlation coefficients (CCCs) from these approaches are listed in Table 3.6. The latter two approaches give clear cut-offs on the significance of the canonical correlations to determine the system of order 2. CCA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_f$) cannot give such a clear cut-off since its CVs try to explain part of the future input effects. Here the first two CVs in these approaches are taken as the estimated process states.

Table 3.6 CCCs from different CCA approaches (ARARX process)

CCA approach	1st CCC	2nd CCC	3rd CCC
CCA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_f$)	0.9987	0.7161	0.2050
CCA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_{f\_ro}$)	0.9580	0.8014	0.0618
CCA($\boldsymbol{P}_{IO\_ro}$, $\boldsymbol{Y}_{f\_ro}$)	0.9990	0.8269	0.0635

CCC: Canonical Correlation Coefficient

The errors on the estimated states are calculated by comparing them to the true process states. The variances of the errors from the three approaches are listed in Table 3.7. As shown in Table 3.7, these errors are tremendously higher than those errors by CVA_Hf and CVA_RO algorithms (variances in magnitude of 10^{-6} and 10^{-3} respectively). As discussed before, these errors include some effects of the future inputs. After regressing U_f out of these errors, the variances of the residuals become significantly smaller. It is clear that about 40% to 70% of the errors on the estimated states are directly related to the future inputs. If these estimated states are used for fitting the state-space model, the final model results are shown in Figure 3.4. Compared to those from CVA algorithms showing in Figure 3.3(a), these impulse weights have much larger deviations from the true values, and the bias on the final result becomes obvious.

Table 3.7 State estimation errors from different CCA approaches (ARARX process)

CCA approach	1st Canonical Variate			2nd Canonical Variate		
	Variance of state error	Variance of residual	Explained by U_f	Variance of state error	Variance of residual	Explained by U_f
CCA(P_{IO}, Y_f)	0.00018	0.00011	38.6%	0.9591	0.5747	40.1%
CCA(P_{IO}, Y_{f_ro})	0.0723	0.0438	39.4%	0.1041	0.0640	38.5%
CCA(P_{IO_ro}, Y_{f_ro})	0.1831	0.0526	71.3%	0.1058	0.0312	70.5%

Note: residual = state error – projection of state error onto U_f

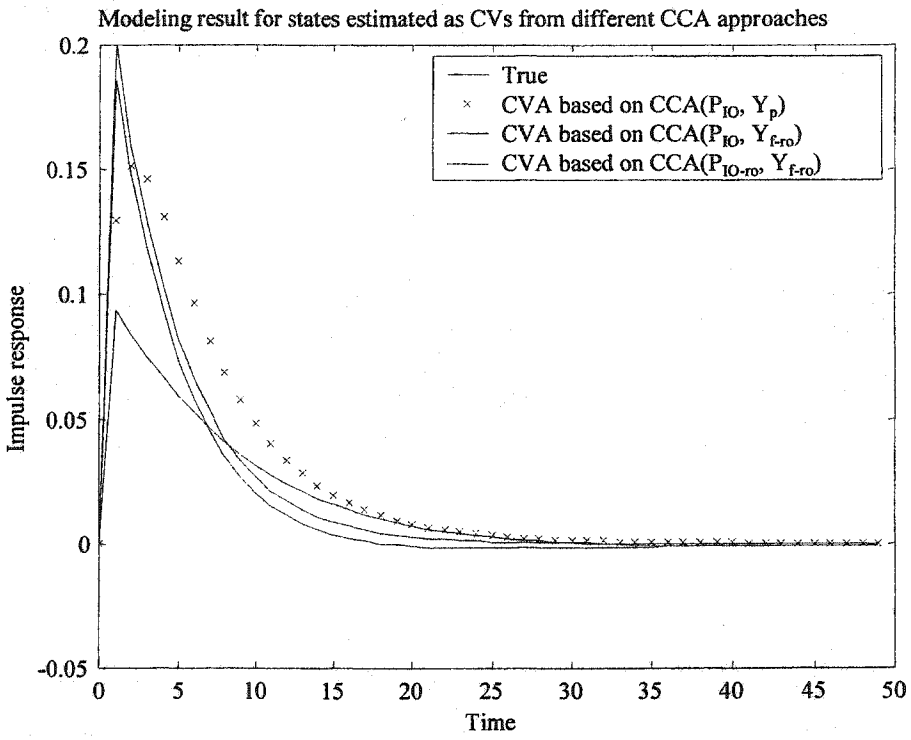


Figure 3.4 Impulse weights of resultant models based on CCA approaches for states

CVA algorithms for an OE process

The OE simulation example (3.2.17) in Section 3.2.3 is also used to demonstrate the CVA_Hf and CVA_RO algorithms. The fitted ARX model, the estimated $\boldsymbol{H}_f$ matrix and the estimated predictable subspace $\boldsymbol{Y}_{f\_e}=\boldsymbol{Y}_f-\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ are the same as shown in Section 3.2.3. The CCCs from CCA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_{f\_e}$) and CCA($\boldsymbol{P}_{IO\_ro}$, $\boldsymbol{Y}_{f\_ro}$) are listed in Table 3.8. The result indicates that the system can be determined of first order, though the cut-off for significance is not as clear as in the ARARX simulation example.

Table 3.8 CCCs from CCA algorithms (OE process)

CVA algorithm	1st CCC	2nd CCC	3rd CCC
CVA_Hf	0.7944	0.0791	0.0410
CVA_RO	0.7260	0.0830	0.0434

For these two CVA algorithms, variances of the estimated states, variances of the state estimation errors, and the *t*-test results on the estimated states are shown in Table 3.9. These two methods give similar accuracy for state estimation (relative error). Both algorithms give biased state estimates for this OE process. The impulse responses of the resultant models are shown in Figure 3.5, and they indicate the bias of the resultant models. Nevertheless, these impulse responses are better than the ARX fitting result (smoother and smaller error).

Table 3.9 State estimated by CVA algorithms (OE process)

CVA algorithm	Variance of state	Variance of state error	Paired *t*-statistic
CVA_Hf	1.0	0.0769	-2.8953
CVA_RO	1.50	0.1176	-2.9292

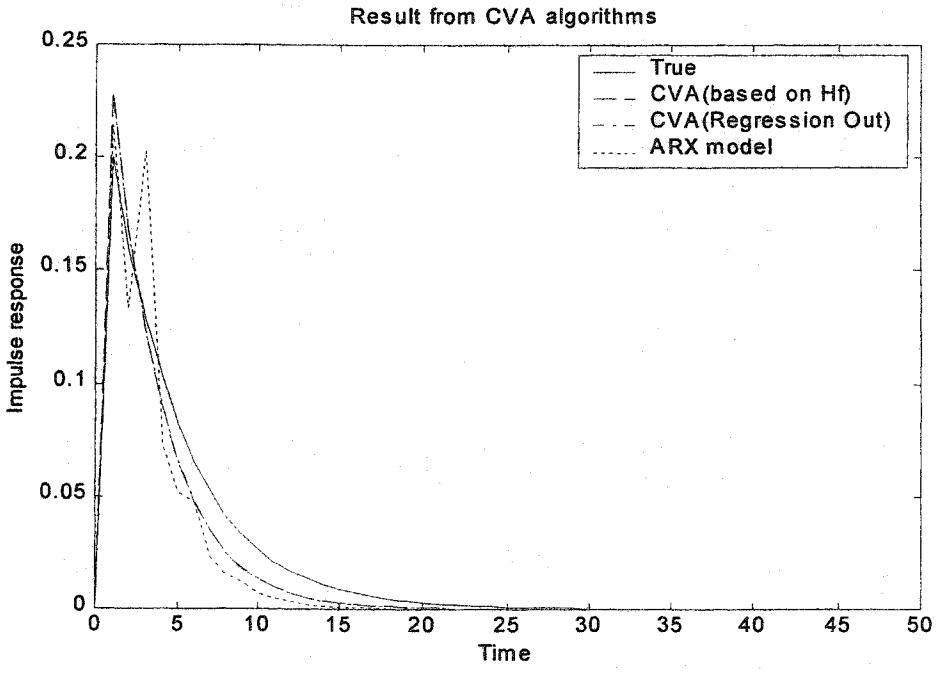


Figure 3.5 Impulse responses from CVA algorithms (OE process)

3.4 MOESP Based on Estimated States

The original MOESP method is based on QR decomposition and SVD for the extended observability matrix Γ_f (see Section 2.4). The use of QR decomposition is in fact an equivalent way of regressing U_f out of Y_f. For white noise inputs, the algorithm (elementary scheme) takes the matrix $R_{22}Q_2$ as an estimate of the predictable subspace.

3.4.1 MOESP Algorithm Based on Estimated States

For the case of white noise inputs, exploration of the MOESP method shows two new algorithms: one based on estimating Γ_f but simpler than the original MOESP, and another one based on estimating the state variables.

The multi-step state model for future outputs can be written in the following form:

$$Y_f = \Gamma_f X_k + H_f U_f + H_{s,f} W_f + V_f \tag{3.4.1}$$

and the QR factorization of U_f and Y_f is:

$$\begin{bmatrix} U_f \\ Y_f \end{bmatrix} = RQ = \begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \tag{3.4.2}$$

Here R_{11} and R_{22} are lower triangular matrices, and $\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \begin{bmatrix} Q_1^T & Q_2^T \end{bmatrix} = \begin{bmatrix} I_{mi} & 0 \\ 0 & I_{li} \end{bmatrix}$. R_{11} is of full rank (the input is persistently excited at a sufficiently high order). Based on (3.4.2), the future outputs in (3.4.1) can be expressed as:

$$\Gamma_f X_k + H_f R_{11} Q_1 + H_{s,f} W_f + V_f = R_{21} Q_1 + R_{22} Q_2 \tag{3.4.3}$$

In the case of white noise inputs, the current state variables are linear combinations of the past input data (in sense of infinite lag steps), and are uncorrelated to the future input data, therefore we have:

$$Eep\{X_k U_f^T\} = Exp\{X_k Q_1^T\} R_{11}^T = 0 \tag{3.4.4}$$

Since $\boldsymbol{Q}_1$ is independent of the stochastic variables ($\boldsymbol{W}_f$, $\boldsymbol{V}_f$) and orthogonal to $\boldsymbol{Q}_2$, we can get the following relationships if the above (3.4.3) is post multiplied with $\boldsymbol{Q}_1^T$ and $\boldsymbol{Q}_2^T$ respectively:

$$H_f R_{11} = R_{21} \tag{3.4.5}$$

Therefore, the coefficient matrix $\boldsymbol{H}_f$ can be estimated as:

$$H_f = R_{21} R_{11}^{-1} \tag{3.4.6}$$

Equation (3.4.5) also indicates that the $\boldsymbol{Q}_1$ term in the left-hand side of (3.4.3) equals to the $\boldsymbol{Q}_1$ term of the right-hand side. What is left in (3.4.3) is the following relationship:

$$\Gamma_f X_k + H_{s.f} W_f + V_f = R_{22} Q_2 \tag{3.4.7}$$

This shows that $\boldsymbol{R}_{22}\boldsymbol{Q}_2$ is essentially an estimate of the predictable subspace $\boldsymbol{\Gamma}_f\boldsymbol{X}_k$. In case of noise free (*i.e.*, $\boldsymbol{W}_f$=$\boldsymbol{V}_f$=0), the above equation gives:

$$\Gamma_f X_k Q_2^T = R_{22} \tag{3.4.8}$$

For a general case, the above relationship approximately holds if the disturbance in the process is not very severe. For precise models, one can project equation (3.4.7) onto the past data in order to remove the effects of the future stochastic signals, and have the following relationship:

$$\Gamma_f X_k / P_{IO} = R_{22} Q_2 / P_{IO} \tag{3.4.9}$$

The left side is essentially the predictable subspace $\boldsymbol{\Gamma}_f\boldsymbol{X}_k$ (ARX process) or its close approximation (other general process).

There are two different schemes for the system matrices: based estimated $\boldsymbol{\Gamma}_f$ and based on estimated states. For the first choice, Equation (3.4.8) or (3.4.9) indicates that the rank of $\boldsymbol{R}_{22}$ is only of system order, and its left factor coincides with $\boldsymbol{\Gamma}_f$. The result of SVD on $\boldsymbol{R}_{22}$ is:

$$R_{22} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ (V_2)^T \end{bmatrix} \approx U_1 S_1 V_1^T \tag{3.4.10}$$

The number of the dominant singular values (in diagonal of $\boldsymbol{S}_1$) is an estimate of the system order, denoted as $\hat{n}$ (or use AIC to determine the system order). Matrix $\boldsymbol{U}_1$ can be

taken as an estimate of Γ_f (i.e., Γ_f=U_1), which is a basis for estimation of system matrices A and C (the original MOESP, refer to (2.4.4)). System matrices B and D can be obtained from the estimated H_f matrix in (3.4.6): D is the average of block elements on the estimated H_f, and B can be obtained by LS method based on the estimated Γ_f and lower diagonal part of the estimated H_f. Here the computation for system matrices B and D is simpler than that of the original MOESP. Matrices B and D can also be obtained by pre-multiplying (3.4.1) with U_2^T then solving a LS regression (U_2^T is orthogonal to the estimated Γ_f (refer to Verhaegen, 1992).

The second scheme is based on the estimated states. Based on (3.4.7) to (3.4.10), the process states can be estimated as:

$$\hat{X}_k = S_1 V_1^T Q_2 \quad \text{or} \quad \hat{X}_k = S_1 V_1^T Q_2 / P_{IO} \tag{3.4.11}$$

Since this estimated state sequence is a linear combination of Q_2 (the first n row vectors), they satisfy the requirement shown in (3.4.4) (i.e., the orthogonality between X_k and Q_1). Then this estimated state sequence can be used to fit the state-space model for system matrices as in N4SID and CVA. The variance matrices of stochastic variables can be estimated by the fitting residuals.

In the two MOESP algorithms shown above, both Γ_f and states X_k are estimated from the same estimated predictable subspace. In Γ_f-based algorithm, there are $l(f\text{-}1)$ fitting data points in LS regression fitting for system matrices A and C (l is the number of outputs). In state-based algorithm, the number of fitting data points in LS regression for system matrices is essentially the effective data length, which is usually much longer than $l(f\text{-}1)$. Therefore, in estimating system matrices, this may result in smaller estimation errors than the Γ_f-based algorithm.

For general auto-correlated input signals, $R_{22}Q_2$ is no longer an estimate of the predictable subspace. In fact, it is Y_{f_ro}, an estimate of $\Gamma_f X_{k\_ro}$ (refer to (3.3.1). Now estimates of Γ_f and X_{k_ro} (not X_k) can be obtained based on SVD of R_{22}. Coefficient matrix H_f and state X_k can be estimated with some other techniques, such as the coefficient matrix for P_{IO_ro} is used for P_{IO} (refer to CVA_RO in Section 3.3.1; Van

Overschee and De Moor, 1995), pre-multiplying (3.4.1) by U_2 (Verhaegen, 1992, 1994) or using P_{IO} as instrumental variables for (3.4.9) (see also Li and Qin, 2000).

3.4.2 Simulation Studies for MOESP Algorithms

MOESP algorithms for an ARARX process

The first simulation example is the ARARX model (3.2.8) in section 3.2.3 but with a white noise input. 100 simulations are done for each of 5 different noise levels. The average SNRs are 105.5, 33.15, 10.0, 3.28 and 1.03 respectively. In each simulation, 500 data points are collected.

The Γ_f-based MOESP algorithm and the state-based MOESP algorithm discussed above are applied for this simulation example. Both the past and the future horizons are of 8 lag steps. The system order is determined to be 2. In order to compare the modeling results, the first 50 steps of impulse weights of the final model are compared with the true values, and the sum of the absolute errors (SAE) is used as an index for model accuracy. The results of SAE for different SNR cases are summarized in Table 3.10. For this simulation example, the state-based MOESP algorithm is better than the Γ_f-based MOESP algorithm at high noise level (i.e., small SNR, say SNR<10); at low level of noise, the Γ_f-based MOESP algorithm gives a better SAE. Student's t-test of these results confirms the conclusion.

Table 3.10 Results of MOESP algorithms from Monte Carlo simulations (ARARX process)

SNR level	105.5	33.15	10.0	3.28	1.03
Average SAE for state-based algorithm	0.1113	0.1515	0.2735	0.6148	1.2970
Average SAE for original algorithm	0.0682	0.1428	0.3163	0.9327	2.3673
Difference of the average SAE (row1 –row2)	0.0432	0.0086	-0.0428	-0.3178	-1.0704
STD of the SAE difference	0.0063	0.0142	0.0206	0.0404	0.0600
t-statistic for the difference	6.86	0.61	-2.08	-7.87	-17.84

SAE: sum of the absolute errors of the first 50 impulse weights

MOESP algorithms for an OE process

The OE model (3.2.17) in Section 3.2.3 is also employed in Monte Carlo simulations to compare the two MOESP algorithms. Similar simulation conditions as for the ARARX process are used in the study, and the average SNRs are 117.2, 36.79, 11.29, 3.61 and 1.13 respectively for different noise levels. The results are listed in Table 3.11 and confirm the conclusion that state-based MOESP gives a better result at high noise level (e.g., say SNR<1.0).

Table 3.11 Results of MOESP algorithms from Monte Carlo simulations (OE process)

SNR level	117.2	36.79	11.29	3.61	1.13
Average SAE for state-based algorithm	0.0619	0.0598	0.0750	0.0829	0.1177
Average SAE for original algorithm	0.0117	0.0208	0.0404	0.0737	0.1397
Difference of the average SAE (row1 –row2)	0.0502	0.0390	0.0346	0.0093	-0.0220
STD of the SAE difference	0.0047	0.0042	0.0062	0.0071	0.0086
t-statistic for the difference	10.68	9.29	5.58	1.31	-2.56

3.5 Application of SIMs to Industrial Data

A practical data set from a plant test of a packed-bed reactor (from Derrick Kozub) is employed to test the SIM algorithms and compare with other system identification methods. The data (uofhid1.dat) is from an SISO process and there are 251 data points in total. The input and output signals (mean-centered) are shown in Figure 3.6. Here the input signal is close to a PRBS, and the output is corrupted with a large disturbance. The input-output data plot indicates that the process has an inverse response with a positive steady state gain of about 1.0, relatively fast response with short (or no) delay.

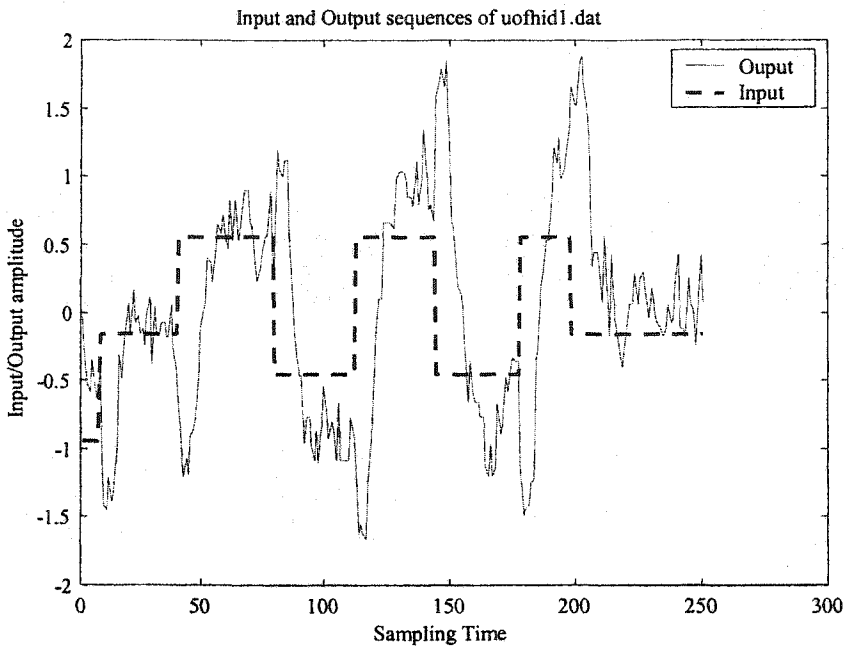


Figure 3.6 Input and output data of a plant test data (uofhid1.dat)

The original N4SID algorithm (M file in System Identification Toolbox for MATLAB) is applied for this data set. The past and future horizons are of 15 lag steps. The singular values (in logarithm) of the estimated predictable subspace are shown in Figure 3.7, and the system order is determined to be 3 (first 3 singular values are considered significant). The resultant state space model is (in the innovation form):

$$x_{k+1} = \begin{bmatrix} 0.9456 & 0.2854 & 0.0960 \\ -0.1348 & 0.7669 & -0.4422 \\ -0.0397 & -0.0825 & 0.8127 \end{bmatrix} x_k + \begin{bmatrix} 0.2707 \\ 0.8653 \\ 0.2944 \end{bmatrix} u_k + \begin{bmatrix} 0.8704 \\ 0.0187 \\ -0.4170 \end{bmatrix} e_k$$

$$y_k = \begin{bmatrix} 0.2903 & 0.8653 & 0.2944 \end{bmatrix} x_k + e_k$$

The corresponding transfer functions for the input and the noise are:

$$y_k = \frac{-0.5714z^{-1} + 1.2290z^{-2} - 0.6551z^{-3}}{1 - 2.5252z^{-1} + 2.1227z^{-2} - 0.5951z^{-3}} u_k$$

$$y_{n,k} = \frac{1 - 2.0905z^{-1} + 1.3792z^{-2} - 0.2678z^{-3}}{1 - 2.5252z^{-1} + 2.1227z^{-2} - 0.5951z^{-3}} e_k \qquad \text{var}(e_k)=0.0422$$

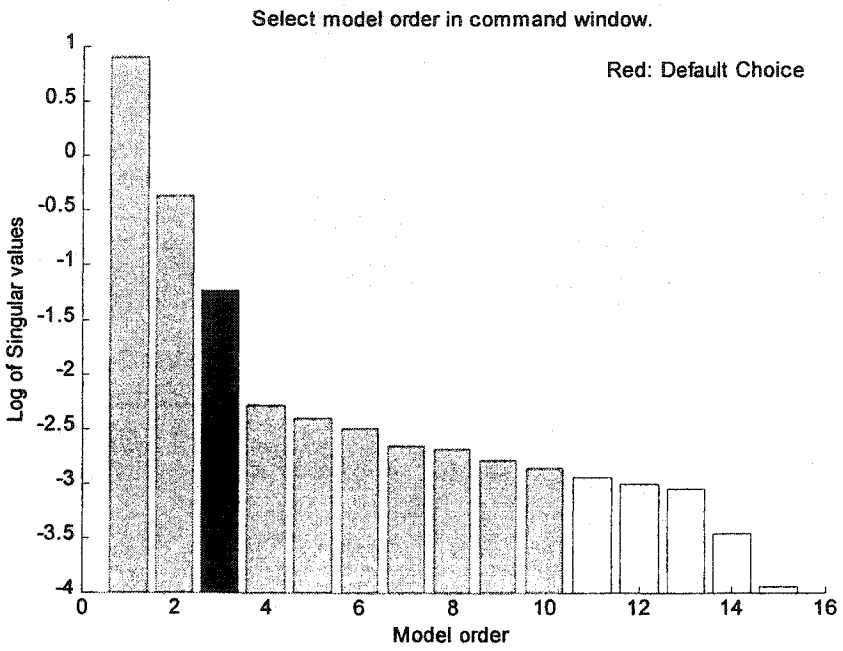


Figure 3.7 Singular values in the N4SID algorithm (uofhid.dat)

The CVA method (Adaptix software) is also applied for this data set. The optimal lag step is determined to be 9. The first 6 canonical correlation coefficients are 1.0, 0.9379, 0.5120, 0.4519, 0.3719 and 0.3540 respectively, and the system order is determined to be 3 based on AIC. The final identified model is (in transfer function form):

$$y_k = \frac{-0.7037z^{-1} + 1.3223z^{-2} - 0.5850z^{-3}}{1 - 2.2059z^{-1} + 1.6231z^{-2} - 0.3961z^{-3}} u_k$$

$$y_{n,k} = \frac{1 - 1.6653z^{-1} + 0.8522z^{-2} - 0.0906z^{-3}}{1 - 2.2059z^{-1} + 1.6231z^{-2} - 0.3961z^{-3}} e_k \qquad \text{var}(e_k)=0.0500$$

These results from SIM algorithms are compared with those from PEM and AIDA (weighted k-step ahead prediction error parametric identification, by Derrick Kozub). The steps responses are shown in Figure 3.8. All these methods show the inverse response clearly, and the steady-state gain is around 1.5. It appears that the results from SIMs are similar to those from the traditional prediction error methods.

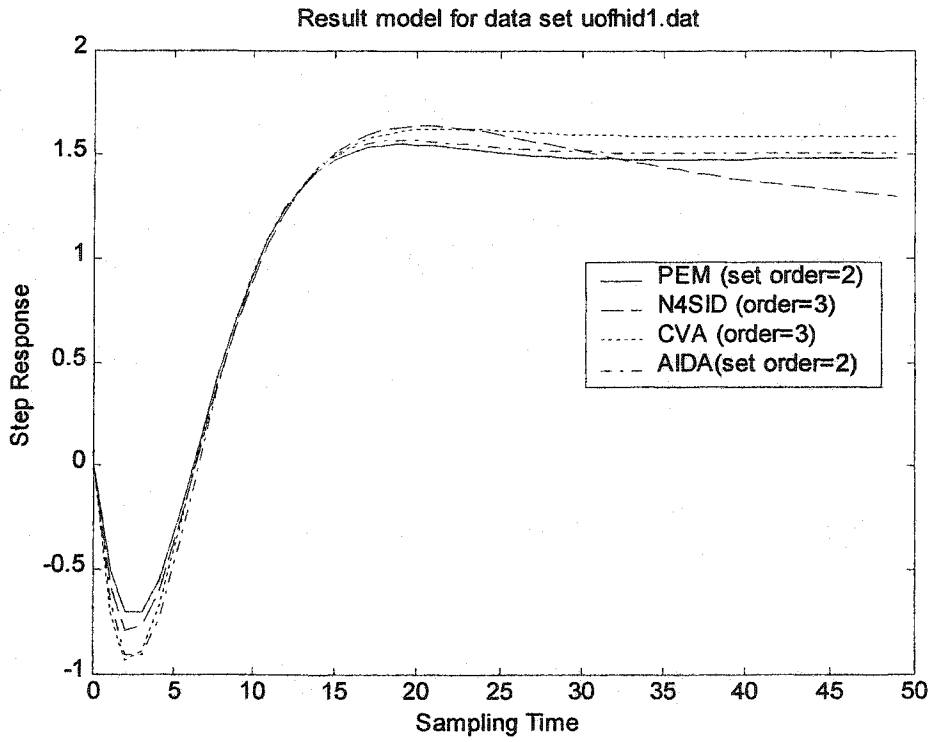


Figure 3.8 Step responses from different methods (uofhid.dat)

3.6 Conclusions

This chapter focuses on the analysis of the fundamental aspects of different SIM algorithms and on the development of the new algorithms. The essential conclusions and the key contributions of this chapter are as follows:

A multi-step state space-model is proposed in this chapter as a basis for comparing and analyzing SIMs. It clearly depicts the relationships among the current state sequence, the past data and the future data: the current state sequence is a linear combination of multiple steps of the past input and output data; the future outputs consist of the predictable subspace, the future input effects and the future stochastic signal effects; the future output data has a linear relationship with the past inputs, past outputs,

and the future inputs. This multi-step state-space model provides a uniform analytic basis for analysis of the SIM algorithms.

Based on the multi-step state-space model, the basic principles and the bias of the original N4SID algorithm are analyzed. The first step in N4SID is the oblique projection of the future outputs along the future inputs onto the past data. This result is in fact an estimate of the predictable subspace. SVD on this subspace gives the estimates of the extended observability matrix Γ_f and process states X_k. If the true process is an ARX process, the estimated predictable subspace and the estimated states are unbiased. For a general process with Box-Jenkins model (ARMAX model other than ARX), the bias on the estimated predictable subspace and states comes from the approximation error of the process with a high (but limited) order ARX model. The resultant model is asymptotically unbiased with infinite number of steps for the past horizon. Two new N4SID algorithms are also studied: the recursive one-step-ahead ARX prediction algorithm (proposed by Ljung for closed-loop case; here it is extended to the open loop case), and a proposed new algorithm based on using an estimated H_f matrix.

The basic principles are also analyzed based on the multi-step state-space model. The correct procedures for the CVA regressing U_f out algorithm (CVA_RO) are summarized. A new CVA algorithm (CVA_Hf) is proposed to remove the effects of future inputs based on an estimated H_f matrix. Both CVA algorithms are shown to give unbiased state estimates for an ARX process and biased state estimates for a general process (other than ARX process). The CVA_Hf algorithm is shown to have a better signal-to-noise ratio (SNR) than the CVA_RO algorithm.

New MOESP algorithms based on estimated Γ_f and process states X_k respectively are proposed in this chapter. The simulations indicate that the state-based MOESP algorithm gives a better result at low SNRs, which are more likely in process industries.

All these conclusions are demonstrated by simulation examples. SIMs are shown to give similar modeling results to those from the traditional system identification methods.

Appendix A3.1 Regression of Y_f against $[Y_p; U_p; U_f]$

The relationship between Y_f and $[Y_p; U_p; U_f]$ is clearly described by (3.1.9) as

$$\begin{aligned} Y_f &= \Gamma_f A^p \Gamma_p^{+} Y_p + \Gamma_f (\Omega_p - A^p \Gamma_p^{+} H_p) U_p + H_f U_f \\ &\quad + \Gamma_f (\Omega_{s,p} - A^p \Gamma_p^{+} H_{s,p}) W_p - \Gamma_f A^p \Gamma_p^{+} V_p + H_{s,p} W_f + V_f \end{aligned} \tag{3.1.9}$$

It can be written in the following brief form:

$$\begin{aligned} Y_f &= C_1 Y_p + C_2 U_p + C_3 U_f + E \\ &= \begin{bmatrix} C_2 & C_3 \end{bmatrix} U_{pf} + C_1 Y_p + E \\ &= A_1 X_1 + A_2 X_2 + E \\ &= AX + E \end{aligned} \tag{A3.1.1}$$

Here C_1, C_2 and C_3: the true coefficient matrix in (3.1.9) for Y_p, U_p and Y_f respectively

$E=E_p+E_f$: sum of the past stochastic effects E_p (terms of W_p and V_p) and the future stochastic effects (terms of W_f and V_f) in (3.1.9)

$A=[A_1\ A_2]$ $A_1=[C_2\ C_3]$ $A_2=C_1$

$X=[X_1; X_2]$ $X_1=U_{pf}=[U_p; U_f]$ $X_2=Y_p$

Different from normal linear model for ordinary LS, here the noise term E is correlated to regressor Y_p since it includes the effects of W_p and V_p (See (3.1.5)), i.e., $Exp\{EY_p^T\}=Exp\{E_pY_p^T\}\neq 0$ (Expectation). LS regression of Y_f onto $[Y_p; U_p; U_f]$ gives the following result:

$$\hat{Y}_f = L_2 U_p + L_3 U_f + L_1 Y_p \tag{A3.1.2}$$

Here L_1, L_2 and L_3 are estimates of C_1, C_2 and C_3 from LS regression respectively as:

$$\begin{bmatrix} L_2 & L_3 & L_1 \end{bmatrix} = Y_f X^T \left(XX^T\right)^{-1} \tag{A3.1.3}$$

With the result for inverse of block matrix, the above equation can be written as:

$$\begin{aligned}
&\begin{bmatrix} L_2 & L_3 & L_1 \end{bmatrix} = \begin{bmatrix} Y_f X_1^T & Y_f X_2^T \end{bmatrix} \begin{bmatrix} X_1 X_1^T & X_1 X_2^T \\ X_2 X_1^T & X_2 X_2^T \end{bmatrix}^{-1} \\
&= \begin{bmatrix} A_1 X_1 X_1^T + A_2 X_2 X_1^T + E X_1^T & A_1 X_1 X_2^T + A_2 X_2 X_2^T + E X_2^T \end{bmatrix} \cdot \\
&\quad \begin{bmatrix} \left(X_1 X_1^T\right)^{-1} + \left(X_1 X_1^T\right)^{-1} X_1 X_2^T \varphi^{-1} X_2 X_1^T \left(X_1 X_1^T\right)^{-1} & -\left(X_1 X_1^T\right)^{-1} X_1 X_2^T \varphi^{-1} \\ -\varphi^{-1} X_2 X_1^T \left(X_1 X_1^T\right)^{-1} & \varphi^{-1} \end{bmatrix} \\
&= \begin{bmatrix} A_1 - E X_2^T \varphi^{-1} X_2 X_1^T \left(X_1 X_1^T\right)^{-1} & A_2 + E X_2^T \varphi^{-1} \end{bmatrix}
\end{aligned}$$

$$\varphi = Y_p Y_p^T - Y_p U_{pf}^T \left(U_{pf} U_{pf}^T\right)^{-1} U_{pf} Y_p^T \tag{A3.1.4}$$

In open loop case, $Exp\{\boldsymbol{E}\boldsymbol{X}_1{}^{\mathrm{T}}\}=0$ for there is no correlation between inputs and the noise. However, $Exp\{\boldsymbol{E}\boldsymbol{X}_2{}^{\mathrm{T}}\}= Exp\{\boldsymbol{E}\boldsymbol{Y}_{\mathrm{p}}{}^{\mathrm{T}}\}\neq 0$, this leads to all the estimated coefficients becoming biased for a general case. Here ϕ is actually the covariance matrix of the residual from regressing out $\boldsymbol{U}_{\mathrm{pf}}$ of $\boldsymbol{Y}_{\mathrm{p}}$. Using the original variables, the detail regression coefficient matrices are:

$$\begin{aligned}
&L_1 = C_1 + E Y_p^T \varphi^{-1} \\
&\begin{bmatrix} L_2 & L_3 \end{bmatrix} = \begin{bmatrix} C_2 & C_3 \end{bmatrix} - E Y_p^T \varphi^{-1} Y_p U_{pf}^T \left(U_{pf} U_{pf}^T\right)^{-1}
\end{aligned} \tag{A3.1.5}$$

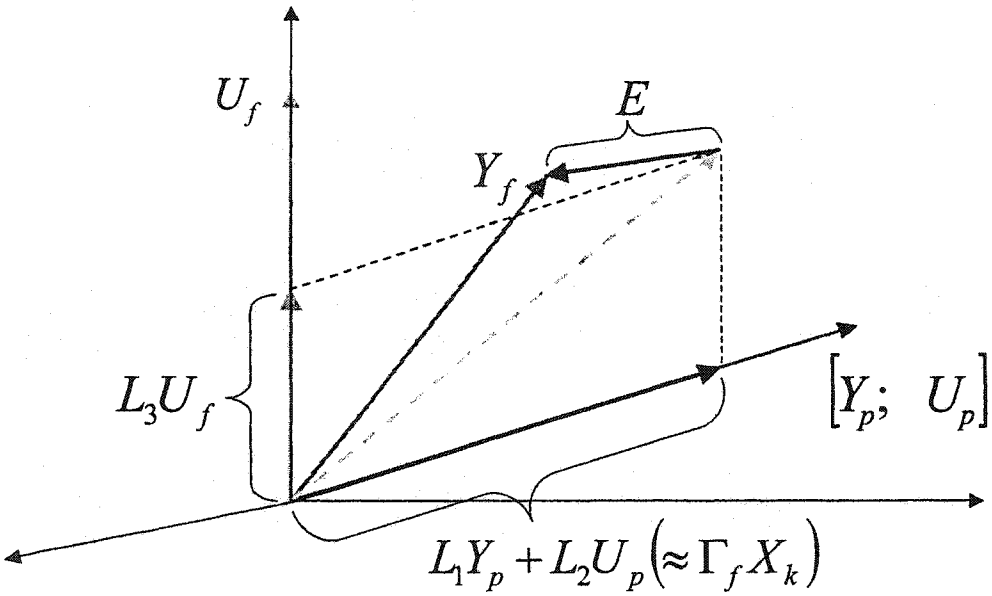
For the special case of an ARX model, i.e, $y_{\mathrm{k}}=\underline{a}^{\mathrm{T}}\underline{y}_{\mathrm{p}}+\underline{b}^{\mathrm{T}}\underline{u}_{\mathrm{p}}+\underline{e}_{\mathrm{k}}$. Here y_{k} does not include the past stochastic signal $\underline{e}_{\mathrm{p}}$ explicitly (null coefficients). Therefore, $\boldsymbol{Y}_{\mathrm{f}}$ can be expressed in $\boldsymbol{Y}_{\mathrm{p}}$, $\boldsymbol{U}_{\mathrm{p}}$ $\boldsymbol{U}_{\mathrm{f}}$ and $\boldsymbol{E}_{\mathrm{f}}$ without including $\boldsymbol{E}_{\mathrm{p}}$. Since $Exp\{\boldsymbol{E}\boldsymbol{X}_2{}^{\mathrm{T}}\}= Exp\{\boldsymbol{E}_{\mathrm{f}}\boldsymbol{Y}_{\mathrm{p}}{}^{\mathrm{T}}\}=0$, there is no bias on the estimated coefficients by LS regression (in sense of expectation):

$$\begin{aligned}
&L_1 = C_1 \\
&\begin{bmatrix} L_2 & L_3 \end{bmatrix} = \begin{bmatrix} C_2 & C_3 \end{bmatrix}
\end{aligned} \tag{A3.1.6}$$

In closed-loop case, $Exp\{\boldsymbol{E}\boldsymbol{X}_1{}^{\mathrm{T}}\}=Exp\{[\boldsymbol{E}\boldsymbol{U}_{\mathrm{p}}{}^{\mathrm{T}}\ \boldsymbol{E}\boldsymbol{U}_{\mathrm{f}}{}^{\mathrm{T}}]\}\neq 0$ for there is correlation between inputs and the past noise. The correlation between an input and the past stochastic signals is built up via the feedback controller (refer to Chapter 6 for more details). The final regression result for a closed-loop case is:

$$\begin{aligned}
&L_1 = C_1 + E Y_p^T \varphi^{-1} - E U_{pf}^T \left(U_{pf} U_{pf}^T\right)^{-1} U_{pf} Y_p^T \varphi^{-1} \\
&\begin{bmatrix} L_2 & L_3 \end{bmatrix} = \begin{bmatrix} C_2 & C_3 \end{bmatrix} - E Y_p^T \varphi^{-1} Y_p U_{pf}^T \left(U_{pf} U_{pf}^T\right)^{-1} + E U_{pf}^T \left(U_{pf} U_{pf}^T\right)^{-1} \left(I + U_{pf} Y_p^T \varphi^{-1} Y_p U_{pf}^T \left(U_{pf} U_{pf}^T\right)^{-1}\right)
\end{aligned} \tag{A3.1.7}$$

These regression coefficients are biased even for an ARX process for the correlation between inputs and the stochastic noise $Exp\{[\boldsymbol{EU}_p^T\ \boldsymbol{EU}_f^T]\}\neq 0$.



Geometric explanation of the projection in N4SID

Appendix A3.2 CVA Regressing U_f Out Algorithm

The CVA regressing U_f out algorithm estimates the process states based on the canonical correlation analysis (CCA) of the past data set and the future outputs, both of which projected to the orthogonal space of the future outputs. The detail procedures include:

- Regress U_f out of the future outputs Y_f and the past input/output data $P_{IO}=[Y_p;\ U_p]$, *i.e.*, $Y_{f\_ro}=Y_f P_{ufo}=Y_f - B_{ro\_Yf}U_f$ and $P_{IO\_ro}=P_{IO}P_{ufo}=P_{IO}-B_{ro\_PIO}U_f$. Here $P_{ufo}=(I-U_f^T(U_fU_f^T)^{-1}U_f)$ is the projection matrix onto the orthogonal space of U_f, $B_{\_ro\_Yf}=Y_fU_f^T(U_fU_f^T)^{-1}$ and $B_{\_ro\_PIO}=P_{IO}U_f^T(U_fU_f^T)^{-1}$.
- Perform CCA on Y_{f_ro} and P_{IO_ro}, denote the canonical coefficient matrix for P_{IO_ro} as J_1. The system order can be determined by the number of the dominant canonical correlation coefficients (CCCs).
- Take estimated process states from $X_k = J_1P_{IO}$ (note, not $J_1P_{IO\_ro}$, the CVs of above CCA)
- Fit the estimated X_k to the state-space model

In this algorithm, CCA is performed on the modified past data and future output data, and the coefficient matrix is used to estimate process states. Proof on the biasness issue of this algorithm is provided below.

The multi-step state space model gives the relationship between the past data and the current state sequence (3.1.6) as:

$$\begin{aligned} X_k &= A^p\Gamma_p^{\ +}Y_p + \left(\Omega_p - A^p\Gamma_p^{\ +}H_p\right)U_p + \left(\Omega_{s,p} - A^p\Gamma_p^{\ +}H_{s,p}\right)W_p - A^i\Gamma_p^{\ +}V_p \\ &= LP_{IO} + L_sS_p \end{aligned}$$

$$\text{here } L = \left[A^p\Gamma_p^{\ +} \quad \Omega_p - A^p\Gamma_p^{\ +}H_p\right] \qquad L^s = \left[-A^i\Gamma_p^{\ +} \quad \Omega_{s,p} - A^p\Gamma_p^{\ +}H_{s,p}\right]$$

$$\text{and } P_{IO} = \begin{bmatrix} Y_p \\ U_p \end{bmatrix} \qquad S_p = \begin{bmatrix} V_p \\ W_p \end{bmatrix} \tag{A3.2.1}$$

that is, the current state sequence is a linear combination of the past data.

The regressing U_f out procedures can be expressed in the following QR decomposition:

$$\begin{bmatrix} U_f \\ P_{IO} \\ Y_f \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \tag{A3.2.2}$$

Substituting results from (A3.2.1) and (A3.2.2) for the mutli-step state-space model for the future outputs (3.1.7), the result will be:

$$\begin{aligned} Y_f &= \Gamma_f L P_{IO} + \Gamma_f L^s S_p + H_f U_f + H_{s,f} W_f + V_f \\ &= \Gamma_f L (R_{21} Q_1 + R_{22} Q_2) + \Gamma_f L^s S_p + H_f H_f R_{11} Q_1 + H_{s,f} W_f + V_f \\ &= (\Gamma_f L R_{21} + H_f R_{11}) Q_1 + (\Gamma_f R_{22} + B_p^s) Q_2 + R_{33} Q_3 \end{aligned} \tag{A3.2.3}$$

Here $\boldsymbol{P}_{IO}$ includes $\boldsymbol{U}_p$ and $\boldsymbol{Y}_p$ (therefore including $\boldsymbol{W}_p$ and $\boldsymbol{V}_p$, i.e., $\boldsymbol{S}_p$). Since past stochastic variable $\boldsymbol{S}_p$ is uncorrelated with future inputs $\boldsymbol{U}_f$ (i.e., $\boldsymbol{Q}_1$), it is represented into $\boldsymbol{Q}_2$ in QR decomposition. Denote $\boldsymbol{B}_p^s \boldsymbol{Q}_2$ as the projection of $\boldsymbol{\Gamma}_f \boldsymbol{L}^s \boldsymbol{S}_p$ onto $\boldsymbol{Q}_2 = \boldsymbol{\Gamma}_f \boldsymbol{L}^s \boldsymbol{S}_p \boldsymbol{Q}_2^T$. $\boldsymbol{R}_{33}\boldsymbol{Q}_3$ includes the projection error and the effects of the future stochastic variables:

$$R_{33}Q_3 = H_f^s W_f + V_f + \Gamma_f L^s S_p - B_p^s Q_2. \tag{A3.2.4}$$

If comparing (A3.2.3) with (A3.2.2), there are the following relationships between $\boldsymbol{R}$ matrix and the system matrices:

$$\begin{cases} R_{31} = \Gamma_f L R_{21} + H_f R_{11} \\ R_{22} = \Gamma_f R_{22} + B_p^s \\ R_{33} \neq 0 \end{cases} \tag{A3.2.5}$$

The results of regressing $\boldsymbol{U}_f$ out of $\boldsymbol{Y}_f$ and $\boldsymbol{P}_{IO}$ are expressed in the QR decomposition as:

$$\begin{cases} P_{IO\_ro} = R_{22} Q_2 \\ Y_{f\_ro} = R_{32} Q_2 + R_{33} Q_3 = R_{Y1} Q_{Y1} \end{cases} \tag{A3.2.6}$$

The result of SVD on $\boldsymbol{Y}_{f\_ro}$ is:

$$\begin{cases} R_{Y1} R_{Y1}^T = R_{32} R_{32}^T + R_{33} R_{33}^T \\ Q_{Y1} = R_{Y1}^{-1} R_{32} Q_2 + R_{Y1}^{-1} R_{33} Q_3 \end{cases} \tag{A3.2.7}$$

The CCA algorithm based on QR decomposition (refer to Appendix A4.3) is used to get the coefficient matrix J_1: $J_1=V_1^{\mathrm{T}}R_{22}^{-1}$, here V_1 comes from SVD of $Q_{Y1}Q_2^{\mathrm{T}}$ as $Q_{Y1}Q_2^T = R_{Y1}^{-1}R_{32} = U_1D_1V_1^T$, i.e., eigenvector of $\left(R_{Y1}^{-1}R_{32}\right)^T\left(R_{Y1}^{-1}R_{32}\right) = R_{32}^T\left(R_{Y1}R_{Y1}^T\right)^{-1}R_{32}$.

Therefore the following relationship holds:

$$R_{32}^T\left(R_{Y1}R_{Y1}^T\right)^{-1}R_{32}\cdot V_1 = V_1\cdot D_1^2 \tag{A3.2.8}$$

Transpose both sides of the above equation and obtain the expression of V_1^{T} as:

$$V_1^T = D_1^{-2}V_1^TR_{32}^T\left(R_{Y1}R_{Y1}^T\right)^{-1}R_{32} \tag{A3.2.9}$$

Denote V_{1n} for the first n vectors in V_1, D_{1n}^{-2} for the first n row vectors in D_1^{-2} The coefficient for the first n CVs is:

$$J_{1n} = V_{1n}^TR_{22}^{-1} = D_{1n}^{-2}V_1^TR_{32}^T\left(R_{Y1}R_{Y1}^T\right)^{-1}R_{32}R_{22}^{-1} = B_1R_{32}R_{22}^{-1} \tag{A3.2.10}$$

Therefore the estimated state sequence in the CVA regression out algorithm is:

$$\begin{aligned}\hat{X}_k &= J_{1n}\cdot P_{IO} = B_1R_{32}R_{22}^{-1}\cdot P_{IO}\\ &= B_1R_{32}R_{22}^{-1}\left(R_{21}Q_1 + R_{22}Q_2\right)\\ &= B_1\left(R_{32}R_{22}^{-1}R_{21}Q_1 + R_{32}Q_2\right)\end{aligned}$$

Consider the relationship in (A3.2.5) and the future outputs in (A3.2.3):

$$\begin{aligned}\hat{X}_k &= B_1\left(\Gamma_fLR_{21}Q_1 + B_p^sR_{22}^{-1}R_{21}Q_1 + \Gamma_fLR_{22}Q_2 + B_p^sQ_2\right)\\ &= B_1\left(\Gamma_fL\left(R_{21}Q_1 + R_{22}Q_2\right) + B_p^s\left(R_{21}Q_1 + Q_2\right)\right)\\ &= B_1\left(\Gamma_fLP_{IO} + \Gamma_fL^sS_p - \Gamma_fL^sS_p + B_p^s\left(R_{21}Q_1 + Q_2\right)\right)\\ &= B_1\left(\Gamma_fX_k + \Gamma_fL^sS_p\left(Q_2^TQ_2 + Q_2^TR_{22}^{-1}R_{21}Q_1 - I_N\right)\right)\end{aligned} \tag{A3.2.11}$$

In general, the second term in the above equation is not null, and CVA_RO algorithm gives a biased estimation of the state vector. The biased term is usually very small compared to the first term, and the estimated states are close approximation of the true process states.

When the true process is an ARX (or ARARX) model, the future outputs can be expressed without involving past stochastic variables (see also Appendix A3.1),

therefore, S_p=0 (refer to equation (A3.2.1)) and the process states can be estimated unbiased.

The conclusion for the algorithm is that if the true process is of ARX (or ARARX) model, CVA_RO gives an unbiased result. For other general process models, CVA_RO gives biased estimates for process states.

If regressing U_f out of Y_f causes rank deficiency, the expectation value of $Q_{Y1}Q_2^T$ has rank less than n, and the rank of D_1 is less than n. This rank deficiency causes the inverse of D_1^2 meaningless (standing only for the noise), and the final estimated states cannot represent all the true states in the process (rank deficiency in B_1 matrix).

Appendix A3.3 CVA Algorithm based on Estimated H_f

This CVA algorithm removes the future input effects from the future outputs (an estimated predictable subspace) based on a estimated H_f, which may come from a fitting ARX model, and then estimates the process states as the canonical variates from CCA on the past data set and the modified future outputs. The detail procedures include:

- Fit ARX model with the current output y_k and the past input and output data $P_{IO}=[Y_p; U_p]$
- Construct the estimated H_f matrix based on the estimated impulse weights from above ARX model, denoted as H_{f_e}, and estimate the predictable subspace as $Y_{f\_e}=Y_f - H_{f\_e}U_f$
- Perform CCA on P_{IO} and Y_{f_e}, the number of dominant canonical correlation coefficients (CCCs) are taken as the system order $\hat{n}$, and the first $\hat{n}$ CVs from P_{IO} are taken as the estimated state sequence $X_k=JP_{IO}$
- Fit the estimated state sequence X_k to the state-space model

This algorithm gives unbiased process state estimates for ARX (ARARX) model process, and unbiased state estimates for other general processes. Proof on the biasness issue of this algorithm is provided below.

Fitting an ARX model with current outputs y_k and past input and output data $P_{IO}=[Y_p; U_p]$ will give an unbiased model if the true process is of ARX or ARARX form. Otherwise, the resultant ARX model will be an approximation of the true process. With longer past horizon, the approximation has less bias but may have larger variance if the past horizon is too long. This small bias error leads to a small bias for the estimated states in CVA.

Suppose the fitting ARX model is an unbiased estimation of the true process model, the impulse weights based on this ARX model and the constructed H_{f_e} will also be unbiased. The estimated future predictable subspace form will also be unbiased:

$$Y_{f\_e} = Y_f - H_{f\_est}U_f = \Gamma_f X_k + N_f \qquad \text{(A3.3.1)}$$

Here N_f is for the effects of the future stochastic driving signals (it also includes the bias effect of bias in the impulse weights for non ARX process).

Decompose the past data set $P_{IO}=[Y_p; U_p]$ (X in CCA) in QR form as:

$$P_{IO} = R_X Q_X$$
$$\text{with } R_X R_X^T = P_{IO} P_{IO}^T \quad Q_X Q_X^T = I \tag{A3.3.2}$$

The true state sequence can be expressed in QR decomposition form as:

$$X_k = RQ \tag{A3.3.3}$$

The multi-step state-space model gives the relationship between the past data and the current state sequence (3.1.6) as:

$$\begin{aligned} X_k &= A^p \Gamma_p^{+} Y_p + \left(\Omega_p - A^p \Gamma_p^{+} H_p\right) U_p + \left(\Omega_{s,p} - A^p \Gamma_p^{+} H_{s,p}\right) W_p - A^i \Gamma_p^{+} V_p \\ &= L P_{IO} + L_s S_p \end{aligned}$$

$$\text{here } L = \left[A^p \Gamma_p^{+} \quad \Omega_p - A^p \Gamma_p^{+} H_p\right] \qquad L^s = \left[-A^i \Gamma_p^{+} \quad \Omega_{s,p} - A^p \Gamma_p^{+} H_{s,p}\right]$$

$$\text{and } P_{IO} = \begin{bmatrix} Y_p \\ U_p \end{bmatrix} \qquad S_p = \begin{bmatrix} V_p \\ W_p \end{bmatrix} \tag{A3.3.4}$$

The above two equations give:

$$\begin{cases} RR^T = X_k X_k^T \\ QQ^T = I_n \\ Q = R^{-1} X_k = R^{-1} L R_X Q_X + R^{-1} L_s S_p \end{cases} \tag{A3.3.5}$$

Denote the QR decomposition of the estimated predictable subspace of the future outputs Y_{f_e} (Y in CCA) as $R_y Q_y$:

$$Y_{f\_e} = R_Y Q_Y \tag{A3.3.6}$$

$$\text{here } R_Y R_Y^T = Y_{f\_e} Y_{f\_e}^T \quad Q_Y Q_Y^T = I$$

$$\text{and } Q_Y = R_Y^{-1} Y_{f\_e} = R_Y^{-1} \Gamma_f R Q + R_Y^{-1} N_f = R_Y^{-1} \Gamma_f L R_X Q_X + R_Y^{-1} \Gamma_f L_s S_p + R_Y^{-1} N_f$$

In CCA for $\boldsymbol{P}_{IO}$ and $\boldsymbol{Y}_{f\_e}$, the canonical variates from $\boldsymbol{P}_{IO}$ is calculated as $J\boldsymbol{P}_{IO}$, where coefficient matrix $J = V^T R_X^{-1}$ and V is the right singular vector matrix from SVD of the covariance matrix:

$$Q_Y Q_X^T = UDV^T \tag{A3.3.7}$$

Now, let's look at the detail of the sample covariance matrix of $\boldsymbol{Q}_Y$ and $\boldsymbol{Q}_X$ in left side of the above equation:

$$Q_Y Q_X^T = \left(R_Y^{-1}\Gamma_f L R_X Q_X + R_Y^{-1}\Gamma_f L_s S_p + R_Y^{-1} N_f\right) Q_X^T = R_Y^{-1}\Gamma_f L R_X + R_Y^{-1}\Gamma_f L_s S_p Q_X^T \tag{A3.3.8}$$

The expectation value of $\boldsymbol{N}_f\boldsymbol{Q}_X^T$ (third term in the above equation) is null for there is no correlation between the future stochastic signals and the past input and output data. So the result of matrix V can be expressed as:

$$V^T = D^{-1}U^T R_Y^{-1}\Gamma_f\left(LR_X + L_s S_p Q_X^T\right) \tag{A3.3.9}$$

Therefore the canonical variates from $\boldsymbol{P}_{IO}$ can be expressed as:

$$\begin{aligned} J \cdot P_{IO} &= V^T R_X^{-1} \cdot R_X Q_X = D^{-1}U^T R_Y^{-1}\Gamma_f\left(LR_X Q_X + L_s S_p Q_X^T Q_X\right) \\ &= D^{-1}U^T R_Y^{-1}\Gamma_f\left(LP_{IO} + L_s S_p - L_s S_p\left(I - Q_X^T Q_X\right)\right) \\ &= D^{-1}U^T R_Y^{-1}\Gamma_f\left(X_k - L_s S_p\left(I - Q_X^T Q_X\right)\right) \end{aligned} \tag{A3.3.10}$$

This CVA algorithm takes the first n canonical variates from $\boldsymbol{P}_{IO}$ as the estimated states. The system order n is estimated by the number of dominant canonical correlation coefficients or is determined by other methods, such as AIC.

When the true process is an ARX (or ARARX) model, the fitted ARX model is unbiased, and so are the impulse weights and the $\boldsymbol{Y}_{f\_e}$. The multiple future outputs and the current states can be predicted in an unbiased way without involving past stochastic variables, therefore $\boldsymbol{L}_s\boldsymbol{S}_p$=0 (refer to equation (A3.3.4)), and the first n canonical variates are linear combinations of the true process states without bias. When the true process is of ARX model, there is an approximation in the fitted ARX model with limited order, and the impulse weights and the estimated states have small bias. The essential bias comes from the correlation between the past stochastic signals and the past outputs.

The conclusion for this CVA algorithm is that if the true process is of ARX (or ARARX) model, CVA gives an unbiased result; for other general process models, CVA gives a biased result for the estimated ARX model and the estimated states.

4 Relationships between SIMs and LVMs

As seen in Chapter 2 and Chapter 3, CVA method is based on CCA method. CVA is one of the SIMs, and CCA is one of the Latent Variable Methods (LVMs, see Section 4.1). Their close relationship naturally incurs curiosity about the general relationship between these two categories of modeling methods; however, SIMs and LVMs are studied by different people and applied in different fields. LVMs are studied and used in statistics and chemometrics, and SIMs are studied and applied in system identification. Up until this time, there has been little exploration of the relationships between LVMs and SIMs. There are many similarities among these methods, such as both methods are used to build process models via a set of much lower dimensional variables as linear combinations of the original variables. These similarities easily cause confusion between these methods and in applications.

The objective of this chapter is to show the relationships between SIMs and LVMs with emphasis on their connections, to find out in particular the connection between N4SID and the LVMs. The differences between these methods will also be studied in order to produce a clear picture of their relationships and to provide a guideline for application of these methods.

4.1 LVMs for Dynamic Process Modeling

4.1.1 A General Introduction to LVMs

In general, Latent Variable Methods (LVMs) relate data sets X ($N\times k$) and Y ($N\times m$). Each row of data set X usually consists of process measurements or operational variables, and each row of Y includes the corresponding quality variables or process outputs. The general latent variable regression model (Martens and Naes, 1991; Burnham

et al., 1999) assumes that both data sets X and Y are linear functions of a smaller number (A, $A \ll k$) of underlying latent variables (LVs):

$$X=TP^{T}+E \tag{4.1.1}$$

$$Y=TC^{T}+F \tag{4.1.2}$$

Here each column of the T ($N \times A$) is a latent variable (LV), generally defined as a linear combination of X variables

$$T=XW \tag{4.1.3}$$

which, in some sense, contains an optimal amount of information for the defined problem. The columns of P, C and W are loading vectors, which indicate independent directions of variation in the process, and **E** and **F** are error matrices. This kind of data structure is common in the real world where processes are driven by only a few events, such as raw material variations, impurities, temperature fluctuation, etc., and the effects of these few variables show up in all the process variables. In practice, there might be several hundred variables in a process, but the plant will vary in only a few independent directions since all the variables are driven by only a limited number of independent disturbances (or events) and constrained by the process dynamic characteristics. The space in which the process varies is defined, in some sense, by the lower dimensional space of the latent variables.

Principal Component Analysis (PCA, and Principal Component Regression, PCR), Partial Least Square or Projection to the Latent Structure (PLS), Canonical Correlation Analysis (CCA) and Reduced Rank Analysis (RRA, also referred to as Redundancy Analysis) can all be referred to as latent variable methods. They all define a subspace of the system in terms of a reduced number of latent variables $T=XW$, but LVs from different LVMs are to optimize different objective functions. The LV in PCA on X is to have maximum variance. In PLS, the paired LVs from X and Y respectively are to have the maximum covariance. RRA is to find a linear combination of X, which can explain the maximum variance of Y. The paired LVs from CCA have the maximum correlation. A common objective function framework for all these latent variable methods has been presented by Burnham et al. (1996), and summarized in Table 4.1. The

definition of the latent variables and the loading vectors for the different methods are given in this table. Perhaps PCR and PLS can be called true latent variable methods since they provide estimates of latent variables (from X space) that span both the X and Y spaces, while CCA and RRA provide latent variables that attempt to span only the Y space (and not the X-space).

Table 4.1 Brief Summary of Latent Variable Methods

Method	PCA/PCR	PLS	RRA	CCA
Objective (for a-th LV dimension a=1, 2, …, A)	Max $\mathbf{p}_a^T\mathbf{X}^T\mathbf{X}\mathbf{p}_a$	Max $\mathbf{w}_a^T\mathbf{X}_a^T\mathbf{Y}\mathbf{c}_a$ $\mathbf{X}_1=\mathbf{X}$ $\mathbf{X}_{a+1}=\mathbf{X}_a-\mathbf{t}_a\mathbf{p}_a^T$ $\mathbf{p}_a=\mathbf{t}_a^T\mathbf{X}_a/\mathbf{t}_a^T\mathbf{t}_a$	Max $\mathbf{w}_a^T\mathbf{X}^T\mathbf{Y}\mathbf{c}_a$	Max $\mathbf{w}_a^T\mathbf{X}^T\mathbf{Y}\mathbf{c}_a$
Constraints	$\mathbf{p}_a^T\mathbf{p}_a=1$	$\mathbf{w}_a^T\mathbf{w}_a=1$, $\mathbf{c}_a^T\mathbf{c}_a=1$	$\mathbf{w}_a^T\mathbf{X}^T\mathbf{X}\mathbf{w}_a=1$, $\mathbf{c}_a^T\mathbf{c}_a=1$	$\mathbf{w}_a^T\mathbf{X}^T\mathbf{X}\mathbf{w}_a=1$, $\mathbf{c}_a^T\mathbf{Y}^T\mathbf{Y}\mathbf{c}_a=1$
LV meaning	Maximize variance of X	Maximize the covariance	Maximize the predictable variance in Y	Maximize the correlation
Calculation of LVs	$\mathbf{t}_a=\mathbf{X}\mathbf{p}_a$	$\mathbf{t}_a=\mathbf{X}\mathbf{w}_a^*$ $\mathbf{W}^*=\mathbf{W}^T(\mathbf{P}^T\mathbf{W})^{-1}$	$\mathbf{t}_a=\mathbf{X}\mathbf{w}_a$	$\mathbf{t}_a=\mathbf{X}\mathbf{w}_a$
Loading vector: is eigenvector of matrix:	$\mathbf{p}_a$ $\mathbf{X}^T\mathbf{X}$	$\mathbf{w}_a^*$ $\mathbf{X}_a^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}_a$	$\mathbf{w}_a$ $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}$	$\mathbf{w}_a$ $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{X}$
Orthogonality ($i\neq j$)	$\mathbf{p}_i^T\mathbf{p}_j=0$ $\mathbf{t}_i^T\mathbf{t}_j=0$	$\mathbf{w}_i^T\mathbf{w}_j=0$ $\mathbf{t}_i^T\mathbf{t}_j=0$	$\mathbf{t}_i^T\mathbf{t}_j=0$	$\mathbf{t}_i^T\mathbf{t}_j=0$
Variance of LV	generally not unit	generally not unit	unit variance	unit variance
Regression coefficient matrix (based on first a LVs)	$\mathbf{P}_a(\mathbf{P}_a^T\mathbf{X}^T\mathbf{X}\mathbf{P}_a)^{-1}\mathbf{P}_a^T\mathbf{X}^T\mathbf{Y}$	$\mathbf{W}_a^*(\mathbf{W}_a^{*T}\mathbf{X}^T\mathbf{X}\mathbf{W}_a^*)^{-1}\mathbf{W}_a^{*T}\mathbf{X}^T\mathbf{Y}$	$\mathbf{W}_a\mathbf{W}_a^T\mathbf{X}^T\mathbf{Y}$	$\mathbf{W}_a\mathbf{W}_a^T\mathbf{X}^T\mathbf{Y}$

Note 1: $\mathbf{t}_a$, $\mathbf{p}_a$,$\mathbf{w}_a$, and $\mathbf{w}_a^*$ are the a-th column vector of **T**, **P**, **W** and **W*** respectively

Note 2: $\mathbf{T}_a$, $\mathbf{P}_a$,$\mathbf{W}_a$,$\mathbf{C}_a$ and $\mathbf{W}_a^*$ are matrices consist of the first a columns of **T**, **P**, **W**, **C** and **W*** respectively

Once the Latent variable structure has been determined by PCA, PLS, RRA or CCA, process outputs can be predicted by the model in latent variable form or by the following regression model form:

$$\hat{Y} = T_a C_a^T = XW_a C_a^T = X\hat{\beta}_a \qquad (4.1.4)$$

This regression procedure is referenced as PCR, PLS(R), RRA or CCR respectively. With the same data set X and Y, the regression coefficient matrix will be different if a different number of LVs are used (i.e., different model result). Suppose there are only n LVs in the true process ($n<<k$), the true process can be described by n LVs by a particular LVM. On one hand, if a LVs are used ($a<n$), the process will be under-modeled and the resultant model will be biased. On the other hand, if more LVs are used than necessary ($a>n$), the extra LVs only describe the noise in the system and cause a large variance for the resultant model. Furthermore, if all the LVs are employed, the resultant model will be equivalent to the LS regression model.

In the application of LVMs to process modeling, the first issue is to decide which LVM should be chosen for the problem, and this is determined by the objective of the problem. The second important issue is to determine how many LVs should be used in the latent model model, and several general criteria have been developed for this purpose, such as significance of the eigenvalue, cross validation, etc. Other issues, such as how to apply LVMs to dynamic processes, what kind of data to use, what kind of model can obtained, and what kind of applications, will be discussed in the next subsection. Discussion of these issues will make the relationship between LVMs and SIMs clearer.

4.1.2 LVMs for Dynamic Process Modeling

Different from a static process, the future outputs of a dynamic process are indeed affected by the past outputs and inputs. In order to capture the dynamic relationship of the process, it is necessary to use the past input and output variables to predict the future outputs. Using lagged process variables for X and Y data sets in LVMs becomes a natural choice, that is,

$$X=[Y_p^T, U_p^T]$$
$$Y=Y_f^T \qquad (4.1.5)$$

Here Y_p, U_p and Y_f are the data matrices defined in Section 2.1. In LVMs, data matrices are conventionally arranged with observations (time points) in rows, while data

observations in columns are more convenient for SIMs. This notation difference does not affect the essence of the date sets.

The lagged process inputs and outputs in the data sets X and Y includes the process dynamics. The past information related to the future outputs dynamics is extracted into the latent variables (LVs), and the process future outputs are predicted based on these LVs by LS regression as in (4.1.4). The process dynamics is expressed by the coefficient matrix in (4.1.4), or written in detail as $Y_f=L_1Y_p+L_2U_p$. In fact, this provides a multi-step-ahead prediction model in ARX form. LVMs have been applied in this approach to obtain dynamic models. Brillinger (1975) discussed the theory and applications of PCA for time series in the frequency domain. A brief discussion of some early publications on the use of lagged variables in PCA for dynamic modeling is contained in Jackson (1991). This approach to dynamic PCA has been used in various applications (Wise et al., 1991; Ku et al., 1995; Hartnett et al., 1999). Use of lagged variables in PLS for dynamic modeling was first suggested by Wold et al. (1983) and has been used in various applications (MacGregor et al., 1991; Dayal et al., 1996; Ergon, 1998). It has been shown that the resulting dynamic models are generally better in the sense of mean square error, etc., than those obtained from least squares and prediction error methods. Box and Tiao (1977) presented a CCA approach to multivariate time series analysis, where the objective was to reduce the dimension of the problem by finding sets of orthogonal latent variables that were highly predictable and others that were unpredictable (white noise). This approach has been used in designing reduced-dimension control by MacGregor and Wong (1980) and Jutan et al. (1984). Negiz and Cinar (1997) used CCA in this approach to build a dynamic model for a milk pasteurization process.

Another approach for incorporating dynamics into latent variable models involves using dynamic models for the inner relationships between the latent vectors of the non-lagged input and output space (Kaspar et al., 1993; Lakshminarayanan et al., 1997). This approach is analogous to using a dynamic transfer function model in place of a

polynomial or other nonlinear model in nonlinear PLS (Wold et al., 1989). This approach is not as popular as the previous one of lagging the input and output variables.

The special objective of each LVM determines the information modeled by this method when applied to $X=[Y_p^T, U_p^T]$ and $Y=Y_f^T$. As shown in Table 4.1, PLS tries to explain both the variations in the past data and the future outputs as well as their correlation. By also modeling the X-space, PLS easily handles missing data and tests for outliers or abnormal observations in the X-space. This is an important feature in process monitoring; however, by compromising to explain the variation in past data, PLS becomes less effective in explaining the relationship between the past data (X) and the future outputs (Y). Section 4.3 will provides more detailed discussion on this point. In dynamic PCA, LVs only explain the past data set without intending to explain the future outputs, and so the efficiency for building a dynamic model becomes even worse.

Both RRA and CCA focus on explaining the future behaviour based on the past data. CCA on $X=[Y_p^T, U_p^T]$ and $Y=Y_f^T$ focuses on maximizing the correlation between the past data and the future outputs, and therefore the LVs form the past data set provide the best predictability of the future outputs. RRA on $X=[Y_p^T, U_p^T]$ and $Y=Y_f^T$ aims to explain as much as possible the variation in future outputs based on the LVs from the past data set. Both RRA and CCA coincide with the goal of the dynamic modeling and provide sound approaches to build dynamic models. The difference between RRA and CCA exists in a different sense of the best predictability of the future outputs based on the past data. RRA is in the sense of absolute variance, while CCA is the relative sense of variance.

Basically two types of data are collected from a dynamic process for building a dynamic model: well-designed experimental data and normal operation data. For the data collected from well-designed experiments, inputs are persistently exciting of a sufficiently high order and there is no correlation (orthogonal) or little correlation between different inputs. These conditions are necessary for obtaining causal dynamic models in system identification. However, in general, there are more data collected under the normal operation condition. In practice, all measured variables (not only the

manipulated variables and the controlled variables, but also many other monitoring variables) may be recorded historically by the control system, and these historical data are usually normal operation data. The variables in normal operation data are usually highly correlated (strong feedback between outputs and inputs or strong correlation between two measurements, e.g., temperatures at trays of a distillation tower). When the process is operating under the normal condition, only a few sources of "common-cause" variations present in the process and drive all the process variables moving up or down in a correlated manner. The number of independent "common-cause" variations is much less than the number of measurements, and the space spanned by the measured variables is essentially never full rank. LVMs are therefore used to extract these common-cause variations into the first few LVs from the highly correlated measurement variables.

The information nature in a data set primarily determines what kind of model can be obtained from the data, causal model or correlation model. If two variables are perfect (or highly) correlated, their causal effects will be completely (or highly) confounded no matter what kind of modeling method is employed. For normal operation data, both SIMs and LVMs can only give correlation models rather than causal models because of the high correlation in the data set.

Causal or correlation model. Even for the well-designed experimental data, the resultant LVM of (4.1.4), based on the dynamic process data of (4.1.5), in general is not a causal model. This is due to the fact that the future outputs data $\boldsymbol{Y}_f$ in $\boldsymbol{Y}$ contains not only the effects of $\boldsymbol{X}=\boldsymbol{P}_{IO}$ (predictable subspace based on past data) but also the future input effects $\boldsymbol{H}_f\boldsymbol{U}_f$ as shown clearly in (3.1.7). The final model of (4.1.4) will include part of the future input effects via the correlation between $\boldsymbol{H}_f\boldsymbol{U}_f$ and $\boldsymbol{P}_{IO}$ if the process inputs are auto-correlated. Therefore, the resultant model will only give a correlation relationship between inputs and outputs instead of a causal relationship. This is why the future input effects are removed from $\boldsymbol{Y}_f$ in SIM algorithms for causal models (refer to Section 3.2 and Section 3.3). In two special cases, the final result models from LVMs will be causal. The first case is where the inputs are not auto-correlated (e.g., white noise signals), and therefore the future input effects have no correlation with the past data. In this situation,

the future input effects would lead to larger variance for the model parameters than if the future input effects had been unaccounted from Y. The second case is where the Y data set contains only outputs of current step (f=1) and no instantaneous reaction is in the process (system matrix D is null in state-space model), and therefore no future input effects exist in Y data set. In fact, in this second case the result is exactly the ARX fitting model if all LVs are used.

Dynamic models based on process data can be used for process control, optimization and monitoring. However, the choice of models or modeling methods must depend on the specific application objective. In the context of control system design, it is necessary to have cause-effect dynamic models between manipulated variables (MVs) and the controlled variables (CVs). The models will be used to predict the transient response of the controlled variables to moves made in the manipulated variables, and the models will also be inverted in the control equations to calculate the manipulated variable changes required to make the controlled variables follow desired trajectories. Similarly, in process optimization, the input variables are adjusted deliberately to achieve the optimal values of the objective function, and the correlation pattern in the input space will be broken deliberately. Therefore causal models (based on experimental data) are required for process control and optimization. System identification methods, including SIMs, are chosen to build the required models, where the causal relationship between the process inputs and outputs is the only objective, without intention of modeling the covariance structures in the inputs or the outputs.

In the case of process monitoring, causal models are usually neither required nor even desirable; rather, a model of the correlation structure presented under normal or "common-cause" operation of the process is required. This is an important but subtle difference of model requirement between process monitoring and process control/optimization. If the sampling intervals are short relative to the settling time of the process, the model must be a dynamic one involving lagged process variables. Under normal operation, there is a certain correlation structure among the variables in past data $X=[Y_p^T;\ U_p^T]$ space (including both contemporaneous and temporal correlation). Any

deviation from this normal correlation structure indicates a potentially abnormal situation; therefore all the correlation structures in X space, in Y space and between the two spaces should be modeled. Therefore, LVMs based on lagged variables are applied for process monitoring, and PCA or PLS would be better choices than RRA and CCA.

Much of the literature on process monitoring of continuous processes assumes that the sampling intervals are long and the process dynamics are not important. In this case, the latent variable models are built using only data available at each individual time interval, that is, the X and Y matrices do not include lagged values of the variables. However, if under common-cause variation, the process data exhibit significant temporal correlations, a dynamic model may be needed. In this situation, the basic philosophy and approach to multivariable SPC (statistical process control) are unaltered. Lagged values of the observations are simply used in the X matrix, and a latent variable model is built. The latent variables defined in (4.1.3) will now be linear combinations of the process variables at current time and at several past lags. The control charts based on the score plots, or Hotelling's T^2, and the Squared Prediction Error (SPE) statistics will again be constructed in the same manner as for steady-state data. LVMs in the approach for process monitoring have become widely accepted (Kresta et al., 1991; Slama et al., 1992; Kourti et al., 1995;). On the other hand, *batch* processes are invariably transient or dynamic in nature, and so the multivariate analysis and monitoring approaches for batch processes have always been based on dynamic PCA and PLS models using lagged variables in the X data (Nomikos et al., 1994, 1995, Kourti et al., 1995).

4.2 Relationship between N4SID and RRA

As shown in Section 3.3, CVA is based on CCA, and this shows a connection between SIM and LVM. As a natural extension to this connection, one might be curious whether a LVM corresponds to N4SID or not, and how they are related to each other. The

goal of this section is to answer to these questions, and the ultimate objective is to explore the relationship between N4SID and LVM.

4.2.1 Equivalency between RRA and LS with PCA

In N4SID, SVD is applied on the oblique project result (part of the LS projection of $\boldsymbol{Y}_f$ on $[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{U}_f]$) to get the estimated states. From Section 4.1, it is clear that SVD is a procedure to perform PCA, which explains the largest possible variance of the oblique projection with a given the number of latent variables. This is similar to the objective of RRA as shown in Section 4.1. This naturally raises the question of whether a general relationship exists between RRA and LS with PCA.

In fact, for general data matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, RRA($\boldsymbol{X}$, $\boldsymbol{Y}$) is equivalent to PCA($\boldsymbol{Y/X}$) where $\boldsymbol{Y/X}$ is the projection of $\boldsymbol{Y}$ onto $\boldsymbol{X}$. Both LVMs explain the same subspace in $\boldsymbol{Y}$, and the latent variables from these two different LVMs are essentially the same. It can be expressed as:

$$\text{PCA}(\boldsymbol{Y/X})=\text{RRA}(\boldsymbol{X},\boldsymbol{Y}) \tag{4.2.1}$$

A strict proof of this general conclusion is included in Appendix 4.1. This was also partially implied in Davies and Tso (1982) for a numerical computation method for RRA.

The conclusion is rational from the explained variance point of view. RRA($\boldsymbol{X}$, $\boldsymbol{Y}$) solves sequentially for a number of orthogonal linear combinations of the $\boldsymbol{X}$ variables with maximum explanation of the total variance in $\boldsymbol{Y}$. The LS projection of the data set $\boldsymbol{Y}$ onto $\boldsymbol{X}$ gives all the possible components in $\boldsymbol{Y}$ that can be modeled by $\boldsymbol{X}$, i.e., the total possible variance of $\boldsymbol{Y}$ can be explained by $\boldsymbol{X}$, and then a PCA on the projection (i.e., PCA($\boldsymbol{Y/X}$)) gives the best approximation (maximum variance) of this projection that can be obtained with a specific number of latent variables as linear combinations of $\boldsymbol{Y/X}$. As one can see, the projection $\boldsymbol{Y/X}$ is a linear combination of $\boldsymbol{X}$ variables, therefore the principal components (PCs) from PCA($\boldsymbol{Y/X}$) are also linear combinations of $\boldsymbol{X}$. In brief, PCA($\boldsymbol{Y/X}$) finds the whole predictable space (full rank) of $\boldsymbol{Y}$ based on $\boldsymbol{X}$, and then PCA shrinks the rank of this predictable space while keeping the variance as large as possible.

RRA(X,Y) does the same thing by sequentially increasing the number of latent variables used until to that specific number (rank constrain). Both methods find out the same rank-specific subspace of Y predictable by X and with the greatest variance, but PCA(Y/X) goes from the full rank to the specific rank while RRA(X,Y) goes from rank one to the specific rank.

The i-th PC from PCA(Y/X) only has a magnitude difference compared to the i-th LV from RRA(X,Y). They are equivalent to each other in the sense of predictability. Though these two methods give the same result in theory, this does not necessarily imply they have the same performance numerically. When the X data set is ill-conditioned, projection of Y onto X will be non-robust to measurement errors, and the projection will have a large variance. In essence, N4SID performs an ill-conditioned projection first, followed by a well-conditioned SVD (PCA) step. However, RRA avoids this ill-conditioned LS regression step by directly selecting the high variance LV space.

4.2.2 Relationship between N4SID and RRA

The original N4SID algorithm performs a LS projection of Y_f onto $[Y_p; U_p; U_f]$:

$$\hat{Y}_f = L_1 Y_p + L_2 U_p + L_3 U_f \tag{3.2.1}$$

Then it takes the n largest PCs from PCA on $L_1Y_p+L_2U_p$ (see (3.2.5)) as the estimated state variables. Based on the property of LS regression, it is obvious that $L_1Y_p+L_2U_p$ is equal to the projection $Y=Y_{f\_e}=Y_f-L_3U_f$ onto the past data $X=P_{IO}=[Y_p; U_p]$, i.e.,

$$Y/X = Y_{f\_e}/P_{IO} = L_1Y_p+L_2U_p=[L_1 \;\; L_2]P_{IO} \tag{4.2.2}$$

Based on the general conclusion about RRA and LS with PCA shown in the last subsection, the following equivalency exists:

$$\text{PCA}(Y_{f\_e}/P_{IO}) = \text{RRA}(P_{IO}, Y_{f\_e}) \tag{4.2.3}$$

This equivalency between N4SID procedures and RRA clearly indicates that the states estimated in N4SID are actually the latent variables from RRA on the past data and the future outputs with the future input effects removed.

As shown in Section 3.1, L3 is an estimate of H_f, and $Y_{f\_e}=Y_f-L_3U_f$ is an estimate of the predictable subspace $\Gamma_f X_k$ with the future noise. Projection of Y_{f_e} onto P_{IO} in (4.2.2) is an estimate of $\Gamma_f X_k$. The first n PCs from the left side of (4.2.3) have the maximum explanation of the total variance of the $\Gamma_f X_k$ estimate for given rank restriction of n, and these PCs are estimate of the state sequence X_k shown as linear combination of P_{IO}. The same state estimation is realized directly by n LVs from right side of (4.2.3) as n linear combinations of P_{IO} with the greatest explanation of total variance of Y_{f_e} under the rank constriction. It is clear that the rank restriction comes from the rank limitation of the predictable subspace $\Gamma_f X_k$, and the system order can be determined by the number of statistically significant PCs from PCA or LVs from RRA in practical dynamic modeling.

4.2.3 Alternative N4SID algorithms based on RRA

Equation (4.2.3) clearly indicates that N4SID is based on RRA just as CVA is based on CCA. This relationship not only shows the basic idea of N4SID but also provides more insight into the relationships between SIMs and LVMs. The first step in the original N4SID algorithm is a LS regression of Y_f onto $[Y_p; U_p; U_f]$, and this step may suffer from an ill-conditioning problem due to the high correlation in $[Y_p; U_p; U_f]$. The relationship (4.2.3) between N4SID and RRA gives heuristic guidance on alternative approaches to improve the performance of N4SID, such as forcing L_3 to follow the structure of H_f or using other methods to estimate H_f. An improved estimate of H_f will lead to a better estimate of the predictable subspace Y_{f_e}, and better state estimates can be obtained by RRA.

N4SID-tri (lower triangle L_3 matrix):

As shown in Section 3.2, the coefficient matrix L_3 in N4SID is an estimate of the H_f (see definition in Section 2.1). H_f is a block lower triangle matrix while its estimate L_3 from N4SID is a full matrix (the upper triangular blocks are not zero due to regression errors). What this lower triangle feature of H_f really means is that inputs at a specific time

point has no effect on the outputs before that time point. This is apparent from the causal-effect point of view. Matrix L_3 should be consistent with the feature. Following this causal-effect feature, the upper triangle blocks of L_3 are set to null. Blocks on the diagonal can also be set to null if there is a priori knowledge that no instantaneous interaction exists in the process. The block entries on a lower diagonal of H_f are equal, and correspond to impulse weights (see Section 2.1). Following this feature, a further step in improving the accuracy of L_3 as an estimate of H_f is to set the element blocks on each lower diagonal of L_3 to their mean values respectively. These mean values should be better estimates of the first *f* step impulse weight blocks. This new estimated H_f is used to remove the effect of the future inputs on Y_f for Y_{f_e}, and LVs from RRA(P_{IO}, Y_{f_e}) will be better estimates for the state variables.

N4SID-Hf (H_f estimated based on ARX model)

Alternatively, as an estimate of H_f, matrix L_3 can be constructed based on the impulse weights from fitted ARX model (refer to Section 2.1), and then $Y_{f\_e}=Y_f-L_3U_f$ can be calculated. This procedure is similar to N4SID-ARX in Section 3.2, where Y_{f_e} is projected onto P_{IO} and states are obtained by SVD (PCA) on this projection result. Here, based on the relationship between N4SID and RRA, the states estimated directly by RRA(P_{IO}, Y_{f_e}).

N4SID-RO (regressing out U_f)

Another method for estimating the predictable subspace is to regress U_f out of both P_{IO} and Y_f to yield P_{IO_ro} and Y_{f_ro} respectively, and then LS regress Y_{f_ro} against P_{IO_ro} for the regression coefficient matrix $L_{\_ro}$. This coefficient matrix $L_{\_ro}$ can be proved to be exactly the same as [L_1 L_2] from N4SID (see Appendix 4.2). This result was also implied in Van Overschee and De Moor (1995) but without a clear explanation. The result of $L_{\_ro}P_{IO}$ (not $L_{\_ro}P_{IO\_ro}$) is an estimate of the predictable subspace, and the first *n* PCs of this subspace are the estimated state variables.

The above LS regression result $L_{\_ro}$ can also be obtained by RRR(P_{IO_ro}, Y_{f_ro}) when all LVs are used. As shown above, only n LVs out of all LVs from RRR(P_{IO_ro}, Y_{f_ro}) are related to process states, and the rest LVs are for the noise in the process. Therefore, the coefficient matrix can be obtained by using only the necessary LVs, and the result may be better than using all the LVs due to leaving out the LVs for noise.

N4SID-ROP (regressing P_{IO} out)

Another method for estimation of the L_3 matrix is to regress P_{IO} out of both U_f and Y_f data sets (result denoted as U_{f_rop} and Y_{f_rop} respectively), and then regress Y_{f_rop} against U_{f_rop}. The result can be shown to be exactly the same as L_3 from N4SID apart from numerical errors (see Appendix 4.2). When the variables in P_{IO} are highly correlated, there is the ill-conditioning problem with the first step of regressing P_{IO} out. Ridge regression can be used for this step for a smaller variance with the expense of introducing some bias. The final result of RRA(P_{IO}, Y_f-L_3U_f) based on the estimated L_3 may give better results if an appropriate ridge parameter is used for the ridge regression.

N4SID-RL3 (L_3 estimated by RRR)

A general method in dealing with the ill-conditioning problem is to use a latent variable regression method (LVR) instead of LS for the first step of N4SID. Reduced rank regression RRR([P_{IO}; U_f], Y_f) is a good choice in the same sense of explaining the greatest total variation but with the limitation of LVs used (rank restriction). The LVs found by RRA are ranked in order of the amount of variance of Y_f explained using [P_{IO}; U_f]. Determining the number of LVs to use for this RRR is a compromise between having small bias and small variance. With more LVs used, the RRR regression coefficient has a smaller bias but a larger variance. In practice, one can use AIC or the significance of a LV in the explanation of the total variance of Y_f to determine the number of LVs to be used. This RRR gives the regression coefficient matrix for P_{IO}, which can be used directly for the estimation of the predictable subspace and the process states (this method

is denoted as N4SID-RLP), and it also gives the regression coefficient matrix for $\boldsymbol{U}_f$, which can be used for calculation of $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ (this method is denoted as N4SID-RL3).

N4SID-Rf3 (L3 estimated by RRR with $\boldsymbol{U}_f$ augmented)

Based on (3.1.7), $\boldsymbol{Y}_f$ includes $\boldsymbol{\Gamma}_f\boldsymbol{X}_k$ and $\boldsymbol{H}_f\boldsymbol{U}_f$, and therefore at least $n+mf$ LVs from RRR($[\boldsymbol{P}_{IO}; \boldsymbol{U}_f]$, $\boldsymbol{Y}_f$) are necessary in order to account for all the effects of $\boldsymbol{X}_k$ and $\boldsymbol{U}_f$. Yet only lf LVs are available to the maximum from this RRR due to the limited number of variables in $\boldsymbol{Y}_f$ (there are usually equal number of inputs and outputs $m=l$ and lf is less than $n+mf$). In order to account for all the effects of $\boldsymbol{X}_k$ and $\boldsymbol{U}_f$, one can augment the $\boldsymbol{Y}$ data matrix with $\boldsymbol{U}_f$ to increase the limit of the possible number of LVs, i.e., performing RRR($[\boldsymbol{P}_{IO}; \boldsymbol{U}_f]$, $[\boldsymbol{Y}_f; \boldsymbol{U}_f]$). In theory, the first $mf+n$ LVs will include both $\boldsymbol{X}_k$ and $\boldsymbol{U}_f$ so that the predictable subspace in $\boldsymbol{Y}_f$ and the effect of $\boldsymbol{U}_f$ in $\boldsymbol{Y}_f$ as well as the augmented $\boldsymbol{U}_f$ can be completely explained. All the remaining LVs are only to model the noise. In practice, AIC or other criteria can be employed to determine the number of LVs to be used for the RRR. The resultant coefficient matrix for $\boldsymbol{P}_{IO}$ from this RRR can be used to estimate the predictable subspace and the process states (this method is denoted as N4SID-Ruf), and the resultant coefficient matrix for $\boldsymbol{U}_f$ can be used for computation of $\boldsymbol{Y}_{f\_e}=\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ to be used in RRA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_{f\_e}$) for estimates of process states (this method is denoted as N4SID-Rf3)

N4SID-RR (RRA based on new regression for $\boldsymbol{L}_3$)

Another way to deal with the ill-conditioning problem in the first step of N4SID caused by the high correlation in $\boldsymbol{P}_{IO}$ is to replace $\boldsymbol{P}_{IO}$ with estimated $\boldsymbol{X}_k$. Equation (3.1.7) gives the theoretical basis for this method. One can use the estimated state sequence $\boldsymbol{X}_{k\_e}$ by the original N4SID algorithm or other methods discussed above. The regression coefficient of $\boldsymbol{Y}_f$ against $[\boldsymbol{X}_{k\_e}; \boldsymbol{U}_f]$ is an estimate of coefficient matrix $[\boldsymbol{\Gamma}_f, \boldsymbol{H}_f]$. This result might exhibit some bias because of the estimation error in $\boldsymbol{X}_{k\_e}$, but the resultant estimate of $\boldsymbol{H}_f$ might be better than $\boldsymbol{L}_3$ in the original N4SID algorithm for much better regression condition.

All these methods discussed above try to improve the performance of N4SID based on a better estimate of H_f matrix and the relationship between N4SID and RRA. The key steps for estimation of the predictable subspace and the process states are summarized in Table 4.2. Most of these methods avoid or lessen the severity of possible ill-conditioning problems in the oblique projection of N4SID; therefore they are promising for improving the performance of N4SID.

Table 4.2 Methods for improving N4SID by RRA

Methods	Estimation of the predictable subspace	Estimation of the process states
N4SID	$L_1Y_p + L_2U_p$	PCA($L_1Y_p+L_2U_p$)
N4SID-L3	$Y_f - L_3U_f$	RRA([Y_p; U_p], $Y_f - L_3U_f$)
N4SID-tri	$Y_f - L_{3\_tri}U_f$; L_{3_tri}=lower triangle of L_3	RRA([Y_p; U_p], $Y_f - L_{3\_tri}U_f$)
N4SID-Hf	$Y_f - H_{f\_e}U_f$; H_{f_e} based on ARX	RRA([Y_p; U_p], $Y_f - H_{f\_e}U_f$)
N4SID-RR	$Y_f - L_{3\_rr}U_f$; [Γ_{f_e} L_{3_rr}]=regress Y_f against [X_{k_e}; U_f], X_{k_e} is from N4SID	RRA([Y_p; U_p], $Y_f - L_{3\_rr}U_f$)
N4SID-RLp	$L_{1\_rlp}Y_p+L_{2\_rlp}U_p$; [$L_{1\_rlp}$ L_{2_rlp}] from RRR([Y_p; U_p], Y_f)	PCA($L_{1\_rlp}Y_p+L_{2\_rlp}U_p$)
N4SID-RUf	$L_{1\_ruf}Y_p+L_{2\_ruf}U_p$; [$L_{1\_ruf}$ L_{2_ruf}] from RRR([Y_p; U_p; U_f], [Y_f; U_f])	PCA($L_{1\_ruf}Y_p+L_{2\_ruf}U_p$)
N4SID-RO	$L_{1\_ro}Y_p + L_{2\_ro}U_p$, [$L_{1\_ro}$ L_{2_ro}] from RRR(P_{IO_ro}, Y_{f_ro})	PCA($L_{1\_ro}Y_p+L_{2\_ro}U_p$)
N4SID-ROP	$Y_f - L_{3\_rop}U_f$, L_{3_rop} by regressing Y_{f_rop} against U_{f_rop}, both matrices from regressing P_{IO} out	RRA([Y_p; U_p], $Y_f - L_{3\_rop}U_f$)
N4SID-RL3	$Y_f - L_{3\_rlp}U_f$, L_{3_rlp} from RRR([Y_p; U_p], Y_f)	RRA([Y_p; U_p], $Y_f - L_{3\_rlp}U_f$)
N4SID-Uf3	$Y_f - L_{3\_ruf}U_f$; L_{3_ruf} from RRR([Y_p; U_p; U_f], [Y_f; U_f])	RRA([Y_p; U_p], $Y_f - L_{3\_ruf}U_f$)

4.3 SIMs Based on Dynamic PLS and PCA

From the discussion in Chapter 3 and earlier sections of this chapter, it is clear that CCA and RRA are the bases for CVA and N4SID respectively. These close relationships between LVMs and SIMs naturally raise the question of whether or not other LVMs, such as PLS and PCA, are applicable for SIMs. This section will explore the possibility of applying dynamic PLS and PCA to extraction of the process states, and will release more connections between LVMs and SIMs.

4.3.1 SIM Based on Dynamic PLS

When CCA and RRA are applied for the past data $\boldsymbol{P}_{IO}$ and the estimated predictable subspace $\boldsymbol{Y}_{f\_e}=\boldsymbol{Y}_f-\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$, the LVs summarize the information from the past data that best predict the process future behavior, and hence are estimates of the process state variables. Analogous to CCA and RRA, PLS can be applied to these data sets to obtain another set of latent variables. However, the relationships between these LVs from PLS and the process states need to be investigated.

As shown in Table 4.1, PLS($\boldsymbol{X}$, $\boldsymbol{Y}$) has the objective of maximizing the covariance of the two data sets. The LVs from PLS($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_{f\_e}$) focus on explaining both the variance of past ($\boldsymbol{P}_{IO}$) and the estimated predictable subspace ($\boldsymbol{Y}_{f\_e}$) as well as their correlation. As a result, the LVs from this PLS are a compromise between explaining the past data and explaining the predictable subspace. This is different from the definition of the process states, which only focus on providing the best explanation of the predictable subspace without any intention of explaining the past data (just as CCA and RRA). The estimated states are linear combinations of the past data and can explain part of the variation in the past data; however, the LVs from PLS are twisted away from the process states by partially compromising toward more explanation of the components of the past data, which is not related to the process states. This turns out that the LVs from PLS include components other than the process states, and therefore PLS are not as efficient as CCA or RRA in extracting the minimum number of process states. As a result of this inefficiency, more LVs from PLS than the system order n are required to describe the process. This point has been discussed by Negiz and Cinar (1997) for time series modeling. All this means that taking the LVs from PLS as the estimated states is feasible for modeling a dynamic process, but it could not be expected to provide a minimal order state-space model.

Since the LVs from PLS depend on the scaling of the data matrices, more attention should be given to the data scaling. If the variances of some outputs or inputs are far great than the others, the LVs from PLS will focus more on those variables with

large variances. This is harmful to the model for the whole process unless there is a justification that those variables with a larger variance are more important than the others.

4.3.2 SIM Based on Dynamic PCA

Dynamic PCA is to perform PCA($[\boldsymbol{Y}_p; \boldsymbol{U}_p]$) (lagging the process input and output variables for p steps) or PCA($[\boldsymbol{Y}_f; \boldsymbol{U}_f; \boldsymbol{Y}_p; \boldsymbol{U}_p]$) (lagging for $p+f$ steps). Based on the objective of PCA, the resultant PCs show the components (or say the directions) with the maximum variances in the data set. The objective of PCA does not show any intent to extract the process state variables. If one wishes that all the information of state variables is included in PCs, many more PCs than the system order must be employed, and these PCs include many components other than the state variables (large estimation errors to the state variables). Therefore dynamic PCA in this approach is inefficient for extracting the process state variables.

There is another way of applying dynamic PCA for the process dynamic based on the least principle components (LPCs) (Ku et al., 1995; Negiz and Cinar, 1997). The number of eigenvalues close to zero indicates the number of linear relationships in the data set (the lagged input and output variables). Their corresponding eigenvectors show the dynamic relationships between these variables. However, these relationships are the repeats and combinations of the true process dynamic model, and these eigenvectors are quite sensitive to the noise. Therefore the result is only acceptable at an extremely low noise level (e.g., the SNR is great than 10000).

Here a SIM algorithm is proposed to estimate the state variables based on the LPCs of dynamic PCA($[\boldsymbol{Y}_f; \boldsymbol{U}_f; \boldsymbol{Y}_p; \boldsymbol{U}_p]$). Suppose the least l PCs associate with eigenvalues close to zero, and the scores of these LPCs are expressed as:

$$\boldsymbol{T}_l = \boldsymbol{L}_{yf}\boldsymbol{Y}_f + \boldsymbol{L}_{uf}\boldsymbol{U}_f + \boldsymbol{L}_{yp}\boldsymbol{Y}_p + \boldsymbol{L}_{up}\boldsymbol{U}_p$$

Then the significant PCs from PCA($\boldsymbol{L}_{yp}\boldsymbol{Y}_p + \boldsymbol{L}_{up}\boldsymbol{U}_p$) are taken as the estimated state variables, and the final dynamic model can be obtained by fitting the estimated state variables to the state-space model.

The idea behind this method is that the scores of these LPCs are essentially zero and this means that the linear combinations of the past data $\boldsymbol{L}_{yp}\ \boldsymbol{Y}_p+\boldsymbol{L}_{up}\boldsymbol{U}_p$ are essentially cancelled with the linear combinations of the future data $\boldsymbol{L}_{yf}\ \boldsymbol{Y}_f+\boldsymbol{L}_{uf}\boldsymbol{U}_f$. Both these linear combinations are essentially the linear combinations of the current states since only the current process states are the connection between the past data and the future data for well-designed experimental data (the cancellation might also happen between $\boldsymbol{U}_f$ and $\boldsymbol{U}_p$ if the inputs are highly correlated). Performing PCA on those state linear combinations from the past data will determine the order of the system, and will improve the quality of the estimated states (in the sense of SNR). The final dynamic model can be obtained by fitting these estimated states to the state-space model.

If the inputs are highly correlated, these estimated states also include the states in the input signal generator, and therefore bear large errors. It turns out that more PCs become necessary to include process states or a less accurate modeling result. Compared to the method directly obtaining the model from the coefficients for the LPCs, this method gives a much better model, decides the system order much easier, and does not have the problem of repeating dynamic relationships. However, the result from this method is still poor compared to CVA or N4SID.

From all the above discussion, it is clear that LVMs are capable of extracting process states from process data for SIM algorithms; however, PLS and PCA are not as efficient as CCA and RRA in the estimation of the process states, though they give a better model for the past data than CCA and RRA. This also indicates a close relationship between LVMs and SIMs.

4.4 Similarities and Differences between SIMs and LVMs

Both LVMs and SIMs use lagged process variables to catch the dynamics in the process, therefore both categories of modeling methods use similar concept of past/future time horizon and similar past/future input and output data set as defined in Section 2.1.

The data sets used in LVMs and SIMs are distinctive from the traditional system identification methods, such as PEM, ML and IV, in which only the current outputs are predicted without involvement of the future steps of the outputs or inputs. However, both LVMs and SIMs use the multiple steps of future outputs (and inputs) where the latent variables or the state variables can be well observed over the future horizon.

The basic ideas behind all these dynamic modeling methods are similar: to predict the process future outputs based on the process past information (past inputs and past outputs). Since long lagged steps of process data are used, these data sets have a high dimension, and the variables in these data sets are highly correlated, utilization of LVM techniques to reduce the dimensionality becomes a natural choice. Both LVMs and SIMs algorithms extract the meaningful information in the time-dependent, high-dimensional, highly correlated past data set down into a reduced dimensional subspace, and the future outputs are then modeled in the terms of latent variables defining this subspace. Different choices of the objective functions, normalizing constraints and rank constraints determine different loading matrices (W in (4.1.3)), which define the latent variables or the estimated state variables in LVMs or SIMs. In brief, both LVMs and SIMs share the basic idea of modeling the process in a lower-dimensional space.

From the discussion in this chapter and the previous chapter, it is clear that SIM algorithms are based on specific LVMs. CVA is based on the CCA result to get an estimate of the process state sequence. N4SID is essentially based on RRA for the state sequence. MOESP is based on PCA (SVD) for the estimate of the extended observability matrix or the state sequence. Naturally, these connections closely link SIMs to LVMs.

Though LVMs and SIMs have strong connections, they have two major differences: in SIMs, the future input effects are removed from the future outputs, and the LVs (estimated states) are fit to the state-space model. The first point has been discussed in Section 4.1. It is the fundamental reason that LVMs and SIMs obtain different kinds of model relationships: LVMs give correlation models while SIMs give causal models in general.

With LVMs the final model is left in the non-parsimonious form of equation (4.1.4), which can always be written in the terms of a high-order ARX model. However, SIM algorithms go one step further and express the final model in the parsimonious state-space form. They are able to do this by defining the latent variables in such a way that they provide a close approximation of the process states. Therefore, the process can always be expressed in the more parsimonious linear state-space model form of equations (3.1.1) and (3.1.2). Estimating the parameters (A, B, C, D) and then casting the model in this form provides certain advantages. Much of the controller design literature is based on state-space models and can thus be employed directly. The more parsimonious model form from SIMs will give smoother predicted dynamic responses, potentially smaller prediction errors with new data, and more robust control. This is entirely analogous to the modeling philosophy of Box and Jenkins (1994) in the input/output space, where preliminary impulse weights or step responses are used to identify and fit more parsimonious transfer function – ARMA models capable of showing similar dynamic behavior. An advantage with SIMs is that, even for MIMO systems, the only major modeling choice is the number of latent variables to use (i.e., order of the state model).

In general, the availability of output (y) data in LVMs for process monitoring situations is usually quite different from that in SIMs for process identification. In the latter, data on y variables is essentially always available from on-line sensors at the same sampling frequency as the manipulated process inputs (u). In process monitoring, the y variables are usually product quality or productivity variables for which observations are available only infrequently from off-line analyses. Process monitoring is therefore often based simply on a PCA model involving only lagged values of the process variables. If y-data is to be used, then the models can only be built from lagged data matrices with rows corresponding to instances at which y's are measured, and the past data matrix X will not contain lagged values of the y's since these are generally unavailable. Dynamic modeling with infrequent y data is also treated by Ergon and Halstensen (2000).

In brief, SIMs are different from LVMs for two distinct features: eliminating the confounded future input effects from the future outputs, and fitting the LVs (estimated

state variables) to the state-space model in a more parsimonious form. In general, LVMs are based on the normal operation data, give the correlation relationship, and are usually applied to process monitoring. SIMs are based on experimental data, give the causal relationship, and are usually applied to system identification.

4.5 Simulation Studies

In this section, a continuous stirred-tank reactor (CSTR) process is used as a simulation example to illustrate the relationship between LVMs and SIMs. There exists a first order chemical reaction in the reactor, and the reaction heat is removed by cooling water from the jacket. The two inputs are the feed flow rate and the coolant temperature, and the two outputs are the product concentration and temperature (see Figure 4.1, for detail parameters, refer to Lakshminarayanan, 1997). The process is linearized at the steady state, and the system matrices of state-space model in the discrete form are shown as below:

$$A=\begin{bmatrix} 0.68654 & -5.046\times10^{-4} \\ 64.182 & 1.0177 \end{bmatrix} \qquad B=\begin{bmatrix} 1.726\times10^{-4} & -1.0647\times10^{-5} \\ -3.5311\times10^{-2} & 4.0478\times10^{-2} \end{bmatrix}$$

$$C=\begin{bmatrix} 1. & 0 \\ 0 & 1 \end{bmatrix} \qquad D=\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

It is a second order dynamic system with poles at 0.8521± 0.0705i. In the following simulations, inputs to the process are independent PRBSs with magnitude of 5 and a switch time period of T_S=5 sampling intervals. Because of the very different magnitudes of the two outputs, the process variables will be scaled to unit variance in all the studies.

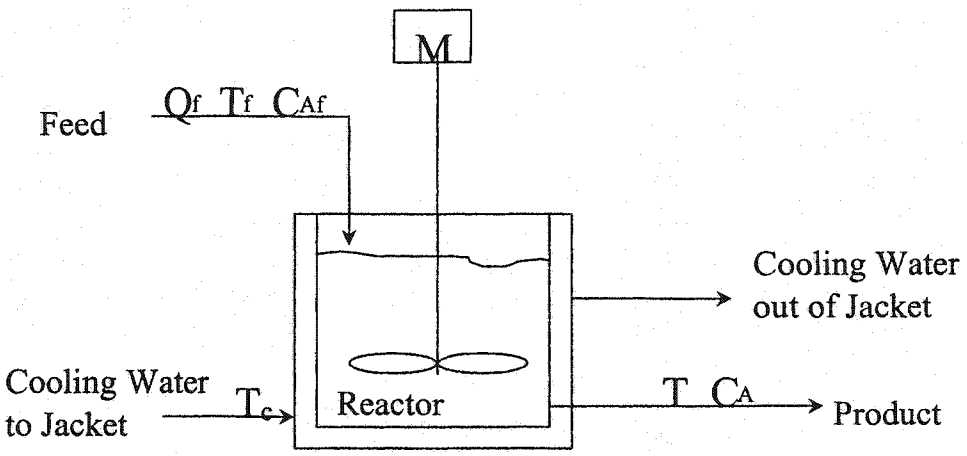


Figure 4.1 Continuous stirred-tank reactor (CSTR)

4.5.1 Comparisons for Case of Noise Free

The purpose of this subsection is to investigate the relationships between the latent variables from LVMs or estimated states from SIMs and the true state variables. To do so in an unambiguous manner, simulation studies on CSTR are performed for the noise free case (w_k=0, v_k=0) with parameters given above. 500 observation data points are collected from this deterministic simulation, and the true state variables are recorded for comparison. For simplicity in the dynamic modeling, both past and future horizons include only 2 lag steps of the input and output variables (this lag structure should be adequate since the true state model is of order 2). LVMs are based on the past data X=[Y_p; U_p] and the future outputs Y=Y_f. PCA uses only the past data X for LVs, which can be used to explain the Y data. SIMs are based on X=[Y_p; U_p] and Y=Y_{f_e}=Y_f - $H_{f\_e}U_f$. In CVA, H_{f_e} is based on the fitted ARX model (In this noise-free case, it equals the true H_f). The CVA_RO algorithm is also applied for this example. In N4SID, H_{f_e} is coefficient matrix L_3 in (3.2.3). The state-based MOESP algorithm is used in this example.

To evaluate how closely the latent variables from these methods coincide with the true state variables of the CSTR process, each latent variable is projected onto the true state variables. The percentage of the variance explained (R^2) by the true state variables is

presented in Table 4.3 (only for the first 4 LVs). The first two LVs from PCA and PLS contain a great amount of state information, but they also contain other components. The later LVs also include some state information. The first two LVs from RRA are very close to the true states but are twisted slightly to explain a little bit of the future input effects in $\boldsymbol{Y}_f$ (if a longer future horizon is used, R^2 will decrease further). The first two LVs from CCA are completely occident with the true states, and the later LVs are total uncorrelated to the state variables. As expected, the first two LVs from SIMs coincide 100% with the true state variables. The later LVs from CVA_RO and state-based MOESP (using regressing $\boldsymbol{U}_f$ out) also contain some state information since they are not orthogonal to the first two estimated states.

Table 4.3 LVs explained by the true states (R^2, PRBS inputs, noise free)

No. LV	PCA	PLS	RRA	CCA	CVA	CVA_ro	N4SID	MOESP_s
1	99.05	98.80	99.28	100	100	100	100	100
2	69.42	80.60	99.26	100	100	100	100	100
3	4.53	1.92	0.65	0	0	12.4	0	80.83
4	17.74	13.85	0.50	0	0	3.55	0	44.11

In MOESP, if $\boldsymbol{H}_{f\_e}$ is taken as the regression coefficient matrix of $\boldsymbol{Y}_f$ against $\boldsymbol{U}_f$, the true states can explain the variance of LVs from SVD($\mathbf{Y}_f - \boldsymbol{H}_{f\_e}\mathbf{U}_f$) 78.81%, 65.54%, 14.16% and 1.31% respectively. This poor result is due to the partial removal of the predictable subspace in eliminating the effects of future inputs $\boldsymbol{U}_f$ from $\boldsymbol{Y}_f$ (QR decomposition). If the input signals had not been serially correlated (i.e., PRBS with a switch time period T_s=1 (uncorrelated) were used rather than autocorrelated PRBS with T_s=5), the LVs from this method would have had the same 100% correlation with the true states.

If the SIM algorithm based on PLS is applied to the example, the LVs from PLS on past data and $\mathbf{Y}_f$-$\boldsymbol{H}_{f\_e}\mathbf{U}_f$ ($\boldsymbol{H}_{f\_e}$ is based on fitted ARX model as in CVA) are explained by the true state variables, the R^2 values are 0.9886, 0.8244, 0.1131 and 0.0354 respectively. The first two LVs contain a great amount of information of the process

states, but they are not coincident with the true states because the objective of PLS includes the compromise of modeling the past data.

If the SIM algorithm based on dynamic PCA (section 4.3.2) is applied to this example, there are 6 zero-eigenvalues in PCA($[\boldsymbol{Y}_f; \boldsymbol{U}_f; \boldsymbol{Y}_p; \boldsymbol{U}_p]$), this means 6 LPCs or 6 linear relationships in the data sets. For these 6 LPCs, PCA on the contribution from the past data will give 2 significant PCs (other PCs corresponds to zero-eigenvalues). This indicates the system has 2 states (order of 2). These 2 PCs are the state estimates, and they coincide 100% with the true process states.

4.5.2 Comparisons for Case with Noise

Here different identification methods are applied for the same CSTR simulation example in the case of white noise added on each output ($v_k \neq 0$). The signal-to-noise ratio (SNR, variance of the true output over variance of the noise) is 10 for each output. 1000 data points were simulated and used for dynamic modeling. Again each process variable is scaled to unit variance. Both past and future input/output data include 13 lag steps (this corresponds to a minimum value of AIC in CVA). As shown before, LVMs are based on the past data $\boldsymbol{X}=[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ and the future $\boldsymbol{Y}=\boldsymbol{Y}_f$. SIMs are based on $\boldsymbol{X}=[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ and $\boldsymbol{Y}=\boldsymbol{Y}_{f\_e}=\boldsymbol{Y}_f - \boldsymbol{H}_{f\_e}\boldsymbol{U}_f$, where $\boldsymbol{H}_{f\_e}$ is based on the fitted ARX model in CVA or coefficient matrix $\boldsymbol{L}_3$ in N4SID.

Depending on the individual method, LVs from LVMs display different efficiency in modeling the past data ($\boldsymbol{X}$) or the future data ($\boldsymbol{Y}$). The total variance percentages of the past data modeled (explained) by LVs (R^2_X, accumulative values) are shown in Figure 4.2, and the total variance percentages of the future data modeled by LVs (R^2_Y, accumulative values) are shown in Figure 4.3. These figures clearly show that, for modeling the past data, PCA is most efficient and RRA is least efficient; for modeling the future outputs, PCA is least efficient and RRA is most efficient; PLS is always between PCA and RRA; CCA is not most efficient in explanation of these variances (but is most efficient in the relative sense in orthogonal directions). All this is due to their

objectives in finding the LVs. This makes PCA and PLS useful in monitoring, and RRA and CCA valuable in system identification.

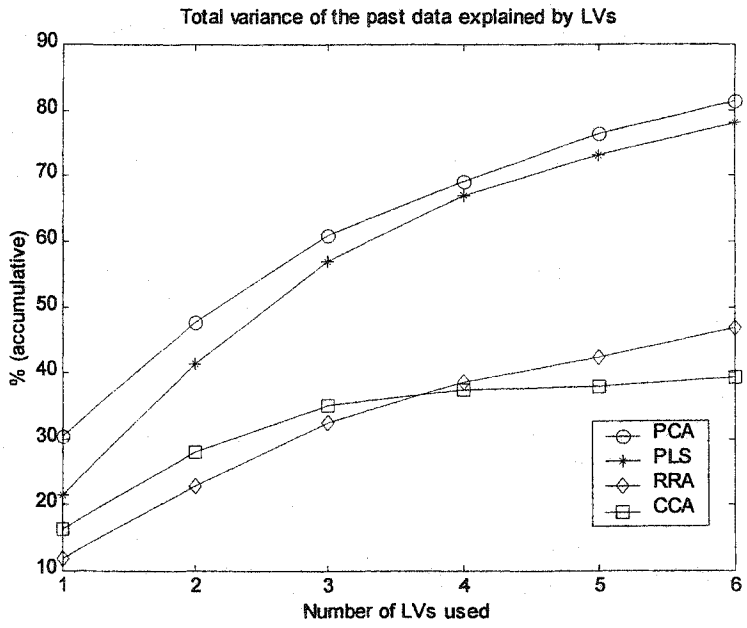


Figure 4.2 Performance of LVMs for modeling the past data

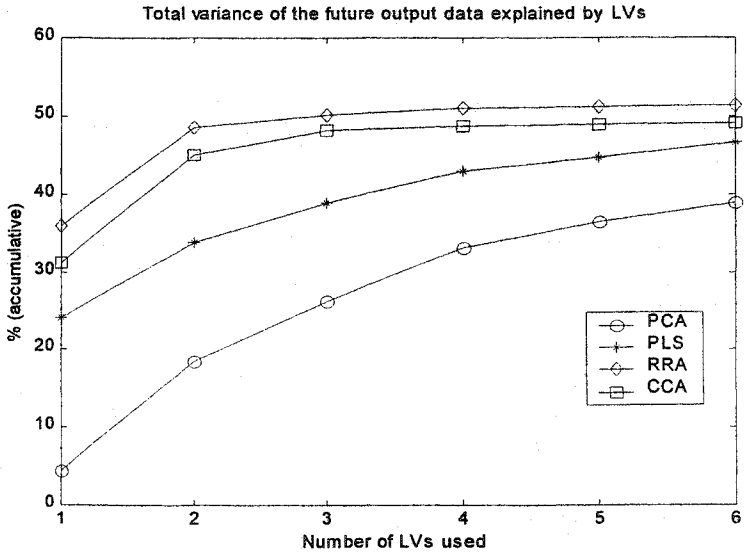
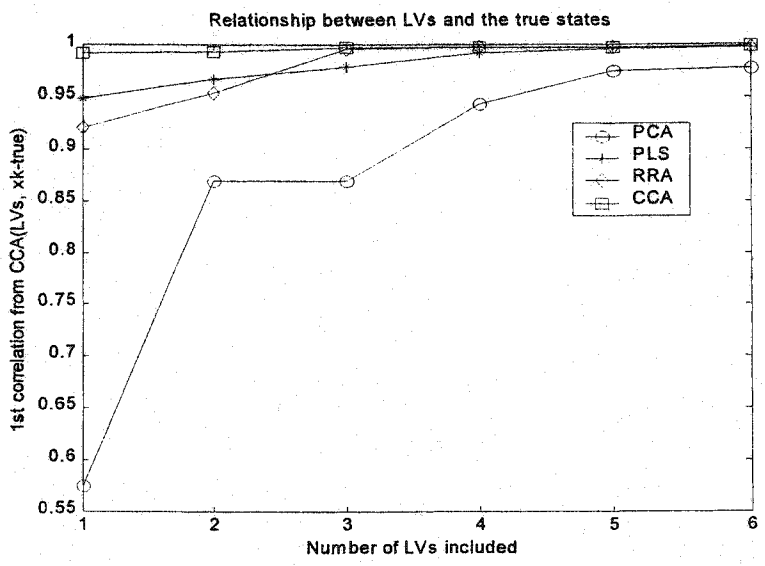


Figure 4.3 Performance of LVMs for modeling the future output data

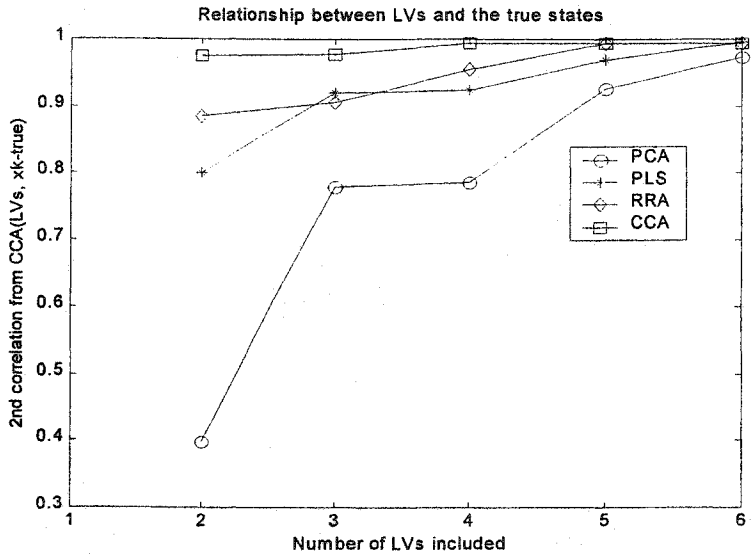
To show the relationships between LVs from LVMs and SIMs and the true state variables, each LV is explained by the true state variables, and the R^2 value shows how close this LV is to the process states (see Table 4.4). In general, the state information in LV decreases with the sequence. The results also clearly indicate that RRA and CCA are more efficient in extracting out the state information than PCA and PLS. However, the first two LVs from SIMs are much closer to the true states than those from LVMs. This conclusion is also shown in the canonical correlation coefficients (CCCs) between cumulative LVs (including 1 to 6 respectively), and the true states are shown in Figure 4.4 (only those from LVMs are shown, and those from SIMs are not shown for too close to 1.0).

Table 4.4 LVs explained by the true states (R^2, PRBS inputs, SNR=10)

No. LV	PCR	PLS	RRA	CCA	CVA	N4SID	MOESP
1	33.42	90.58	84.64	98.26	99.65	99.63	99.65
2	57.60	67.00	84.94	95.62	99.80	99.78	99.55
3	45.53	23.30	11.84	1.39	0.000	0.011	0.015
4	14.53	3.29	8.92	2.85	0.000	0.007	0.009
5	29.22	9.42	7.51	0.10	0.000	0.001	0.012
6	9.64	5.39	0.24	0.05	0.000	0.003	0.049



(a) 1st Canonical Correlation Coefficient (CCC)



(b) 2nd Canonical Correlation Coefficient (CCC)

Figure 4.4 CCCs between the LVs and the true states

As discussed previously, both CVA and CCR use the dominant canonical variates from a CCA to build the dynamic model, but CVA removes the future input effects and fits these LVs to the state-space model resulting in a lower order parsimonious model. These extra procedures eliminate the future input confounding and filter out the erratic spurs or ripples in the responses. The impulse responses of the CCR and CVA models using 2 canonical variates are shown in Figure 4.5. Both of them have the same general trend as the true responses, but the responses from the CVA state-space model are clearly closer to the true model (almost overlap) and are much smoother than those from the CCR model. Fitting the latent variables to the state-space model constrains the impulse and step responses to follow only that behaviour possible for a second-order process and therefore eliminates the impulse-response flexibility (over-parameterization) of the lagged CCR regression model.

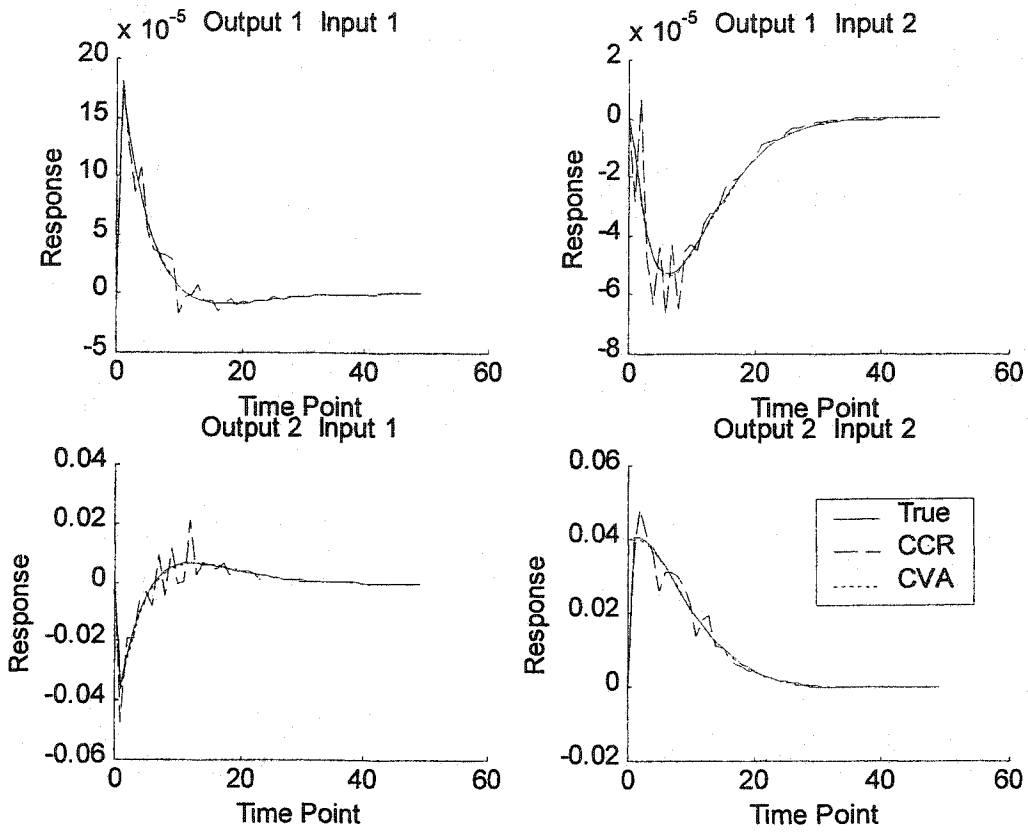


Figure 4.5 Impulse Responses of CCR and CVA (SNR=10)

All the results from different SIMs, CVA, N4SID and MOESP, are so close to the true process model that they overlap on the impulse response plots. In Figure 4.6, their errors on the impulse responses are compared to give an idea about their accuracy. In general, CVA gives a relatively better result, and MOESP gives a relatively larger error in this simulation example.

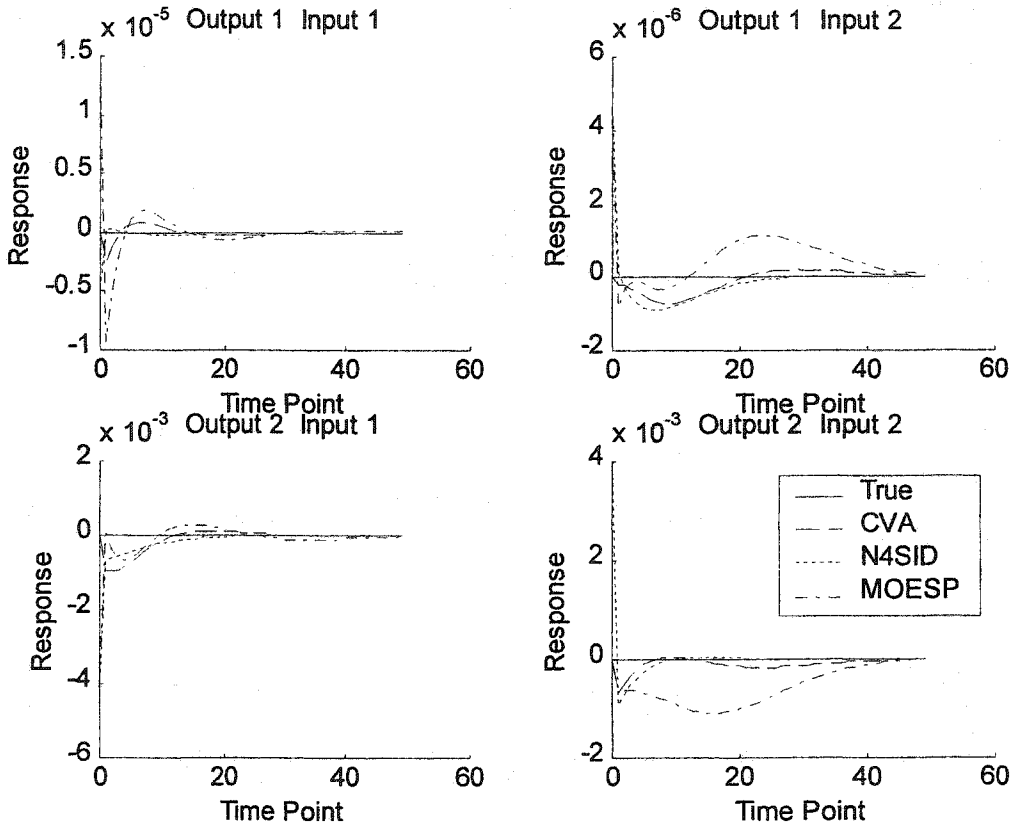


Figure 4.6 Errors of the impulse responses from SIMs (SNR=10)

The SIM algorithm based on dynamic PCA is also applied to this simulation example. PCA($[Y_f; U_f, Y_p; U_p]$) shows many small eigenvalues, and here the contribution from the past data for the last 6 PCs are taken out as the linear combinations of the process states. SVD (PCA) on these components gives singular values: 33.06, 16.10, 7.61, 5.73, 4.82 and 4.68. The R^2 for these components explained by the true states are 0.9898, 0.7091, 0.0496, 0.0189, 0.0041 and 0.0007 respectively. The two components are significant (indicating the system to be order of 2), and they contain a great amount of state information. The impulse response from this method is shown in Figure 4.7. In general, the results are close to the true model; however, the method is apparently worse than CVA, N4SID or MOESP algorithm shown before.

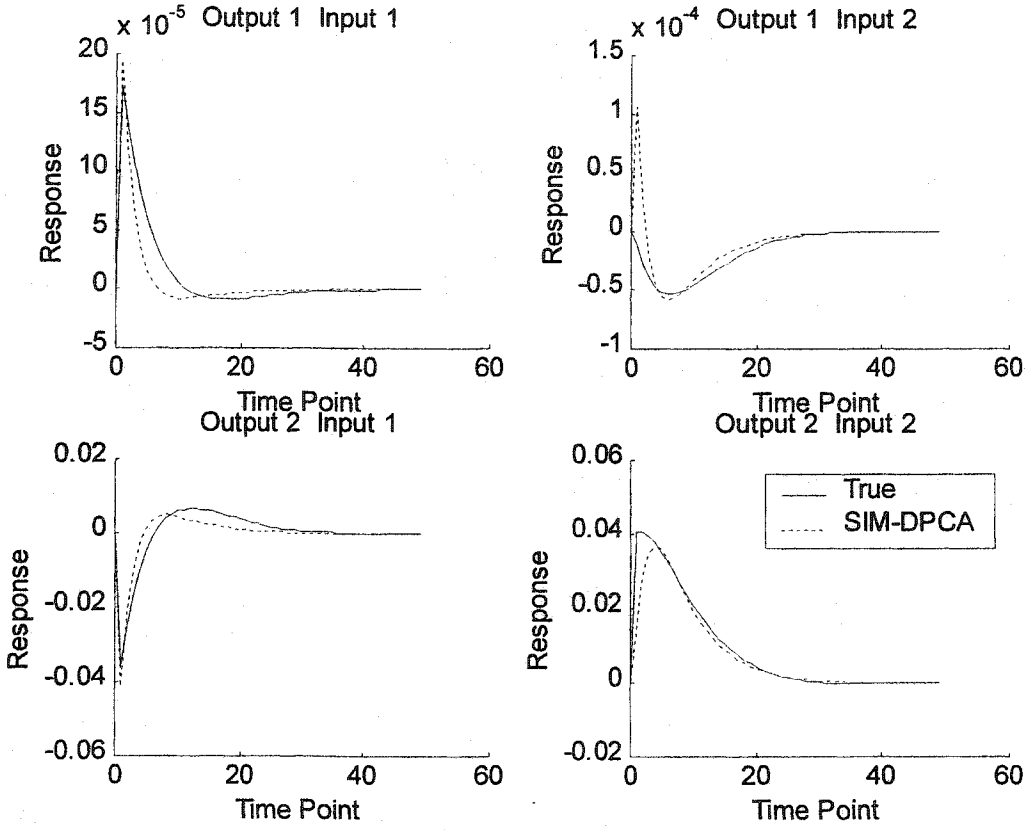


Figure 4.7 Modeling result from SIM based on dynamic PCA

The efficiency of PLS for extracting out the state information is also investigated for this example. 2, 4 and 6 LVs from PLS($[\boldsymbol{Y}_p; \boldsymbol{U}_p]$, $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$) (where $\boldsymbol{H}_{f\_e}$ is constructed based on fitted ARX model) are taken respectively as the estimated states for state-space models, and the results are shown in Figure 4.8. Compared to the result of CVA (with 2 LVs from CCA), PLS is less efficient in extracting the state information. Using more LVs from PLS, however, the results tend to be very close to the true model. In this example, 6 LVs are good enough for the dynamic model. (A similar simulation in Lakshminarayanan, 1997, indicated that over 30 LVs were necessary. The difference might come from no scaling and no elimination of the future input effects.)

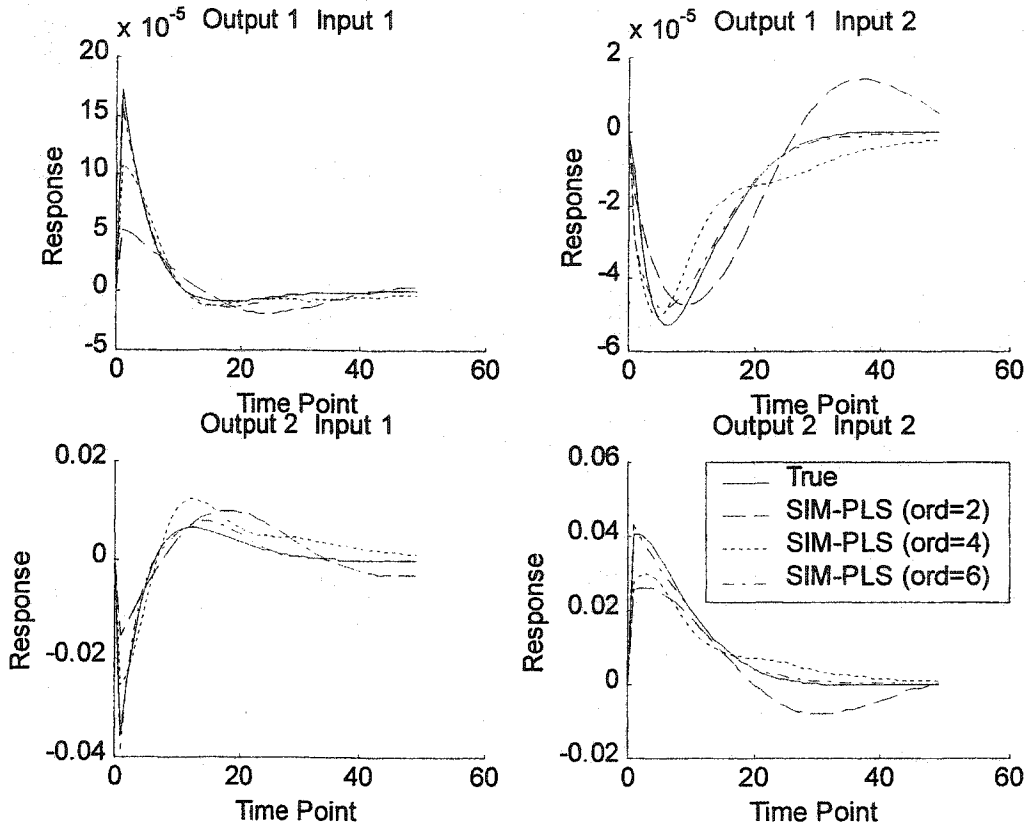


Figure 4.8 Results from SIM algorithm based on different number of LVs from PLS

4.5.3 Improving N4SID Performance with RRA

In this subsection, Monte Carlo simulations are done to demonstrate the relationship between N4SID and RRA and the improvement of N4SID performance by using RRA in various methods. The same CSTR example shown before is employed with SNR=2.0. 500 data points are collected in each simulation, and a total of 200 simulations are done in the Monte Carlo simulation.

As an example, Table 4.5 summarizes the result of the above methods for one typical simulation. In this example, both past and future horizons are of 9 lag steps (determined by CVA according to minimum AIC), and 2 LVs (estimated states) are used for fitting the state-space model. Simulation for this example indicate that, in general, the results from n4sid.m function (based on N4SID unbiased algorithm) in System

Identification Toolbox (version 4.0.5) are numerically better than those from biased, unbiased or robust algorithms in Van Overschee and De Moor (1996), therefore it is used for the comparison study in this section. The results from various methods shown in Section 4.2.3 are compared with that from N4SID in several aspects: ABE_Hf for the total absolute error of the estimated $\boldsymbol{H}_f$, SSE_Yf for the sum of squared error on the predictable subspace (relative to the total sum squares of the true predictable subspace), two CCCs between the estimated states and the true states, IAE and SSE for the Integral Absolute Error (IAE) and Sum Squared Error (SSE) on the impulse responses (first 50 steps) respectively.

The simulation results shown in Table 4.5 clearly indicate that the subspace modeling result (N4SID-L3) based on RRA($\boldsymbol{P}_{IO}$, $\boldsymbol{Y}_f$-$\boldsymbol{L}_3\boldsymbol{U}_f$) is the same as that from N4SID. All the methods listed in Table 4.2 (except N4SID-Uf3) give better estimations of $\boldsymbol{H}_f$. Estimated predictable subspaces based on coefficient matrices $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ from different methods are closer to the true predictable subspace. Those methods based on $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ include all the future noise in the estimated predictable subspace, and therefore indicate a large error on the estimated predictable subspace; however, after projecting to the past data, the errors become rather small. The values of CCCs between the estimated states and the true states are very close to 1.0, and this indicates that the estimated states are very close to the true states. The final models from all these modified methods have smaller errors for both IAE and SSE than the original N4SID.

Table 4.5 Comparison of results from different methods for one typical simulation

Methods	# LVs	ABE_Hf	SSE_Yf	1st CCC	2nd CCC	IAE	SSE
N4SID	27	0.0704	0.1254	0.9906	0.9847	0.1807	0.0272
N4SID-L3	18	0.0704	1.0607	0.9906	0.9847	0.1807	0.0272
N4SID-tri	2	0.0611	1.0605	1.0000	1.0000	0.0449	0.0020
N4SID-Hf	2	0.0214	1.0707	0.9906	0.9848	0.1651	0.0224
N4SID-RR	2	0.0592	1.0323	0.9906	0.9850	0.1752	0.0257
N4SID-RLp	16	/	0.1206	0.9907	0.9849	0.1767	0.0261
N4SID-RUf	20	/	0.0348	0.9905	0.9852	0.1659	0.0234
N4SID-RO	2	/	0.0266	0.9905	0.9853	0.1624	0.0226
N4SID-ROP	K=2	0.0669	1.0519	0.9908	0.9851	0.1709	0.0252
N4SID-RL3	16	0.0662	1.0608	0.9907	0.9849	0.1768	0.0261
N4SID-Uf3	20	0.0729	1.0372	0.9905	0.9851	0.1665	0.0235

The distributions (histograms) of IAE from these algorithms are compared with that from N4SID for the 200 simulations. The IAE results from N4SID and N4SID-L3 are shown in Figure 4.9. It shows clearly that these two methods give the same result. The distribution of IAE from N4SID-tri is compared with that from N4SID in Figure 4.10. It is obvious that N4SID-tri gives a better result than N4SID. This conclusion becomes much more apparent from the distribution of the IAE difference for each simulation in Figure 4.11. The distributions of the IAE differences for other methods are shown in Figure 4.12. These figures clearly show that all these methods of using RRA improve the performance of N4SID.

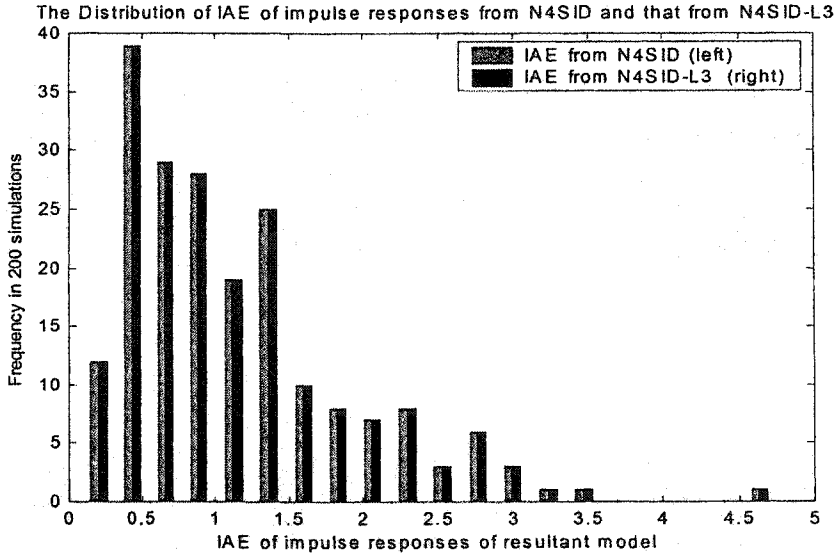


Figure 4.9 Distribution of IAE results from N4SID and N4SID-L3

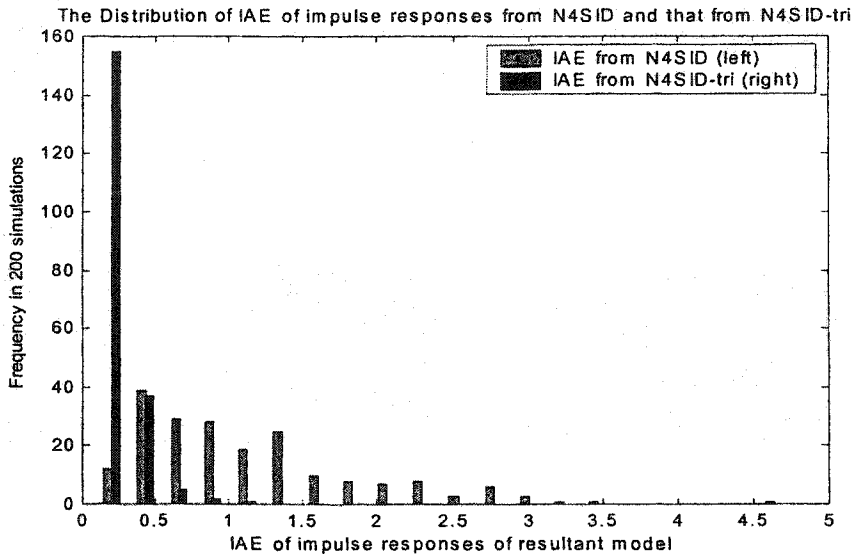


Figure 4.10 Distribution of IAE results from N4SID and N4SID-tri

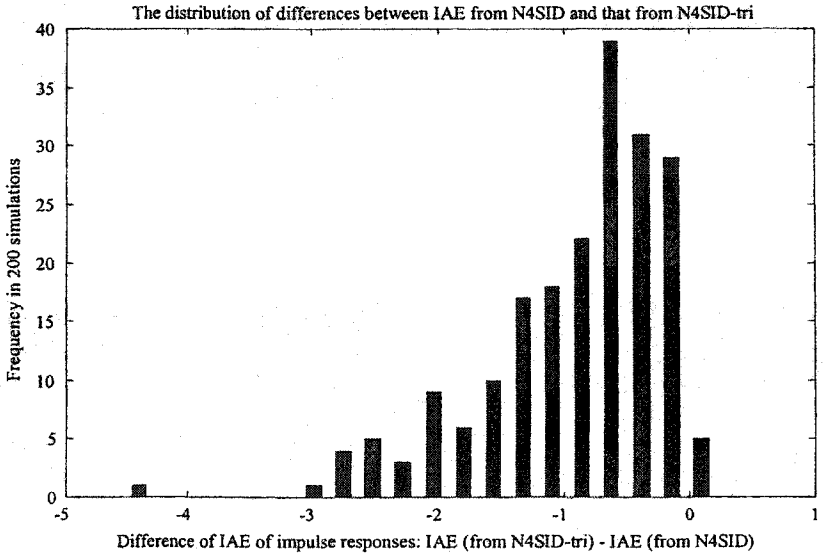


Figure 4.11 Distribution of IAE differences between N4SID and N4SID-tri

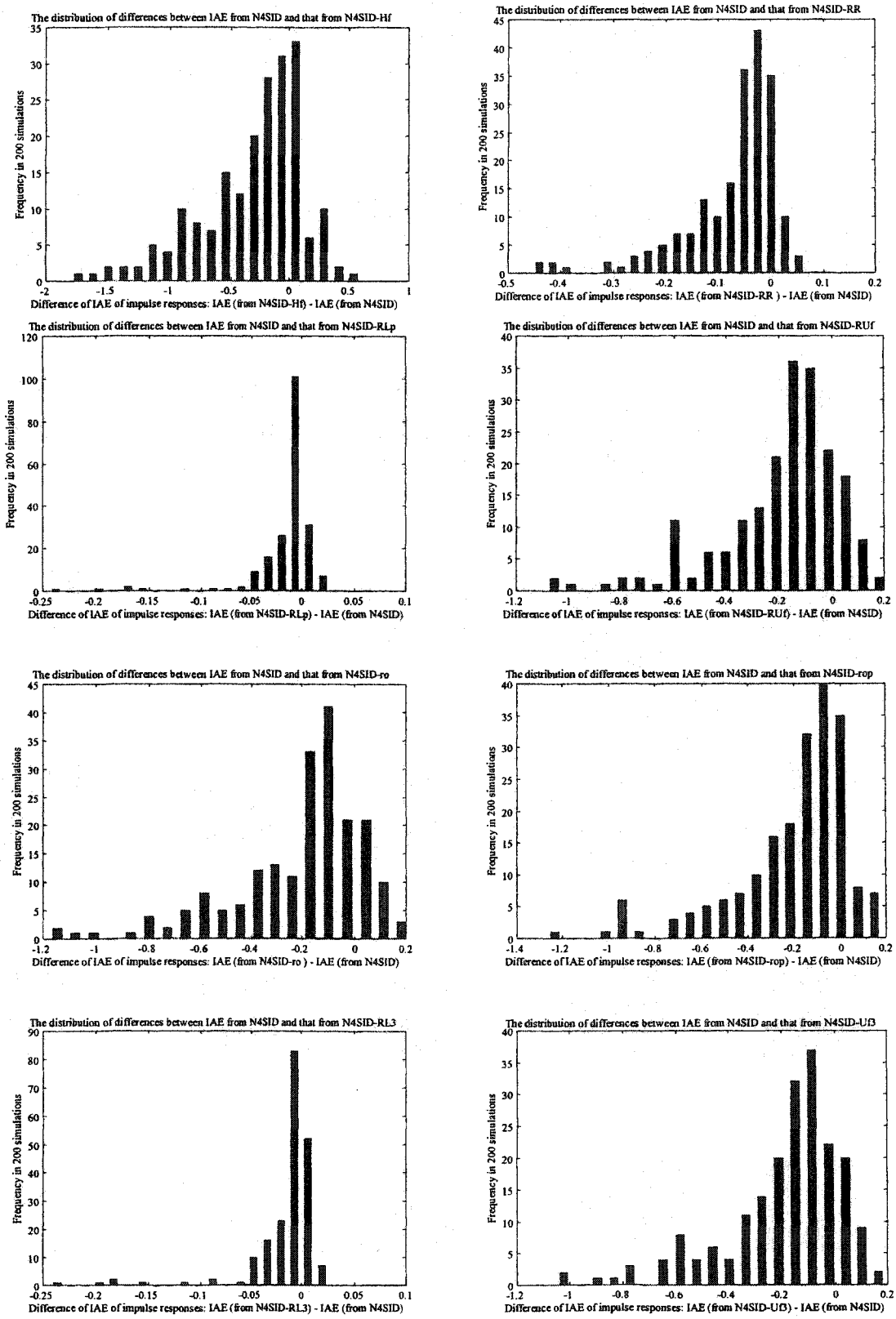


Figure 4.12 Distribution of the IAE differences (compared to N4SID) from various methods
Left column (from top to bottom): N4SID-Hf, N4SID-RLp, N4SID-ro, N4SID-RL3
Right column (from top to bottom): N4SID-RR, N4SID-RUf, N4SID-rop, N4SID-Rf3

The equivalence and the improvement of the listed methods using RRA for N4SID are quantitatively tested in Table 4.6 based on the paired differences of IAE (the impulse weights). The first column lists the number of cases (percentage of the 200 simulations) in which the final IAE from one method is improved relative to that from N4SID. The remaining columns provide statistics on the paired IAE difference (IAE from the modified N4SID method – IAE from N4SID). For N4SID-L3, 53% of the cases are improved (50% is the expected value if there are no numerical errors), and the Student's t-statistic shows that this algorithm is not statistically different from N4SID (critical value of $t_{0.05}$=1.96 for 199 degree of freedom at 0.05 level of significance). All other methods show significant improvement over N4SID according to the Student's t-statistic.

Table 4.6 Improvement of different methods (for 200 simulations)

Methods	Cases improved (%)	Mean of IAE difference	STD of IAE difference	Student's t-statistic
N4SID	0	0	0	0
N4SID-L3	53	0	0	0.31
N4SID-tri	99	-0.9157	0.0524	-17.46
N4SID-ARX	74	-0.3000	0.0304	-9.86
N4SID-RR	82	-0.0722	0.0067	-10.85
N4SID-RLp	76.5	-0.0148	0.0024	-6.10
N4SID-Ruf	83.5	-0.1928	0.0162	-11.91
N4SID-ro	82	-0.2094	0.0180	-11.66
N4SID-rop	86	-0.2028	0.0180	-11.25
N4SID-RL3	76.5	-0.0148	0.0024	-6.11
N4SID-Uf3	83.5	-0.1893	0.0157	-12.05

4.6 Conclusions

This chapter mainly focuses on the relationships between SIMs and LVMs for dynamic process modeling. It is the first time to reveal the connection and the difference between these two categories of methods, and show a clear picture of their relationships.

When the process input and output variables are properly lagged to form the past data and the future output data, LVMs can be applied to these data sets for the dynamic relation between the inputs and the outputs. The final latent variable models are in ARX

model form. These models show the correlation relationship instead of the causal relationship in general, even for experimental data due to the inclusion of the future input effects in the future output data. They may predict the future outputs well if the correlation structure in the inputs remains unchanged.

The N4SID method is proven to be based on RRA just as CVA is based on CCA. The whole procedure of the oblique projection and the SVD in N4SID is equivalent to performing RRA between past data and the estimated predictable subspace. This relationship releases another important connection between SIMs and LVMs. The insights from the relationship provide a variety of methods to improve the performance of N4SID.

Both SIMs and LVMs build the dynamic models using a lower dimensional subspace of the original data sets. Both LVs and estimated states are linear combinations of the past data; however, in SIMs the effect of the future inputs is removed from the future outputs. LVMs are employed for the modified data sets for the process state estimates, and these estimated states (LVs) are further fit to the state-space model for the system matrices. Here removal of the future input effect makes the final result a casual relationship, and fitting to the state-space model makes the final model more parsimonious and with smoother impulse responses.

In estimating the states by LVMs, though PCA and PLS are feasible, they are not as efficient as CCA and RRA. PCA and PLS include more information about the past data set and the correlation structure of the input variables in their LVs, therefore these methods are more suitable for process monitoring. CCA and RRA focus on the prediction of the future outputs based on the past data, and there is no intention to model the past data, therefore LVs from these methods are more efficient in estimating the process states.

Appendix 4.1 Relationship between RRA and LS with PCA

Suppose general data set $X \in \Re^{N \times k}$ and $Y \in \Re^{N \times l}$. Denote SVD on X as:

$$X = U_X S_X V_X^T \text{ with } U_X^T U_X = I_N \tag{A4.1.1}$$

Then the projection of Y onto X will be:

$$Z = Y / X = X\left(X^T X\right)^{-1} X^T Y = U_X U_X^T Y \tag{A4.1.2}$$

Denote the SVD of matrix $U_X{}^T Y$ as $U_X Y = U_1 S V^T$ ($U_1{}^T U_1 = I_k$), then SVD of the above projection result will be: (i.e., PCA result)

$$Z = Y / X = U_X U_X^T Y = U_X U_1 S V^T = U S V^T = T P^T \tag{A4.1.3}$$

Here $\boldsymbol{T}=\boldsymbol{US}$ and $\boldsymbol{U}=\boldsymbol{U}_X\boldsymbol{U}_1$ ($\boldsymbol{U}^T\boldsymbol{U}=\boldsymbol{U}_1{}^T\boldsymbol{U}_X{}^T\boldsymbol{U}_X\boldsymbol{U}_1=\boldsymbol{I}_k$). Columns in $\boldsymbol{P}=\boldsymbol{V}$ are the loading vectors for PCA($\boldsymbol{Z}$), and columns in $\boldsymbol{T}$ are the scores for the PCs. Scores in $\boldsymbol{T}$ are columns in $\boldsymbol{U}$ weighted by diagonal elements (singular values) of $\boldsymbol{S}$ respectively.

In PCA or SVD of $\boldsymbol{Z}$, $\boldsymbol{U}$ is the eigenvector matrix of ZZ^T, i.e., $\boldsymbol{ZZ}^T\boldsymbol{U}=\boldsymbol{US}^2$, that is,

$$X\left(X^T X\right)^{-1} X^T Y \cdot Y^T X\left(X^T X\right)^{-1} X^T U = U S^2 \tag{A4.1.4}$$

Pre-multiply the above equation by $(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$, and denote $\boldsymbol{W}=(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{U}$, then the above equation becomes:

$$\left(X^T X\right)^{-1} X^T Y \cdot Y^T X W = W S^2 \tag{A4.1.5}$$

This indicates that $\boldsymbol{W}$ is the eigenvector matrix of $(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{X}=\boldsymbol{S}_{XX}{}^{-1}\boldsymbol{S}_{XY}\boldsymbol{S}_{YX}$. However, this is the same matrix for the LVs of RRA are the eigenvectors (refer to Table 4.1). The latent variables from RRA are:

$$U_R = XW = X \cdot \left(X^T X\right)^{-1} X^T U = U_X U_X^T U_X U_1 = U_X U_1 = U \tag{A4.1.6}$$

Therefore, PCA($\boldsymbol{Y/X}$) and RRA($\boldsymbol{X}$, $\boldsymbol{Y}$) find the same latent variables with only a scaling difference. Therefore, these two methods are theoretically equivalent to each other.

What the first a PCs from PCA($\boldsymbol{Y/X}$) can explain in Y subspace is:

$$\hat{Y}_{PCA} = T_a P_a^T = U S_a V_a^T = U_a S_{aa} V_a^T \tag{A4.1.7}$$

where the subscript a stands for the matrix with only first a column vectors, and S_{aa} is the diagonal square matrix with first a singular values. The first a latent variables from RRA(X, Y) can explain the following Y subspace:

$$\hat{Y}_{RRA} = U_a U_a{}^T Y = U_a U_{1a}^T U_X^T Y = U_a U_{1a}^T U_1 S V^T = U_a \begin{bmatrix} I_a & 0 \end{bmatrix} S V^T = U_a S_{aa} V_a^T \tag{A4.1.8}$$

That is, RRA(X, Y) and PCA(Y/X) theoretically explain exactly the same components in Y data set.

Appendix 4.2 Regression Out Method for N4SID

The regression model in N4SID is as following (3.2.1):

$$Y_f = L_1 Y_p + L_2 U_p + L_3 U_f + E$$

If $\boldsymbol{P}_{IO}=[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ denotes the past data and $\boldsymbol{L}_{12}$ denotes the corresponding coefficient matrix $[\boldsymbol{L}_1\ \boldsymbol{L}_2]$, the above model can be written as:

$$Y_f = L_{12} P_{IO} + L_3 U_f + E \tag{A4.2.1}$$

If regressing $\boldsymbol{U}_f$ out of both sides of the above equation, the result will be:

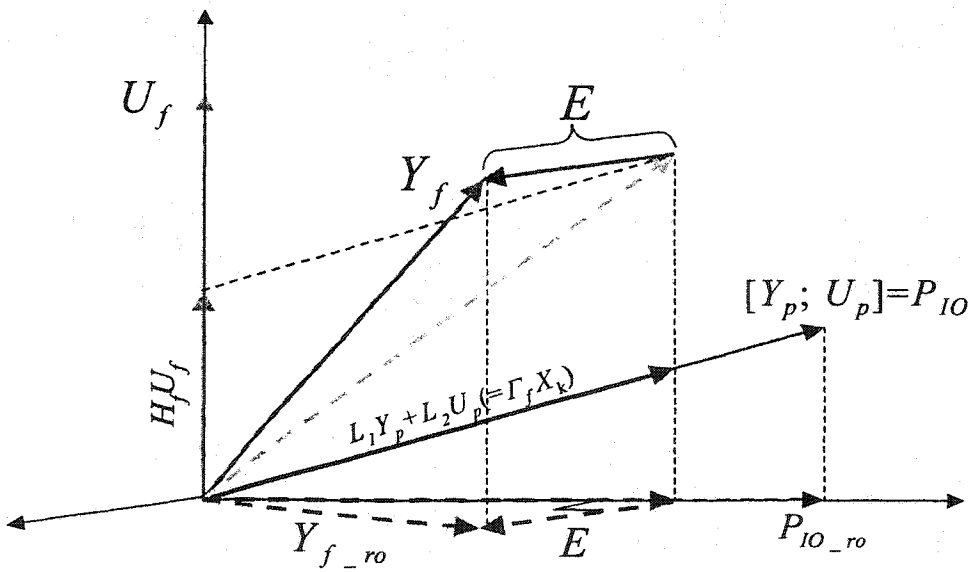
$$Y_{f\_ro} = L_{12} P_{IO\_ro} + E \tag{A4.2.2}$$

Where $\boldsymbol{Y}_{f\_ro}$ is for the result from regressing $\boldsymbol{U}_f$ out of $\boldsymbol{Y}_f$, and $\boldsymbol{P}_{IO\_ro}$ is for the result from regressing $\boldsymbol{U}_f$ out of $\boldsymbol{P}_{IO}$. Both these two matrices are known and the only unknown coefficient matrix $\boldsymbol{L}_{12}$ can be obtained by regressing $\boldsymbol{Y}_{f\_ro}$ against $\boldsymbol{P}_{IO\_ro}$ as:

$$L_{12} = Y_{f\_ro} P_{IO\_ro}^T \left(P_{IO\_ro} P_{IO\_ro}^T\right)^{-1} \tag{A4.2.3}$$

The predictable subspace is:

$$L_1 Y_p + L_2 U_p = L_{12} P_{IO}$$



Geometric explanation of the N4SID regression out algorithm

The above regression-out operation can also be performed for the past data $\boldsymbol{P}_{IO}$. If $\boldsymbol{P}_{IO}$.is regressed out of both sides of (A4.2.1), the result will be:

$$Y_{f\_rop} = L_3 U_{f\_rop} + E_f \tag{A4.2.4}$$

Where $\boldsymbol{Y}_{f\_rop}$ is for the result from regressing $\boldsymbol{P}_{IO}$ out of $\boldsymbol{Y}_f$, and $\boldsymbol{U}_{f\_rop}$ is for the result from regressing $\boldsymbol{P}_{IO}$ out of $\boldsymbol{U}_f$. Both these matrices are known, and the only unknown coefficient matrix $\boldsymbol{L}_3$ can be obtained by regressing $\boldsymbol{Y}_{f\_rop}$ against $\boldsymbol{U}_{f\_rop}$ as:

$$L_{12} = Y_{f\_rop} U_{f\_rop}^T \left(U_{f\_rop} U_{f\_rop}^T\right)^{-1} \tag{A4.2.5}$$

The results from (A4.2.3) and (A4.2.5) are equivalent to those from N4SID in the sense of expectation value.

Appendix 4.3 Algorithms for CCA Computation

This appendix lists several algorithms often used in the computation of CCA. For data sets X and Y (dimension: dim(X), dim(Y). d=min(dim(X),dim(Y))), CCA is to find d canonical variates (CVs) $U_x=XJ$ and $V_y=YL$ from X and Y respectively, so that $U_x^TU_x=I_d$, $V_y^TV_y=I_d$, and $U_y^TV_y=D$=diag(r_i, i=1, 2, …, d), i.e., to find the orthonormal variable sets (CVs) by linear combinations of X and Y respectively, the corresponding paired CVs are most correlated and with no correlation with other CVs. The key issue is to find the coefficient matrix J and L. Here the covariance matrices of X and Y are denoted as S_{xx}, S_{yy}, S_{xy}, and S_{yx} respectively. $S_{xy}=(S_{yx})^T$, and S_{xx} and S_{yy} are symmetric.

Eigenvector decomposition

$[J, D]$=eig($S_{xx}^{-1}S_{xy}S_{yy}^{-1}S_{yx}$) and $[L, D]$=eig($S_{yy}^{-1}S_{yx}S_{xx}^{-1}S_{xy}$).

Note, the matrices for eigenvector decomposition (ED) are not symmetric, and numerical errors may cause complex numbers in the result.

ED of S_{xx} and S_{yy} and SVD

$[V_x, D_x]$=eig(S_{xx}), and let $B=V_xD_x^{-1/2}$, so $BS_{xx}B^T=I$

$[V_y, D_y]$=eig(S_{yy}), and let $C=V_yD_y^{-1/2}$, so $CS_{yy}C^T=I$

$[U, D, V]$=svd($B^TS_{xy}C$)

$J=BU$ and $L=CV$

Square root factor and SVD

$[V_x, D_x]$=eig(S_{xx}), and $S_{xx}^{-1/2}=S_{xx\_srf}=V_xD_x^{-1/2}V_x^T$

$[V_y, D_y]$=eig(S_{yy}), and $S_{xx}^{-1/2}=S_{yy\_srf}=V_yD_y^{-1/2}V_y^T$

$[U, D, V]$=svd($S_{yy}^{-1/2}S_{yx}S_{xx}^{-1/2}$)

$J=S_{xx}^{-1/2}V$ and $L=S_{yy}^{-1/2}U$

SVD method

$[\boldsymbol{U}_x, \boldsymbol{S}_x, \boldsymbol{V}_x]=\text{svd}(\boldsymbol{X})$, and $\boldsymbol{X}_{\_srf}=\boldsymbol{V}_x\boldsymbol{S}_x^{-1}$

$[\boldsymbol{U}_y, \boldsymbol{S}_y, \boldsymbol{V}_y]=\text{svd}(\boldsymbol{Y})$, and $\boldsymbol{Y}_{\_srf}=\boldsymbol{V}_y\boldsymbol{S}_y^{-1}$

$[\boldsymbol{U}, \boldsymbol{D}, \boldsymbol{V}]=\text{svd}(\boldsymbol{Y}_{\_srf}^{T}\boldsymbol{S}_{yx}\boldsymbol{X}_{\_srf})$

$\boldsymbol{J}=\boldsymbol{X}_{\_srf}\boldsymbol{V}$ and $\boldsymbol{L}=\boldsymbol{Y}_{\_srf}\boldsymbol{U}$

QR decomposition and SVD

$[\boldsymbol{Q}_x, \boldsymbol{R}_x]=\text{qr}(\boldsymbol{X})$, $[\boldsymbol{Q}_y, \boldsymbol{R}_y]=\text{qr}(\boldsymbol{Y})$

$[\boldsymbol{U}, \boldsymbol{D}, \boldsymbol{V}]=\text{svd}(\boldsymbol{R}_y^{-T}\boldsymbol{S}_{yx}\boldsymbol{R}_x^{-1})$

$\boldsymbol{J}=\boldsymbol{R}_x^{-1}\boldsymbol{V}$ and $\boldsymbol{L}=\boldsymbol{R}_y^{-1}\boldsymbol{U}$

5 A Framework for SIMs

5.1 Introduction

5.1.1 Objectives

N4SID, CVA and MOESP are quite different from each other in their development and computational methods, but they also show some commonalities in their basic ideas, for example, a set of lower dimensional linear combinations of the past data is taken as the estimated states in all these methods. However, little research has been done on the commonalities or the similarities among SIMs. The SIM algorithms are usually treated individually, and the connections among these methods are not clear.

The objective of this chapter is to investigate the commonalities of SIMs in both their philosophy and their computational approaches. A statistical framework will be proposed to show these commonalities and reveal the essential nature behind the computation procedures of SIM algorithms. By casting SIM algorithms into such a framework, these methods are expected to become much easier for understanding, and their similarities as well as differences should become clearer. This framework should also help the exploration for new SIM algorithms.

After a brief review of literature in the next subsection, a general framework for SIMs will be set up in Section 5.2. The following three sections provide the details of the framework and show how SIM algorithms fit to this framework. A simulation study will illustrate some key points of this framework. The last section includes a summary of this framework.

5.1.2 Literature Review

SIMs have been developed only over the past decade, and they are still immature in many aspects; therefore not much research has been done to unify or to build a framework for SIMs. An extensive search turned up only one published paper on this aspect (Van Overschee and De Moor, 1995). In this paper, a unifying theorem was set up by casting CVA and MOESP to the N4SID procedures, which act as a common computational framework. It was shown that the different SIM methods could be obtained by using different specific values for the left and right weighting matrices W_1 and W_2 in the reduction of the high-order oblique projection result $O=[(Y_f/U_f^{\perp})(P_{IO}/U_f^{\perp})^T][(P_{IO}/U_f^{\perp})(P_{IO}/U_f^{\perp})^T]^{-1}P_{IO}$ by a rank-limited approximation R (using the matrix notations defined in Section 2.1):

$$\underset{R}{Min}\left\|W_1(O-R)W_2\right\|^2 \qquad (5.1.1)$$

s.t. rank(R)=n

Here $/U_f^{\perp}$ stands for projection onto the orthogonal space of future inputs, i.e., operation of post-multiplying by $P_{ufo}=I-U_f^T(U_fU_f^T)^{-1}U_f$. In fact, $Y_f/U_f^{\perp}=Y_{f\_ro}$, $P_{IO}/U_f^{\perp}=P_{IO\_ro}$ and projection O is the estimated predictable subspace $L_1Y_p+L_2U_p$ by the regressing U_f out method shown in Appendix A4.2. Different SIMs correspond to different values of weighting matrices W_1 and W_2 as listed in Table 5.1. Apparently, in N4SID, the solution for the order-reduced projection R in (5.1.1) is a PCA (taking n significant components of SVD). In MOESP and CVA, it is a special weighted SVD involving the inverse of W_1 and W_2. These weighting matrices W_1 and W_2, as well as their inverses, are obscure in their physical meanings.

Table 5.1 Weighting matrices W_1 and W_2 in different SIMs

	CVA	N4SID	MOESP
W_1	$[(Y_f/U_f^{\perp})^T(Y_f/U_f^{\perp})]^{-1/2}$	I	I
W_2	P_{ufo}	I	P_{ufo}

Note: $P_{ufo}=I-U_f^T(U_fU_f^T)^{-1}U_f$

The above unifying theorem casts MOESP and CVA into the frame of N4SID and treats other SIMs as a special case of N4SID. For these methods, the similarities and

differences are hidden in the specific right and left weighting matrices. The physical meaning of these weighting matrices is not clear. By focusing on the detailed calculation procedures, this unifying theorem provides insight into their mathematical computation but does not reveal the common basic ideas behind these methods. Therefore, it does not provide a general conceptual framework for SIMs, nor does it help people understand the nature of SIMs, nor does it provide clear guidelines for developing new SIM algorithms.

5.2 A General Statistical Framework for SIMs

Each of the SIMs looks quite different from the other in the concepts, the computation methods and the interpretations. As shown, MOESP (elementary algorithm) does a QR decomposition on $[U_f; Y_f]$ and then a SVD on R_{22} (part of the R matrix). Part of the singular vector matrix is taken as an estimate of Γ_f. The A and C matrices of the state-space model are then estimated from this estimate Γ_f. The B and D are then estimated through another LS fitting with estimated A and C matrices. N4SID performs a SVD on the oblique projection of Y_f along U_f onto $[Y_p; U_p]$, and the right singular vectors are taken as estimates of the state variables, which are then used to fit to the state-space model. The estimated states in N4SID are interpreted as the results of a series of non-stationary Kalman filters. CVA uses CCA to estimate the state variables and then fits them to the state-space model. It is interpreted in terms of the maximum likelihood principle. As for the detailed algorithms, the difference between these SIMs seems so large that it is hard to find the similarities between them.

These SIM algorithms show some commonalities in their basic ideas and computation methods. In fact, the QR decomposition in MOESP is an alternative way to perform LS regression of Y_f against U_f, and $R_{21}R_{11}^{-1}$ is an estimate of H_f (unbiased if the inputs are white noise). $R_{21}Q_1$ is an estimate of the future input effects on the future horizon, and $R_{22}Q_2$ is an estimate of the predictable subspace. PCA (SVD) on $R_{22}Q_2$ provides the estimated Γ_f and estimated states. In N4SID, the coefficient matrix L_3 from regressing Y_f against $[Y_p; U_p; U_f]$

is an estimate of H_f, and L_3U_f is an estimate of the effects of future inputs on the future horizon. $L_1Y_p+L_2U_p$ or $Y_f-L_3U_f$ is the estimated predictable subspace. Based on this, PCA or RRA gives the estimated Γ_f and estimated states. The estimated process states are then used for fitting the state-space model. In CVA, the H_f matrix is estimated based on the fitted ARX model, and $Y_f-H_fU_f$ is the estimated predictable subspace. CCA on past data and this estimated predictable subspace gives estimates of the process states, which are used to fit the state-space model. From the above analysis, if the basic ideas and the computation methods behind all these methods are scrutinized from the viewpoint of statistical regression, these SIMs appear to be very similar, and follow a common framework. The framework consists of three steps:

i) Estimate the predictable subspace Γ_fX_k by a linear regression method based on the past data or by removing the future input effects from the future outputs

ii) Estimate the state variables based on the estimated subspace by one of the latent variable methods (LVMs)

iii) Then fit the estimated states to the state-space model to obtain the system matrices and noise covariance matrices

The first step in the framework is to estimate the predictable subspace Γ_fX_k, which is included in the future output space as shown in (3.1.7). This subspace can be predicted based on the past data as shown in (3.1.8). Removing H_fU_f from Y_f eliminates the confounding between the past data effects (Γ_fX_k) and the future input effects (H_fU_f). The second step of the framework is to estimate the process states based on the estimated predictable subspace. Both the estimated predictable subspace and the past data space are of high dimension but are highly correlated. LVMs on these two data spaces find out the linear combinations of the past data with the best predictability of the estimated predictable subspace; that is, the past history of a process is summarized into a few variables with the best information for the prediction of the future behaviour of the process. These linear combinations can therefore be interpreted as estimates of the process states. In the third step, the estimated process states are fitted to the

state-space model by some form of LS regression to obtain the system matrices. The regression residuals are the estimates of the stochastic noise variables, and their covariance matrices can be easily estimated.

This framework points out the function of each step and general methodology in each step; yet many different detailed approaches exist for each of these steps, especially for the first two steps. The overall picture of the framework is shown in Figure 5.1, and detailed discussions of these steps are provided in the next three sections. The major differences between SIMs are found in the choices of specific approaches in each of these steps. The original MOESP algorithm extracts Γ_f from the estimated predictable subspace. From the same subspace, process states can be estimated (refer to Section 3.4, and Van Overschee and De Moor, 1995). The state-based MOESP algorithms fit well to the framework.

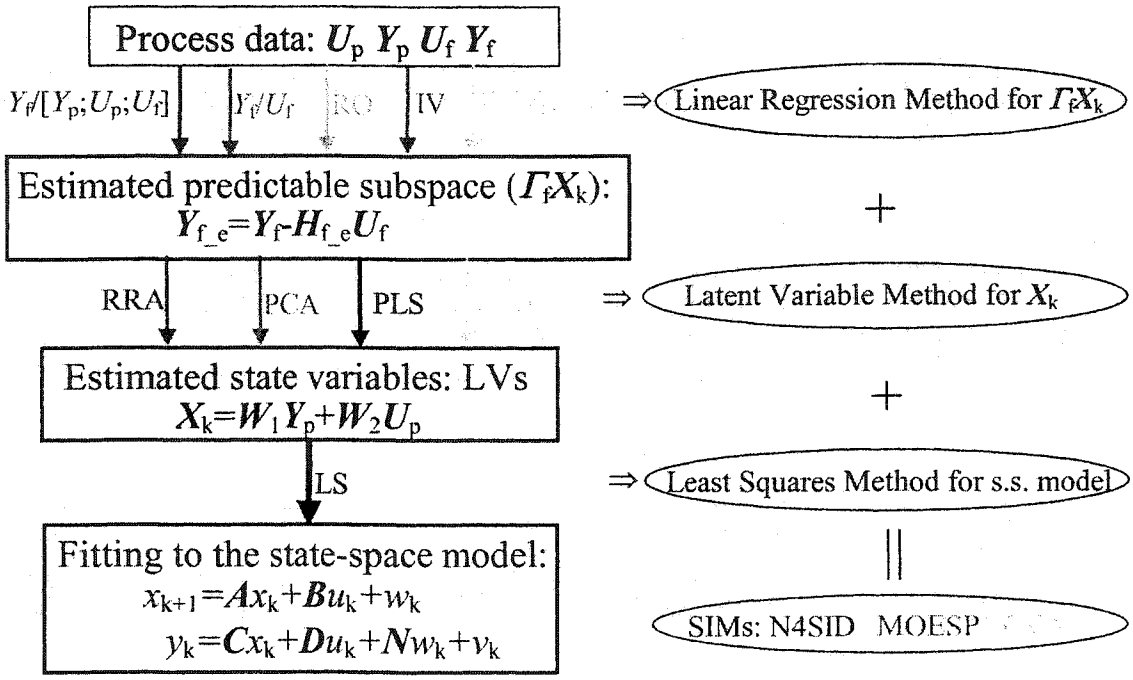


Figure 5.1 The framework for the SIM algorithms

The computational framework by Van Overschee and De Moor (1995) fits well into this framework, but is more restrictive since it forces all SIMs into the N4SID format. The oblique projection $\boldsymbol{O}$ in (5.1.1) is actually an estimate of the predictable subspace based on

the past input and output data. The resultant $\boldsymbol{R}$ in (5.1.1) is an approximation of $\boldsymbol{O}$ obtained by PCA, which is one of the many LVMs included in this framework. If looked at from this framework, the rank restriction on $\boldsymbol{R}$ in (5.1.1) is naturally imposed by the rank limitation of the true predictable subspace $\Gamma_f X_k$. This is not clear in the unifying theorem of Van Overschee and De Moor (1995). Their unifying theorem was based on regressing U_f out, which is only one of the many approaches summarized in this framework. This framework shows clearly the purpose as well as the physical meaning of the computation procedures; therefore, it reveals the fundamental ideas and the nature of SIM algorithms. The connections and differences among SIMs become apparent under this framework. In the unifying theorem of Van Overschee and De Moor (1995), the physical meaning of the weighting matrices as well as their effects on the final result is not clear, and therefore little guidance for new algorithms could be gained. This framework provides clear guidelines on the choice of SIM algorithms and strong heuristic directions for developing new algorithms.

System matrices of the state-space model can be estimated based on the estimated process states or the estimated observability matrix Γ_f (also refer to Viberg, 1995). The framework proposed in this chapter is for the state-based methods, but the first two steps of this framework are also helpful in understanding the Γ_f-based methods.

5.3 Estimation of the Predictable Subspace

5.3.1 Linear Regression for H_f to Estimate $\Gamma_f X_k$

In SIMs, the predictable subspace $\Gamma_f X_k$ should be estimated first in order to have a basis for estimation of the state sequence X_k or the extended observability matrix Γ_f. As shown in (3.1.7), $\Gamma_f X_k$ is a component of the future outputs Y_f, which also includes the future input effects $H_f U_f$. If the input signals are auto-correlated, $H_f U_f$ is correlated with $\Gamma_f X_k$ and therefore causes difficulties in separating these two components. The key problem in

estimation of the predictable subspace $\Gamma_f X_k$ is to remove $H_f U_f$ away from Y_f. Since the future inputs U_f are known, only the coefficient matrix H_f is unknown and needs to be estimated.

H_f shows the effects of U_f on Y_f and consists of the first f steps of impulse weights on lower diagonals (for SISO) or block weights on block lower diagonals (for MIMO). The true H_f is a lower block triangular matrix as shown in Section 2.1. These features are informative; however, most algorithms do not make full use of these features. Different algorithms use different methods to estimate H_f from the input and output data sets. These methods for estimating this coefficient matrix belong to the class of linear regression methods.

Once H_f is estimated, say H_{f_e}, then $Y_f - H_{f\_e} U_f$ is taken as an estimate of the predictable subspace. This estimate includes the effects of the estimation errors in H_{f_e} and the effects of future stochastic signals, which can be removed away by projection onto the past data. This projection procedure may induce some error; however, in most cases the error is much less than the effects of the future stochastic signals. Some subspace identification methods, such as N4SID, do the estimation of H_f (L_3) and projection onto the past data sets in one step.

5.3.2 Methods Used to Estimate H_f

There are many different approaches for estimating the H_f matrix to remove the future input effects from the future outputs in order to obtain the predictable subspace. The discussion in this subsection will focus on the differences among these approaches in the basic ideas, the choices of the data sets, the properties of the result, and the conditions for application. Some of the approaches have been employed in the existing SIMs, and others are novel approaches.

Regressing Y_f against U_f

Since H_f is the coefficient matrix relating U_f to Y_f, it is natural to estimate H_f by directly performing LS regression of Y_f against U_f. The basis for this regression is (3.1.7):

$$\begin{aligned} Y_f &= \Gamma_f X_k + H_f U_f + H_{s,f} W_f + V_f \\ &= H_f U_f + \left(\Gamma_f X_k + H_{s,f} W_f + V_f\right) \end{aligned} \tag{3.1.7}$$

and the result is:

$$H_{f\_est} = Y_f U_f^T \left(U_f U_f^T\right)^{-1} \tag{5.3.1}$$

A basic assumption for an unbiased result is that the future inputs are uncorrelated with the regression error, which includes both the effect of state variables $\Gamma_f X_k$ and the effects of future stochastic signals. Once H_f is estimated, the predictable subspace is estimated as $Y_f - H_{f\_e} U_f$. The original MOESP elementary algorithm (Verhaegen, 1992) uses this approach to implicitly estimate H_f and the predictable subspace via QR decomposition on $[U_f; Y_f]$. In one early CVA algorithm, the effects of future inputs were removed by this method (Larimore, 1990). This approach for estimation of H_f is unbiased only for cases where the inputs are not auto-correlated signals, such as independent white noise signals or PRBS with switching time period of 1. If the input sequences are auto-correlated (making $\Gamma_f X_k$ correlated with U_f), this method regresses part of the state effect away and gives a biased result for the predictable subspace. The estimated H_f will have large variance in any case since the regression error term (which includes $\Gamma_f X_k$) have a large variance.

Regressing of Y_f against $[Y_p; U_p; U_f]$

Based on (3.1.6), we know the predictable subspace $\Gamma_f X_k$ in (3.1.7) can be expressed as a linear combination of the past inputs U_p and past outputs Y_p as shown in (3.1.9). Therefore, it is a natural choice to regress Y_f against $[Y_p; U_p; U_f]$ for estimates of $\Gamma_f X_k$ and the future input effects $H_f U_f$. The regression result is shown in (3.2.1). Here the regression coefficient (L_3) for U_f is an estimate of H_f (H_{f_e}) and the part corresponding to the past data ($L_1 Y_p + L_2 U_p$) is an estimate of the predictable subspace, which is equivalent to the projection of $Y_f - H_{f\_e} U_f$ onto the past data (refer to Section 4.2). This approach is employed in N4SID. In practice, the regression can be realized by QR decomposition of $[U_f; P_{IO}; Y_f]$, which is also employed in the PO-MOESP (Verhaegen and Dewilde, 1994, past output (PO-) MOESP).

As shown in Section 3.2, the estimates of $\Gamma_f X_k$ and H_{f_e} will have slight bias if the true process does not follow an ARX model. In practice, the bias is very small if a long past horizon is used (although this will lead to increase in the variance). However, the variance of the estimation result is much smaller than that from regressing Y_f against U_f since the predictable subspace is not included in the error term in this regression.

Constructing H_f from estimated impulse weights

As shown in Section 2.1, the trapezoid matrix H_f consists of the first f impulse weights $w_0=D$, $w_i=CA^{i-1}B$. These impulse weights can be estimated from a fitted ARX model, which is obtained by regressing y_k against u_k (if $D\neq 0$), past inputs (U_p) and past outputs (Y_p). These estimated impulse weights can therefore be employed to construct the estimated H_f as:

$$H_{f\_e}=\begin{pmatrix} \hat{w}_0 & 0 & 0 & \cdots & 0 \\ \hat{w}_1 & \hat{w}_0 & 0 & \cdots & 0 \\ \hat{w}_2 & \hat{w}_1 & \hat{w}_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & 0 \\ \hat{w}_{f-1} & \hat{w}_{f-2} & \hat{w}_{f-3} & \cdots & \hat{w}_0 \end{pmatrix} \tag{5.3.2}$$

The predictable subspace then is estimated as $Y_f - H_{f\_e}U_f$. This estimate includes all the future noise. N4SID_Hf and CVA_Hf algorithms proposed in Chapter 3 employ this approach to estimate H_f and the predictable subspace.

It is worth mentioning that the first f impulse weights can also be estimated via a FIR model by regressing y_k against u_k (if $D\neq 0$) and the long lag steps of the past inputs.

Regression out method

U_f can be regressed out of both sides of (3.1.9) by projecting all the terms onto the orthogonal space of U_f, i.e., by post-multiplying both sides with $P_{ufo}=I-U_f^T(U_fU_f^T)^{-1}U_f$. This procedure removes the U_f term from the equation, and the coefficient matrices for past data in (3.1.9) can be obtained by regressing $Y_{f\_ro}=Y_fP_{ufo}$ against $P_{IO\_ro}=P_{IO}P_{ufo}$:

$$[L_1 \quad L_2] = Y_{f\_ro} P_{IO\_ro}^T \left(P_{IO\_ro} P_{IO\_ro}^T\right)^{-1} = Y_{f\_ro} P_{IO\_ro}^T \left(P_{IO\_ro} P_{IO\_ro}^T\right)^{-1} \qquad (5.3.3)$$

The regression coefficient is equivalent to the coefficient for the past data from regressing $\boldsymbol{Y}_f$ against $[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{U}_f]$, i.e., that in N4SID (refer to Appendix A4.2), and the predictable subspace can be estimated by multiplying this coefficient matrix to the past data $\boldsymbol{P}_{IO}$ (equal to $\boldsymbol{O}$ in (5.1.1)). This approach was implied and employed in the unifying theorem (Van Overschee and De Moor, 1995) and other places (Carrette, 2000). This approach is also used in the CVA_RO algorithm.

Another similar approach is to regress past data $\boldsymbol{P}_{IO}$ out of both sides of (3.1.9) (projecting to the orthogonal space of $\boldsymbol{P}_{IO}$, i.e., post-multiplied by $\boldsymbol{P}_{po}=\boldsymbol{I}-\boldsymbol{P}_{IO}^T(\boldsymbol{P}_{IO}\boldsymbol{P}_{IO}^T)^{-1}\boldsymbol{P}_{IO}$). This procedure removes the $\boldsymbol{P}_{IO}$ term from the equation, and the coefficient matrices for the future inputs in (3.1.9) can then be obtained by regressing $\boldsymbol{Y}_f\boldsymbol{P}_{po}$ against $\boldsymbol{U}_f\boldsymbol{P}_{po}$ as:

$$H_{f\_e} = Y_f P_{po} P_{po}^T U_f^T \left(U_f P_{po} P_{po}^T U_f^T\right)^{-1} = Y_f P_{po} U_f^T \left(U_f P_{po} U_f^T\right)^{-1} \qquad (5.3.4)$$

The future input effects can be estimated as $\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$, and the predictable subspace is estimated as $\boldsymbol{Y}_f-\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$.

Instrumental variable method

If a set of variables are fully correlated with $\boldsymbol{U}_f$ but are not correlated with $\boldsymbol{X}_k$ and the future noise, these variables can be used as the instrumental variables (IVs) and an unbiased $\boldsymbol{H}_f$ can be estimated based on (3.1.7) by the Instrumental Variable Method (IVM). For auto-correlated inputs, $\boldsymbol{U}_f$ correlates with $\boldsymbol{X}_k$ through its correlation with the past input data, which contribute to $\boldsymbol{X}_k$. The part of $\boldsymbol{U}_f$, which is in the orthogonal space of the past input data, has no correlation with $\boldsymbol{X}_k$. This part of $\boldsymbol{U}_f$ can be calculated by regressing $\boldsymbol{U}_p$ out of $\boldsymbol{U}_f$ and taken as the IVs:

$$IV = U_f - U_f U_p^T \left(U_p U_p^T\right)^{-1} U_p \qquad (5.3.5)$$

and then $\boldsymbol{H}_f$ is easily estimated by multiplying the transpose of the IVs on both sides of (3.1.7). The result is:

$$H_{f\_e} = Y_f \cdot IV^T \left(U_f \cdot IV^T\right)^{-1} \qquad (5.3.6)$$

The predictable subspace can then be easily estimated as Y_f-$H_{f\_e}U_f$.

Other regression methods

In Section 4.2, several approaches are listed to estimated H_f matrix based on relationships between data sets and prior knowledge of H_f, such as LVMs, ridge regression, forcing the L_3 matrix in N4SID to follow the features of H_f matrix (N4SID_tri). Many other methods are available to estimate H_f matrix.

All the above methods for estimation of H_f matrix are variants of the linear regression method: they differ only in their choice of the independent and dependent variables, and the degree of utilization of prior knowledge about the features of H_f. The key problem is the correlation between U_f and X_k, which arises from auto-correlation of the input signals in the open loop case and arises naturally as a result of feedback in the closed-loop case (see Chapter 6). The estimation accuracy (bias and variance) in each method depends on the input signal, the true model structure, and the signal-to-noise ratio (SNR).

5.4 Estimation of the State Variables

Estimation of the state variables is a key step in all SIM algorithms. Process states can be estimated based on the estimated predictable subspace and the past data, and fit to the state-space model to get the system matrices. The method used for state variable estimation is generally one of the LVMs.

5.4.1 Latent Variable Methods for State Estimation

The predictable subspace estimated by the linear regression methods discussed in the last section is a high-dimensional space (far larger than the system order n) consisting of highly correlated variables. If there were no error terms, this subspace should be only of rank n, and any n independent variables in the space or their linear combinations could be taken as the state variables. However, the estimation error generally makes the estimated subspace full

rank. Directly taking any arbitrary n variables as the process states will lose the useful state information in other variables and introduce large estimation error. It is therefore desirable to extract only n linear combinations from this highly correlated high-dimensional space and keep as much state information as possible. This is exactly the ultimate goal of LVMs, and the situation of high dimension and high correlation is what LVMs were developed to deal with. Therefore SIMs employ LVMs to estimate the state variables from the estimated predictable subspace and the past data.

Latent variables (LVs) are linear combinations of the original variables for optimization of a specific objective function. There are a variety of latent variable methods based on different optimization objectives. As shown in Section 4.1, PCA, PLS, CCA, and RRA are LVMs that maximize variance, covariance, correlation, and predictable variance respectively. Different LVMs are used in different SIMs to estimate the state variables.

5.4.2 Methods Used for State Estimation

Though the computation methods for state estimation in different SIMs and their corresponding explanations seem quite different from each other, these methods are essentially LVMs. All these LVMs can be used for state estimation, although each method may have unique accuracy and efficiency in state estimation.

Principal Component Analysis (PCA)

The estimated predictable subspace $\boldsymbol{Y}_f - \boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ consists of $\Gamma_f \boldsymbol{X}_k$ and the effects of the future stochastic signals. $\Gamma_f \boldsymbol{X}_k$ is the major component in the subspace with high correlation. PCA (a SVD procedure) on the estimated predictable subspace will pick up the state information in the significant principal components (PCs) and these PCs are taken as the estimated process sates. The high correlation in $\Gamma_f \boldsymbol{X}_k$ greatly increases the information (variance) of the process states contained in these PCs, while the stochastic noise information, if random in nature, is largely rejected by eliminating the PCs associated with

the small singular values. Therefore, the SNR in these dominant (n) PCs is much better than that in the estimated predictable subspace ($\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$).

On the other hand, the estimated predictable subspace is usually projected onto the past data before performing the PCA (as in N4SID where $\boldsymbol{L}_1\boldsymbol{Y}_p+\boldsymbol{L}_2\boldsymbol{U}_p$ is used as the predictable subspace). Only $\Gamma_f\boldsymbol{X}_k$ in the estimated predictable subspace is correlated to the past data and can be predicted based on the past data. This projection removes the effects of the future stochastic signals, and the result is a better estimation of the true predictable subspace $\Gamma_f\boldsymbol{X}_k$. PCA on this projection result gives better estimates of the process states, and the estimated states are linear combinations of the past data.

In order to ensure the first n PCs are unbiased estimates of the process states, there are three assumptions on the estimated predictable subspace:

- The estimation errors are insignificant compared to $\Gamma_f\boldsymbol{X}_k$
- The estimation errors are not correlated with $\Gamma_f\boldsymbol{X}_k$
- There is no severe correlation among the estimation errors.

These assumptions are to ensure that the first n PCs show the process states instead of the noise, and these PCs are unbiased estimates of the process states (not twisted by the estimation errors) in the SVD procedure. The first assumption is well satisfied if the signal-to-noise ratio (SNR) of the process data is large enough. The second assumption holds generally since the deterministic part is un-correlated to the stochastic part. The third assumption is not satisfied if there exist high correlation among the effects of the future stochastic signals. However, the projection of the estimated predictable subspace onto the past data can satisfy this assumption approximately with only small projection error.

The elementary MOESP algorithm directly uses PCA (SVD) on the estimated predictable subspace $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ and PCs are estimates of process states. These estimates bear large estimation errors since this estimated predictable subspace includes the effects of the future stochastic signals. The PO-MOESP applies PCA to the projection of estimated predictable subspace onto the past data space, therefore the result is generally improved.

N4SID applies PCA (SVD) on the oblique projection of $\boldsymbol{Y}_f$ along $\boldsymbol{U}_f$ onto $[\boldsymbol{Y}_p;\boldsymbol{U}_p]$. As discussed in Section 4.2, this result can be deemed the predictable subspace projection of $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ onto the past data. The significant PCs are better estimates of process states than those directly from PCA on $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$.

The first n PCs have the best predictability of the future outputs based on the past data in the sense of total variance, i.e., $\min trace\left\{(Y-\hat{Y})(Y-\hat{Y})^T\right\}$ (Y is the estimated predictable subspace and $\hat{Y}$ is its prediction based on the first n PCs). This objective implies that the final result of the PCA depends on the relative magnitudes of the outputs; therefore the different outputs need to be scaled appropriately.

Reduced Rank Analysis (RRA)

As shown in Section 4.2, N4SID is equivalent to performing RRA on the past data $\boldsymbol{P}_{IO}=[\boldsymbol{Y}_p;\ \boldsymbol{U}_p]$ and the estimated predictable subspace $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ ($\boldsymbol{H}_{f\_e}$ is the coefficient matrix $\boldsymbol{L}_3$ in (3.2.1)), and the significant LVs from this RRA are taken as the estimates of process states. These LVs (estimated states) are linear combinations of the past data $\boldsymbol{P}_{IO}$ and have the best predictability of $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$, that is, these LVs summarize the process history and have the best predictability of the process future behavior. This best predictability is in the sense of total predictable variance, i.e., $\min trace\left\{(Y-\hat{Y})(Y-\hat{Y})^T\right\}$ where $\boldsymbol{Y}=\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ and $\hat{Y}$ is its prediction based on the estimated states. This RRA directly picks up LVs with the most significant predictability. Therefore, RRA is more efficient in estimation of the process states than the original N4SID, where all the predictable subspace is estimated by LS regression (possible ill-conditioned) first, and then PCA is performed on this subspace to get the most significant LVs.

From the discussion in the last section, it is clear that several methods are available for estimation of matrix $\boldsymbol{H}_f$ other than that used by N4SID, and therefore several other methods are available to estimate the predictable subspace as $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$. The significant LVs from RRA on past data $\boldsymbol{P}_{IO}$ and this estimated predictable subspace $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ also provide estimates of the process states.

Canonical Correlation Analysis (CCA)

In this approach, CCA is applied on the past data $\boldsymbol{P}_{IO}=[\boldsymbol{Y}_p;\ \boldsymbol{U}_p]$ and the estimated predictable subspace $\boldsymbol{Y}=\boldsymbol{Y}_f\text{-}\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$. The canonical variables (CVs) are linear combinations of the past data, and the canonical correlations are ranked from the highest correlation to the lowest. The LVs corresponding to significant canonical correlations are taken as the estimated process states. These estimated states maximize the variance percentages in orthogonal directions of $\boldsymbol{Y}$ explained the past data, that is, $\min trace\left\{\left(Y-\hat{Y}\right)\left(YY^T\right)^{-1}\left(Y-\hat{Y}\right)^T\right\}$ where $\boldsymbol{Y}=\boldsymbol{Y}_f\text{-}\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ and $\hat{Y}$ is its prediction based on the estimated states. Here the weighting matrix $(\boldsymbol{YY}^T)^{-1}$ makes the LV explain maximum percentage of variance (relative to the total variance in the direction) rather than the absolute value of the variance. In other words, LVs from $\boldsymbol{P}_{IO}$ are the best estimates of independent variates in $\boldsymbol{Y}_f\text{-}\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$. Since this best predictability is in the relative basis, the CCA result does not depend on the scaling of the data sets. These CVs summarize the information in the past data having the best predictability of the future outputs, and hence they are consistent with the definition of process states.

CCA can also be applied to the results of regressing $\boldsymbol{U}_f$ out both the past data and the future outputs, i.e., to $\boldsymbol{P}_{IO\_ro}$ and $\boldsymbol{Y}_{f\_ro}$; however, the direct CVs $\boldsymbol{J}_1\boldsymbol{P}_{IO\_ro}$ are estimates of $\boldsymbol{X}_{k\_ro}$ instead of $\boldsymbol{X}_k$. Here the coefficient matrix $\boldsymbol{J}_1$ should be applied to the original past data to get state estimates $\boldsymbol{J}_1\boldsymbol{P}_{IO}$, and these estimates are no longer orthogonal (refer to Appendix A3.2 for detailed proof). However, since part of the state signal is removed by regressing $\boldsymbol{U}_f$ out while the noise is kept intact, the data set $\boldsymbol{Y}_{f\_ro}$ has a worse SNR than $\boldsymbol{Y}_f\text{-}\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ in general.

Partial Least Squares (PLS)

PLS is another LVM and a possible choice for state estimation in SIMs. In this approach, PLS is applied to the past data $\boldsymbol{P}_{IO}=[\boldsymbol{Y}_p;\ \boldsymbol{U}_p]$ and the estimated predictable subspace $\boldsymbol{Y}_f\text{-}\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$, and the significant LVs are taken as the estimated process states. However, since the objective of PLS is to model the covariance of $\boldsymbol{P}_{IO}$ and $\boldsymbol{Y}_f\text{-}\boldsymbol{H}_f\boldsymbol{U}_f$, i.e., maximizing both the

variance in $\boldsymbol{P}_{IO}$ and $\boldsymbol{Y}_f$-$\boldsymbol{H}_f\boldsymbol{U}_f$ as well as their correlation, the LVs are partially twisted towards the large variance directions in the past data. Therefore the LVs are a compromise between modeling the past data $\boldsymbol{P}_{IO}$ and modeling the relationship between the past data and the predictable subspace. The past data includes the past inputs, which come from an experimental design and are not a property of the process. Therefore, the LVs from PLS include not only the process state information but also information on the structure of the input signals. In order to get an adequate process model, PLS therefore needs more LVs than the system order to include the state information, and will not provide a minimal order state-space model in general (refer to Section 4.3 and simulation examples in Section 4.5). From the objective of PLS, it is also clear that the PLS result is dependent on the data scaling. In brief, PLS is a feasible way to estimate the process states, but it is not as efficient as RRA and CCA at extracting only the system states.

Determination of the system order

The final system order is determined by how many LVs are taken as the estimated states. Many criteria can be employed for the system order determination. One generic criterion is to look at the significance of these LVs based on the corresponding eigenvalues, singular values or the canonical correlations. Cross-validation is one such procedure for assessing significance (Wold, 1978). Another method is to check the cross-correlation between the residuals and the inputs to test whether the deterministic model is adequate. AIC is a quantitative method for system order determination with consideration of both the model adequacy and the model complexity. (See Section 2.1 for the definition of AIC. A correction factor is added to the expression of AIC for a small sample case, refer to Hurvich and Tsai, 1989)

In the previous discussion of the framework for SIMs, there are several approaches for estimation of the predictable subspace and for the extraction of the process states. Each SIM algorithm corresponds to a special combination of the approaches used in the two steps. For example, CVA_Hf algorithm uses a fitted ARX model to construct $\boldsymbol{H}_f$ for estimation of the predictable subspace and then uses CCA for estimation of the process states. It is worthy

to point out that by using various combinations of the approaches for the predictable subspace estimation and the approaches for the state variable estimation, a whole variety of new SIM algorithms can be formulated, such as SIM-IV-CCA, SIM-ARX-RRA, SIM-ARX-PLS, and SIM-$\boldsymbol{Y}_f/[\boldsymbol{P}_{IO};\boldsymbol{U}_f]$-CCA, etc. (each SIM algorithm is based on a pair of specific approaches for the first and second steps respectively of the framework). Some of the new algorithms give similar results to those from N4SID or CVA, and they will be evaluated in the simulation study in Section 5.6.

5.5 Fitting to the State-space Model

After estimation of the predictable subspace and the process states, the last step of SIMs is to fit the estimated process states to the state-space model. In this step, the system matrices are estimated by LS regression, and the noise covariance is estimated based on the regression residuals. This is a common step for SIM algorithms. For a clearer picture of the SIM algorithms, several issues will be discussed in this section, such as the forms for state-space model and the effects of the estimation errors in the state variables.

5.5.1 State-space Model Forms Employed

Before fitting the estimated state variables to the state-space model, the state-space model form should be determined. There are several types of state-space model forms. The state-space model shown by (2.1.1) and (2.1.2)

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{2.1.1}$$

$$y_k = Cx_k + Du_k + Nw_k + v_k \tag{2.1.2}$$

has the process noise and the output measurement noise with parameters in system matrices. This state-space model can also be presented in the innovation form as:

$$x_{k+1} = Ax_k + Bu_k + K\alpha_k \tag{5.5.1}$$

$$y_k = Cx_k + Du_k + \alpha_k \tag{5.5.2}$$

Echelon form is a special case of the above innovation form with the following parameterization of matrix A and C:

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} 0 & I_{n-l} \\ & A_2 \end{bmatrix} \qquad C = [I_l \quad 0] \tag{5.5.3}$$

Here block matrix $A_2 \in \Re^{l \times n}$ is fully parameterized. The echelon form is a rearrangement of the observer canonical form (refer to Ljung, 1999) and has the minimum number of parameters to be estimated.

Different state-space model forms associate with different state bases and with different numbers of parameters. The form in (2.1.1) and (2.1.2) is based on an arbitrary state basis, and all the elements in A, B, C and D are assumed unknown before fitting to this model structure. Therefore there are $n^2+n(m+l)+ml$ $+n(n+1)/2+l(l+1)/2$ parameters including the noise covariance matrices (m is the number of inputs, l is the number of outputs). It is referred to here as the full matrix form. However, not all the elements in the system matrices are independent since different sets of system matrices are similar to each other with a full rank transform matrix, which is also the transform matrix between different state bases. The number of the independent parameters is less than the total number of the elements. In the canonical (echelon) form, some of the elements in matrix A and C are set to be 0 and others to be 1. In the canonical form, and there are only $n(2l+m)+ml+n(n+1)/2$ parameters to be determined. However, the basis of the state variables in the canonical form is not an arbitrary one. This state basis includes the independent current output variables and their independent future predictions. The basis is not unique though the total number of states is set.

The canonical form has an attractive advantage over the full matrix form, namely that fewer parameters need to be determined by the LS regression. This advantage will reduce the confidence limits of the estimated parameters. However, the state basis is usually ill-conditioned for variables in the basis (the independent current output variables and their independent future predictions) are usually highly correlated. The condition number in the LS regression will be very large, and the regression result will be very sensitive to the noise. This

ill-conditioning problem will lead to large variances for the estimated parameters. In this sense, the advantage of fewer parameters is cut back by the ill-conditioning problem more or less. In addition to that, one needs to devote special efforts to determining this basis before fitting to the canonical state-space form. The estimated states obtained from the LMVs are usually orthogonal to each other, and this is a very attractive feature for the LS regression. Considering all these advantages and disadvantages, the full matrix is generally employed by SIM algorithms to the estimated state variables to the state-space model.

5.5.2 Fitting to the State-space Model

Fitting the estimated state variables to the state-space model (2.1.1) and (2.1.2) in the full matrix form is a common step for SIM algorithms. It gives a parsimonious, lower-order model with smoother impulse responses as shown in Section 4.4 and 4.5. The fundamental reasons and advantages of this procedure will be discussed in this section, and several computation issues related to this LS regression will also be discussed.

Estimation of X_{k+1}

In fitting the estimated states to the state-space model in (2.1.1) and (2.1.2), system matrices A, B, C and D are obtained by LS regressing $[X_{k+1}; y_k]$ against $[X_k; u_k]$. In SIM algorithms, X_k is estimated by LVMs, but X_{k+1} needs to be determined before performing the regression.

In practice, there are at least three different approaches to get the value of X_{k+1}. The first approach is to shift the value of estimated X_k by one time point to form the estimate of X_{k+1}. The total N-p-f+1 time points in X_k correspond to k=p+1, p+2, …, N-f+1 (N, p and f are the total data points, the past lag steps and the future lag steps respectively). The values for k=p+2, p+3, …, N-f+1 in X_k are taken as the first N-p-f values of X_{k+1} (no value corresponds to the last time point in X_k). The shifting leads to one effective data point lost for the LS regression, and this has little effect on the final result for a long data set. In general, this approach is simple and effective. The second approach is based on the coefficient matrix

between the estimated states and the past data, such as the J matrix in CCA_Hf algorithm (where $X_k=JP_{IO}$) and the similar coefficient matrix from RRA in N4SID_Hf algorithm. For the estimate of X_{k+1}, the current time point is extended for one data point in the past data, e.g., in $X_k=JP_{IO}$, the current time points correspond to $k=p+1, p+2, \ldots, N-f+1$; if P_{IO} collects the past data corresponding to $k=p+2, p+3, \ldots, N-f+2$, this relationship will give the result for X_{k+1}. In fact, only the state vector for the last time point $k=N-f+2$ is to be estimated based on the known X_k. The third approach for X_{k+1} is to increase the past horizon by one step and decrease the future horizon by one step, i.e., $p+1$ steps in past horizon from $k-p$ to k and $f-1$ step in future horizon from $k+1$ to $k+f-1$ (for X_k, the past horizon is from $k-p$ to $k-1$ and the future horizon is k to $k+f-1$). This usually leads to unequal past and future horizon lengths. Similar procedures as those for X_k can be preformed to obtain the estimate of X_{k+1}. This approach is employed in the original N4SID algorithm, and it is applicable for other SIM algorithms. This approach increases the computation load significantly compared to the other two approaches.

In fact, the second approach can extend to more data points in the past data (the last data point corresponding to $k=N$), and therefore the effective data points for the LS regression can extend to $N-p$ from $N-p-f$ as in the first approach. This increase of the total effective regression data length will reduce the variances of the parameters. This improvement may not be significant for long data sets, but it becomes more significant when the collected data is short and a relatively long future horizon is used in a SIM algorithm. In the third approach shown above, X_k and X_{k+1} are estimated by two different LS regressions.

Effects of estimation errors in state variables

In the ordinary LS regression, only the regressed variable is assumed to be affected by noise, and the regressor variables are assumed to be noise-free (refer to (2.1.9)). Here in the LS regression of fitting the estimated process states to the state-space model, $[X_{k+1}; y_k]$ is regressed against $[X_k; u_k]$. It differs from the ordinary LS regression in that there are estimation errors in the estimated state variables (X_k). This error-in-variable LS regression theoretically gives biased results for the system matrices, and it is equivalent to ridge

regression if the errors in the estimated states are uncorrelated white noise with equal variance (Wang, 2000). In practice, this bias effect becomes noticeable only when the estimation errors in the estimated states become very large and significant compared to the magnitude of the true states. This happens only when the SNR is very low in the collected data set, and therefore there are large estimation errors on the estimated states by the LVMs. When the SNR is reasonable large (such as 1.0), the bias is small and does not cause noticeable problems on the final result.

The bias problem can be avoided if an IV method is used in the procedure for the system matrices instead of the LS regression. The instrumental variables (IVs) should be correlated to both $[X_k; u_k]$ and $[X_{k+1}; y_k]$ but not correlated with the noise, that is, the IVs should closely relate to X_k but with no noise. One simple way is to take only the contribution of the past inputs (of p steps) in the LVs (estimated states) as IVs. These IVs can be far away from the true states in the magnitude, and will increase the variances of the final parameters. However, the IVs constructed in these approaches are only related to the deterministic states without correlation with the stochastic states, and so the IV method may not be applicable to the situations with stochastic states in the process.

It is worth mentioning that the error-in-variable problem also exists in the LS regression for the A and C matrices based on the estimated extended observability matrix (Γ_f) as in the original MOESP method. However, it is hard to construct IVs in the Γ_f-based method to avoid the bias problem.

After the system matrices are obtained by this LS regression or IV method, the residuals are estimates of the stochastic noise w_k and v_k in (2.1.1) and (2.1.2). The covariance matrices of the stochastic variables can be estimated based on these residuals. Variances of the state-space model parameters can be estimated approximately based on these residuals as in ordinary LS regression, or by numerical methodologies such as Jackknife and Bootstrapping methods (refer to Efron and Tibshirani, 1993). The final state-space model in the full form can be transformed to other state-space model forms, such as the innovation form or the canonical form, or to other representations, such as transfer functions.

Parsimonious model from fitting to the state-space model

If the latent variables (estimated states) from LVMs are fitted only to the output equation (2.1.2), the result will be latent variable regression (LVR) models. Since the LVs (estimated states) in LVR are expressed as linear combinations of the past inputs and outputs over a long past horizon, the final LVM model will be a high-order ARX form (of order *lp*). On one hand, so many parameters cause over-fitting and noisy impulse (step, frequency) responses. On the other hand, these LVs (estimated states) usually have strong temporal correlations, but they are treated independently without consideration of the temporal correlations. Furthermore, any common dynamics (common poles) among the transfer functions in MIMO system are treated independently as different dynamics (different poles).

Fitting the estimated state variables (LVs) to the state-space model in fact imposes a lower order and more parsimonious structure on the model. These state variables do not evolve in an arbitrarily free manner. Equation (2.1.1) shows that the states of two adjacent time points are related to each other via matrix $\boldsymbol{A}$, and the dynamic characteristics of the states is summarized in the system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. Fitting the estimated process states to the state equation (2.1.1) essentially is to compute the process states recursively, i.e., only based on the states and input of the previous time point. This fitting procedure in fact is to impose a parsimonious structure of order n onto the noisy estimated states. This procedure trims down the over-fitting in the estimated states and results in a parsimonious model with removal of the erratic noise away from the impulse (step and frequency) responses. It is analogous to the Box-Jenkins methodology of imposing a lower order transfer-function model structure on the noisy estimated FIR weights. Here in SIMs, this structure is naturally decided by the estimated system order. This structure summarizes all the common dynamics among the transfer functions in a MIMO process. This feature is particularly useful for modeling a piece of equipment (such as a reactor, a distillation tower) or several pieces of equipment with strong interactions. In these cases, the input/output dynamics generally have some common dynamics (poles) since all the input-output transfer functions are closely related to the same process.

As for the accuracy aspect of the calculated state variables, on one hand, the erratic errors caused by over-fitting in the estimated state variables are smoothed out (filtered out) by this LS fitting procedure; on the other hand, the fitting errors on the states at one time point will propagate to the future steps via matrix A, however, these errors will soon die out for the eigenvalues of A (system poles) are within a unit circle for a stable process.

5.6 Simulation Study

In this section, a simple simulation example is used to illustrate the various approaches for each step of the SIM framework as well as their similarities and differences. The example is a first order SISO process with AR(1) noise:

$$y_k = \frac{0.2z^{-1}}{1-0.8z^{-1}}u_k + \frac{1}{1-0.95z^{-1}}e_k$$

The input signal u_k is a PRBS signal with magnitude of 4.0 and switching time period T_s=5. In the simulation, 1000 data points are collected with var(e_k)=1.0, and SNR is about 0.93 (in sense of variance at the output). Both u_k and y_k are scaled to unit variance in the computation and 7 lag steps are used for both the past and future horizons in each method.

5.6.1 Estimation of the Predictable Subspace

As shown in Section 5.3, the predictable subspace can be obtained based on the estimated H_f matrix. In this subsection, different approaches for H_f matrix estimation are applied to the simulation example, and the results are compared to the true result. The first 7 true impulse weights are 0, 0.2, 0.16, 0.128, 0.1024, 0.0819, and 0.0655 respectively, and the true H_f consists of these impulse weights in its lower diagonals in the Toeplitz matrix form (see Section2.1):

0	0	0	0	0	0	0
0.2	0	0	0	0	0	0
0.16	0.2	0	0	0	0	0
0.128	0.16	0.2	0	0	0	0
0.1024	0.128	0.16	0.2	0	0	0
0.0819	0.1024	0.128	0.16	0.2	0	0
0.0655	0.0819	0.1024	0.128	0.16	0.2	0

The estimated H_f matrix by regressing Y_f directly against U_f (MOESP elementary algorithm) has the following result (converted back to the original variable units to be comparable to the true H_f given above):

0.6357	-0.0284	-0.0260	-0.0258	-0.0645	0.1209	-0.0422
0.6847	-0.0188	-0.0174	-0.0260	-0.0258	0.1130	-0.0578
0.5360	0.1676	-0.0039	-0.0174	-0.0260	0.1132	-0.0499
0.4110	0.1380	0.1805	-0.0039	-0.0174	0.0806	-0.0173
0.3276	0.0986	0.1425	0.1805	-0.0039	0.0675	-0.0020
0.2605	0.0818	0.0992	0.1425	0.1805	0.0637	0.0132
0.2135	0.0622	0.0784	0.0992	0.1425	0.2359	0.0333

This method gives very biased results because of the strong auto-correlation in the PRBS signal.

Regressing Y_f against $[Y_p; U_p; U_f]$ gives the following estimated H_f (L_3 in the original N4SID algorithm):

-0.0035	0.0048	-0.0126	0.0275	0.0246	-0.0380	0.0080
0.1866	0.0026	-0.0089	0.0091	0.0546	-0.0109	-0.0241
0.1558	0.1832	-0.0008	0.0078	0.0391	0.0211	-0.0295
0.1056	0.1514	0.1810	0.0180	0.0351	0.0056	-0.0016
0.0736	0.1092	0.1407	0.1949	0.0499	0.0023	0.0070
0.0640	0.0789	0.0963	0.1537	0.2283	0.0165	0.0118
0.0510	0.0696	0.0659	0.1029	0.1906	0.1969	0.0287

This result is much closer to the true value. The elements on each lower diagonal are close to each other, and they are estimates of the corresponding impulse weights. The upper triangle part is close to null but still significant (some elements are as large as about 30% of the largest impulse weight in the magnitude and similar to the smallest impulse weight). If the elements on the upper triangle part are set 0, and the elements on each lower diagonal are set to their mean value respectively, the result is the estimated $\boldsymbol{H}_f$ matrix used in N4SID_tri algorithm in Section 4.2. The method based on regressing $\boldsymbol{P}_{IO}$ out gives the same result as above with differences due to very small numerical errors.

Fitting y_k against $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ gives the following ARX model:

$$y_k=0.9464y_{k-1}+0.0045y_{k-2}+0.0332y_{k-3}+0.0151y_{k-4}+0.0229y_{k-5}-0.0552y_{k-6}-0.0261y_{k-7}+$$
$$0.1997u_{k-1}-0.0344u_{k-2}-0.0386u_{k-3}-0.0238u_{k-4}-0.0301u_{k-5}-0.0069u_{k-6}-0.0113u_{k-7}$$

and the first 7 impulse weights from this model are 0, 0.1997, 0.1546, 0.1086, 0.0863, 0.0602, and 0.0610 respectively. These impulse weights are used to construct the following estimated $\boldsymbol{H}_f$ matrix (used in algorithms such as CVA_Hf):

0	0	0	0	0	0	0
0.1997	0	0	0	0	0	0
0.1546	0.1997	0	0	0	0	0
0.1086	0.1546	0.1997	0	0	0	0
0.0863	0.1086	0.1546	0.1997	0	0	0
0.0602	0.0863	0.1086	0.1546	0.1997	0	0
0.0610	0.0602	0.0863	0.1086	0.1546	0.1997	0

It is very close to the true $\boldsymbol{H}_f$ and consistent with the features of the true $\boldsymbol{H}_f$ matrix: a lower triangle matrix, the same value for the elements on each lower diagonal, and values on these diagonals are the (estimated) impulse weights. In this approach, knowledge that the instantaneous effect of u_k on y_k is zeros (i.e., $\boldsymbol{D}$=0) can be easily incorporated, but it is harder for other approaches to use this knowledge.

By regressing U_p out of U_f and taking the remainder of U_f as instrumental variables (IVs) as shown in (5.3.5), the IV approach gives the following estimated H_f matrix as the result of (5.3.6):

-0.0014	0.0136	-0.0003	0.0415	-0.0096	-0.0397	0.0375
0.1923	0.0103	0.0031	0.0198	0.0221	-0.0096	0.0018
0.1636	0.1931	0.0107	0.0161	0.0075	0.0221	-0.0039
0.1129	0.1637	0.1943	0.0248	0.0027	0.0075	0.0224
0.0784	0.1195	0.1574	0.2004	0.0188	0.0027	0.0309
0.0707	0.0868	0.1112	0.1620	0.1959	0.0188	0.0338
0.0559	0.0792	0.0779	0.1112	0.1618	0.1959	0.0516

In general, this result has minuscule deviations from the true values, and it is close to the result from regressing Y_f against $[Y_p; U_p; U_f]$.

The means of elements on lower diagonals of the estimated H_f can be taken as the estimated impulse weights. The total absolute error on these estimated impulse weights shows roughly how close the estimated H_f is to the true H_f. The estimated impulse weights and the total absolute error for each approach discussed are listed in Table 5.2. The results by regressing Y_f directly onto U_f are clearly the farthest from the true values because of the bias due to the strong auto-correlation in the input PRBS signal. The results of the estimated H_f from all other methods are similar to each other and very close to the true value.

Table 5.2 The impulse weights in estimated H_f

Method	True	Y_f/U_f	$Y_f/[P_{IO}; U_f]$	ARX	IV
w_0	0	0.1003	0.0159	0	0.0191
w_1	0.2	0.2716	0.1951	0.1997	0.1953
w_2	0.16	0.2203	0.1585	0.1546	0.1617
w_3	0.128	0.1770	0.1035	0.1086	0.1137
w_4	0.1024	0.1626	0.0728	0.0863	0.0811
w_5	0.0819	0.1613	0.0668	0.0602	0.0750
w_6	0.0655	0.2135	0.0510	0.0610	0.0559
Sum of Abs. Err.	0	0.5688	0.1061	0.0675	0.0778

Note: impulse weights $w_0, w_1, \ldots, w_6$ are the means of the lower diagonals of estimated H_f

Based on the estimated H_f matrix, the predictable subspace can be estimated as the result of Y_f-$H_{f\_e}U_f$. This estimate includes the effects of the future stochastic signals. Projecting this estimated predictable subspace onto the past data removes this part of the future noise and usually leads to a better estimate of the predictable subspace. In some cases, such as using PCA for state estimation, this projection is necessary for a better result. Yet LVMs based on both X and Y spaces, such as RRA and CCA, do not need this projection since the prediction and the state estimation are completed simultaneously in one step.

5.6.2 Estimation of State Variables

Based on the estimated predictable subspace, different approaches are available for the estimation of the state variables as shown in Section 5.4. In this example, MOESP (elementary algorithm), N4SID and CVA use PCA or CCA to estimate the states. CCA is also used to estimate process states based on the estimated predictable subspace by the IV method. PLS and RRA are also applied to estimate states based on the estimated predictable subspace by the ARX method. Here all these methods give an obvious cut-off on the number of large singular values, canonical correlation coefficients (CCCs) or the predictabilities after the second latent variable. This indicates a clear determination of the system order to be 2, for both the combined deterministic and stochastic states. AIC also indicates the system order to be 2.

There are many different ways to show how close the estimated states are to the true states (more accurately, how the estimated state space is to the true state space). One good measure is to compute the CCCs between the estimated states and the true states. The results are listed in Table 5.3. The closer the CCCs are to 1.0, the more the estimated states are consistent with the true state space. MOESP using the direct regression Y_f/U_f clearly gives poor results. The two estimated states by PLS are close to the true states, but are relatively degraded compared to those from CCA or RRA based on the same estimated predictable subspace by ARX method. The estimated states from N4SID, CVA, SIM-IV-CVA and SIM-ARX-RRA are very close to the true states. Similar conclusions are also indicated by the

squared multiple correlation R^2, which shows how much of the total sum squares of each true state can be explained by the estimated states. R^2 for both states (scaled to same variance) is also listed in Table 5.3 for comparison of these SIM algorithms.

Table 5.3 Comparison of the estimated states and the true states

Method	MOESP	N4SID	CVA	SIM- IV-CCA	SIM-ARX-PLS	SIM-ARX-RRA
Predictable Subspace Esti.	Y_f/U_f	$Y_f/[P_{IO};U_f]$	ARX	IV	ARX	ARX
State Variable Estimation	PCA	PCA	CCA	CCA	PLS	RRA
1st Canonical Correlation	0.8680	0.9993	0.9997	0.9995	0.9500	0.9993
2nd Canonical Correlation	0.2599	0.9623	0.9613	0.9600	0.9122	0.9618
R^2 for 1st state	0.1534	0.9625	0.9641	0.9631	0.8947	0.9616
R^2 for 2nd state	0.6528	0.9599	0.9569	0.9550	0.8386	0.9596
R^2 for both states	0.4031	0.9612	0.9605	0.9590	0.8667	0.9606

5.6.3 Final Identified Models

If the estimated states are used to fit to the state-space model by LS regression, each SIM algorithm can obtain an identified model in the state-space model form. The impulse responses calculated from some of these fitted models are shown in Figure 5.2, and their errors are show in Figures 5.3 and 5.4. MOESP elementary algorithm gives a poor result for this example. The result from SIM-ARX-PLS has a large error but can be improved to match the others by using more LVs. The results from other SIMs are very close to the true model. The impulse response from the fitted ARX model shows somewhat large erratic errors (refer to Figure 5.4). All SIM algorithms give smooth responses by fitting the LVs to the state equation.

This simulation example is also applied for another SIM algorithm based on the regressing out method for the predictable subspace estimation (equivalent to $Y_f/[P_{IO}; U_f]$) and CCA for the state estimation, i.e., SIM-$Y_f/[P_{IO}; U_f]$-CCA. It can be deemed a combination of N4SID and CVA. The result is shown in Figure 5.5, and it indicates that this algorithm is valid and gives similar results as those from CVA and N4SID.

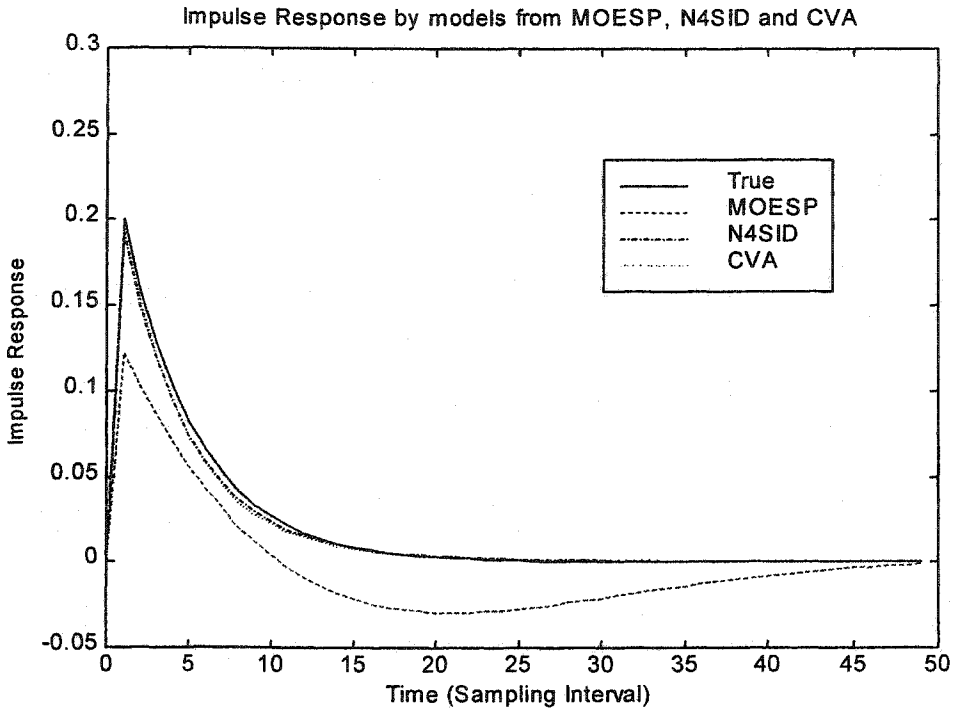


Figure 5.2 Impulse response from SIMs

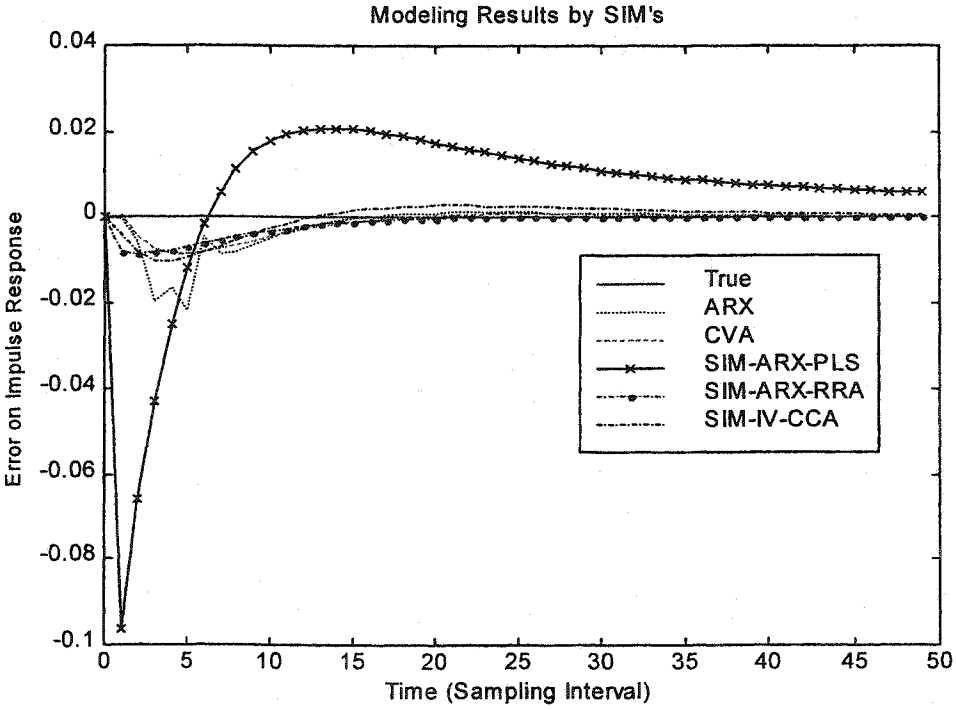


Figure 5.3 Errors on the impulse responses

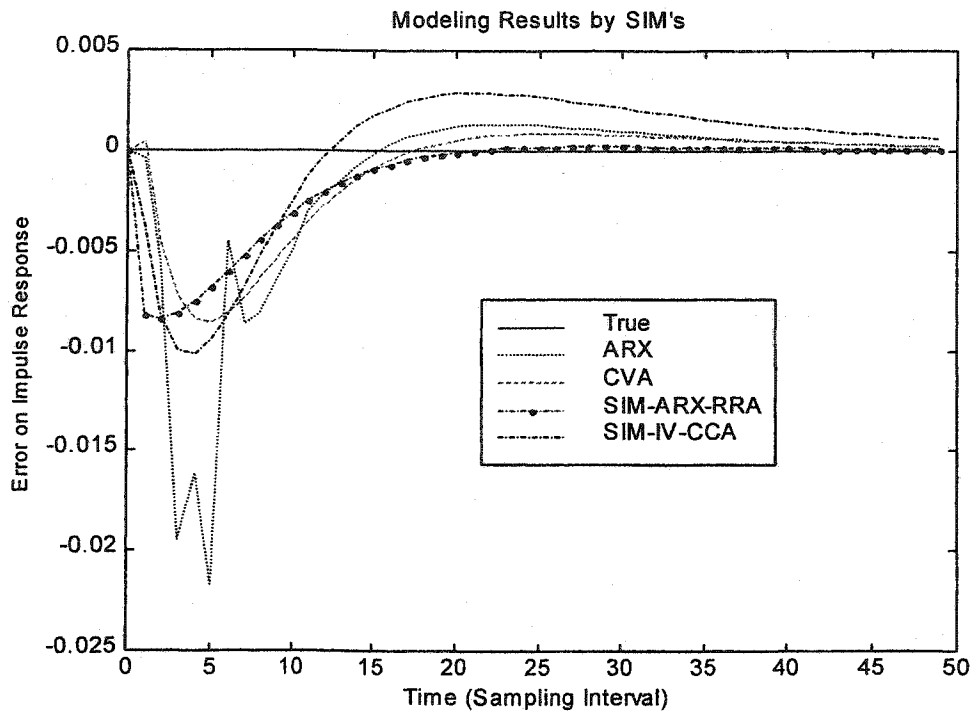


Figure 5.4 Errors on the impulse responses (without SIM-ARX-PLS)

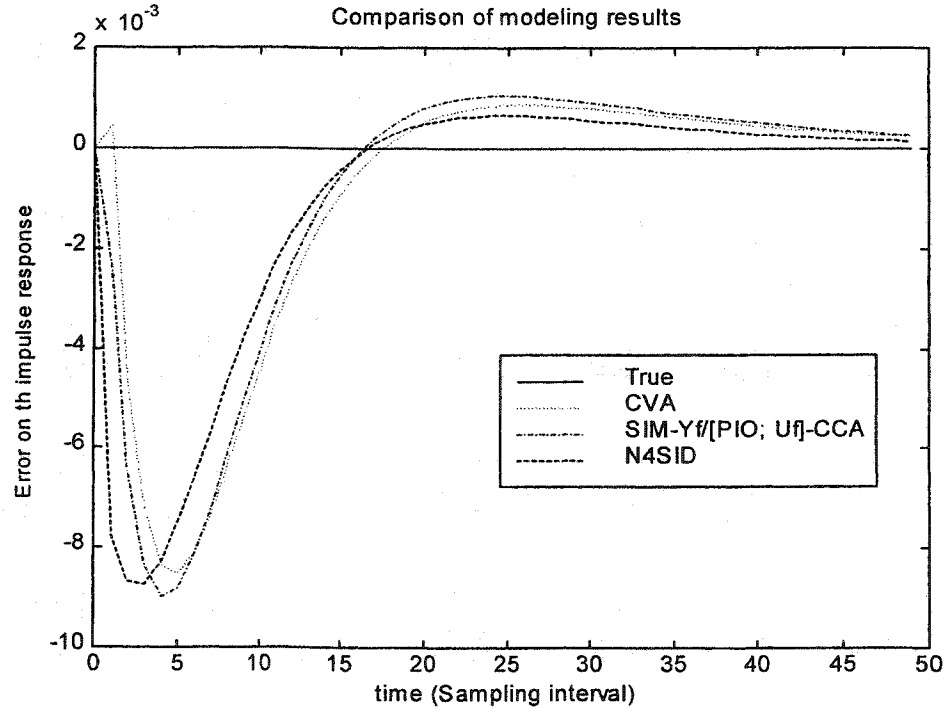


Figure 5.5 Errors on the impulse responses from different SIM algorithms

5.7 Conclusions

Although SIMs (MOESP, N4SID and CVA) are quite different in their concepts and algorithms, they follow a statistical framework set up in this chapter: (1) use of a linear regression method to estimate $\boldsymbol{H}_f$ and the predictable subspace; (2) use of a latent variable method for estimation of a minimal set of the state variables; and (3) then fitting to the state-space model. The major differences among these subspace identification methods lie in the first two steps, and the third step is common for all the SIMs.

The predictable subspace is a key component and the starting point in SIMs. It can be estimated by removing the future input effects from the future outputs based on estimated $\boldsymbol{H}_f$ matrix. There are several approaches to estimate the $\boldsymbol{H}_f$ matrix, such as regressing $\boldsymbol{Y}_f$ against $\boldsymbol{U}_f$ (if no auto-correlation in the input signals), regressing $\boldsymbol{Y}_f$ against $[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{U}_f]$, constructing $\boldsymbol{H}_f$ based on the impulse weights from fitted ARX model and regressing out $\boldsymbol{P}_{IO}$. A new approach based on the IV method is proposed in this chapter. All these approaches are based on linear regression and are easy to understand from the viewpoint of statistics.

Process state variables are estimated by LVMs based on the estimated predictable subspace and the past data. It is the second step in the framework. PCA, RRA, and CCA can be directly employed to estimate a minimum set of state variables. PLS is a feasible method but not as effective as other LVMs for partially modeling the past data space. The significant LVs are taken as the estimated process states. All these methods find out a lower dimensional subspace (estimated state space) in the past data with the best predictability of the future outputs (the best predictability may appear in different senses in different LVMs).

This framework reveals a whole variety of new SIM algorithms just by combinations of the approaches in the first two steps. Simulation study shows that these new SIM algorithms have the similar performance as the existing SIMs.

The third step in this framework is to fit the estimated states to the state-space model. The state-space model parameterized in full matrix form is generally employed in this

procedure. The resultant state-space model is much more parsimonious than the LVR model. Several computational issues of this procedure are also discussed, such as obtaining the estimated X_{k+1}, the number of effective data points and the effects of the estimation errors in the states.

This chapter compares the existing methods by examining them in a common framework involving linear regression and latent variable estimation. The framework reveals clearly the nature and fundamental ideas of SIMs, and therefore SIM algorithms become easier to understand. In this framework, the similarities and differences among SIMs become apparent. SIM algorithms based on combinations of the approaches for the first two steps of the framework are feasible and give similar results as CVA and N4SID.

6 SIMs for Closed-loop Data

6.1 Introduction and Motivations

The purpose of closed-loop system identification is to estimate dynamic models based on data collected from processes with feedback controls. Figure 6.1 shows a schematic diagram of a general closed-loop system, where $G(z)$ is the model of the dynamic process, N_k stands for unmeasured disturbances to the process, and $C(z)$ is the feedback controller to reduce the effect of disturbances on the outputs y_k. In system identification, well-designed independent external signals as dither d_k or setpoint s_k are usually added to the process for persistent perturbation of the system in order to get informative data (y_k and u_k) for identifying $G(z)$. In some cases, the perturbation signals are also available and can be used for system identification. Effects of perturbations s_k are equivalent to perturbations of $C(z)s_k$ at d_k, therefore using only d_k in analysis does not lose any generality.

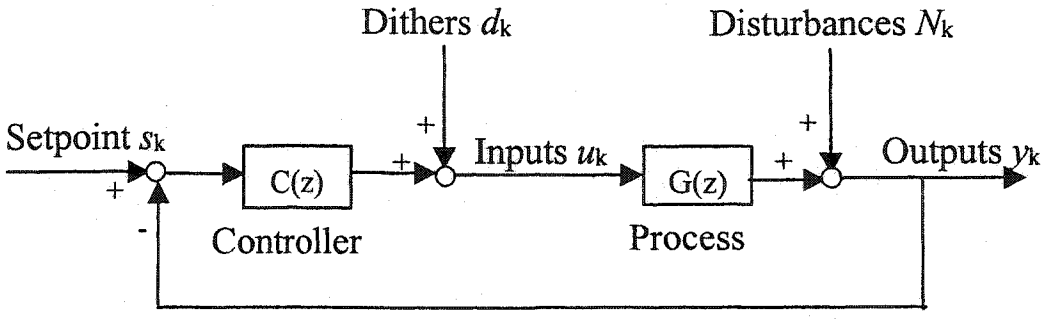


Figure 6.1 A schematic diagram of a general closed-loop system

System identification based on closed-loop data is very important for practical applications. Many systems require closed-loop identification. It is hard to perform open loop experiments for unstable systems. For some systems with inherent feedbacks (e.g., social, economic, or biological systems), closed-loop experiments are the only choice. For

some systems, open loop experiments will cause safety or environmental hazards, and therefore closed-loop experiments become mandatory. In other situations, open loop experiments are possible; however, this will cause products to be out of specifications and result in a profit loss. In all these cases, closed-loop experimentation and closed-loop identification is the preferred approach for estimating process dynamic modeling.

In model identification for controller design, it has been shown that the closed–loop data gives a model more suitable for robust control than open loop data if the same output variance is allowed for both cases (Esmaili et al., 2000). In the closed-loop case, the input signal can have a larger perturbation than in the open loop case when the output variance is constant. Data collected under feedback resemble the normal operation better (in terms of frequency components). As a result, closed-loop data is more suitable for the purpose of controller design. This has prompted studies on control relevant identification in recent years (e.g., Hjalmarsson et al., 1996).

Although closed-loop system identification is important to practical applications and may give better models for controller design, the presence of feedback in the system makes the identification more complicated. In a closed-loop system (See Figure 6.1), the feedback controller makes the process inputs correlate with the process outputs and the disturbances. This is fundamentally different from the open loop case where the inputs are well-designed signals independent of disturbances. The correlation raises the identifiability problem and applicability problem of system identification methods to closed-loop data.

Identifiability is an important issue for dynamic modeling with closed-loop data. A system is identifiable if the corresponding parameter estimates are consistent. There is considerable literature discussing this issue (e.g., Söderström, 1989). Identifiability for closed-loop data is guaranteed under the following conditions: (i) an adequate model structure is used that includes the true process, (ii) the system is persistently excited by adding adequate dither signals or shifting between a sufficient number of controllers, and (iii) there is at least one delay step in the closed-loop. In this chapter, all the discussion is under these conditions, and therefore identifiability is not considered.

Extensive research has resulted in clear conclusions on the traditional methods for closed-loop data (e.g., a survey by Forssell and Ljung, 1999). However, SIMs for closed-loop data have not been well studied, and some disputes still exist over the applicability of SIMs for closed-loop data. Jha and Georgakis (1996) applied N4SID method for closed-loop data. Ljung and McKelvey (1996) pointed out that N4SID gives biased results for closed-loop data. They implied that other SIMs had the same problem if used for closed-loop data. On the other hand, Larimore (1997) claimed that CVA could be used for closed-loop data, but Van Overschee and De Moor (1996) disagreed with this statement. Lakshminarayanan et al. applied CVA to an industrial process for closed-loop identification (2001). Verhaegen (1993) indicated that MOESP only applied to open loop identification problems, and he recast the closed-loop identification to an open loop problem, and applied MOESP to this open loop problem.

The objectives of this chapter are to analyze subspace identification methods (SIMs) for closed-loop identification, clarify their applicability for closed-loop data, and explore possible new SIM algorithms for closed-loop data. N4SID and CVA algorithms will be investigated respectively in the next two sections. Then some simulations will be used to illustrate the main points.

In general, there are three different approaches to apply system identification methods to closed-loop data: the direct, indirect and joint input-output approaches. The direct approach treats the closed-loop data as open loop data and fits the input-output model directly to the data. Indirect approaches get the model of the closed-loop system first and then extract the process model out of the closed-loop model with the prior knowledge of the controller model. The joint input-output method approach builds a joint multivariable time series model for both the input and output variables, and then back calculates both the process model and the controller model. Only the direct approach uses identification methods directly for the closed-loop data, and other approaches essentially transform the closed-loop identification into an open loop identification problem and then open loop identification methods are used. Under asymptotic conditions, all these approaches have been shown to perform the same (Gevers, Ljung and Van den Hof,

1996). However, for a finite data set, direct method has been shown to give results that are better than or equal to the other approaches (Esmaili et al., 2000). Therefore, this chapter will only focus on SIMs (N4SID and CVA) in the direct approach.

6.2 N4SID for Closed-loop Data

Due to their linear model structure and ease of computation, FIR and ARX models are often used in open loop or closed-loop identification. They are also employed in SIM algorithms. In this section, after a brief review of using FIR and ARX models for closed-loop identification, the original N4SID algorithm is analyzed in terms of input-output model and multi-step state-space model respectively, and then existing N4SID algorithms for closed-loop data are discussed and three new N4SID algorithms for closed-loop identification are proposed.

6.2.1 FIR and ARX Models for Closed-loop Data

In closed-loop case, the input and output variables have the following basic relationships (refer to Figure 6.1):

$$y_k = G(z)u_k + N_k \tag{6.2.1}$$

$$u_k = -C(z)y_k + d_k \tag{6.2.2}$$

With notation $\boldsymbol{S}(z)=(\boldsymbol{I}+\boldsymbol{C}(z)\boldsymbol{G}(z))^{-1}$ for the sensitivity function, the above relationships can be rewritten as the following relationships in terms of the external signals (dithers and the noise):

$$u_k=\boldsymbol{S}(z)d_k-\boldsymbol{S}(z)\boldsymbol{C}(z)N_k \tag{6.2.3}$$

$$y_k=\boldsymbol{G}(z)\boldsymbol{S}(z)d_k+(\boldsymbol{I}-\boldsymbol{G}(z)\boldsymbol{S}(z)\boldsymbol{C}(z))N_k \tag{6.2.4}$$

Equation (6.2.3) clearly shows that inputs are always correlated with the process disturbances through feedback control, while in the open loop case, inputs are usually

well-designed signals and are independent of process disturbances. This is a fundamental difference between open loop data and closed-loop data, and this difference causes potential problems in modeling the dynamic process model with closed-loop data.

If FIR model is used to represent the process model, it has the following form (with prior knowledge of no instantaneous action):

$$y_k = \sum_{i=1}^{p} g_i u(k-i) + N_k \tag{6.2.5}$$

Where g_i is the i-th step impulse weight and p is assumed sufficiently large. If prediction error methods (PEM) are used for closed-loop identification, it has been shown that the noise must also be adequately modeled and identified simultaneously with the process model; otherwise, the identified process model will be biased (MacGregor and Fogal, 1995).

ARX models are also used for dynamic modeling. An ARX model can be represented in time domain as:

$$y_k = \sum_{i=1}^{p} a_i y_{k-i} + \sum_{j=1}^{p} b_j u_{k-j} + e_k \tag{6.2.6}$$

The ARX model structure provides models for both the dynamic process and the noise, and these models are identified simultaneously. If the true process can be adequately represented by an ARX model and the process is excited by dither signals (d_k), then unbiased coefficients in the above model can be estimated by LS regression since the stochastic signal e_k in the above equation is un-correlated to the past inputs and outputs (regessors). Here the correct prior knowledge of the process model structural is crucial for this parametric model in low order ARX form. If the true process is a general process with Box-Jenkins model (not an ARX model form) and an ARX model is used to fit the closed-loop data, the resultant model is a biased approximation of the true process model as in the open loop case (see Section 3.2). The fitted ARX model is asymptotically unbiased if the model order is infinitely high. The correlation between the inputs and outputs via feedback showing in (6.2.3) increases the correlation between the regressors and thus increases the variance of the parameters given the same input excitation. In

practice, a high-order ARX model is usually employed, and the resultant model is a close approximation of the true process model.

If the dither signals $\boldsymbol{d}_k$ are available, they can be used as instrumental variables (past dither data $\boldsymbol{D}_p$, i.e., past p steps of dithers), and unbiased results for FIR or ARX models can be obtained by IVM with closed-loop data because the dither signals are correlated with the input and output variables but are un-correlated with the process disturbances.

6.2.2 Analysis of N4SID for Closed-loop data

Analysis of N4SID in terms of input/output model

From the viewpoint of Infinite Impulse Response (IIR), Ljung and McKelvey (1996) summarized SIMs (focusing on N4SID) and investigated the applicability of these methods for closed-loop identification. They pointed out that the correlation between future inputs and the noise made N4SID biased when applied to closed-loop data. They also suggested an algorithm to circumvent the correlation problem by recursively using a fitted ARX model to predict the multiple-step future outputs (for detailed procedures, refer to N4SID_ARX algorithm in Section 3.2). The analysis in that paper was originally based on an IIR model and changed to an ARX model later. Here N4SID is analyzed directly in ARX model form for easier understanding; however, the essential idea remains unchanged.

The first step of the original N4SID algorithm (Van Overschee and De Moor, 1994) is to perform a LS regression of $\boldsymbol{Y}_f$ onto [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{U}_f$]:

$$Z_f = Y_f / \begin{bmatrix} Y_p \\ U_p \\ U_f \end{bmatrix} = L_1 Y_p + L_2 U_p + L_3 U_f \qquad (3.2.1)$$

This projection is essential to build models for multiple step ahead predictions. It consists of a series of LS regressions of future outputs at $k+j$ ($0 \le j < f$) time points (y_{k+j}) against [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{U}_f$]:

$$y_{k+j} = \sum_{i=1}^{p} L_{j1} y_{k-i} + \sum_{m=1}^{p} L_{j2} u_{k-m} + \sum_{n=0}^{f-1} L_{j3} u_{k+n} + \varepsilon_{k+j} \tag{6.2.7}$$

The value for the left-hand side of (6.2.7) has the following general relationship with the past inputs, outputs, future inputs and the stochastic signals:

$$y_{k+j} = \sum_{i=1}^{r} a_{ji} y_{k-i} + \sum_{m=1}^{r} b_{jm} u_{k-m} + \sum_{n=0}^{j} c_{jn} u_{k+n} + \sum_{s=0}^{j} d_{js} e_{k+s} + \sum_{l=1}^{q} f_{jl} e_{k-l} \tag{6.2.8}$$

As seen in (6.2.3), in the closed-loop case, inputs at one time point are correlated to the stochastic signals before that time point, i.e., u_{k+n} is correlated to e_{k+n}, e_{k+n-1}, e_{k+n-2}, ... (for $n \leq j$), and u_{k+n} is correlated to e_{k+j}, e_{k+j-1}, e_{k+j-2}, ... (for $j<n<f$). Therefore, the future inputs U_f (3rd term of (6.2.7)) in the LS regression (3.2.1) are correlated to the noise term (4th term in (6.2.8)), and therefore the LS regression results from (6.2.7) will be biased. As a result, the estimated predictable subspace $L_1 Y_p + L_2 U_p$ in N4SID is a biased estimate, and the estimated states from PCA on this estimated predictable subspace are also biased. This bias comes from the correlation between the future inputs and the noise. It does not vanish with the increase of the number of data points or the number of lag steps (p). In fact, equation (6.2.8) indicates that the correlation becomes stronger if a longer future horizon is used or the SNR decreases.

Analysis of N4SID in terms of multi-step state-space model

Similar to the open loop case, the multi-step state-space model can also be used to show the relationships in closed-loop data, and therefore provides a basis for analyzing SIM algorithms for closed-loop data. In the closed-loop case, the dynamic process has the same relationships between future/past input and output variables as in the open loop case, except that the inputs are determined by the feedback controller and the dither signals. The controller has the following state-space model (superscript c stands for the controller):

$$x_{k+1}^c = A^c x_k^c + B^c y_k \tag{6.2.9}$$

$$u_k = C^c x_k^c + D^c y_k + d_k \tag{6.2.10}$$

x_k^c is the controller state vector with dimension of n^c. The process outputs are the inputs to the controller, and the controller outputs are inputs to the process (dither d_k can be deemed as measured or unmeasured disturbances).

Through a similar conduction as in Section 3.1, it is easy to express the current states of the controller in the following multi-step state-space model ($\boldsymbol{D}_p$ is the matrix for the dither over the past horizon):

$$\begin{aligned} X_k^c &= \left(A^c\right)^p \Gamma_p^{c+} U_p + \left(\Omega_p^c - \left(A^c\right)^p \Gamma_p^{c+} H_p^c\right) Y_p - \left(A^c\right)^p \Gamma_p^{c+} D_p \\ &= L^c P_{IO} + L^d D_p \end{aligned} \tag{6.2.11}$$

This relationship indicates that the controller states are also a linear combination of the past data $\boldsymbol{P}_{IO}=[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ and past dither $\boldsymbol{D}_p$. The future inputs (controller outputs) have the following relationship with the current controller states, the future process outputs and the future dither ($\boldsymbol{D}_f$ is for the dither in the future horizon):

$$U_f = \Gamma_f^c X_k^c + H_f^c Y_f + D_f \tag{6.2.12}$$

This shows that the future inputs are correlated to the future outputs, which includes the future noise $\boldsymbol{H}_{s,f}\boldsymbol{W}_f+\boldsymbol{V}_f$ (as shown in (3.1.7)):

$$Y_f = \Gamma_f X_k + H_f U_f + H_{s,f} W_f + V_f \tag{3.1.7}$$

that is, $\boldsymbol{U}_f$ is correlated to the future noise $\boldsymbol{H}_{s,f}\boldsymbol{W}_f+\boldsymbol{V}_f$.

Consider the basis for the first step of LS regression in N4SID method:

$$\begin{aligned} Y_f = {} & \Gamma_f A^p \Gamma_p^{+} Y_p + \Gamma_f (\Omega_p - A^p \Gamma_p^{+} H_p) U_p + H_f U_f \\ & + \Gamma_f (\Omega_{s,p} - A^p \Gamma_p^{+} H_{s,p}) W_p - \Gamma_f A^p \Gamma_p^{+} V_p + H_{s,f} W_f + V_f \end{aligned} \tag{3.1.9}$$

Because of the correlation between future inputs $\boldsymbol{U}_f$ and the future noise $\boldsymbol{H}_{s,f}\boldsymbol{W}_f+\boldsymbol{V}_f$ as shown in (6.2.12), LS results of regressing $\boldsymbol{Y}_f$ against $[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{U}_f]$ in the first step of N4SID turn out to be biased. Refer to Appendix A3.1 for the detailed regression results. This confirms the conclusion drawn from Ljung's analysis of N4SID for closed-loop data and the previous analysis in terms of the input/output model.

6.2.3 N4SID algorithms for Closed-loop Data

This section will summarize the existing N4SID-related algorithms for closed-loop data and propose three new N4SID algorithms based on the above analysis and on the framework for SIMs presented in Chapter 5.

Multiple-step future predictions based on fitted ARX model (N4SID-ARX)

Ljung and McKelvey (1996) proposed an algorithm to circumvent the correlation problem for N4SID with closed-loop data: fit a high-order ARX model with the closed-loop data, assuming no instantaneous action in the process, then recursively use this ARX model to predict multiple steps of future outputs with setting the future inputs to null, and then perform PCA on these predictions for estimated states.

This provides an unbiased N4SID algorithm for closed-loop data. As shown in Section 6.2.1, the fitted high-order ARX model is asymptotically unbiased. In prediction of the multi-step future outputs, setting the future inputs to null serves to eliminate the effects of future inputs from the future outputs, and this multiple-step future output prediction is essentially the estimated predictable subspace. As shown in Section 3.2, this algorithm is also applicable to open loop data.

A similar N4SID algorithm is also available by estimating the multi-step future output predictions based on a valid FIR model, which can be obtained from closed-loop data by an IVM as mentioned in Section 6.2.1. Different from the algorithm based on ARX model, this algorithm can only estimate the deterministic states. Therefore, it can only give the process dynamic model, and the noise model needs to be estimated separately thereafter.

Modification of the future inputs based on controller model

Van Overschee and De Moor (1996) admitted the analysis of N4SID for closed-loop data by Ljung and McKelvey and pointed out that the conclusion was also applicable to MOESP and CVA. They proposed another algorithm via modification of the future

inputs based on the prior knowledge of the controller model. The impulse weights of the controller were used to construct the H_f^c matrix; then the future inputs were modified as $M_f=U_f-H_f^c Y_f$ (see (6.2.14)), that is, removing away the component in U_f that correlated to Y_f (i.e., correlated with the future noise signals), then performed oblique regression of Y_f onto $[Y_p; U_p]$ along M_f. The rest of the steps were similar to those in original N4SID procedures.

By modifying future input data, this algorithm avoids the correlation between the future inputs and the future noise signals. This algorithm does not require the dither signals but requires prior knowledge of the controller model, and can be deemed as applying N4SID in an indirect approach without identifying the closed-loop model. The prior knowledge of the controller model is a very strong requirement, one that is not available in many practical applications. Even if the controller equation is known, this algorithm assumes the controller is implemented perfectly and does not allow for input saturation, valve sticking, measurement errors on u_k, etc.

N4SID in a joint input-output approach

Jha and Georgakis (1996) used N4SID for closed-loop data in a joint input-output modeling approach. The d_k is treated as the inputs to the joint system (see (6.2.3) and (6.2.4)), and both u_k and y_k are treated as the outputs of the joint system. N4SID was applied to the joint system. The states estimated by N4SID include both the process states and the controller states, and the system from d_k to u_k was inverted to get the process model and the controller model (corresponding to the inversion of transfer function in SISO case). This approach involves the inversion of MIMO system, which is realized in the state-space model.

This algorithm does not conflict with Ljung's analysis or the analysis in Section 6.2.2. In this approach, N4SID is applied to the system from dither signals (inputs) to the process input and output variables (outputs), which is an open loop identification problem. The direct resultant model from N4SID is only the closed-loop system model, not the process model itself. In this approach, N4SID guarantees the same denominator

for both transfer functions from $\boldsymbol{d}_k$ to $\boldsymbol{y}_k$ and from $\boldsymbol{d}_k$ to $\boldsymbol{u}_k$. This leads to denominator-numerator cancellations in extracting the process model, and therefore a relatively lower order model for the final result. This is an advantage of SIMs (including N4SID) for MIMO system compared to traditional methods (e.g., PEM) in joint input-output identification approach, where a series of MISO identifications usually leads to different denominators for systems from $\boldsymbol{d}_k$ to $\boldsymbol{y}_k$ and from $\boldsymbol{d}_k$ to $\boldsymbol{u}_k$, and gives an unnecessary high-order model for the dynamic process.

N4SID-RRA based on FIR or ARX model (N4SID-RRA)

Based on the analysis in Section 6.2.1 and the framework for SIMs in Chapter 5, a generic N4SID algorithm for closed-loop data is proposed (an extension of the N4SID_Hf algorithm in Section 3.2): fit a high-order ARX model by LS regression or IVM (or fit an FIR model by IVM); obtain the estimated impulse weights from the fitted ARX or FIR model and construct the estimated coefficient matrix $\boldsymbol{H}_f$, then estimate the predictable subspace by removing the future input effects $\boldsymbol{H}_f\boldsymbol{U}_f$ away from the future outputs $\boldsymbol{Y}_f$; take the dominant LVs from RRA on $\boldsymbol{P}_{IO}=[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ and $\boldsymbol{Y}_f$-$\boldsymbol{H}_f\boldsymbol{U}_f$ as estimated estates and fit to the state-space model for the process model. Here $\boldsymbol{U}_f$ is no longer a regressor, and the correlation problem with the original N4SID algorithm is avoided.

In fact, this same algorithm is applicable for both open loop data and closed-loop data. For closed-loop identification, the only precaution is to use an ARX model of sufficiently high order to ensure unbiased estimation of the predictable subspace.

N4SID algorithm based on IV method (N4SID-IV)

As shown in Section 6.2.2, the problem with N4SID for closed-loop data originates from the bias result in the first step of LS regression (the oblique projection step). This bias results from the correlation between the future inputs and the future noise signals. Using an IV method instead of LS regression is a promising way to avoid the correlation problem.

The key issue in any IVM is to choose desirable instrumental variables (IVs). There are two basic requirements for the instrumental variables (e.g., Ljung, 1999): no correlation with the noise and the covariance matrix with regressors is nonsingular. The choice of IVs depends on the regressors. Looking at the basis for the first step of N4ISD:

$$\begin{aligned} Y_f &= \Gamma_f A^p \Gamma_p^+ Y_p + \Gamma_f (\Omega_p - A^p \Gamma_p^+ H_p) U_p + H_f U_f \\ &\quad + \Gamma_f (\Omega_{s,p} - A^p \Gamma_p^+ H_{s,p}) W_p - \Gamma_f A^p \Gamma_p^+ V_p + H_{s,f} W_f + V_f \end{aligned} \tag{3.1.9}$$

If the purpose is to estimate the $\boldsymbol{H}_f$ matrix (e.g., to be used in constructing $\boldsymbol{Y}_f$-$\boldsymbol{H}_f\boldsymbol{U}_f$), $\boldsymbol{U}_f$ is the regressor and the noise includes terms of $\boldsymbol{Y}_p$, $\boldsymbol{U}_p$ and stochastic signals. If the purpose is to estimate all the coefficient matrices, [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{U}_f$] is the regressor and the noise only includes the stochastic terms.

In a well-designed experiment for closed-loop identification, the dither signals are independent external variables added to the system, and $\boldsymbol{D}_f$ is un-correlated to the stochastic terms. Furthermore, if the dither signals are not auto-correlated, such as white noise or a RBS (Random Binary Sequence), then $\boldsymbol{D}_f$ is not correlated to the $\boldsymbol{Y}_p$ and $\boldsymbol{U}_p$ terms in (3.1.9). Since $\boldsymbol{D}_f$ is a part of $\boldsymbol{U}_f$, then their covariance matrix is also nonsingular (dither signals are persistently exciting of high order). Therefore, $\boldsymbol{D}_f$ can be applied to (3.1.9) as IVs to estimate $\boldsymbol{H}_f$ by IVM:

$$\hat{H}_f = Y_f D_f^T \left(U_f D_f^T \right)^{-1} \tag{6.2.13}$$

This estimated $\boldsymbol{H}_f$ can be used to estimate the predictable subspace, and RRA can be employed to estimate the process states. If the dither signals are auto-correlated, such as a PRBS with switching time period T_s>1, $\boldsymbol{D}_f$ is correlated to $\boldsymbol{D}_p$, which has significant correlation with $\boldsymbol{Y}_p$ and $\boldsymbol{U}_p$, and therefore (6.2.13) cannot be used directly for this situation.

For auto-correlated dither signals, a component of the future dither signals can be used as IVs: this component can be obtained by projecting $\boldsymbol{D}_f$ onto the orthogonal space of $\boldsymbol{D}_p$, i.e., $\boldsymbol{D}_{f\_m}$=$\boldsymbol{D}_f$-$\boldsymbol{D}_f$/$\boldsymbol{D}_p$. As long as the number of past lag steps is greater than the switching period (T_s) of the PRBS (or the significant auto-correlation period for other

types of signals), this IV variable $\boldsymbol{D}_{f\_m}$ will be orthogonal to $\boldsymbol{D}_p$, and therefore un-correlated with the past inputs and outputs. Using $\boldsymbol{D}_{f\_m}$ as the IVs, $\boldsymbol{H}_f$ can be estimated as:

$$\hat{H}_f = Y_f D_{f\_m}^T \left(U_f D_{f\_m}^T\right)^{-1} \tag{6.2.14}$$

For estimation of all the coefficient matrices in (3.1.9), [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{U}_f$] is the regressor, and the noise for this regression comes only from the terms of stochastic signals. Both [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{D}_f$] and [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{D}_{f\_m}$] are uncorrelated to the future stochastic noise terms $\boldsymbol{H}_{s,f}\boldsymbol{W}_f+\boldsymbol{V}_f$. Either of these two data sets can be used as IVs to estimate the coefficient matrices by IVM. This avoids the correlation between $\boldsymbol{U}_f$ and $\boldsymbol{H}_{s,f}\boldsymbol{W}_f+\boldsymbol{V}_f$ in closed-loop data. Using IVs=[$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{D}_f$] or IVs=[$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{D}_{f\_m}$], the result by IVM is:

$$\begin{bmatrix} L_1 & L_2 & L_3 \end{bmatrix} = Y_f \cdot IVs \left\{ \begin{bmatrix} Y_p & U_p & U_f \end{bmatrix} \cdot IVs^T \right\}^{-1} \tag{6.2.15}$$

Here $\boldsymbol{L}_1$, $\boldsymbol{L}_2$ and $\boldsymbol{L}_3$ are coefficient matrices for $\boldsymbol{Y}_p$, $\boldsymbol{U}_p$ and $\boldsymbol{U}_f$ respectively, as in (3.2.1). The resultant L_3 is an estimate of $\boldsymbol{H}_f$, and it can be used to estimate the predictable subspace as $\boldsymbol{Y}_f$-$\boldsymbol{L}_3\boldsymbol{U}_f$. Alternatively, $\boldsymbol{L}_1\boldsymbol{Y}_p+\boldsymbol{L}_2\boldsymbol{U}_p$ is another estimate of the predictable subspace as in the original N4SID algorithm, but without the future stochastic components.

Compared to using an IVM for the estimation of $\boldsymbol{H}_f$ only, the IVM for estimation of all the coefficient matrices gives a better result in the sense of a smaller variance of the estimated predictable subspace. This is due to a much smaller magnitude of noise in the latter method.

N4SID algorithm based on estimated joint states (N4SID-Joint)

Based on the conclusion that the bias of N4SID comes from the correlation between future inputs and future noise, the basic idea for a modified N4SID closed-loop algorithm is to use the future dither data $\boldsymbol{D}_f$ instead of the future input data $\boldsymbol{U}_f$ to show the effects of $\boldsymbol{U}_f$.

Based on the multi-step state-space model for the controller, the future inputs $\boldsymbol{U}_f$ have the following relationship with the current controller states, the future outputs and the future dither signals:

$$U_f = \Gamma_f^c X_k^c + H_f^c Y_f + D_f \tag{6.2.14}$$

Substitute the above relationship into the multi-step state-space model for the future outputs:

$$Y_f = \Gamma_f X_k + H_f U_f + H_{s,f} W_f + V_f \tag{3.1.7}$$

and the result is:

$$Y_f = \Gamma_f X_k + H_f \Gamma_f^c X_k^c + H_f H_f^c Y_f + H_f D_f + H_{s,f} W_f + V_f \tag{6.2.15}$$

The first two terms on the right-hand side of the above equation show the predictable subspace of the process and the predictable subspace of the controller, respectively. Linear combination of these two predictable subspaces is a joint predictable subspace, which has as a basis the joint process-controller states $[X_k; X_k^c]$, with a rank of $n+n^c$ (total system order of process and controller). Moving the Y_f term on the right-hand side to the left-hand side, the result is:

$$\left(I - H_f H_f^c\right) Y_f = \Gamma_f X_k + H_f \Gamma_f^c X_k^c + H_f D_f + H_{s,f} W_f + V_f \tag{6.2.17}$$

Matrix $(I-H_fH_f^c)$ is invertible (lower triangle matrix with identity matrix on the diagonal), and Y_f can be expressed as:

$$\begin{aligned} Y_f = {} & \left(I - H_f H_f^c\right)^{-1} \Gamma_f X_k + \left(I - H_f H_f^c\right)^{-1} H_f \Gamma_f^c X_k^c + \left(I - H_f H_f^c\right)^{-1} H_f D_f \\ & + \left(I - H_f H_f^c\right)^{-1} \left(H_{s,f} W_f + V_f\right) \end{aligned} \tag{6.2.18}$$

Considering that both the process states and the controller states are linear combinations of the past data, i.e., (3.1.6) and (6.2.11), the above relationship can be written as:

$$\begin{aligned} Y_f = {} & \left(I - H_f H_f^c\right)^{-1} \left(\Gamma_f A^p \Gamma_p^{+} + H_f \Gamma_f^c \Omega_p^c - H_f \Gamma_f^c A^{cp} \Gamma_p^{c+} H_p^c\right) Y_p \\ & + \left(I - H_f H_f^c\right)^{-1} \left(\Gamma_f \Omega_p - \Gamma_f A^p \Gamma_p^{+} H_p + H_f \Gamma_f^c A^{cp} \Gamma_p^{c+}\right) U_p \\ & - \left(I - H_f H_f^c\right)^{-1} H_f \Gamma_f^c A^{cp} \Gamma_p^{c+} D_p \\ & + \left(I - H_f H_f^c\right)^{-1} H_f D_f \\ & + \left(I - H_f H_f^c\right)^{-1} \left(\Gamma_f (\Omega_{s,p} - A^p \Gamma_p^{+} H_{s,p}) W_p - \Gamma_f A^p \Gamma_p^{+} V_p + H_{s,f} W_f + V_f\right) \end{aligned} \tag{6.2.19}$$

The first three terms on the right-hand side of the above equation coincide with the joint predictable subspace. Equation (6.2.19) indicates that $\boldsymbol{Y}_f$ can be predicted by [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{D}_p$; $\boldsymbol{D}_f$]. If $\boldsymbol{Y}_f$ is projected onto [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$; $\boldsymbol{D}_p$; $\boldsymbol{D}_f$], the sum of the first three terms will give an unbiased estimate of the joint predictable subspace; performing an SVD on this part can give the estimates of the joint states. This new N4SID algorithm for closed-loop data gives both the process model and the controller model. To some degree, it resembles N4SID in the joint input-output approach. However, they are completely different in both the fundamental idea and the computation.

By directly fitting the joint states to a state-space model, both the resultant models for the process and the controller are of order $n+n^c$ (not a minimum realization of the process of order n, or the controller of order n^c). A lower order model for the process or the controller is available through model-order reduction techniques or state separation techniques, such as zero-pole cancellation and balanced truncation.

6.3 CVA for Closed-loop Data

There are two basic CVA algorithms for open loop data, i.e., CVA_RO and CVA_Hf (refer to Section 3.3). When applied directly to closed-loop identification, these algorithms need to be investigated first due to the presence of correlation between inputs and disturbances in closed-loop data. In this section, these CVA algorithms will be analyzed in the multi-step state-space model with emphasis on the model bias and possible new algorithms.

6.3.1 CVA_RO Algorithm for Closed-loop data

The applicability of CVA algorithms for closed-loop identification will be analyzed based on the basic relationships between process states, inputs and outputs. In

the multi-step state-space model, the future outputs have the following relationship with the current states and the future inputs:

$$Y_f = \Gamma_f X_k + H_f U_f + H_{s,f} W_f + V_f \tag{3.1.7}$$

In this relationship, the current state sequence X_k is independent of the effects of future stochastic signals $H_{s,f}W_f+V_f$. In the open loop case, with well-designed external independent signals, U_f is un-correlated with the effects of future stochastic signals $H_{s,f}W_f+V_f$. However, in the closed-loop case, the future input data U_f is correlated to $H_{s,f}W_f+V_f$ (refer to (6.2.14) and discussions). Denote the projection of $H_{s,f}W_f+V_f$ on U_f as $B_{Nf}U_f$ and the residual as $H_{s,f}W_f+V_f-B_{Nf}U_f$ (B_{Nf} is the regression coefficient matrix), then regressing U_f out of both sides of the equation (3.1.7) gives:

$$Y_{f\_ro} = \Gamma_f X_{k\_ro} + H_{s,f} W_f + V_f - B_{Nf} U_f \tag{6.3.1}$$

where X_{k_ro} is the result of projecting X_k onto the orthogonal space of U_f, i.e., regressing U_f out of X_k.

The current state sequence X_k is a linear combination of the past inputs and outputs as shown in Section 3.1:

$$X_k = A^p \Gamma_p^+ Y_p + (\Omega_p - A^p \Gamma_p^+ H_p) U_p + (\Omega_{s,p} - A^p \Gamma_p^+ H_{s,p}) W_p - A^p \Gamma_p^+ V_p \tag{3.1.6}$$

This relationship can be simply rewritten as (S_p stands for the past stochastic signals [W_p; V_p]):

$$X_k = \begin{bmatrix} L_1 & L_2 \end{bmatrix} \begin{bmatrix} Y_p \\ U_p \end{bmatrix} + \begin{bmatrix} L_1^s & L_2^s \end{bmatrix} \begin{bmatrix} W_p \\ V_p \end{bmatrix} = L P_{IO} + L^s S_p \tag{6.3.2}$$

The above coefficient matrix L shows the relationships between current states and the past input/output data P_{IO}, and is the ultimate goal of CCA in CVA for state estimation. Regressing U_f out of both sides of above equation gives:

$$X_{k\_ro} = L P_{IO\_ro} + L^s S_{p\_ro} \tag{6.3.3}$$

Where $P_{IO\_ro}=P_{IO}-B_{PIO}U_f$, and B_{PIO} is the coefficient matrix of regressing P_{IO} against U_f; $S_{p\_ro}=S_p-B_{sp}U_f$, and B_{sp} is the coefficient matrix of regressing S_p against U_f;

In the CVA_RO algorithm, the coefficient matrix J comes from CCA on Y_{f_ro} and P_{IO_ro}, and J is an estimate of the above L matrix and used to estimate states $X_k=JP_{IO}$ (refer to Section 3.3.1). If X_{k_ro} is the only common variation between Y_{f_ro} and P_{IO_ro}, the coefficient matrix J will be an unbiased estimate of L. For closed-loop data, however, the common variation between Y_{f_ro} and P_{IO_ro} includes not only X_{k_ro} but also the common variation between $H_{s,f}W_f+V_f-B_{Nf}U_f$ and P_{IO_ro}. Consider the following: $B_{Nf}U_f$ is uncorrelated to P_{IO_ro} (which is in the orthogonal space of U_f); however, $H_{s,f}W_f+V_f$ is correlated with $P_{IO\_ro}=P_{IO}-B_{PIO}U_f$ because of the correlation between future noise $H_{s,f}W_f+V_f$ and U_f ($H_{s,f}W_f+V_f$ is un-correlated to P_{IO}). Therefore, the coefficient matrix J from CCA will be affected by the common variation between $H_{s,f}W_f+V_f-B_{Nf}U_f$ and P_{IO_ro}, thus the estimated state sequence based on J will be a biased estimate of L.

This bias of CVA_RO for closed-loop data is due to the correlation between future inputs U_f and the future noise $H_{sf}W_f+V_f$, and does not vanish if the number of data points tends to infinity. The correlation between $H_{sf}W_f+V_f$ and U_f becomes stronger if the noise in the closed-loop data becomes larger, or if a longer future horizon is used in the algorithm. Therefore, the bias from CVA_RO is expected to increase if the number of future lag steps increases or the SNR of the closed-loop data becomes poorer.

6.3.2 CVA_Hf Algorithm for Closed-loop Data

In this method, an ARX model is fitted with the closed-loop data and the impulse weights from this ARX model are used to construct an estimated H_f matrix, say H_{f_e}. The predictable subspace is estimated as $Y_f-H_{f\_e}U_f$, and then CCA is applied on this subspace and the past data to estimate the process states. The key issue in this algorithm is to have an unbiased estimation of the coefficient matrix H_f from the closed-loop data.

As shown in Section 6.2, the fitted ARX model based on closed-loop data is asymptotically unbiased for an infinite long past horizon. In practice, a high-order ARX is usually a close approximation of the true process, and the estimated impulse weights based on the fitted ARX model are adequately close to the true ones. As discussed in

Section 3.3, however, a longer past horizon also results in higher variances for the estimated parameters in the ARX model. These variances decrease as the number of data points increases (or if we use other estimation techniques than least squares to reduce the variances, such as PCR, ridge regression, etc., at the expense of a small increase in bias). The impulse weights from this ARX model are used to estimate the future input effects $\boldsymbol{H}_f\boldsymbol{U}_f$ on the future outputs. The estimated predictable subspace by this method, $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$, is an estimate of $\boldsymbol{\Gamma}_f\boldsymbol{X}_k+\boldsymbol{H}_{s,f}\boldsymbol{W}_f+\boldsymbol{V}_f$, and it is valid for use in the next step for estimation of the process states.

CCA on the estimated predictable subspace $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ and the past data $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ picks up the process states. This procedure does not suffer any problem from the correlation between $\boldsymbol{U}_f$ and the future noise $\boldsymbol{H}_{s,f}\boldsymbol{W}_f+\boldsymbol{V}_f$ since the effects of $\boldsymbol{U}_f$ have been removed from $\boldsymbol{Y}_f$. $\boldsymbol{Y}_f$-$\boldsymbol{H}_{f\_e}\boldsymbol{U}_f$ is an estimate of $\boldsymbol{\Gamma}_f\boldsymbol{X}_k+\boldsymbol{H}_{s,f}\boldsymbol{W}_f+\boldsymbol{V}_f$, which has only the current states $\boldsymbol{X}_k$ as the common variation with the past data $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ (the effects of future stochastic signals $\boldsymbol{H}_{s,f}\boldsymbol{W}_f+\boldsymbol{V}_f$ have no correlation with the past data).

In summary, providing a reasonably unbiased ARX, the CCA_Hf approach can be applied directly to closed-loop data to obtain an unbiased state-space model.

6.3.3 CVA Algorithms for the Joint States

To avoid the correlation between the future inputs $\boldsymbol{U}_f$ and the future noise in closed-loop data, one possible way is to use only the future dither signals $\boldsymbol{D}_f$ instead of $\boldsymbol{U}_f$. As shown in Section 6.2.3, the future output data $\boldsymbol{Y}_f$ has the following relationship with the process states $\boldsymbol{X}_k$, the controller states $\boldsymbol{X}^c_k$ and $\boldsymbol{D}_f$:

$$\left(I - H_f H_f^c\right)Y_f = \Gamma_f X_k + H_f \Gamma_f^c X_k^c + H_f D_f + H_{s,f} W_f + V_f \qquad (6.2.17)$$

The first two terms on the right-hand side of the above equation are the effects of the joint process-controller state sequence $X_k^J=[X_k; X_k^c]$ with rank of $n+n^c$. X_k is a linear combination of the past data $\boldsymbol{P}_{IO}$ as shown in (6.3.2), and X_k^c is a linear combination of the past data $\boldsymbol{P}_{IO}$ and past dither $\boldsymbol{D}_p$ as shown in (6.2.11). Therefore, joint state sequence

X_k^J is a linear combination of P_{IO} and D_p. Equation (6.2.17) is analogous to equation (3.1.7) with the joint states corresponding to the process states and D_f corresponding to U_f. Following the same idea of regressing U_f out as in the CVA_RO algorithm, regressing D_f out and performing CCA should give a way to estimate the joint states X^J_k.

Denote Y_{f_roDf}, P_{IO_roDf}, X_{k_roDf} and $X^c_{k\_roDf}$ as the result of regressing D_f out of Y_f, P_{IO}, X_k and X^c_k respectively. Regressing D_f out of equation (6.2.17) gives:

$$\left(I - H_f H_f^c\right) Y_{f\_roDf} = \Gamma_f X_{k\_roDf} + H_f \Gamma_f^c X^c_{k\_roDf} + \left(H_{s,f} W_f + V_f\right) \tag{6.3.5}$$

Here the past and future stochastic signals are un-correlated to the D_f. Equation (6.3.5) indicates that Y_{f_roDf} is only related to X_{k_roDf} and $X^c_{k\_roDf}$, which are related to the past data in (6.3.2) and (6.2.11) by regressing D_f out respectively:

$$X_{k\_roDf} = LP_{IO\_roDf} + L^s S_p \tag{6.3.6}$$

$$X^c_{k\_roDf} = L^c P_{IO\_roDf} + L^d D_{p\_roDf} \tag{6.3.7}$$

Equations (6.3.5), (6.3.6) and (6.3.7) indicate that X_{k_roDf} and $X^c_{k\_roDf}$ are the only common variation between $[P_{IO\_roDf}; D_{p\_roDf}]$ and Y_{f_roDf}. Therefore the dominant CVs (computed as $J[P_{IO\_roDf}; D_{p\_roDf}]$) from CCA on these two data sets will coincide with $X^J_{k\_roDf}=[X_{k\_roDf}; X^c_{k\_roDf}]$, and therefore $J[P_{IO}; D_p]$ is an unbiased estimate of the joint states X^J_k.

The above analysis can be summarized into the following procedures for a new CVA algorithm based on estimating the joint process states and controller states (CVA_Joint):

- Regress D_f out of Y_f and $[Y_p; U_p; D_p]=[P_{IO}; D_p]$, to get $Y_{f\_roDf}=Y_f P_{Dfo}$ and $[P_{IO\_roDf}; D_{p\_roDf}]=[P_{IO}; D_p]P_{Dfo}$, where $P_{Dfo}=(I - D_f^T(D_f D_f^T)^{-1} D_f)$
- Perform CCA on $[P_{IO\_roDf}; D_{p\_roDf}]$ and Y_{f_roDf}, and take the number of dominant CVs as the estimated order of the joint system. J is the coefficient matrix for $[P_{IO\_roDf}; D_{p\_roDf}]$ for dominant CVs.
- Estimate the joint process-controller states as $X_k^J = J[P_{IO}; D_p]$

- Fit the estimated X_k^J to the state-space model for process dynamic model and controller model; for controller model, $\boldsymbol{u}_k$ are the output variables and $\boldsymbol{y}_k$ are the input variables.

As in N4SID-Joint algorithm, both the resultant models for the dynamic process and the controller will be order of $n+n^c$. To obtain a minimum order model for the dynamic process (order n), one could perform pole-zero cancellation in the transfer function form or other model order reduction techniques. Another possible way is to extract an n-dimensional basis just for the process states from the above estimated joint states ($n+n^c$ dimensions), and then fit to the state-space model form.

In the multi-step state-space model, the controller can be expressed in the following relationship:

$$\left(I - H_f^c H_f\right) U_f = \Gamma_f^c X_k^c + H_f^c \Gamma_f X_k + D_f + H_f^c H_{s,f} W_f + H_f^c V_f \qquad (6.3.8)$$

This relationship is a counterpart of (6.3.5) for the controller. Regressing $\boldsymbol{D}_f$ out of both $\boldsymbol{U}_f$ and $[\boldsymbol{P}_{IO};\ \boldsymbol{D}_p]$ and performing CCA on the resultant data sets also provide a feasible CVA algorithm to estimate the joint states. The computation procedure is the same as for the above CVA-Joint algorithm except using $\boldsymbol{U}_f$ instead of $\boldsymbol{Y}_f$ for the future data.

Considering that both $\boldsymbol{Y}_f$ and $\boldsymbol{U}_f$ data sets contain the joint state information, another CVA algorithm is to join these two future data sets together, that is, regressing $\boldsymbol{D}_f$ out of both $[\boldsymbol{Y}_f;\ \boldsymbol{U}_f]$ and $[\boldsymbol{P}_{IO};\ \boldsymbol{D}_p]$, performing CCA on these modified data sets. The detailed computation procedures are similar to the above CVA-Joint algorithm except using $[\boldsymbol{Y}_f;\ \boldsymbol{U}_f]$ instead of $\boldsymbol{Y}_f$ for the future data

The joint-state CVA algorithm based on $\boldsymbol{U}_f$ data set gives a better estimation of the controller states since $\boldsymbol{U}_f$ contains more information of the controller states than the process states. Therefore, the joint-state CVA algorithm based on $\boldsymbol{Y}_f$ should be adopted if the main purpose of the closed-loop identification is to get the process dynamic model. The joint-state CVA algorithm based on $[\boldsymbol{Y}_f;\ \boldsymbol{U}_f]$ essentially is equivalent to using CVA_RO algorithm in the joint input-output approach, where y_f and u_f are the output variables of the joint system, and d_f is the inputs to the joint system.

In summary, these algorithms estimate the joint process-controller states from the future and past data sets using the same idea as the CCA_RO algorithm (here regressing out the future dither data), and fit to the state-space model form for both the process model and the controller model.

6.4 Simulation Studies

The simulation study in this section is to illustrate the analysis of SIM algorithms for closed-loop data and to demonstrate the theoretical conclusions in the previous sections. For ease of illustration, simple examples are used in simulations. The first simulation example is an ARARX process with model:

$$y(k)=\frac{0.2z^{-1}}{1-0.8z^{-1}}u(k)+\frac{1}{(1-0.95z^{-1})(1-0.8z^{-1})}e(k)$$

The steady state gain of the process is 1.0 and the dynamic time constant is about 5 sampling time periods. The noise is an AR(2) process, which has a common pole with the dynamic process. The controller used in the feedback loop is a tight PI controller with transfer function:

$$u(k)=-\frac{1.4-1.2z^{-1}}{1-z^{-1}}y(k)+d(k)$$

The dither signal added to the output of controller is a PRBS with magnitude of 4 and switching time period T_s=5. The variance of noise signal is adjusted to have different SNR levels at the output, such as SNR of 10, 3.1 and 1.0 (ratio of the variance resulting from the dither to that resulting from the noise at output). 100 simulations are done for each SNR level, and 10,000 data points are collected in each simulation.

Another simulation example used in this section is a simple general process with the following Box-Jenkins (BJ) model:

$$y(k)=\frac{0.2z^{-1}}{1-0.8z^{-1}}u(k)+\frac{1}{1-0.95z^{-1}}e(k)$$

The controller, dither signal and SNR are the same as in the first example. These simulation examples have been used in Chapter 3 and Chapter 5 for open loop cases and now are used again for easy comparison.

With the above simulation conditions for the BJ model, the cross-correlation between the current input and the past disturbance is investigated. Figure 6.2 shows the mean correlations at different lag steps and their 95% confidence limits from 100 Monte Carlo simulation data sets for SNR=10.6, 3.37 and 1.06 (average value) respectively. The severe correlation between the current input and past disturbance can last for 30 to 50 steps, about 5 to 10 times of the process time constant. It clearly shows that the correlation increases with the decrease of SNR, i.e., the poorer the SNR is, the stronger the correlation is. This correlation has a significant effect on the identification result.

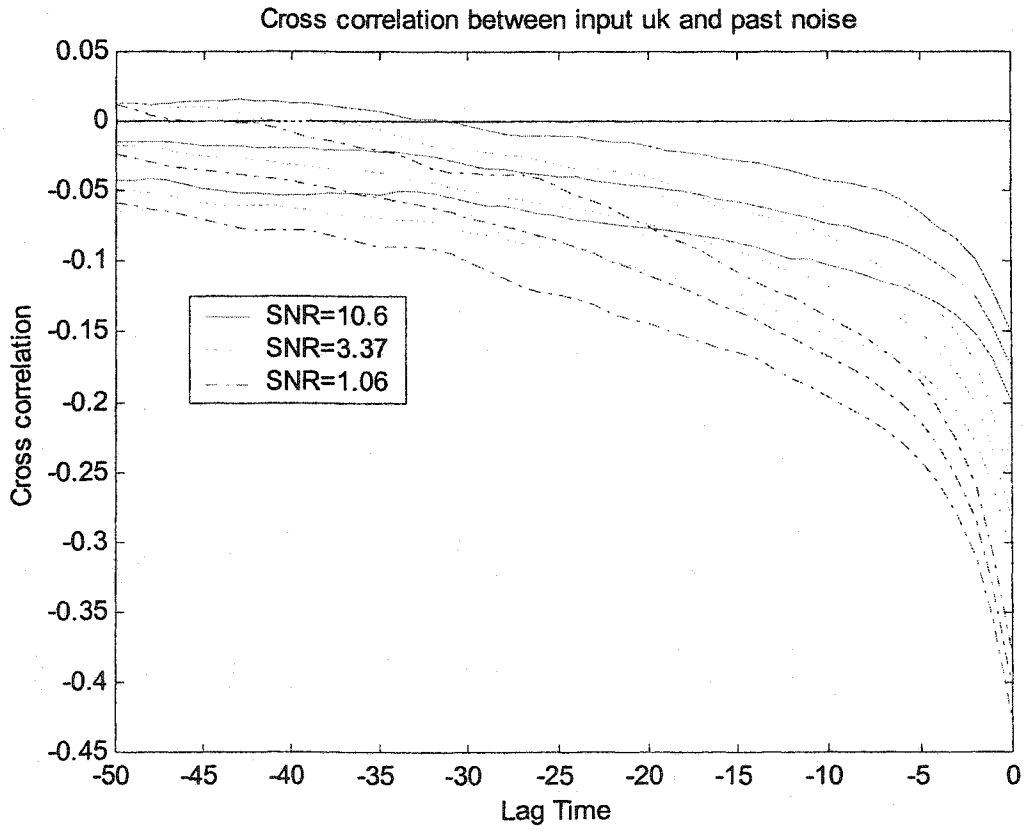


Figure 6.2 Correlation between the input and disturbance in closed loop data

In order to illustrate conclusions on applications of N4SID and CVA algorithms for closed-loop data, the above ARARX process and the controller are used for closed-loop Monte Carlo simulations. To show the relationship between the bias and the SNR, three different levels of SNR are selected by setting different values for the variance of noise signal e_k. The mean values of SNRs over 100 simulations are 10.4, 3.3 and 1.03 at the output, and 4.5, 1.4 and 0.45 at the input respectively. Each data set is used to build the dynamic model by various N4SID and CVA algorithms discussed in the last two sections. Table 6.1 gives a brief summary of the algorithms used. In order to show the relationship between the bias and the length of the future horizon, each algorithm uses 2, 6 and 12 lag steps for both the past and future horizon. For fair comparison, 9970 effective data points are used for all the cases. The system order is chosen to be 3 for joint-state-based algorithms, and order of 2 for all other algorithms. In all the SIM algorithms listed in Table 6.1, only the original N4SID and CVA_RO algorithms use $\boldsymbol{U}_f$ directly in LS regression involving the correlation between future inputs and the noise. These two algorithms are expected to give biased results based on previous analysis. All other algorithms are expected to give unbiased results.

Table 6.1 N4SID and CVA algorithms used in Mote Carlo simulations

Algorithm	1st Step	2nd Step	Order	Notes
N4SID	$\boldsymbol{Y}_f/[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{U}_p]$	PCA	2	M file in Matlab Identification toolbox
N4SID_IV	IV Method	PCA	2	Section 6.2, IVM for 1st step of N4SID
N4SID_ARX	Recursive ARX	PCA	2	Ljung: k-step predictions by recursive ARX
N4SID_RRA	ARX	RRA	2	Section 4.2.2, $\boldsymbol{H}_f$ from ARX model
N4SID_Joint	$\boldsymbol{Y}_f/[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{D}_p; \boldsymbol{D}_f]$	PCA	3	Section 6.2, Joint process-controller states
CVA_RO	RO ($\boldsymbol{U}_f$)	CCA	2	Section 3.3.1, Regress $\boldsymbol{U}_f$ out of $\boldsymbol{Y}_f$ and $\boldsymbol{P}_{IO}$
CVA_Hf	ARX	CCA	2	Section 3.3.2, $\boldsymbol{H}_f$ from ARX model
CVA_Joint	RO ($\boldsymbol{D}_f$)	CCA	3	Section 6.3, Joint process-controller states

1st step: Method for estimation of the predictable subspace

2nd step: Method for estimation of the states

For the Monte Carlo simulation for the case of SNR=10.4, the frequency responses and their 95% confidence regions from these SIM algorithms with 2 lag steps are shown in Figure 6.3. The impulse response curves are so close to the true that it is hard to distinguish them, so they are not shown here. Joint state N4SID and CVA algorithms are not applicable for this situation since at least 3 lag steps are required. For the results from the original N4SID algorithm and CVA_RO algorithm, the 95% confidence regions do not include the true process response. This clearly shows that these two algorithms give biased results for closed-loop data. The results from all other algorithms do not show any bias.

For the same simulation data sets, the results from SIM algorithms with 6 and 12 lag steps are shown in Figures 6.4 and 6.5 respectively. Compared to those results with 2 lag steps shown in Figure 6.3, the biases from the original N4SID and CVA_RO algorithms increases as the number of lag steps increases. This verifies the conclusion in previous analysis that the bias increases with the number of lag steps used. The results from all other algorithms do not show any statistically significant bias in these figures.

In order to illustrate the effects of SNR on the identified results, these N4SID and CVA algorithms are also applied to simulations with SNR=3.3. Figures 6.6 and 6.7 show the frequency responses and impulse responses as well as their 95% confidence regions respectively for the case of 6 lag steps used in these algorithms. The original N4SID and CVA_RO algorithms show clear bias on these results, and all other algorithms show unbiased results (some upper 95% confidence limits are marginally close to the true response, and the reason is explained below). Compared to those results for the case of SNR=10.4 (with 6 lag steps, showing in Figure 6.4), the biases from the original N4SID and CVA_RO algorithms become worse. This verifies the conclusion in previous analysis, that is, larger disturbance causes larger bias due to the stronger correlation between the future inputs and the effects of the future stochastic signals in the original N4SID and CVA_RO algorithms.

For the case of SNR=1.03, the results of frequency responses and impulse responses are shown in Figure 6.8 and Figure 6.9 respectively. The results from N4SID and CVA_RO show a larger bias; however, all other algorithms also show some bias (though not as large as from those two algorithms). This is due to the large estimation errors in the estimated states caused by the enormous noise, and these estimation errors introduce bias in fitting the estimated states to the state-space model (the third step of the framework for SIMs, refer to Section 5.5.2). Even if the significant LVs provide an unbiased estimate of the true state space, they still have estimation errors. Large errors in regressor variables (estimated states) in LS regression usually lead to slower dynamics and smaller steady state gains for the identified model (Wang, 2000; Höskuldsson, 1996). The noise in this simulation case is so large that the bias introduced in the third step of LS regression shows significantly in all the SIM algorithms. This is the reason that the upper 95% confidence limits in the results for SNR=3.3 and 1.03 become close to the true responses or even below the true response (showing bias). Nevertheless, the biases from the original N4SID algorithm and CVA_RO algorithm are still much larger than the results from other algorithms. In fact, the results from N4SID-Joint and CVA-Joint algorithms show larger variance than other algorithms and tend to show larger bias (for higher SNR cases). It is also due to the larger errors in the estimated states — the estimated controller state is also deemed estimation error for the process states in the LS regression procedure.

In these Monte Carlo simulations, the identified models from the original N4SID algorithm and CVA_RO algorithm show bias, and the results from other algorithms included in Section 6.2.3 and 6.3.3 do not show bias. These Monte Carlo simulations illustrate the analysis in the last two sections and verify conclusions about the relationships between bias and future lag steps as well as the SNR when SIM algorithms are applied to closed-loop data. The simulation results for cases with very low SNRs also show the effects of state estimation errors in fitting the estimated states to the state-space model.

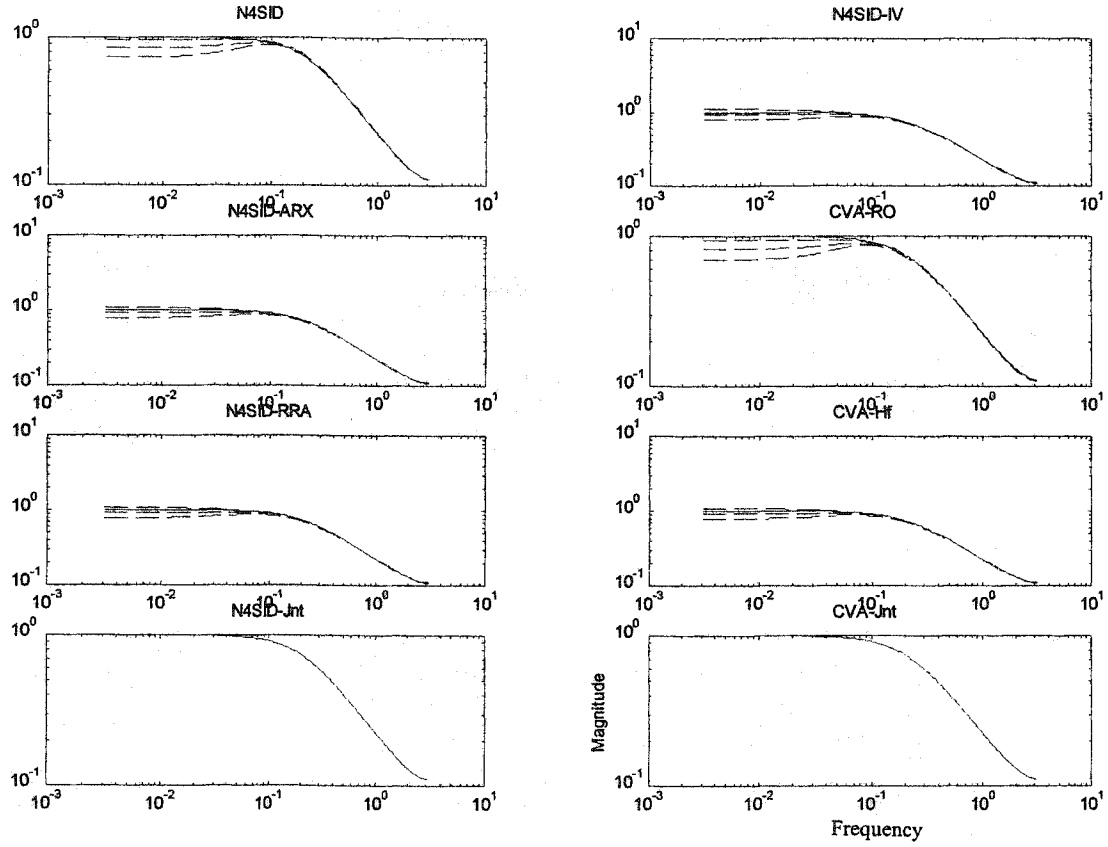


Figure 6.3 Frequency response results from N4SID and CVA algorithms (SNR=10.4, lags=2)

Solid line: true response, dashed lines: identified result (mean response and 95% confidence limits)

(No result for algorithms based on joint states for only 2 lag steps used)

N4SID algorithms: left column and top plot in right column

CVA algorithms: lower three plots in right column

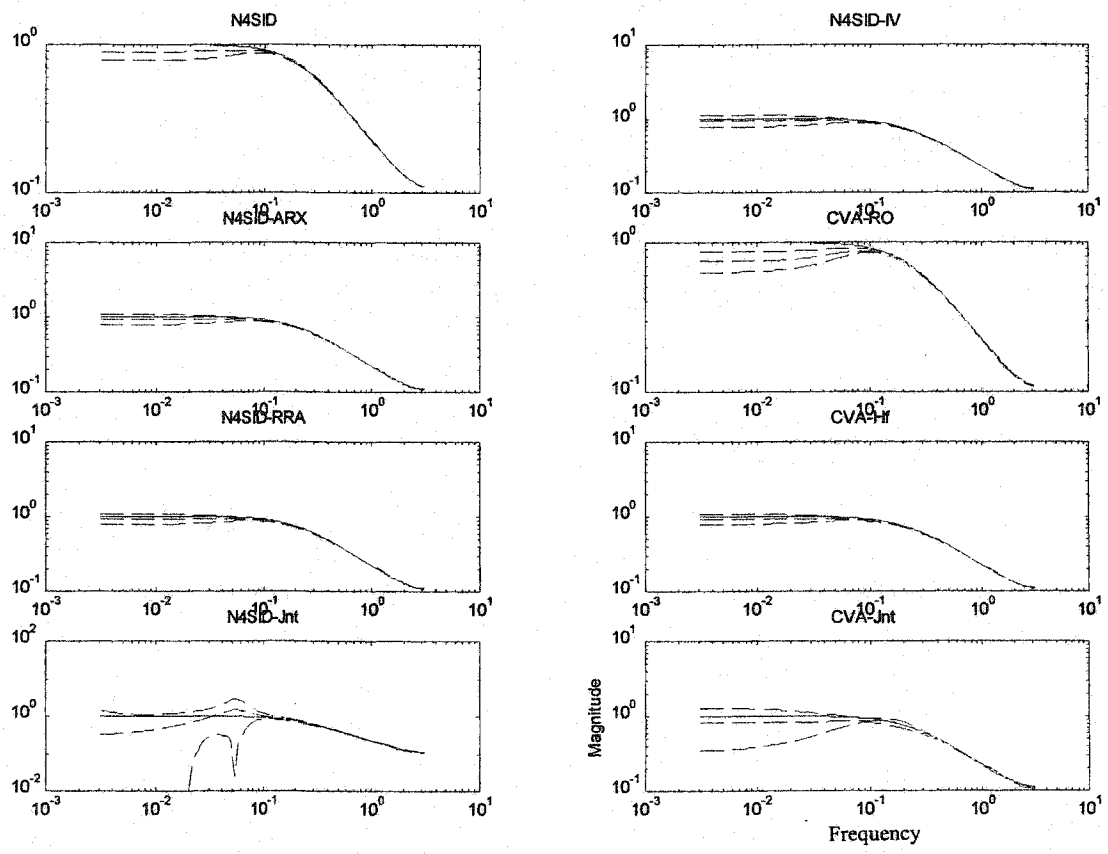


Figure 6.4 Frequency response results from N4SID and CVA algorithms (SNR=10.4, lags=6)

Solid line: true response, dashed lines: identified result (mean response and 95% confidence limits)

N4SID algorithms: left column and top plot in right column

CVA algorithms: lower three plots in right column

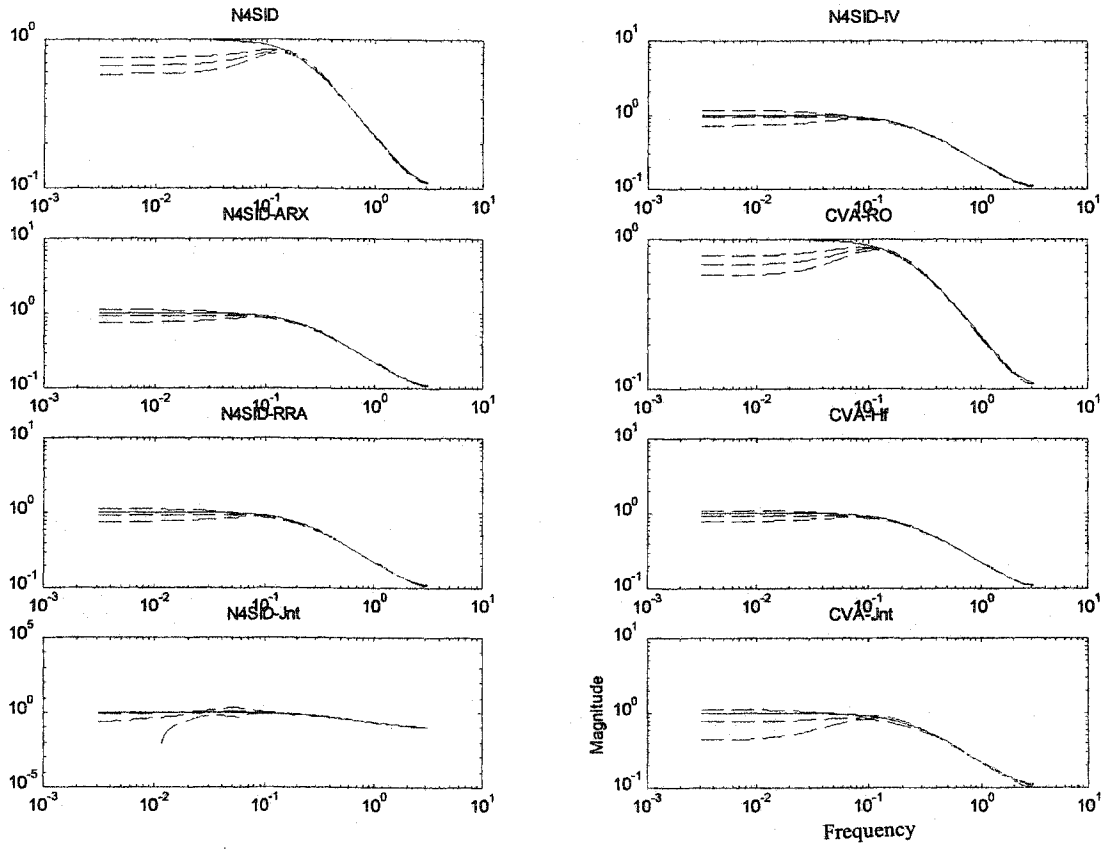


Figure 6.5 Frequency response results from N4SID and CVA algorithms (SNR=10.4, lags=12)

Solid line: true response, dashed lines: identified result (mean response and 95% confidence limits)

N4SID algorithms: left column and top plot in right column

CVA algorithms: lower three plots in right column

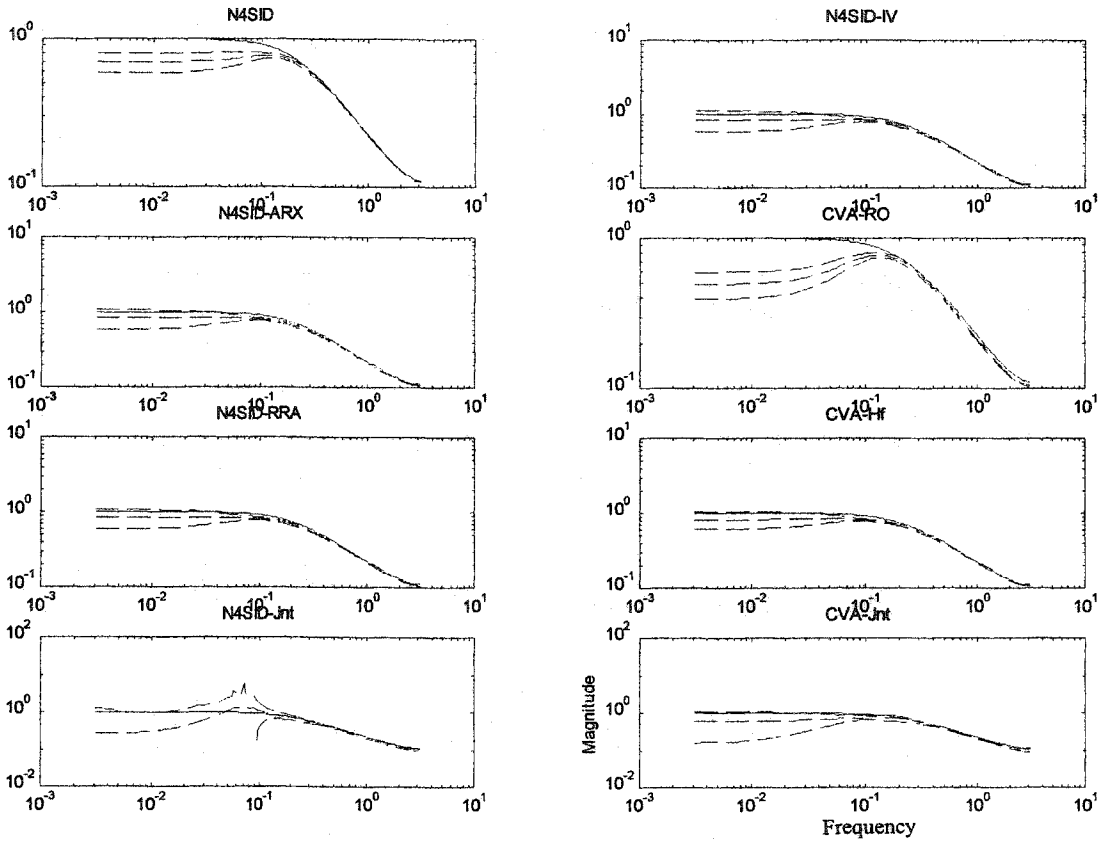


Figure 6.6 Frequency response results from N4SID and CVA algorithms (SNR=3.3, lags=6)

Solid line: true response, dashed lines: identified result (mean response and 95% confidence limits)

N4SID algorithms: left column and top plot in right column

CVA algorithms: lower three plots in right column

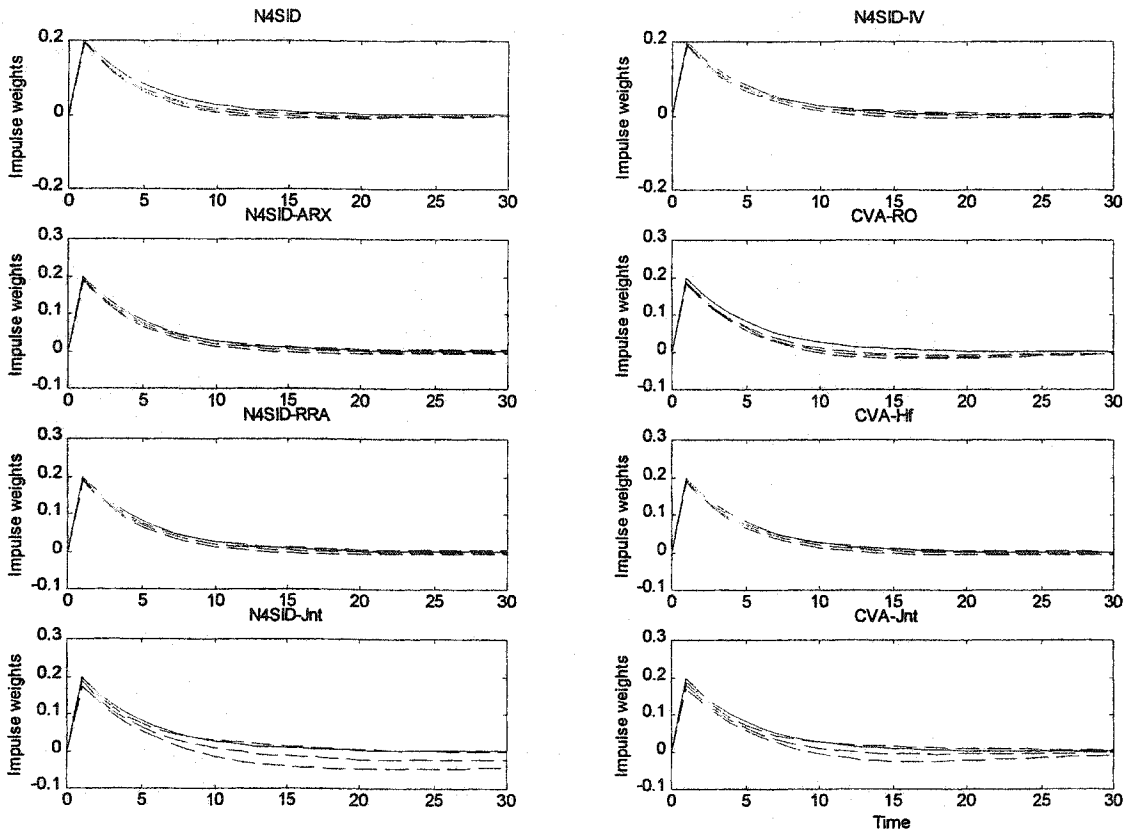


Figure 6.7 Impulse response results from N4SID and CVA algorithms (SNR=3.3, lags=6)

Solid line: true response, dashed lines: identified result (mean response and 95% confidence limits)

N4SID algorithms: left column and top plot in right column

CVA algorithms: lower three plots in right column

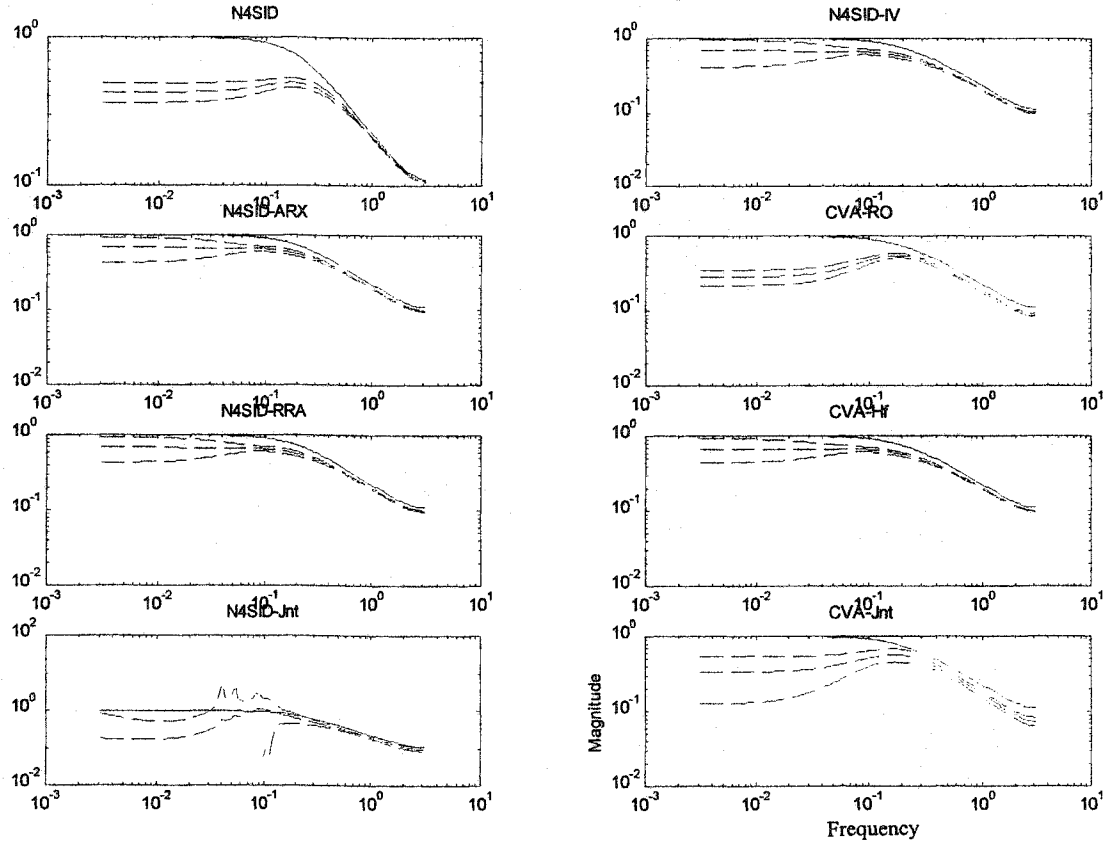


Figure 6.8 Frequency response results from N4SID and CVA algorithms (SNR=1.03, lags=6)

Solid line: true response, dashed lines: identified result (mean response and 95% confidence limits)

N4SID algorithms: left column and top plot in right column

CVA algorithms: lower three plots in right column

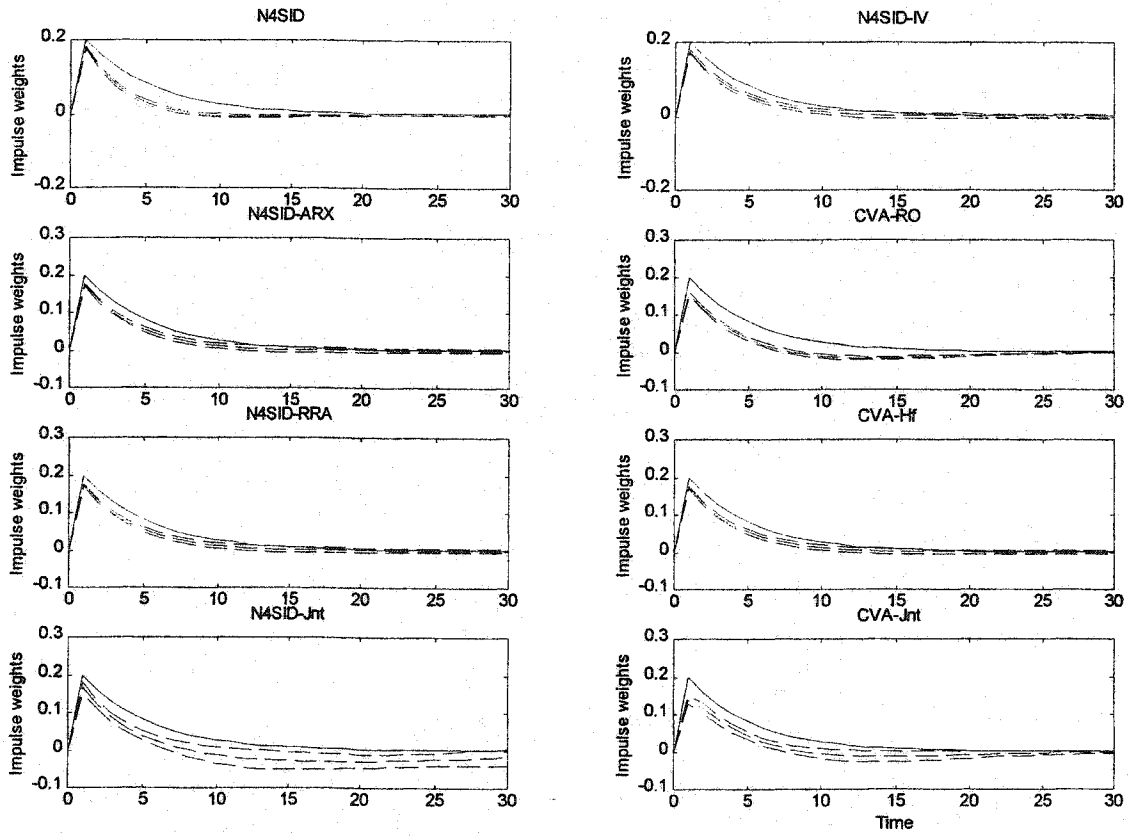


Figure 6.9 Impulse response results from N4SID and CVA algorithms (SNR=1.03, lags=6)

Solid line: true response, dashed lines: identified result (mean response and 95% confidence limits)

N4SID algorithms: left column and top plot in right column

CVA algorithms: lower three plots in right column

6.5 Conclusions

This chapter investigated the applicability of existing SIM algorithms for closed-loop data, clarified the disputes on this issue among researchers, and proposed several new SIM algorithms for closed-loop data.

The fundamental difference between open loop data and closed-loop data is the correlation between the inputs and the disturbance in closed-loop data. This raises the bias issue when SIM algorithms are applied to closed-loop identification.

For closed-loop data, the original N4SID algorithm was analyzed in both time domain and multi-step state-space model. The correlation between future inputs and the future noise causes bias in the first step of N4SID, where future inputs are directly used in the LS regression. This analysis not only confirmed the analysis in the viewpoint of infinite impulse weights in literature but also provided basis for new SIM algorithms for closed loop data.

Several new N4SID algorithms were proposed for closed-loop data. N4SID-RRA, proposed for open loop data, is also applicable for closed-loop data with correctly fitted FIR or high-order ARX model. N4SID-IV avoids the correlation problem by using instrumental variable method (IVM) instead of LS regression in the first step of N4SID. N4SID joint state algorithm is to perform oblique projection of the future outputs onto the past input, output and dither data along the future dither data, and get estimates of both the process states and the controller states. This algorithm is able to give both the process model and the controller model in a high order.

For closed-loop data, CVA_RO algorithm was analyzed in the multi-step state-space model. This algorithm gives biased results due to the correlation between the future inputs and the future noise. The CVA_Hf algorithm was shown to give asymptotically unbiased results for general closed-loop systems. With availability of dither signals, new CVA algorithms based on estimation of the joint process-controller states were proposed

for closed-loop data. Both the process model and the controller model can be identified by these CVA algorithms.

In general, whether a subspace identification method is applicable for closed-loop data depends on the specific algorithm, in fact, applicability depends on how the future inputs are used in the detailed computation procedure. One SIM algorithm is applicable for closed-loop data only when the correlation between future inputs and the future noise is avoided.

7 Practical Issues and Application Guidelines

7.1 Introduction

While the fundamental research on SIMs is progressing and making advances, SIMs have been employed for many applications, such as modeling of a distillation column (Schaper 1990), identification and control of unstable aircraft flutter (Peloubet et al., 1990). SIMs have shown great potential for practical applications in control design and process monitoring. Yet in many applications, SIMs need to deal with special issues, such as process delays, common model structures in MIMO systems, and non-stationary and co-integrating disturbance problems (see Sections 7.4 and 7.5). Such practical problems are common in chemical processes. These practical problems have a great impact on the successful application of SIMs, and they also critically affect the applicable scope of SIMs. Solving these problems may not involve as deep fundamental research as in previous chapters, but the solutions are often not trivial. In this chapter, these practical issues will be discussed and analyzed, and their effects or suggested solutions are demonstrated by simulation examples.

As shown in Section 2.5, SIMs have their strengths and limitations when compared to traditional system identification methods. Therefore, SIMs should be applied to the situations or processes where their strengths can be used and their shortcomings can be avoided. This chapter will discuss some general guidelines for applying SIMs based on the conclusions in the previous chapters and on research experience gained during this study of SIMs. These general guidelines are not rigorous proofs or conclusions, and they should only be considered as suggestions for the application of SIMs.

7.2 SIMs for Processes with Delays

7.2.1 Effects of Process Delays on SIMs

Pure delays from process inputs to process outputs are common in practical applications, especially for the chemical processes, where delays are much more common than in electrical or mechanical systems. For example, flowing materials may need some time to travel from the measurement points to reactors (distance velocity delay). Delays have significant impact on many practical applications, such as on the design of controllers (e.g., minimum variance controller, Smith predictors, Dahlin's controller) and on controller performance monitoring. Therefore information on process delay is vital for many practical applications.

In some applications, one may have the prior knowledge of the process delays or a rough estimation of the process delays. In such cases, one can remove the effect of delays by shifting the inputs sequence for the corresponding delays before performing system identification. The known delays can be incorporated directly into the final identified model.

In many applications, however, delays in a process are not clear and need to be determined from the experimental data. In traditional system identification methods, such as fitting FIR models, delays can be estimated from the estimated impulse weights or step responses of the final identified model based on the number of near zero weights in the initial period of step response.

Consider a SISO process with unknown b steps of pure delays. The process can be represented in the following state-space model (for simplicity, only the deterministic part is shown):

$$x_{k+1} = Ax_k + Bu_{k-b}$$
$$y_k = Cx_k$$

The current states x_k are effects of the past inputs up to u_{k-b-1}, and the delayed inputs (u_{k-b} to u_{k-1}) will show their effects on the future horizon. The future outputs are shown in the following relationships:

$$
\begin{aligned}
y_k &= Cx_k \\
y_{k+1} &= CAx_k + CBu_{k-b} \\
y_{k+2} &= CA^2x_k + CABu_{k-b} + CBu_{k-b+1} \\
&\cdots\cdots \\
y_{k+b} &= CA^bx_k + CA^{b-1}Bu_{k-b} + \ldots + CABu_{k-2} + CBu_{k-1} \\
y_{k+b+1} &= CA^{b+1}x_k + CA^bBu_{k-b} + \ldots + CA^2Bu_{k-2} + CABu_{k-1} + \underline{CBu_k} \\
&\cdots\cdots \\
y_{k+f-1} &= CA^{f-1}x_k + CA^{f-2}Bu_{k-b} + \; + CA^{f-b-1}Bu_{k-1} + \underline{CA^{f-b-2}Bu_k + CA^{f-b-3}Bu_{k+1} + . + CBu_{k+f-b-2}}
\end{aligned}
\tag{7.2.1}
$$

From the above equations, it is clear that the effects of the delayed inputs (terms related with u_{k-b} to u_{k-1}) are included in the future outputs. Removing the effects of the future inputs $\boldsymbol{H}_f\boldsymbol{U}_f$ (underlined terms in equation (7.2.1)) does not eliminate the effects of delayed inputs in the future outputs. Therefore the data matrix $\boldsymbol{Y}_{f\_e}=\boldsymbol{Y}_f-\boldsymbol{H}_{f\_est}\boldsymbol{U}_f$ (the estimated predictable subspace as in CVA_Hf algorithm) contains not only the predictable subspace $\Gamma_f X_k$ (the effects of current states) but also the effects of delayed inputs (and the future noise if there is disturbance to the process). In the method of projecting $\boldsymbol{Y}_f$ against $[\boldsymbol{Y}_p; \boldsymbol{U}_p; \boldsymbol{U}_f]$ (as in the original N4SID algorithm), the delayed inputs are included in the past inputs $\boldsymbol{U}_p$, therefore the effects of the delayed inputs are included in the oblique projection results $\boldsymbol{L}_1\boldsymbol{Y}_p+\boldsymbol{L}_1\boldsymbol{U}_p$.

In SIMs, estimated states are taken as the significant LVs from a LVM (e.g., CCA or RRA) on data matrix $\boldsymbol{Y}_{f\_e}=\boldsymbol{Y}_f-\boldsymbol{H}_{f\_est}\boldsymbol{U}_f$ and $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$. As a result of including the effects of the pure delays in $\boldsymbol{Y}_{f\_e}$, the LVs from the above LVM are linear combinations of $[\boldsymbol{Y}_p; \boldsymbol{U}_p]$ (the delayed inputs are included in $\boldsymbol{U}_p$), and will not only predict the effects of the current states but will also predict the effects of the delayed inputs. Therefore, the significant LVs (estimated states) include essentially the true process states and the pure delayed inputs. In other words, the estimated states from SIM algorithms contain both the process true states and the delayed inputs.

For a SISO process, with adequate lag steps in the past data set, the number of significant LVs (estimated states) from SIM algorithms will include both the true process order and the number of pure delays:

$$\textbf{\# of significant LVs = true system order + \# of pure delays} \tag{7.2.2}$$

For a SIMO system, the delays to different outputs are usually different. The LVs only need to memorize the maximum number of delayed inputs, therefore the number of significant LVs will be:

$$\textbf{\# of significant LVs = true system order + max(delays for outputs)} \tag{7.2.3}$$

For a MISO system, the numbers of delays from different inputs to the output are usually different. The overall effect of the delays on a future output is a linear combination of the delayed inputs. For each step within the maximum number of delays, a new linear combination of these delayed inputs shows in the future output by involving new delayed input signals (refer to (7.2.1)). Beyond the maximum number of delays, no more linear combinations are independent of previous linear combinations (no more new delayed input signal; see also the simulation example for MISO case in Section 7.2.3). Therefore, the maximum number of delays of different inputs determines the number of linear combinations required to show the effects of delays. The delayed inputs are memorized in the estimated states in the form of linear combinations, not the individual delayed inputs of each input variable. The number of significant LVs is:

$$\textbf{\# of significant LVs = true system order + max(pure delays of inputs)} \tag{7.2.4}$$

For a MIMO system, SIM algorithms join all the outputs together. Therefore a MIMO system can be deemed as the combination of a series of MISO systems, and in general the number of significant LVs is:

$$\begin{aligned}\text{Max}(&\textbf{\# of significant LVs) = true system order} \\ &\textbf{+ sum (maximum of delays over all inputs for one output)}_{\textbf{over all outputs}}\end{aligned} \tag{7.2.5}$$

When a series of MISO systems are put together, the linear combinations of delayed inputs for different outputs might become collinear, therefore the estimated order of a MIMO system might be lower than the maximum from (7.2.5).

By increasing the number of estimated states, SIM algorithms are able to deal with the process with unknown delays. However, the effect of pure delays decreases the accuracy of the estimated final model. First, the increased model order leads to a larger number of parameters to be fit in LS estimation for the system matrices and therefore the variance of the estimated parameters will increase. For example, the number of parameters in the A matrix increases with the square of system order. Secondly, the delayed inputs memorized in the estimated states lead to extra state estimation errors, and thus an increase in the variance of the estimated model parameters and introduces extra bias in the fitted state-space model by error in the regressor variables (x_k) in the LS regression for system matrices.

7.2.2 Detection of Process Delays and Solutions for SIMs

The delayed inputs memorized in the estimated states are combined with the estimates of the true process states and cannot be recognized or separated out directly. If an ARX or FIR model is fitted during estimation of the predictable subspace (as in CVA_Hf algorithm), the delays can be estimated based on the calculated impulse responses. The same method can be used to detect the delays based on the impulse responses of the high-order state-space model obtained from SIM algorithms.

Here an approach is proposed to detect and diagnose the delayed inputs memorized in the estimated states: perform a CCA between the estimated states and past inputs over a sufficient horizon, the number of CCCs (canonical correlation coefficients) significantly close to 1.0 indicates the number of state variables used to model the delays in the system. In practice, these CCCs will be slightly below 1.0 due to the noise in the system. If the horizon of past inputs is beyond the delays, these CCCs indicate both the delayed inputs memorized in the estimated states, and the contribution of far past inputs (beyond delays) on the process states. In fact, the true states are the result of a long

history of past inputs (a period of 3 to 4 times of the system time constant) beyond the delay period. The immediate past inputs beyond the delayed inputs usually do not have a large contribution to the true states (unless the time constant is very short). The large CCCs (close to 1.0) are mainly due to the delayed inputs memorized in the estimated states, and thus the number of CCCs close to 1.0 should be a good indication of the maximum number of delays in the system, and the corresponding coefficient vectors of the past input data (vectors in $\boldsymbol{J}$ matrix in CCA) show the delay structure in the inputs. The past inputs corresponding to non-zero elements (noise-free case) or major elements (case with noise) of these loading vectors are the delayed inputs memorized in the estimated states. The CCC's closeness to 1.0 and the significance of the elements in loading vectors depend on the system structure, SNR and the number of the collected data points. The criteria are somewhat subjective; nevertheless this CCA is a simple and effective approach to detect and diagnose the delay structure.

After detecting the delay structure in the system, the delayed inputs can be eliminated from the past data $\boldsymbol{U}_p$, and the LVMs on Y_f-$\boldsymbol{H}_f\boldsymbol{U}_f$ and modified past data will not catch the effects of delayed inputs in their LVs. This approach works well for SISO and MISO cases. In SIMO and MIMO cases, one input variable may have a different number of delays to different outputs. The number of delays common to all input variables can be eliminated from the $\boldsymbol{U}_p$ data set. The estimated states will contain no (for SISO and MISO) or fewer delayed inputs (for SIMO and MIMO), and will reduce the estimated system order; therefore the number of parameters in system matrices to be estimated is reduced effectively.

7.2.3 Simulation Examples

In this subsection, various simulations for simple and complex cases are used to illustrate the effects of delayed inputs on SIMs, and to illustrate how to use CCA to detect and diagnose the delay structure in a process.

Delays in a deterministic SISO system

To illustrate the effects of delays clearly, a simple deterministic process is used as the simulated example:

$$y_k = \frac{0.2z^{-3}}{1-0.8z^{-1}}u_k \tag{7.2.6}$$

The process is a first-order system with 2 delays. In simulation, the input is a PRBS with switching time period of 5 and magnitude of 4, and 1000 data points are collected.

The CVA_Hf algorithm is used for the simulation data with 6 lag steps for the past and future horizons respectively. The impulse weights from a fitted ARX model are estimated correctly in this noise-free case. The rank of $\boldsymbol{Y}_{f\_e}=\boldsymbol{Y}_f-\boldsymbol{H}_{f\_est}\boldsymbol{U}_f$ is 3 though there are 6 lagged variables. CCA between the past data and $\boldsymbol{Y}_{f\_e}$ gives 3 perfect CCCs (1.0), and the system is determined to be of order 3. Take the first 3 CVs as the estimated states x_{k_est} and the resultant state-space model is:

$$\begin{aligned} x_{k+1} &= \begin{bmatrix} 0.2599 & -0.1385 & 0.6012 \\ 1.7765 & -0.4487 & -1.0308 \\ -0.1697 & -0.0434 & 0.9888 \end{bmatrix} x_k + \begin{bmatrix} 0.0069 \\ -0.0161 \\ 0.0001 \end{bmatrix} u_k \\ y_k &= \begin{bmatrix} -2.8287 & -1.5567 & -98.5368 \end{bmatrix} x_k \end{aligned} \tag{7.2.7}$$

This state-space model can be transformed into the following form:

$$\begin{aligned} z_{k+1} &= \begin{bmatrix} 0.8 & -0.0029 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} z_k + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_k \\ y_k &= \begin{bmatrix} -69.0855 & 0 & 0 \end{bmatrix} z_k \end{aligned} \tag{7.2.8}$$

through a transform of $z_k=\boldsymbol{T}x_k$ with matrix $\boldsymbol{T}$ as:

$$T = \begin{bmatrix} 0.6470 & 0.2938 & 0.0069 \\ 0.3702 & 0.9514 & -0.0161 \\ 0.6665 & 0.0921 & 0.0001 \end{bmatrix}$$

In (7.2.8), it is clear that the past two inputs are memorized in the last two states. If the two delays are shifted to the input, the model can be simplified as (first order system):

$$z_{1,k+1} = 0.8z_{1,k} - 0.0029u_{k-2}$$
$$y_k = -69.0855z_{1,k} \tag{7.2.9}$$

To detect the delays memorized in the estimated states, CCA is performed between the estimated states and the past inputs. The CCCs between past inputs [u_{k-1}, u_{k-2}, u_{k-3}] and the estimated states x_{k_est} are 1.0, 1.0 and 0.6023. The number of CCCs equal to 1.0 clearly indicates that the two delayed inputs are included in the estimated states x_{k_est}. The coefficient matrix for the past inputs is (columns are for different CVs, and rows 1, 2 and 3 correspond to variable u_{k-1}, u_{k-2} and u_{k-3} respectively):

-0.0138	-0.0108	-0.0010
0.0172	0.0034	0.0165
0.0000	0.0000	-0.0175

It is clear that the first 2 CVs (corresponding to perfect CCCs) come solely from u_{k-1} and u_{k-2}, and the past input u_{k-3} has no contribution to these two CVs. This indicates that two of the three estimated states are to memorize the two delayed past inputs u_{k-1} and u_{k-2}.

The original N4SID algorithm is also applied to the simulation data. The first three singular values from SVD (PCA) on $L_1Y_p+L_2U_p$ are 202.6741, 27.8965 and 5.2661, and the remaining three are zeros. The system is determined clearly to be of order 3. The state-space model from N4SID is:

$$x_{k+1} = \begin{bmatrix} 0.8960 & 0.1746 & -0.0244 \\ -0.2648 & 0.3734 & 0.3279 \\ -0.2156 & -0.6718 & -0.4694 \end{bmatrix} x_k + \begin{bmatrix} 0.0157 \\ 0.0356 \\ 0.0371 \end{bmatrix} u_k$$
$$y_k = \begin{bmatrix} 6.7601 & -4.0283 & 1.0077 \end{bmatrix} x_k \tag{7.2.10}$$

This model can have similar forms as (7.2.8) and (7.2.9) through transforms. CCCs between the past inputs and the estimated states indicate two delay steps in the process.

Delays in a SISO system with noise

The second example is a combined deterministic-stochastic process. The deterministic part is the same as (7.2.3), and a white noise (unit variance) is added on the output as the measurement noise. The SNR is about 10.

The CVA_Hf algorithm is applied for this example with 6 lag steps for the past and future horizons respectively. The fitted ARX model is:

$$y_k = .0506y_{k-1} + .0327y_{k-2} + .1256y_{k-3} + .1108y_{k-4} + .1095y_{k-5} + .0005y_{k-6} - .0139u_{k-1} - .0370u_{k-2} + .2323u_{k-3} + .1602u_{k-4} + .1249u_{k-5} + .0844u_{k-6} + \varepsilon_k \quad (7.2.11)$$

The first 8 estimated impulse weights are 0, -0.0139, -0.0377, 0.2299, 0.1689, 0.1346, 0.1201 and 0.0531. The two small impulse weights at the initial period (lags 1 and 2) indicate 2 delay steps in the process. The 6 CCCs from CCA on past data and Y_{f_e} are 0.9873, 0.7090, 0.2487, 0.1434, 0.1346 and 0.1049 respectively. There are three dominant CVs and the system are determined to be order of 3 by AIC. The identified model by CVA has the following transfer function for the deterministic part (with one real pole at 0.8036 and a pair of insignificant complex poles):

$$y_k = \frac{-0.0320z^{-1} + 0.0080z^{-2} + 0.2365z^{-3}}{1 - 0.7528z^{-1} + 0.0156z^{-2} - 0.0453z^{-3}} u_k \quad (7.2.12)$$

Its impulse response is shown in Figure 7.1.

To detect the delays in the process, CCA is performed between the estimated states and the past inputs [u_{k-1}, u_{k-2}, u_{k-3}]. The CCCs are 0.9977, 0.9052 and 0.6461. This indicates that essentially 2 delayed inputs are memorized in the estimated states. The corresponding coefficient matrix is (columns are for different CVs, and rows 1, 2, and 3 corresponds to variables u_{k-1}, u_{k-2} and u_{k-3} respectively):

0.0047	0.0168	0.0016
0.0039	-0.0167	-0.0168
-0.0005	-0.0004	0.0175

The large elements in the first two loading vectors indicate that the delayed inputs u_{k-1} and u_{k-2} are memorized in the estimated states.

After detection of the 2 delay steps, one can shift the input for 2 time steps: take u_{k-2} as current input, and u_{k-2} and u_{k-1} are taken into the future input data set (Y_p and Y_f remain unchanged). The CCCs from the CCA in CVA_Hf algorithm are 0.9790, 0.1584, 0.1441, 0.1283, 0.1076 and 0.0997 respectively. It clearly indicates the model to be order 1. The CCC between the estimated state and the true state is 0.9980, and this indicates an

excellent estimation of the process state. The identified model has the following transfer function for the dynamic part (impulse response is shown in Figure 7.1):

$$y_k = \frac{0.2075z^{-1}}{1-0.7948z^{-1}}u_{k-2} \tag{7.2.13}$$

Based on the same treatment on the input, the CVA_RO algorithm gives the following transfer function for the dynamic part (impulse response is shown in Figure 7.1):

$$y_k = \frac{0.2139z^{-1}}{1-0.7888z^{-1}}u_{k-2} \tag{7.2.14}$$

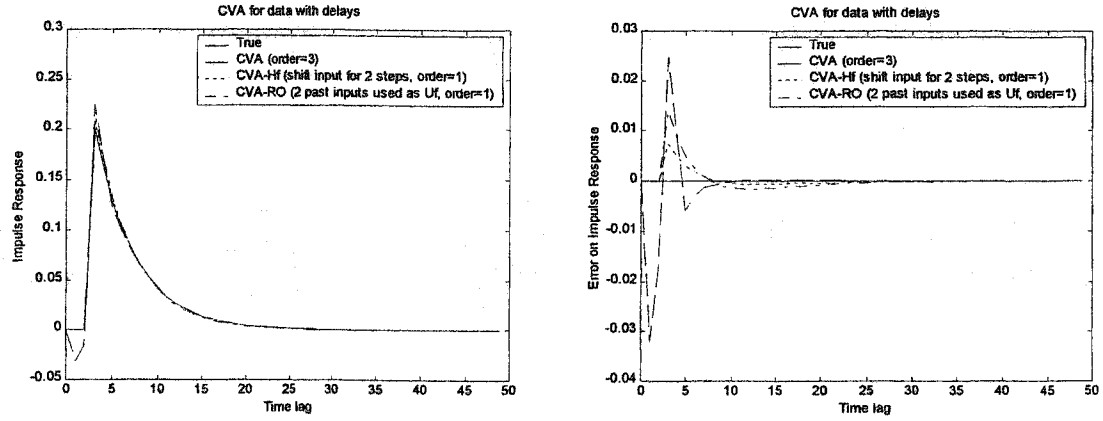


Figure 7.1 CVA algorithms for process with delays (impulse response and the errors)

The original N4SID algorithm is also applied for the simulation data with 6 lag steps for the past and future horizons. The singular values from PCA are 204.1816, 32.0449, 7.7955, 4.6428, 4.2914 and 3.3504 respectively. The first three are dominant, but the cut-off is not so obvious. The order is estimated to be 3 by AIC. The identified model has the following transfer function for the dynamic part (one real pole is at 0.8067, and a pair of insignificant complex poles):

$$y_k = \frac{-0.0392z^{-1}+0.0255z^{-2}+0.2153z^{-3}}{1-0.8613z^{-1}+0.1673z^{-2}-0.0994z^{-3}}u_k \tag{7.2.15}$$

If the delayed inputs are taken into the future input data set, N4SID give the following result for the dynamic part:

$$y_k = \frac{0.2142z^{-1}}{1-0.7885z^{-1}}u_{k-3} \tag{7.2.16}$$

Impulse responses based on the identified models from N4SID method are shown in Figure 7.2. The CCCs between the three estimated states and past inputs [u_{k-1}, u_{k-2}, u_{k-3}], the CCCs are 0.9975, 0.8826 and 0.6093. This indicates 2 delay steps in the estimated states.

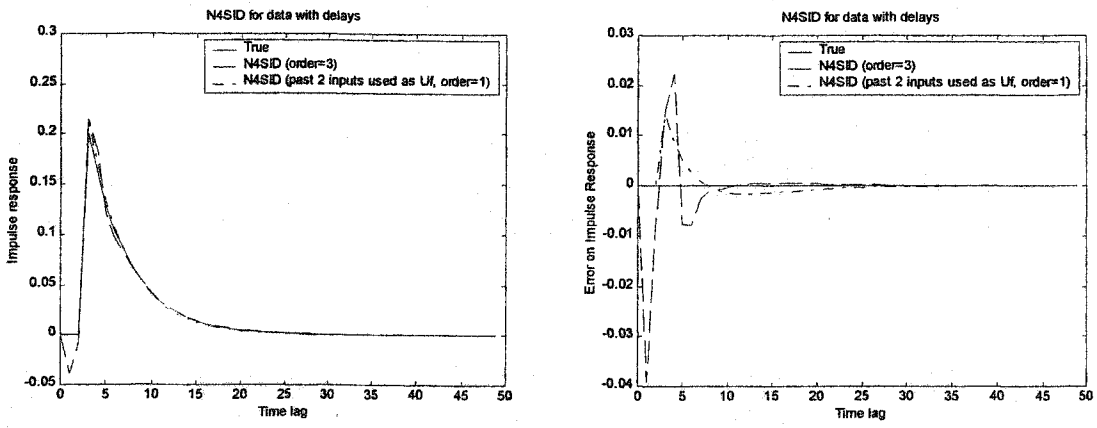


Figure 7.2 N4SID for data with delays (impulse response and the errors)

All the above results show that both CVA and N4SID can deal with the process delays; the delayed inputs are memorized in the estimated states. CCA can be used as an approach to detect the delays in the estimated states, and a lower-order model with better accuracy can be obtained by appropriate modification of the SIM algorithms.

Delays in MIMO systems

Simulation studies for SIMO, MISO and MIMO systems in this part are to illustrate the effects of delays on SIM algorithms, and the approach to detect the delays. The simulation example has 2 inputs and 3 outputs:

$$\begin{cases} y_{1,k} = \dfrac{0.8z^{-3}}{1-0.8z^{-1}}u_{1,k} + \dfrac{0.4z^{-2}}{1-0.95z^{-1}}u_{2,k} + a_{1,k} \\ y_{2,k} = \dfrac{0.2z^{-4}}{1-1.8z^{-1}+0.9z^{-2}}u_{1,k} + \dfrac{0.6z^{-3}}{1-0.9z^{-1}}u_{2,k} + a_{2,k} \\ y_{3,k} = \dfrac{1.2z^{-5}}{1+0.5z^{-1}}u_{1,k} + \dfrac{0.4z^{-4}-0.2z^{-5}}{1-1.7z^{-1}+0.85z^{-2}}u_{2,k} + a_{3,k} \end{cases} \tag{7.2.17}$$

All the transfer functions have different poles. The transfer function from $u_{1,k}$ to $y_{2,k}$ has poles at $0.9\pm0.3i$, and the transfer function from $u_{2,k}$ to $y_{3,k}$ has poles at $0.85\pm0.3571i$. The transfer function from u_1 to y_3 has a pole at –0.5 (ringing response). The disturbance for each output is independent white noise.

The two inputs signals are independent PRBS sequences with magnitude of 4 and switch time period T_s=5. In each simulation, 5000 points are collected. Simulations are performed for different SNR cases (including a noise-free case). For fair comparison, 15 lag steps are used for both past and future horizons in all case studies. Table 7.1 summarizes the results of the CVA_Hf algorithm for SIMO, MISO and MIMO systems at SNR of 10. Some of the results are shown briefly as demonstration below.

Table 7.1 CVA results for delays in MIMO systems (SNR=10)

Cases	Inputs	Outputs	Estimated Order	CCA between past inputs and estimated states		
				# of past input steps	# of CCCs close to 1.0	Delays determined by coefficient matrix
SIMO	u_1	y_1, y_2, y_3	8	6	4	$u_{1,k-1}, u_{1,k-2}, u_{1,k-3}, u_{1,k-4}$
SIMO	u_2	y_1, y_2, y_3	7	6	3	$u_{2,k-1}, u_{2,k-2}, u_{1,k-3}$
MISO	u_1, u_2	y_1	4	5	2	$u_{1,k-1}, u_{1,k-2}, u_{2,k-1}$
MISO	u_1, u_2	y_2	5	3	2	$u_{1,k-1}, u_{1,k-2}, u_{2,k-1}, u_{2,k-2}$
MISO	u_1, u_2	y_3	7	5	4	$u_{1,k-1}$ to $u_{1,k-4}$, $u_{2,k-1}$ to $u_{2,k-3}$
MIMO	u_1, u_2	y_1, y_2	9	3	4	$u_{1,k-1}, u_{1,k-2}, u_{2,k-1}, u_{2,k-2}$
MIMO	u_1, u_2	y_1, y_3	10	5	5*	$u_{1,k-1}$ to $u_{1,k-4}$, $u_{2,k-1}$ to $u_{2,k-3}$
MIMO	u_1, u_2	y_2, y_3	12	5	6*	$u_{1,k-1}$ to $u_{1,k-4}$, $u_{2,k-1}$ to $u_{2,k-3}$
MIMO	u_1, u_2	y_1, y_2, y_3	14	5	6*	$u_{1,k-1}$ to $u_{1,k-4}$, $u_{2,k-1}$ to $u_{2,k-3}$*

Note: Column 5: the number of past steps of input(s) used for the CCA
Column 6: the number of significant CCCs (close to 1.0).
Column 7: past inputs determined to be pure delay based on the coefficient matrix
*: not a clear cut-off for the CCC significance, or delayed input not clearly determined.

For the SIMO system for u_1 to all the three outputs, the CCCs from CCA are 0.996, 0.991, 0.984, 0.977, 0.967, 0.950, 0.925, 0.707, 0.370, 0.358, 0.339 and 0.335. There is a clear cut-off after the eighth CCC on the significance, and the first 8 CVs are taken as the estimated states. To detect how many delays exist in the system, CCA is

performed on the past inputs (6 lag steps) and the estimated states, and CCCs are 0.9983, 0.9982, 0.9968, 0.9965, 0.9665 and 0.7781. The significance of these CCCs and the coefficient matrix for past inputs indicates that 4 delayed inputs are memorized in the estimated states.

For MISO systems, the results from the CVA_Hf algorithm for the system from u_1 and u_2 to output y_1 are shown as an example. The CCCs for the CCA are 0.9852, 0.8986, 0.8425, 0.6764, 0.2251, 0.2103, 0.2006 and 0.1933. Based on the significance of these CCCs, the system is estimated of order 4. To detect the delays in the system, CCA is performed between the past inputs (3 lag steps of the two inputs) and the estimated states (4 variables). The CCCs are 0.9951, 0.9817, 0.6476 and 0.4600. The first two CCCs are very close to 1.0, and this indicates that two estimated states mainly come from these past inputs. The coefficient matrix for past inputs is (6 rows correspond to $u_{1,k-1}$, $u_{2,k-1}$, $u_{1,k-2}$, $u_{2,k-2}$, $u_{1,k-3}$ and $u_{2,k-3}$ respectively):

-0.0223	-0.0232	-0.0030	-0.0027
-0.0075	0.0098	-0.0154	0.0107
-0.0187	0.0230	0.0047	-0.0082
-0.0003	-0.0002	-0.0053	0.0193
-0.0013	-0.0006	0.0265	0.0135
-0.0002	0.0006	-0.0047	0.0182

The first two CVs are essentially only contributions from $u_{1,k-1}$, $u_{2,k-1}$ and $u_{1,k-2}$. This indicates that $u_{1,k}$ has 2 delays and $u_{2,k}$ has one delay for the output. The delay structure is correctly detected by CCA between past inputs and the estimated states.

For MIMO systems, the estimated system order is the sum of the true order and the maximum number of delays of all inputs, not the sum of all delay numbers of different inputs. The estimated states memorize the linear combinations of the delayed inputs, instead of the individual delayed inputs (see (7.2.4)). This can be seen more clearly in this example (for simplicity, using deterministic part only). The 5 steps of future outputs have the following relationships with the past data:

$$y_{1,k}=1.75y_{1,k-1}-0.76y_{1,k-2}+0.8u_{1,k-3}-0.76u_{1,k-4}+0.4u_{2,k-2}-0.32u_{2,k-3}$$

$$y_{1,k+1}=1.75\underline{y_{1,k}}-0.76y_{1,k-1}+0.8u_{1,k-2}-0.76u_{1,k-3}+0.4u_{2,k-1}-0.32u_{2,k-2}$$

$$y_{1,k+2}=1.75\underline{y_{1,k+1}}-0.76\underline{y_{1,k}}+0.8u_{1,k-1}-0.76u_{1,k-2}+\underline{0.4u_{2,k}}-0.32u_{2,k-1}$$

$$y_{1,k+3}=1.75\underline{y_{1,k+2}}-0.76\underline{y_{1,k+1}}+\underline{0.8u_{1,k}}-0.76u_{1,k-1}+\underline{0.4u_{2,k+1}}-\underline{0.32u_{2,k}}$$

$$y_{1,k+4}=1.75\underline{y_{1,k+3}}-0.76\underline{y_{1,k+2}}+\underline{0.8u_{1,k+1}}-0.76\underline{u_{1,k}}+\underline{0.4u_{2,k+2}}-\underline{0.32u_{2,k+1}}$$

The double-underlined output terms are the previous outputs, and underlined input terms are eliminated by removing the effects of future inputs (e.g., in CVA_Hf algorithm). It is clear that both $y_{1,k}$ and $y_{1,k+1}$ are completely explained by the past data. With the additional linear combination $0.8u_{1,k-1}-0.76u_{1,k-2}-0.32u_{2,k-1}$, output $y_{1,k+2}$ can be completely explained. With the additional variable $u_{1,k-1}$, output $y_{1,k+3}$ can be completely explained. Outputs beyond $y_{1,k+3}$ can be completely explained. It is clear that 4 LVs (linear combinations of the past data) are adequate to explain all the variables in $\boldsymbol{Y}_f$-$\boldsymbol{H}_f\boldsymbol{U}_f$. 2 LVs are necessary for the process states and the 2 extra LVs are for the delays in the MISO system. The delayed inputs show in terms of linear combinations, not individually.

The CVA_Hf algorithm is also applicable for MIMO processes with delays. For example, consider the system from u_1 and u_2 to y_1 and y_2. The order of the system is estimated to be 9, which is just the sum of the estimated orders for output y_1 and y_2. The estimated states include the true system states (5) and 4 linear combinations of past two steps of inputs. The CCCs between past inputs (3 lag steps) and the estimated states are 0.9949, 0.9866, 0.9770, 0.9261, 0.7242 and 0.5213. The corresponding coefficient matrix for past inputs indicates that the delays are from $u_{1,k-1}$, $u_{2,k-1}$, $u_{1,k-2}$ and $u_{2,k-2}$. Some of the listed MIMO systems face difficulties in determining the number of significant CCCs or the major elements in the coefficient matrix. This is partially due to output y_3 having long delays and a fast decaying pole (-0.5). If SNR becomes higher or longer data is used, conclusions become clearer.

As mentioned in Section 7.2.1, the combinations of the delayed inputs in the estimated states for one output may be collinear with those for the other outputs. This happens for the 2 input-3 output MIMO system. This system is estimated to be order of 14, which is less than the sum of estimated orders of the individual outputs (4 for y_1, 5 for y_2 and 7 for y_3). This is confirmed by the simulation for the purely deterministic situation.

7.3 Effects of Model Structure on SIMs

The model structure of a system directly affects the result from SIMs. This section is to analyze the effects of model structure, especially the common dynamics, on the system order and the number of model parameters from SIMs. Simulations are used to show the effects of common dynamics on SIMs.

7.3.1 Analysis of Effect of Model Structure on SIMs

For a SISO process, the order of the identified model by a SIM algorithm, in general, is the sum of the orders for the deterministic part and the stochastic part. The estimated process states (LVs) include both the deterministic states and the stochastic states. If the deterministic and the stochastic parts have common dynamics (poles), these two parts can share the same dynamics represented by the common states variable. Therefore, the system order may be reduced to a lower order rather than the sum of the orders for the two parts individually, and the number of parameters required to represent the system can be reduced automatically in SIM algorithms.

In practice, the situation that deterministic and stochastic parts share common dynamics is not rare since the input signal and the disturbance can easily have common paths through the process. For traditional system identification methods, such as PEM or ML, common dynamics cannot be detected easily in the identification procedure. These methods either employ prior knowledge of common dynamics (poles) to parameterize a simpler model structure, or perform model-order reduction after obtaining the identified model. However, it is a natural process for SIM algorithms to catch the common states in the state estimation by LVMs. In N4SID, RRA tries to catch as much variation of the estimated predictable subspace as possible by as few LVs (states) as possible. In CVA, CCA tries to have as much predictability of the estimated predictable subspace as possible by as few CVs (states) as possible. As a result of the optimality of the LVMs used in state estimation, common variations in the deterministic part and the stochastic

part are automatically represented by one latent variable (estimated state). This leads to more parsimonious models and a smaller variance for the estimated parameter.

For a MIMO system, the states of the whole system in theory include the states for each output. The system order becomes the sum of orders for each output if there are no common states for different outputs. In this situation, the state-space model employed by SIMs does not show any advantage with respect to the total number of parameters required to represent the system.

In practice, it is highly possible for different outputs in a MIMO process to have common dynamics (poles). For example, in the 2 input-2 output CSTR example used in Section 4.5 the mixing dynamics are common to both outputs; and furthermore, the temperature and the concentration in the reactor have mutual interaction. Any perturbation in the temperature will lead to changes in the concentration, which in return will affect the temperature; therefore both outputs have some common dynamics (poles). This kind of situation is common in practical applications, especially for cases where the inputs and outputs associate with the same operational unit, or several units but with strong interactions.

With common dynamics in a MIMO system, SIMs can automatically recognize the common dynamics in the system and combine the common parameters (poles). In other words, SIMs naturally impose the most parsimonious structure for the final model. As a result, SIMs potentially give a lower-order state-space model. The number of parameters becomes less, and this leads to a higher accuracy for the parameters. This result is consistent with the conclusion shown in Box and Draper (1965), that one can better estimate common parameters from multiple outputs than from only one output.

7.3.2 Simulation Studies of Common Dynamics for SIMs

In this subsection, simulations are performed for a SISO and a MIMO process to show that SIMs can summarize the common dynamics in a process and give a better estimation of model parameters.

Common dynamics in a SISO process

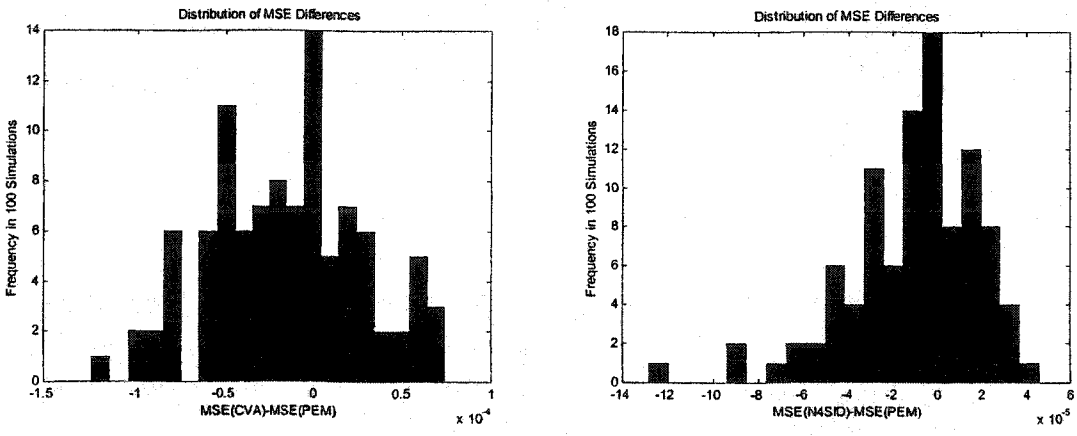
The first example is a simple SISO process with common dynamics in the deterministic part and the stochastic part:

$$y_k = \frac{0.2z^{-1}}{1-0.8z^{-1}}u_k + \frac{1}{1-0.8z^{-1}}e_k \tag{7.2.18}$$

The process has a common pole of 0.8 for both the deterministic and stochastic parts. The SNR is about 10.0 in the Monte Carlo simulation. In each of the 100 simulations, 200 data points are collected.

PEM, CVA_Hf and N4SID algorithms are applied to the simulation data sets. Based on AIC, both CVA_Hf and N4SID algorithms determine the system order to be 1. PEM uses a Box-Jenkins model form (separately parameterized dynamic and noise models), and both the dynamic model and the noise model are assumed to be a first order transfer function. The dynamic poles (mean and standard deviation) from PEM, CVA and N4SID are estimated as 0.7880±0.0374, 0.7889±0.0318 and 0.7914±0.0331 respectively. SIMs give a slightly better estimation of the pole than PEM. Here the accuracy of estimated models are measured by the Mean Squared Error (MSE) of their impulse responses (first 50 steps). The distribution of the paired MSE differences between CVA and PEM is shown in Figure 7.3(a), and that distribution related to N4SID and PEM is shown in Figure 7.3(b). It is clear that both CVA and N4SID show smaller MSE than PEM. The same conclusion can be drawn from Student's *t*-test on these paired MSE differences. The mean and standard deviation of the differences between CVA and PEM are -1.5437×10^{-5} and 4.3523×10^{-6} respectively (Student's *t*-test is -3.55). The mean and standard deviation of the differences between N4SID and PEM are -1.0325×10^{-5} and 2.9305×10^{-6} respectively (Student's *t*-test is -3.52). The critical *t* value is –1.96 for 95% confidence.

This example illustrates that SIM algorithms can perform better than using PEM with independently parameterized dynamics and noise models when, in fact, the models have common dynamics. This results from the SIMs being able to recognize the common pole.



(a) differences between CVA and PEM **(b) differences between N4SID and PEM**

Figure 7.3 Distribution of paired MSE differences for a SISO process

Common dynamics in a MIMO case

The second simulation example is a 5-input 5-output process with ARMAX structure and poles at 0.8 and/or 0.95 for all transfer functions, and some of the transfer functions are null (no effects) or simple first-order systems:

$$y_{1,k} = \frac{2.5z^{-1}}{1-0.8z^{-1}}u_{1,k} + \frac{z^{-1}}{1-0.8z^{-1}}u_{2,k} + \frac{z^{-1}}{1-0.8z^{-1}}u_{4,k} + \frac{1.2z^{-1}}{1-0.8z^{-1}}u_{5,k}$$

$$y_{2,k} = \frac{0.5z^{-1}}{1-0.95z^{-1}}u_{1,k} + \frac{z^{-1}}{1-0.95z^{-1}}u_{2,k} + \frac{z^{-1}}{1-0.95z^{-1}}u_{3,k} + \frac{0.4z^{-1}}{1-0.95z^{-1}}u_{5,k}$$

$$y_{3,k} = \frac{1.5z^{-1}-1.3875z^{-2}}{(1-0.8z^{-1})(1-0.95z^{-1})}u_{1,k} + \frac{z^{-1}-0.875z^{-2}}{(1-0.8z^{-1})(1-0.95z^{-1})}u_{2,k} + \frac{0.5z^{-1}}{1-0.95z^{-1}}u_{3,k} + \frac{0.5z^{-1}}{1-0.8z^{-1}}u_{4,k} + \frac{0.8z^{-1}-0.73z^{-2}}{(1-0.8z^{-1})(1-0.95z^{-1})}u_{5,k}$$

$$y_{4,k} = \frac{2z^{-1}-1.975z^{-2}}{(1-0.8z^{-1})(1-0.95z^{-1})}u_{1,k} - \frac{0.15z^{-2}}{(1-0.8z^{-1})(1-0.95z^{-1})}u_{2,k} - \frac{z^{-1}}{1-0.95z^{-1}}u_{3,k} + \frac{z^{-1}}{1-0.8z^{-1}}u_{4,k} + \frac{0.8z^{-1}-0.82z^{-2}}{(1-0.8z^{-1})(1-0.95z^{-1})}u_{5,k}$$

$$y_{5,k} = \frac{2.2z^{-1}-2.06z^{-2}}{(1-0.8z^{-1})(1-0.95z^{-1})}u_{1,k} + \frac{1.2z^{-1}-1.08z^{-2}}{(1-0.8z^{-1})(1-0.95z^{-1})}u_{2,k} + \frac{0.4z^{-1}}{1-0.95z^{-1}}u_{3,k} + \frac{0.8z^{-1}}{1-0.8z^{-1}}u_{4,k} + \frac{1.12z^{-1}-1.04z^{-2}}{(1-0.8z^{-1})(1-0.95z^{-1})}u_{5,k} \quad (7.2.19)$$

In the simulation, the disturbance to each output has poles at 0.8 and 0.95. The 5 inputs are independent PRBSs with magnitude of 1 and switching time period T_s=5. SNR is around 20 for the process (20.7, 15.7, 23.6, 9.5 and 26.9 for the 5 outputs respectively). In the Monte Carlo simulation, 500 data points are collected in each of the 200 simulations.

For the simulation example, the CVA_Hf algorithm is applied in 5 MISO identifications and a MIMO identification to see the effects of combining common

dynamics in SIMs. Compared to individual MISO identifications, CVA in the MIMO identification can catch the common dynamics in different outputs collectively.

In MISO identification, CVA determines output y_1 and y_2 to be 1st order in all 200 simulations. For y_3, 140 cases are determined to be order 2, the rest to be 1st order (effect of noise and short data length). For y_4, 139 cases are determined to be order 2, the rest to be 1st order. For y_5, 40 cases are determined to be order 2, and 160 cases to be 1st order. In MIMO identification, CVA determines 197 cases to be order 2 (3 cases for order 3). In the majority of cases, CVA gives the correct model order. If the common dynamics were not combined, the system order would be 4 (for y_1 and y_2) or 8 (for y_3, y_4 and y_5), or even higher (in the MIMO case).

The final identification results are compared in the MSE of the impulse responses (first 50 steps). The Student-t tests on the paired MSE differences between MIMO identification and MISO identifications are shown in Table 7.2 (the former is significantly better than the latter with 95% confidence limit if the t statistic is lower than $t_{0.025}$=-1.96). The majority of the transfer functions indicate that the results from MIMO identification are significantly better than those from MISO identifications. The 2 adverse cases are for output y_1 and y_2, which in fact are of 1st order rather than estimated order 2; however, the results for y_3, y_4 and y_5 are greatly improved in MIMO identification than in MISO identification.

Table 7.2 Paired Student-t test on paired impulse response of CVA results (MSE_{MIMO}-MSE_{MISO})

	u_1	u_2	u_3	u_4	u_5
y_1	-5.1242	1.3682	3.9721	-6.1245	-4.5850
y_2	4.0634	-2.6793	0.2840	-2.4603	-2.6361
y_3	-12.7197	-13.6049	-14.1649	-14.0777	-12.9112
y_4	-10.4071	-10.3834	-9.5953	-11.8850	-11.7861
y_5	-22.5117	-32.7109	-36.3997	-23.6584	-18.8407

The estimated poles and their standard deviations (STD) in MISO identifications are also shown in Table 7.2 (only considering cases estimated of order 2 for y_3, y_4 and y_5). The two poles are estimated to be 0.7983±0.0071 and 0.9494±0.0040 in MIMO

identification, which are much better than those from y_3, y_4 and y_5 in MISO identifications, and are similar to those from MISO identification for output y_1 or y_2, where only the information of one pole is provided. Based on the sampled covariance matrix and assuming a normal distribution of the estimated poles, the 95% confidence regions of these estimated poles from individual y's in MISO and jointly in MIMO are shown in Figure 7.4. It is clear that by joining the outputs with common dynamics, CVA in MIMO identification brings a great advantage to the estimation: a more accurate estimation of common parameters in both the mean and the variance.

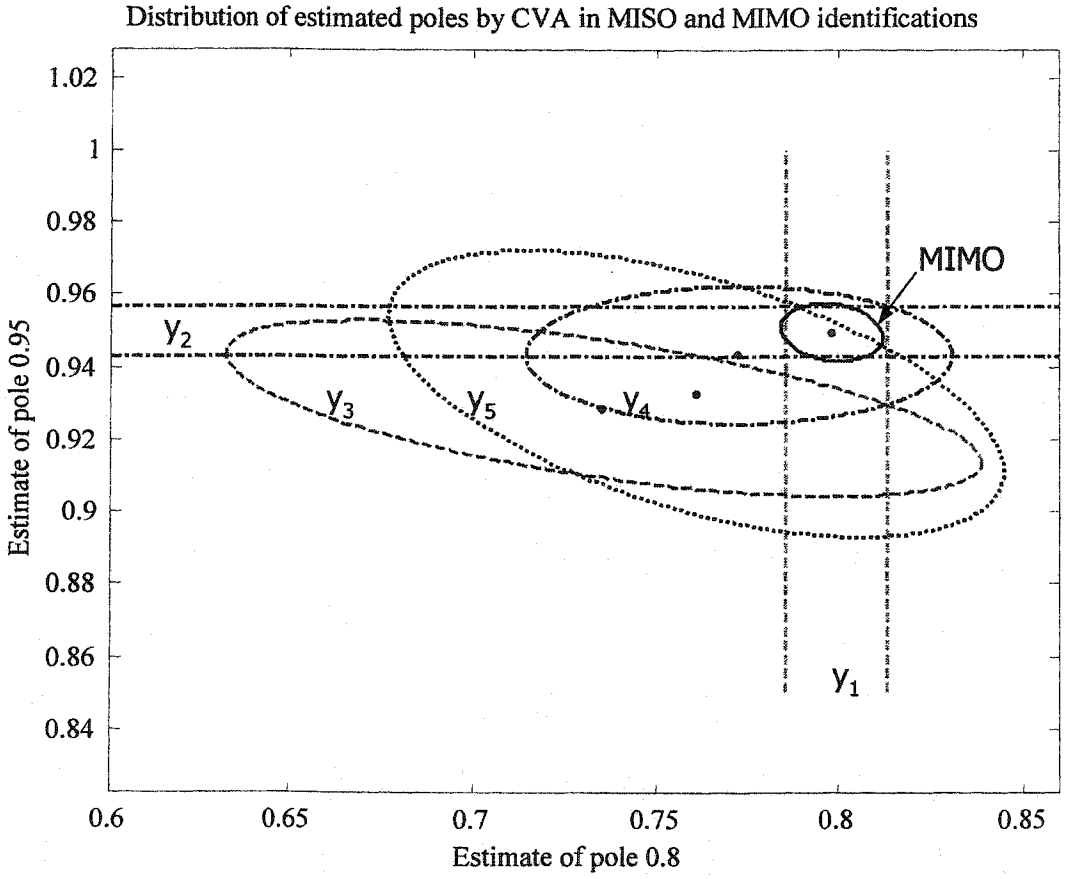


Figure 7.4 Distribution of estimated poles by CVA in MISO and MIMO identifications

Note for lines: MISO: y1 (dashed straight); y2 (dashdot straight); y3 (dashed); y4 (dashdot), y5 (dotted).
MIMO: all y's (solid)

7.4 SIMs for Non-stationary Disturbances

Non-stationary disturbances are common in practical applications, and they have special effects on the identified models from SIMs. This section is to investigate the applicability of SIM algorithms for non-stationary disturbances, analyze the effects of non-stationary disturbances on the modeling results, and propose approaches to deal with the possible problems.

7.4.1 Introduction

A non-stationary disturbance has no fixed value for its mean or variance. A common type of non-stationary disturbance is a random-walk time series (drifting around) with no stationary level for the mean. Nearly non-stationary disturbances are common in practice, especially in chemical processes. Another type of non-stationary disturbance has non-stationary variance, i.e., the variance is a function of the mean level of the disturbance or the variance changes with time. Box and Cox Transformation can be used to stabilize the variance (refer to Taylor, 1993).

The random-walk type of non-stationary disturbance is considered in this section. This kind of disturbance has unity poles in its model and infinitely large variance in theory. In practice, the disturbance variance is extensive, and this leads to large regression error if one is fitting the data to a FIR model. Differencing the data is a traditional way to deal with non-stationary disturbances. Data differencing makes fitting FIR models feasible. It is, in fact, a data pre-filtering procedure, and this procedure is fully based on the prior knowledge of non-stationarity in the disturbance. This procedure decreases the disturbance variance, but it usually decreases the variance of the input signals as well (e.g., PRBS becomes a series of spikes after differencing).

A high-order ARX model can catch the dynamics in a non-stationary disturbance. The unit pole of the stochastic part is also modeled into the ARX model. In order to get a parsimonious and stable process model, the fitted ARX model usually has to perform

pole-zero cancellation (near unity). Otherwise, the slight error in the unit pole may drive the identified ARX model unstable.

7.4.2 SIMs for Non-stationary Disturbance

Before considering the combined deterministic-stochastic system, a relatively simpler pure stochastic system with non-stationary disturbance will be analyzed. The simplest non-stationary disturbance is a random walk process represented by the following model with a unit pole:

$$y_k = \frac{1}{1-z^{-1}} e_k \tag{7.4.1}$$

where e_k is a white noise signal. In the time domain, this process has the following relationships for the future outputs:

$$\begin{aligned} y_k &= y_{k-1} + e_k \\ y_{k+1} &= y_{k-1} + e_k + e_{k+1} \\ &\cdots\cdots \\ y_{k+f-1} &= y_{k-1} + \sum_{j=0}^{f-1} e_{k+j} \end{aligned} \tag{7.4.2}$$

This means that the immediate past output y_{k-1} is the best prediction of the future outputs, and is a state variable for current time. In both the CCA and N4SID procedures, the past data (y_{k-1}) can explain all the variation of the future outputs except the future stochastic signals. Therefore y_{k-1} can be picked up easily as the estimated state, and the unit pole is included in the final state-space model.

If a non-stationary disturbance is a combination of mean shifting/trend shifting and other stationary noise (AR or ARMA model), the only difference from (7.4.2) is the involvement of more steps of past data in the estimation of stochastic states. In general, along with other states, the non-stationary states can be estimated naturally and automatically in SIMs.

For the combined deterministic-stochastic system, SIMs can easily pick up both the deterministic states and the stochastic state by LVs from the past data [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$] and the estimated predictable subspace $\boldsymbol{Y}_{f\_e}=\boldsymbol{Y}_f-\boldsymbol{H}_{f\_est}\boldsymbol{U}_f$. The effects of the future inputs $\boldsymbol{H}_{f\_est}\boldsymbol{U}_f$ do not involve the stochastic part; therefore $\boldsymbol{Y}_{f\_e}$ includes the effects of the future outputs of the non-stationary disturbance. Suppose the non-stationary disturbance is a random-walk process, then the stochastic state is the stochastic part of y_{k-1}, which can be well predicted by the linear combination of the past data up to time point k-2. Therefore, as linear combinations of the past data [$\boldsymbol{Y}_p$; $\boldsymbol{U}_p$], LVs from CCA or RRA will include both deterministic states and the non-stationary stochastic state to have the best prediction of $\boldsymbol{Y}_{f\_e}$. With other types of non-stationary disturbances, SIMs can also estimate the non-stationary states together with other states naturally and automatically.

Since the deterministic states and the non-stationary stochastic state are estimated simultaneously in SIM algorithms, the final identified state-space model will include both the stable poles and the unit pole. Due to estimation errors, the unit pole may be slightly different from 1.0. If the estimated pole is slightly smaller than 1.0, the identified model is still stable. However, the transfer function for the dynamic process involves non-perfect pole-zero cancellation. Since the estimated pole and corresponding zero are very close to 1.0, this non-perfect zero-pole cancellation could greatly distort the response on the very-low frequency region and the steady state gain (there is a long tail in the impulse response, i.e., not returning to zero until long lag steps). For example, if the unit pole and corresponding zero are estimated as 0.9999 and 0.9990 respectively, the steady state gain will be increased to 10 folds of the original values ((1-0.9990)/(1-0.9999)=0.0010/0.0001=10)! If the zero is estimated as 1.0015, the estimated steady state gain will be 15 folds of the value gain and with an opposite sign (the result always is of opposite sign if the estimated zero and the estimated pole are in the opposite site of 1.0). If this model is used for the design of the controller, the closed system will be unstable. If the pole-zero cancellation were not close to 1.0, the error would usually be small enough to ignore. If the pole were 0.8 as in the above example with the same magnitude of estimation error (0.7999 and 0.7990 for the pole and zero respectively), the

corresponding error on the steady state gain would be only 0.45% ((1-0.7990)/(1-0.7999)=0.2010/0.2001 ≈1.0045). Because of the presence of the unit pole in a process, the identified model is greatly affected by the non-perfect zero-pole cancellation around this pole. In addition, if the estimated pole is slightly larger than 1.0 due to the estimation error, the identified model will be unstable.

Several methods can fix the above non-perfect pole-zero cancellation problem, e.g., deliberately drop off those two terms in the transfer function in zero-pole form. For the estimated unstable pole, a simple method is available to stabilize the identified state-space model: perform eigenvalue decomposition of the fitted system matrix A, the eigenvalue closest to 1.0 is set to 1.0 and then back calculate a new matrix A by the modified eigenvalues and the original eigenvectors. With the prior knowledge of the non-stationary disturbance, SIMs can also be applied to the data after differencing.

7.4.3 Simulation Examples

In this subsection, CVA and N4SID methods are applied to simulation examples with non-stationary disturbances, focusing on the effects of the non-stationary disturbance and the non-perfect pole-zero cancellation problem.

SIMs for a non-stationary time series

A simple non-stationary time series process is simulated to illustrate the capability of SIMs to catch the non-stationary states. The example is a random-walk process as shown in (7.4.1). In the simulation, the variance of e_k is 0.01 and 5000 data points are collected. Both past and future horizons are of 3 lag steps, and future data and past data only consist of the output variable.

In CVA, the three CCCs from CCA on $\boldsymbol{Y}_p$ and $\boldsymbol{Y}_f$ are 0.9997, 0.0140 and 0.0136 respectively. This gives a clear cut-off for determining the process to be of order 1. The first CV from past data has almost 100% prediction of the first CV from the future data.

The first CV is taken as the estimated state, and the pole of the resultant model from CVA is 0.9999.

In N4SID, RRA is performed on the past outputs and the future outputs. The first LV from the past data can explain (predict) 99.89% the variance of the future output, and the remaining two LVs essentially predict nothing of the future outputs. This also gives a clear determination of the process to be of order 1. The first LV is taken as the estimated state, and the pole of the fitted model from N4SID is 0.9999.

N4SID for a process with a non-stationary disturbance

The simulation example for this case has the following state-space model form:

$$\begin{cases} x_{k+1} = 0.8x_k + u_k + \eta_k \\ y_k = x_k + \eta_k \\ \eta_k = \eta_{k-1} + e_k \end{cases} \tag{7.4.3}$$

The pole for the deterministic part is 0.8 and the steady state gain is 5.0. Here e_k is white noise and $\eta_k = e_k/(1-z^{-1})$ is a non-stationary disturbance (random walk, or integrated white noise). The process has the following transfer function model:

$$y_k = \frac{z^{-1}}{1-0.8z^{-1}}u_k + \frac{1+0.2z^{-1}}{(1-0.8z^{-1})(1-z^{-1})}e_k \tag{7.4.4}$$

In the simulation, input u_k is a random binary signal (RBS) with magnitude of 1.0, var(e_k)=1.0 and 500 data points are collected.

Directly use N4SID (M file in System Identification Toolbox of MATLAB) for the collected data with lag steps of 4 for both the future and past horizons. The order is correctly determined to be 2, and the identified model has the following transfer function:

$$y_k = \frac{1.0262z^{-1} - 1.0222z^{-2}}{1-1.8117z^{-1}+0.8116z^{-2}}u_k + \frac{1+0.1498z^{-1}+0.0611z^{-2}}{1-1.8117z^{-1}+0.8116z^{-2}}e_k \tag{7.4.5}$$

The zero for the deterministic part is 0.9961, and the two poles are 0.8114 and 1.0003 respectively. The direct result of the deterministic part is far from the true dynamic model for including an unstable pole. Estimation errors cause the stochastic pole a slight deviation from 1.0 to be unstable. The effects of the unstable pole are ignorable in the

future horizon of 4 steps but are unacceptable in predicting a long future horizon. In fact, N4SID catches both the deterministic and stochastic states. The impulse response of the dynamic part is shown in Figure 7.5. The result is close to the true impulse response, especially in the initial period. However, there is a long tail not returning to zero (making the estimated steady state gain infinity). The problem is not due to the N4SID algorithm itself, but is due to the non-perfect pole-zero cancellation near the unit pole (See Section 7.4.2). Once the real problem is recognized, the dynamic model can be approximated to the following transfer function by simply dropping the pole and zero terms to be cancelled:

$$y_k = \frac{1.0262z^{-1}}{1-0.8114z^{-1}}u_k \tag{7.4.6}$$

This model is stable with pole at 0.8114 and steady state gain of 5.441. The impulse response of this model is shown in Figure 7.5, and it is closer to the true response and returns to zero at the tail part. One may use other model order reduction techniques to get a better approximate model.

If N4SID is used for the data after differencing, the model order is determined to be 1, and the resultant model has the following dynamic transfer function:

$$y_k = \frac{1.0245z^{-1}}{1-0.8162z^{-1}}u_k \tag{7.4.7}$$

The impulse response of this model is also shown in Figure 7.5. The result is stable, but it is slightly worse than the model (7.4.6) from directly applying N4SID to the original data followed by a pole-zero cancellation (refer to Figure 7.5(b)).

The same simulation example was used in literature (Amirthalingam, et al., 1998, 2000) where SIMs were alleged to require the stochastic part of the system to be stationary. In fact, SIMs do well in catching the non-stationary state (and the unit pole). The only problem is the non-perfect zero-pole cancellation near the unit pole in the process dynamic model. After paying attention to this problem, SIMs can give accurate results for data with non-stationary disturbances.

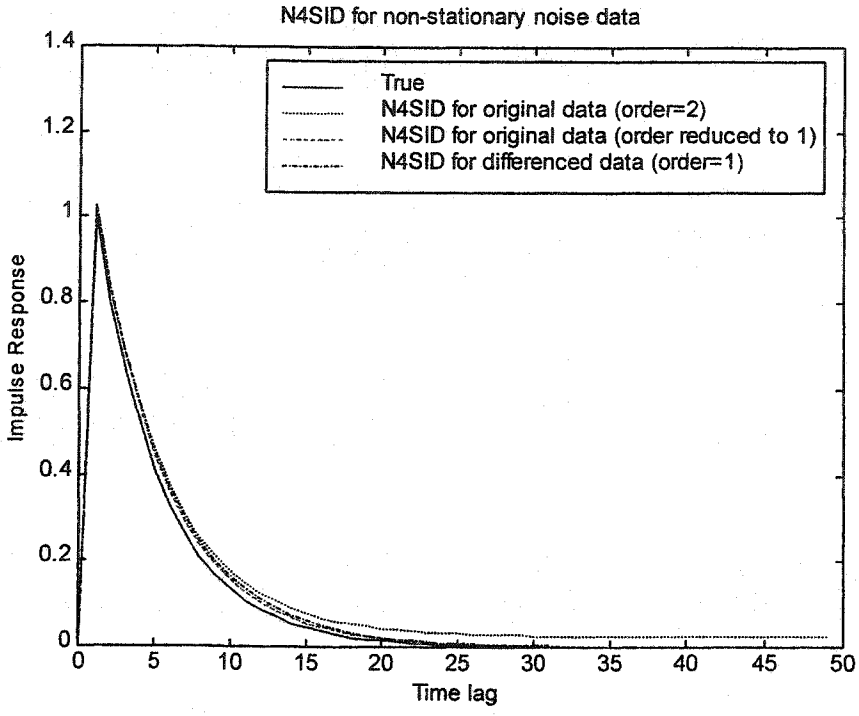


Figure 7.5(a) Impulse responses from N4SID for non-stationary data

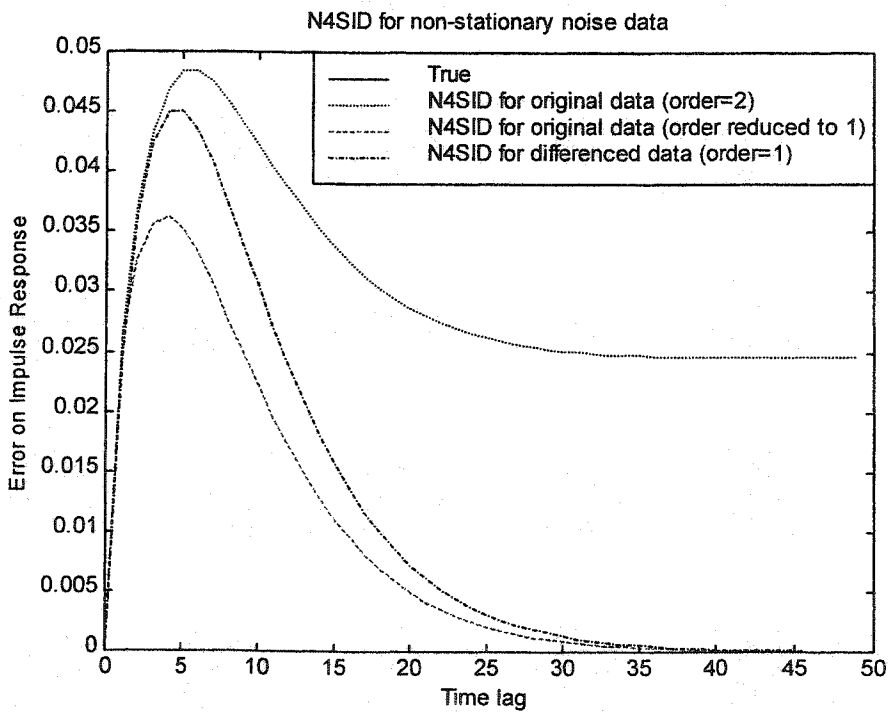


Figure 7.5(b) Error on impulse responses from N4SID for non-stationary data

CVA for a process with a non-stationary disturbance

Since a CVA_Hf algorithm is based on using the impulse weights from a fitted ARX model, an ARX model is applied to the simulation data first. Two lag steps are used, and the resultant ARX model is:

$$y_k = 1.8657y_{k-1} - 0.8658y_{k-2} + 1.0309u_{k-1} - 1.0828u_{k-2} + \varepsilon_k \qquad (7.4.8)$$

The poles for this model are 0.9989 and 0.8668 and zero is 1.0504. The steady state gain is -341.5, far from the true value. However, the first 6 impulse weights are 0, 1.0309, 0.8406, 0.6757, 0.5328 and 0.4091 respectively, and they are close to those true impulse weights 0, 1.0, 0.8, 0.64, 0.512 and 0.4096, though the fitted ARX model has a long tail in the impulse response as shown in Figure 7.6. The long tail can be removed by just dropping the zero and pole terms to be canceled, and the resultant model is:

$$y_k = 0.8668y_{k-1} + 1.0309u_{k-1} + \varepsilon_k \qquad (7.4.9)$$

Its impulse response is also shown in Figure 7.6, and it has a greater error in the fast changing period due to the poor estimated pole. Other model-order reduction techniques may lead to better results.

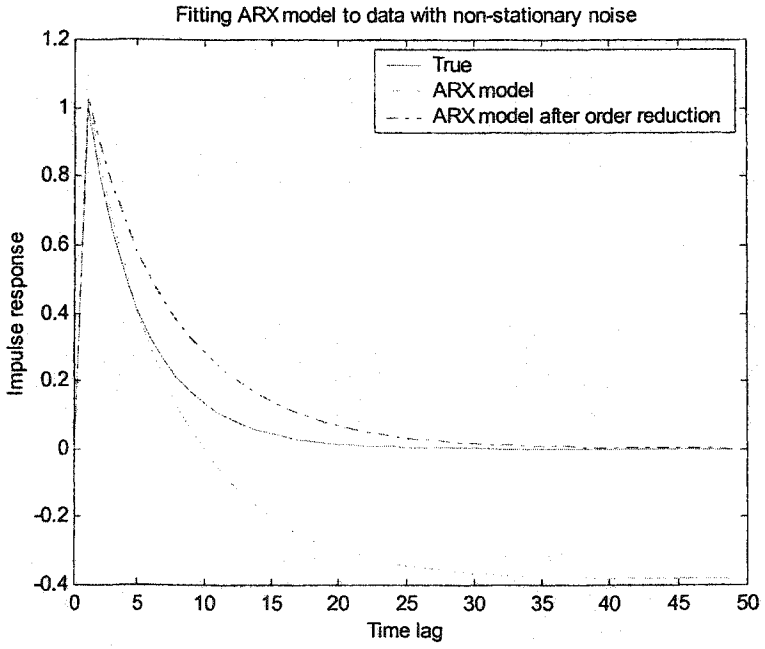


Figure 7.6 ARX model from fitting non-stationary disturbance data

The CVA_Hf algorithm is then applied to the simulation data with 4 lag steps for both the future and past horizons. The impulse weights used to construct $\boldsymbol{H}_f$ are based on the fitted ARX model above (without model-order reduction). The CCCs from CCA are 1.0000, 0.7369, 0.1687 and 0.1037 respectively. The order is determined to be 2 by AIC. The identified model has the following transfer function:

$$y_k = \frac{1.1175z^{-1} - 1.0605z^{-2}}{1 - 1.8149z^{-1} + 0.8149z^{-2}} u_k + \frac{1 + 0.1498z^{-1} + 0.0611z^{-2}}{1 - 1.8149z^{-1} + 0.8149z^{-2}} e_k \qquad (7.4.10)$$

The two poles are 0.9998 and 0.8151 respectively (not 1.0 and 0.8149 from the presented denominator due to the number of significant digitals), and the zero for the deterministic part is 0.9489. The steady state gain is 1560.4 if calculated directly from the above model (larger if compared to that from N4SID since the pole is closer to 1.0 and the zero is farther from 1.0 in the non-perfect zero-pole cancellation). The impulse response is shown in Figure 7.7, and it is far from the true response. The dynamic model has the following transfer function by simply dropping the pole and zeros terms to be cancelled:

$$y_k = \frac{1.1175z^{-1}}{1 - 0.8151z^{-1}} u_k \qquad (7.4.11)$$

This model is stable with the pole at 0.8151 and a steady state gain of 6.044. Its impulse response is shown in Figure 7.7, and it is much closer to the true than that from (7.4.10). Other model-order reduction techniques may lead to better results.

If the CVA_Hf algorithm is used for the data after differencing, past horizon and future horizons are determined to be 3 lag steps by the minimum AIC. The canonical correlation coefficients from CCA are 0.9035, 0.1354 and 0.0977 respectively. The order is determined to be 1 by AIC. The identified model has the following dynamic transfer function (the impulse response is shown in Figure 7.7, worse than the model (7.4.11)):

$$y_k = \frac{1.0605z^{-1}}{1 - 0.8343z^{-1}} u_k \qquad (7.4.12)$$

These simulations show that SIMs can be applied to processes with non-stationary disturbances, and reasonably good dynamic transfer functions can be obtained after pole-zero cancellations.

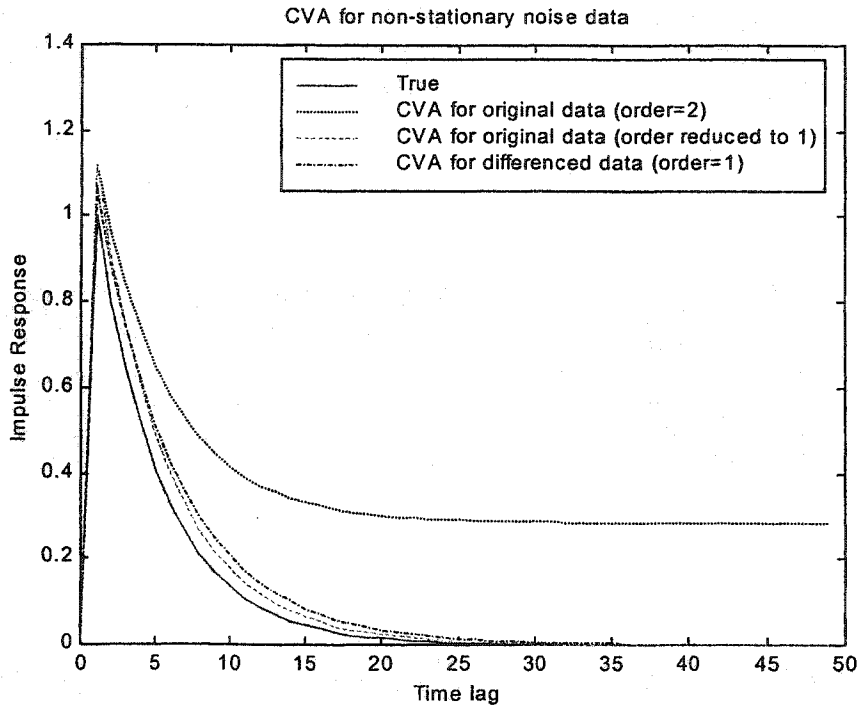


Figure 7.7 (a) Impulse responses from CVA for non-stationary data

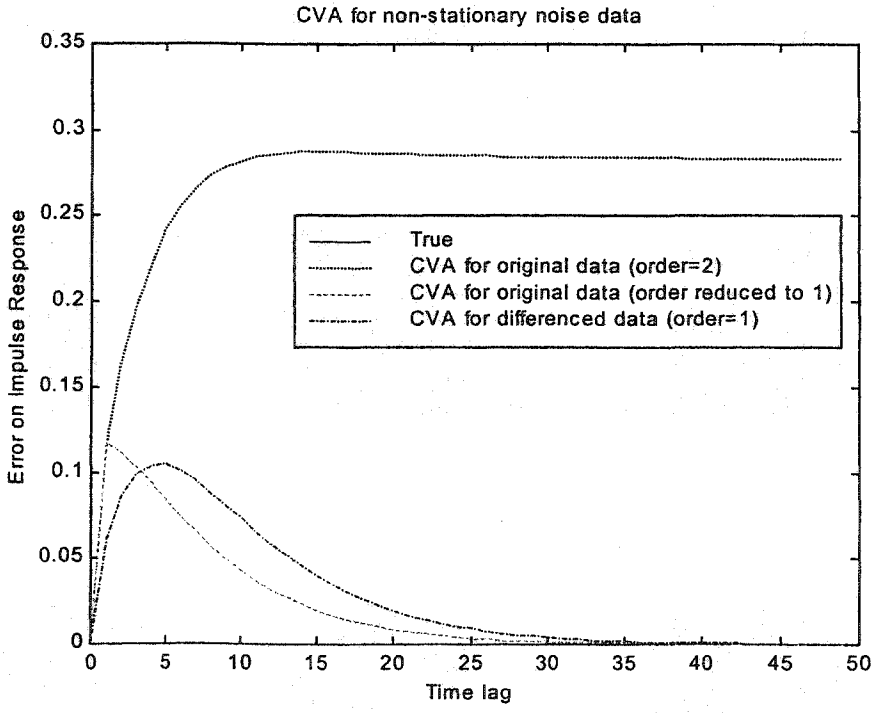


Figure 7.7(b) Error on impulse responses from CVA for non-stationary data

7.5 SIMs for Processes with Co-integrating Disturbances

This section is to investigate the applicability of SIMs for co-integrating disturbances and the effects on the identified models. Simulations are used to illustrate the analysis in this section.

7.5.1 Analysis of Effects of Co-integrating Disturbances

Co-integrating disturbances occur when multiple outputs are affected by the same non-stationary disturbance. This kind of disturbance is common in practice, especially in chemical processes; for example, a non-stationary disturbance in raw material properties or in the feed flow will affect many process outputs.

In MISO identification for each output, the disturbance is considered to be a non-stationary one without considering the common source of these non-stationary disturbances. From the viewpoint of SIM algorithms, the nature of co-integrating disturbances is nothing else than having common non-stationary dynamics in the stochastic parts of different outputs. Its effect on more than one output should enhance the information of the stochastic state in both $\boldsymbol{Y}_p$ and $\boldsymbol{Y}_f$ (therefore $\boldsymbol{Y}_{f\_e}$), and therefore should improve the accuracy of the estimated non-stationary state. This is again analogous to the idea shown in Box and Draper (1965) that common parameters are better estimated from several outputs than from only one output.

SIM algorithms can also be used for the data with co-integrating disturbances after differencing, and there will be no non-stationary state in the differenced data. However, this presumes prior knowledge about the existence of non-stationary or co-integrating disturbances. There is the possibility of over-differencing for several cases: i) the noise part is a non-stationary disturbance plus other general noise, such as white noise, which will have doubled variance after differencing; ii) differencing all outputs where only some of the outputs are affected by the co-integrating disturbance; iii) the

output looks like it is affected by the non-stationary disturbance while the true pole for the noise is close but not equal to the unit, such as 0.97 or 0.98, and there will still be some correlation in the data after differencing. In these cases, over-differencing the outputs could lead to relatively poor estimates of their dynamic responses. By directly applying to the original data, SIM algorithms do not need any prior knowledge and can avoid the problem of over-differencing.

7.5.2 Simulation Studies

Process with only integrating disturbances

The first simulation example is a simple 1-input 2-output process with a random-walk disturbance added onto both outputs. The process has the following transfer function model:

$$\begin{cases} y_{1,k} = \dfrac{0.2z^{-1}}{1-0.8z^{-1}}u_k + \dfrac{1}{\left(1-z^{-1}\right)}e_k \\ y_{2,k} = \dfrac{0.05z^{-1}}{1-0.95z^{-1}}u_k + \dfrac{1}{\left(1-z^{-1}\right)}e_k \end{cases} \tag{7.5.1}$$

The input to the process is a PRBS with magnitude of 4 and switching time period T_s=5. 1000 data points are collected. The variance of e_k is chosen to have about equal values of mean squares between the dynamic part and the stochastic part over the collected time period.

The CVA_Hf algorithm is used for the above simulation data with both outputs. The past and future horizons are determined to be of 3 lag steps by AIC. The CCCs from CCA are 1.0, 1.0, 0.9979, 0.8912, 0.0901 and 0.0483 respectively. The system order is determined to be 3 by minimum AIC, although the 4th CCC is also very large. The transfer function in zero-pole form for the dynamic part is:

$$\begin{cases} y_{1,k} = \dfrac{0.2068\, z^{-1}\left(1-0.9487\, z^{-1}\right)\left(1-z^{-1}\right)}{\left(1-0.8z^{-1}\right)\left(1-0.95\, z^{-1}\right)\left(1-0.9983\, z^{-1}\right)}u_k \\ y_{2,k} = \dfrac{0.0568\, z^{-1}\left(1-0.8094\, z^{-1}\right)\left(1-1.0092\, z^{-1}\right)}{\left(1-0.95\, z^{-1}\right)\left(1-0.8z^{-1}\right)\left(1-0.9983\, z^{-1}\right)}u_k \end{cases} \tag{7.5.2}$$

If zero-pole cancellation is performed for the above model, the result is close to the true process model.

The canonical correlation between the three estimated states and the three true states are essentially all 1.0, and the poles are 0.8000, 0.9500 and 0.9983. These results are better than those from the CVA_Hf algorithm for individual outputs as shown in Table 7.3, especially the estimated states and the estimated unit pole in co-integrating case are improved compared to those estimated in case of individual output. All these show the positive contribution from the common dynamics in the co-integrating disturbance as analyzed in Section 7.5.1. This example demonstrates that CVA is applicable for co-integration problems, and the result is improved by using all outputs for better estimation of the common non-stationary stochastic model.

It is interesting to see that the residuals of the two outputs in fitting separate ARX models are collinear in this simulation example. This is due to the disturbances in the two outputs being driven by the same stochastic signal e_k. This also leads to the large value for the 4th CCC in the above state estimation in applying CVA. This point is also verified by adding additional white noise with variance of 0.01 onto y_2. In this situation, the first 3 CCCs are almost unchanged, but the other 3 CCCs are now between 0.40 and 0.36.

Table 7.3 CVA_Hf algorithm for co-integrating disturbances

System	Estimated order	CCCs between x_{k_est} and true x_k	Estimated poles
u-y1	2	1.0, 0.9991	0.7992, 0.9979
u-y2	2	1.0, 0.9392	0.9261, 0.9935
u-y1,y2	3	1.0, 1.0, 1.0	0.8000, 0.9500, 0.9983

The N4SID algorithm is also applied for the simulation data with 3 lag steps. The singular values for PCA in N4SID are 71.6210, 13.7007, 3.5342, 0.1070, 0.0598 and 0.0. This indicates the process of order 3 with none of the ambiguity that existed in CVA. This is because N4SID looks at the variance of the outputs as well as the correlations.

The identified model has poles at 0.7825, 0.9585 and 0.9981 respectively, and the transfer function for the dynamic part is:

$$\begin{cases} y_{1,k} = \dfrac{0.2085z^{-1}\left(1-0.9472z^{-1}\right)\left(1-z^{-1}\right)}{\left(1-0.7825z^{-1}\right)\left(1-0.9585z^{-1}\right)\left(1-0.9981z^{-1}\right)}u_k \\ y_{2,k} = \dfrac{0.0552z^{-1}\left(1-0.7938z^{-1}\right)\left(1-1.0088z^{-1}\right)}{\left(1-0.7825z^{-1}\right)\left(1-0.9585z^{-1}\right)\left(1-0.9981z^{-1}\right)}u_k \end{cases} \tag{7.5.3}$$

This model is very close to the true process model if zero-pole cancellation is performed on the above form. The CCCs between the estimated states and the true states are 1.0, 1.0 and 0.9991 (slightly worse than those from the CVA method). This result is better than the CCCs (0.9999 and 0.9989) for applying N4SID for only one output (y_1). All these indicate that N4SID is also applicable for co-integrating disturbances, and the result is improved by using all outputs to estimate the common dynamics in the co-integrating noise.

Process with co-integrating and additional disturbances

The second simulation example is similar to the previous one, but a third output is added with the co-integrating disturbance filtered by an AR(1) process, and all outputs have additional independent white noise signals (variance of 0.01). The process has the following model form:

$$\begin{cases} y_{1,k} = \dfrac{0.2z^{-1}}{1-0.8z^{-1}}u_k + \dfrac{1}{\left(1-z^{-1}\right)}e_k + a_{1,k} \\ y_{2,k} = \dfrac{0.05z^{-1}}{1-0.95z^{-1}}u_k + \dfrac{1}{\left(1-z^{-1}\right)}e_k + a_{2,k} \\ y_{3,k} = \dfrac{0.11z^{-1}-0.1z^{-2}}{\left(1-0.8z^{-1}\right)\left(1-0.95z^{-1}\right)}u_k + \dfrac{0.2}{\left(1-z^{-1}\right)\left(1-0.7z^{-1}\right)}e_k + a_{3,k} \end{cases} \tag{7.5.4}$$

The same PRBS is used for the input, and 1000 data points are collected.

The CVA_Hf algorithm is applied to the simulation data, and the optimal lag steps for past and future horizons are determined to be 10 by AIC. The first 9 CCCs are 0.9997, 0.9980, 0.9947, 0.5893, 0.3520, 0.3383, 0.2852, 0.2783 and 0.2610 respectively. The first 4 CCCs are significantly larger than the others, and the process is determined

correctly to be order of 4 by AIC. The CCCs between the estimated states and the true states are 1.0, 0.9998, 0.9983 and 0.9442. This indicates accurate estimates of the process states. The identified model has 4 poles at 0.8011, 0.9476, 0.9981 and 0.7300 respectively, and the dynamic part of the identified model has the following transfer function in zero-pole from:

$$\begin{cases} y_{1,k} = \dfrac{0.2054z^{-1}\left(1-0.9484z^{-1}\right)\left(1-0.7264z^{-1}\right)\left(1-z^{-1}\right)}{\left(1-0.8011z^{-1}\right)\left(1-0.9476z^{-1}\right)\left(1-0.73z^{-1}\right)\left(1-0.9981z^{-1}\right)}u_k \\ y_{2,k} = \dfrac{0.0561z^{-1}\left(1-0.8215z^{-1}\right)\left(1-0.7204z^{-1}\right)\left(1-z^{-1}\right)}{\left(1-0.8011z^{-1}\right)\left(1-0.9476z^{-1}\right)\left(1-0.73z^{-1}\right)\left(1-0.9981z^{-1}\right)}u_k \\ y_{2,k} = \dfrac{0.1113z^{-1}\left(1-1.9138z^{-1}+0.9169z^{-2}\right)\left(1-0.7134z^{-1}\right)}{\left(1-0.8011z^{-1}\right)\left(1-0.9476z^{-1}\right)\left(1-0.73z^{-1}\right)\left(1-0.9981z^{-1}\right)}u_k \end{cases} \tag{7.5.5}$$

It is close to the true dynamic model in (7.5.4) after a pole-zero cancellation. These results show that CVA works well with co-integrating disturbances and other disturbances.

N4SID was also applied for the simulation data. The process is determined to be order of 4, and the CCCs between estimated states and the true states are 1.0, 0.9993, 0.9983 and 0.7767 respectively. This indicates that 3 states are estimated accurately and the other one is not so well estimated but is still acceptable. The identified model has poles at 0.8116, 0.9451, 0.9977 and 0.6992, close to the true poles. The dynamic part of the identified model has the following transfer function:

$$\begin{cases} y_{1,k} = \dfrac{0.2027z^{-1}\left(1-0.9430z^{-1}\right)\left(1-0.7057z^{-1}\right)\left(1-z^{-1}\right)}{\left(1-0.8116z^{-1}\right)\left(1-0.9451z^{-1}\right)\left(1-0.6992z^{-1}\right)\left(1-0.9977z^{-1}\right)}u_k \\ y_{2,k} = \dfrac{0.0537z^{-1}\left(1-1.4767z^{-1}+0.5481z^{-2}\right)\left(1-1.0261z^{-1}\right)}{\left(1-0.8116z^{-1}\right)\left(1-0.9451z^{-1}\right)\left(1-0.6992z^{-1}\right)\left(1-0.9977z^{-1}\right)}u_k \\ y_{3,k} = \dfrac{0.1138z^{-1}\left(1-0.8602z^{-1}\right)\left(1-0.7363z^{-1}\right)\left(1-1.0238z^{-1}\right)}{\left(1-0.8116z^{-1}\right)\left(1-0.9451z^{-1}\right)\left(1-0.6992z^{-1}\right)\left(1-0.9977z^{-1}\right)}u_k \end{cases} \tag{7.5.6}$$

The above model is very close to the true dynamic model in (7.5.4) after a model-order reduction. These results indicate that N4SID also works well with co-integrating disturbances and other disturbances.

7.6 Application Guidelines for SIMs

Based on the analysis in this chapter and the results from previous chapters, SIMs clearly show great advantages for practical applications with the following features:

- No prior knowledge about delays in the process is available. SIMs can incorporate the delayed inputs into the estimated process states automatically; therefore the delays will be included in the final model with a higher order. Separation of the delayed inputs from the estimated states is feasible if a lower order model is desirable.
- No prior knowledge about the system order is available. In SIMs, the estimated states (LVs from LVMs) are ranked in their importance, and the system order can be easily determined by some criteria based on their predictabilities (e.g., AIC).
- There are common dynamics (common poles) in the process. SIMs are able to catch these unknown common dynamics and give a better estimation of the common poles.
- There is non-stationary or co-integration disturbance in the process. SIMs can easily pick up the non-stationary state in the process.
- There are multiple outputs in the process. SIMs are particularly well suited for MIMO systems. They can also deal with the collinear problem in the outputs.
- There is feedback in the process. With the proper choice of algorithms, SIMs can be applied to closed-loop data,

In practice, if the process has one or more of the above features, SIMs will be a good choice for dynamic process modeling. Considering the characteristics of SIMs shown in Section 2.5, the following are some application directions where SIMs can exhibit their advantages and avoid their shortcomings:

- A quick method with little input from the user is desirable. SIMs can be applied to dynamic process modeling with little prior knowledge and can give a reasonably good model quickly for application. If a more accurate model is required, one can use the model structure and model parameters provided by SIMs as initial values for other system identification methods, such as PEM, for a better parameter estimation with the price of non-linear optimization (refer to Ljung, 1997).

- A model identified for model-based controller design, especially for model predictive control (MPC) is desired. The final model from SIMs is in state-space form; in this model form it is easy to obtain the multiple-step-ahead predictions of the future outputs, which is a key component for MPC. In fact, SIM and MPC are a dual problem. They involve a prediction of the multi-step future outputs based on the multi-step past data as well as the multi-step future inputs. In SIMs, the future inputs are known and the model is to be determined; while in MPC, the model is known and the future inputs are to be determined. Some software packages, such as SMOC (Shell Model-based Optimal Controller) and ASPEN Target, use state-space models as a basis for MPC. The strengths and weaknesses of MPC based on a state-space model have been discussed in literature (e.g., Vogel and Downs, 2001).

- SIMs should be suitable for on-line application, such as adaptive control, because SIMs are reliable and stable in numerical implementation with predictable computation load and without involving a nonlinear search (Ljung, 1999; Viberg, 1995). One such application has been reported for on-line adaptive control of unstable aircraft wing flutters (Peloubet, et al., 1990).

- The models from SIMs can be used as soft sensors. SIMs can give the final state-space model in Kalman filter form, and can be used to infer any unmeasured outputs or properties (assuming they were available in the model-building phase), or to update the future predictions based on past measurements.

- The model from SIMs can be used for process monitoring. The model from SIMs provides information about the process states, though they may or may not correspond to physical variables measured from the process; and these estimated states can be used to monitor the status changes in a process. They are also suitable for detecting changes in the process dynamics based on the causal model from experiments. On the other hand, PCA and PLS are usually based on historical data and do not need experiments, and they show more advantages in detecting the change in correlation structure in the process inputs or disturbances. Combining SIMs with PCA/PLS will be a promising approach for process monitoring.

These are some general guidelines for practical applications of SIM, especially for dynamic process modeling, control design and process monitoring. There might be many other applications where SIMs can give satisfactory results.

8 Conclusions and Open Issues

The first part of this thesis (Chapters 3 through 5) focuses on fundamental research on Subspace Identification Methods (SIMs), including the basic principles and properties (such as bias) of the algorithms, their relationships to other methods and the relationships among themselves. The second part of this thesis (Chapters 6 through 7) concentrates on applications, including SIMs for closed-loop data and some issues arising from practical applications.

This chapter will first recap the general conclusions drawn from the research completed in this thesis, and will then point out some open issues to be explored in future research.

In Chapter 3, a multi-step state-space model was set up first as a uniform analytical basis for analyzing all SIM algorithms. This model shows that the current state sequence is a linear combination of the past input and output data, and the future outputs consist of the predictable subspace, the future input effects and the future stochastic signal effects. The subspaces $\boldsymbol{L}_1\boldsymbol{Y}_p+\boldsymbol{L}_2\boldsymbol{U}_p$ in N4SID and $\boldsymbol{Y}_f-\boldsymbol{H}_{f\_est}\boldsymbol{U}_f$ in CVA in fact are estimates of the predictable subspace. For a general open-loop process, most SIM algorithms give asymptotically unbiased estimates of the predictable subspace. For a truly ARX process, these methods give unbiased results. PCA in N4SID and CCA in CVA are to estimate the process states. Several new algorithms were proposed in this chapter, such as N4SID algorithm based on recursive one-step-ahead predictions, N4SID and CVA algorithms based on an estimated $\boldsymbol{H}_f$ matrix from a fitted ARX model, and MOESP algorithm based on estimated states.

The contributions of this chapter are as follows. First, a multi-step state-space model is established as a systematic approach to analyze the SIM algorithms. Second, the basic principles behind the computation steps of SIM algorithms are revealed. Third, the conditions for the algorithms to be biased or unbiased are provided. Finally, several new

algorithms are proposed in this chapter, and some mistakes with CVA regression-out algorithm in literature are pointed out.

Chapter 4 mainly focused on the relationships between SIMs and LVMs. The whole procedure of the oblique projection and the SVD in N4SID has been shown to be equivalent to a RRA between the past data and the estimated predictable subspace. The N4SID method was proven to be based on RRA, just as CVA is based on CCA. Insights from this relationship led to a variety of approaches with improved performance. This chapter discussed the similarities and differences between SIMs and LVMs. Both SIMs and LVMs build process dynamic models using a lower dimensional subspace of the original data sets, and SIMs use LVMs to estimate the process states. Both LVs and estimated states are linear combinations of the past data. Different from LVMS, SIMs remove the future input effects from the future outputs to get an estimate of the predictable subspace, and fit the significant LVs (estimated states) to the state-space model instead of to the outputs. Removal of the future input effect makes the final models from SIMs give casual relationships, and fitting to the state space model makes the final models from SIMs much more parsimonious and with smoother responses. For estimation of the process states, CCA and RRA have been shown to be more efficient than PCA and PLS.

This chapter serves to build a bridge between SIMs and LVMs, and links the research in the isolated fields together. It revealed the theoretical basis of N4SID in LVMs, and proposed a series of approaches to improve the performance of N4SID. This chapter drew a clear picture on the connections and differences between SIMs and LVMs, and gave a conclusion on the efficiency of LVMs for state estimation. The research in this chapter should help people understand the nature of these methods and provide guidance for the application of these methods.

In Chapter 5, a general statistical framework was proposed for SIMs. This framework shows that SIMs can be broken down into three steps: (1) use of a linear regression method to estimate the predictable subspace; (2) use of a latent variable method to estimate of a minimal set of the state variables; and (3) then fitting the

estimated process states to the state-space model. The major differences among SIMs lie in the first two steps, and the third step is common for the SIMs. The predictable subspace can be estimated by removing the future input effects from the future outputs based on estimated $\boldsymbol{H}_f$ matrix. Several approaches for estimation of matrix $\boldsymbol{H}_f$ were discussed, and a new approach based on the IV method was suggested. Process state variables can be estimated by LVMs based on the estimated predictable subspace and the past data. PCA, RRA and CCA can be directly employed to estimate a minimum set of state variables. PLS is a feasible method but not as efficient as other LVMs. The significant LVs are taken as the estimated process states. By fitting the LVs (estimated states) to the state-space model in the third step, the resultant state-space model becomes much more parsimonious than the corresponding LVR model.

This chapter proposed a general framework for SIMs. This framework clearly reveals the nature of the computation steps in SIM algorithms and the fundamental ideas behind the SIM algorithms. It also clearly reveals the relationships among different SIM algorithms. This chapter also points out the bias problem caused by the errors in the estimated states as well as the possible solutions based on the IV method. Combining the approaches in the first two steps leads to a whole variety of new SIM algorithms. Simulation study shows that these new SIM algorithms perform similarly to existing SIMs.

Chapter 6 investigated the applicability of SIM algorithms for closed-loop data. The fundamental difference between open loop and closed-loop data is the correlation between inputs and disturbances in closed-loop data. The original N4SID algorithm was analyzed in both the input-output model and the multi-step state-space model, and the correlation between future inputs and the future noise was shown to cause bias in the first step of N4SID (oblique projection). For closed-loop data, the CVA_RO algorithm was analyzed in the multi-step state-space model, and was shown to give biased results due to the correlation between the future inputs and the future noise. The CVA_Hf algorithm was shown to give asymptotically unbiased results for general closed-loop systems. In general, the applicability of SIMs for closed-loop data is algorithm-specific. Whether a

SIM algorithm is applicable for closed-loop data depends on how the future inputs are used in the detailed computation procedure. A SIM algorithm is applicable for closed-loop data only when the correlation between future inputs and the future noise is avoided. Several new N4SID and CVA algorithms were proposed for closed-loop data in this chapter, such as N4SID-RRA, N4SID-IV and joint state algorithms in N4SID and CVA, and these algorithms were shown to be applicable for closed-loop data.

Contributions from this chapter are as follows. First, this chapter provided a clear answer to the applicability of SIMs for closed-loop data, which had been controversial and totally unclear in the literature with disputes among researchers. Second, the existing SIM algorithms have been carefully analyzed, and conclusions have been drawn on which algorithms are applicable and which algorithms introduce bias. Third, new SIM algorithms are proposed for closed-loop data.

Chapter 7 analyzed several practical issues arising in the application of SIMs. In particular, SIMs were shown to be able to handle the delays in the process by increasing the number of estimated states. This detailed effect of delays and possible solutions were investigated for SISO, SIMO, MISO and MIMO cases. SIMs also were also shown to be able to capture the common dynamics easily, resulting in a more parsimonious model structure in the final state-space model. SIMS were also shown to handle the non-stationary as long as one is careful to treat the non-perfect pole-zero cancellation in the resulting dynamic model from SIMs. For co-integrating disturbances, SIMs are shown to offer advantages on capturing the common non-stationary states by simultaneously using multiple outputs as compared to using one output at a time. Some general guidelines for the application of SIMs were discussed in this chapter, and situations where SIMs could show their advantages were indicated.

This chapter provided clear conclusion about the ability of SIMs to handle several practical issues such as delays, common dynamics, non-stationary and co-integrating disturbances. It also provided some general guidelines for the application of SIMs.

This thesis covered many issues of SIMs; however, many other interesting issues have not been explored. Parameter variance of the identified model from SIMs is

discussed in Section 5.5, but not in detail. This is very important in practical application, and more research should be conducted on this issue. In SISO, SIMO and MISO cases, the effects of delays in the process can be eliminated effectively by shifting the input variables; however, for MIMO cases, this approach is not very efficient and other approaches need to be explored. In some applications, it might be desirable to separate the estimated states into different components, such as the deterministic states and the stochastic states, the process states and the controller states (joint state methods for closed-loop data), and therefore more research should be conducted on this issue. The resultant state-space model from SIMs is parsimonious; but for a specific input-output transfer function, an even lower model order might be adequate. Efficient methods would be required to reduce the model order. This would also improve lower frequency responses for models from data with non-stationary disturbances.

Some new ideas for SIM algorithms are worth for future exploration. The identified result from SIMs is expected to be improved based on the iterative estimate of $\boldsymbol{H}_f$. The idea is that a new $\boldsymbol{H}_f$ matrix can be constructed with the impulse from the resultant state-space model, and that a new predictable subspace can be estimated based on this new constructed $\boldsymbol{H}_f$. The Wilks statistic test can be used for the significance detection of the canonical correlation coefficients (CCCs); therefore it should be able to determine the system order in the CVA method. This is an issue for future research on SIMs. SIM algorithms for pure feedback closed-loop data are expected to give the most parsimonious models for the process and controller; however, this is still an open issue for future research.

SIMs have not reached maturity yet, and many issues still remain for deep exploration. With more researchers attracted to the field, SIMs will be well established and become more prevail in the future.

References

Andersen, H., K. Rasmussen and S. Jørgensen, Advances in Process Identification, Proceedings of 4th Int'l Conf. on Chemical Process Control -CPC IV, South Padre Island, TX 1991. Ed. Y. Arkun and W. Ray

Akaike, H., A New Look at the Stochastic Model Identification, IEEE Tran. on Automatic Control, Vol.AC-19, No.6, 1974

Amirthalingam, R., S. W. Sung and J. H. Lee, Two-step Procedure for Data-based Monitoring for Inferential Control Applications, AIChE Journal, Vol.46, No.10, pp.1974-88, 2000

Amirthalingam, R., J. H. Lee, K.S. Lee and D.R. Young, Inferential Model Predictive Control with Subspace Identification: Practical Issues and an Application to a Multi-component Distillation Column, AIChE Annual Meeting, 1998

Aström, K.J., Introduction to stochastic Control Theory, Academic Press, 1970

Basseville, M., M. Abdelghani and A. Benvenste, Subspace-based Fault Detection Algorithms for Vibration Monitoring, Automatica, Vol.36, pp.101-109, 2000

Bauer, D., M. Deistler, M., and W. Scherrer, Consistency and Asymptotic Normality of some Subspace Algorithms for Systems without Observed Inputs, Automatica, Vol.35, pp.1243-54, 1999

Bittanti, S. and M. Lovera, Bootstrap-based Estimates of Uncertainty in Subspace Identification Methods, Automatica, Vol.36, No.11, pp.1605-1615, 2000

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

Box, G.E.P. and G.C. Tiao, A Canonical Analysis of Multiple time Series, Biometricka, Vol.64, No.2, pp.355-65, 1977

Box, G.E.P., Jenkins, G.M. and Reinsel, G.C., "Time Series Analysis", Prentice-hall, 1994.

Brillinger, D.R., Time Series: Data Analysis and Theory, Holt, Rinehart and Winston, Inc., 1975

Burnham, A.J. R. Viveros and J.F. MacGregor, Frameworks for Latent Variable Multivariate Regression, Journal of Chemometrics, V10, pp.31-45, 1996

Burnham, A.J., J.F. MacGregor and R. Viveros, Latent Variable Multivariate Regression Modeling, Chemometrics Intell. Lab. Syst., Vol.48, pp.167-80, 1999

Candy, J.V., T.E. Bullock and M.E. Warren, Invariant System Description of the Stochastic Realization, Automatica, Vol.15, pp.493-95, 1979

Carrette, P., personal communication, notes for CVA, 2000

Carrette, P. and T. McKelvey, Model Parameter Gradients in Prediction Identification of State-space Systems, 37th Conference on Decision and Control, also see http://www.control.isy.liu.se

Cho, Y. M. and Kailath, T., Fast Subspace-based System Identification: An Instrumental Variable Approach, Automatica, Vol. 31, No. 6, pp.903-905, 1995

Chou C.T. and M. Verhaegen, Subspace Algorithms for the Identification of Multivariable Dynamic Error-in-Variables Methods, Automatica, Vol. 33 No. 10, pp.1857-69, 1997

Chui, N.L.C. and J.M. Maciejowski, Realization of Stable Model with Subspace Methods, Automatica, Vol.32, No.11, pp.1587-95, 1996

Dayal, B.S., Multivariable Statistical Regression Methods for Process Modeling and Experimental Design, Ph.D. Thesis, McMaster University, 1996

Dayal, B.S. and J.F. MacGregor, Identification of FIR Models: Methods and Robustness Issues, Ind. Eng. Chem. Res., Vol. 35, No.11, 1996

Davies, P.T., and M. K-S. Tso, Procedures for Reduced-rank Regression, Appl. Statist., Vol.31, No.3, pp.244-255, 1982

Deistler, M., K. Peternell and W. Scherrer, Consistency and Relative Efficiency of Subspace Methods, Automatica, Vol.31, No.12, pp.1865-75, 1995

Di Ruscio, D., A Method for Identification of Combined Deterministic-stochastic Systems, Proc. 3rd European Control Conference, Rome, Italy, 1995

Efron B. and R. Tibshirani, An introduction to the bootstrap, Chapman and Hall, 1993

Ergon, R., Dynamic System Multivariate Calibration, Chemometrics Intell. Lab. Syst., Vol.44, pp.135-46, 1998

Ergon, R. and M. Halstensen, Dynamic System Multivariate Calibration using High Sampling Rate Acoustic X data and Low Sampling Rate Y Data, J. Chemometrics, Vol.14, 2000

Esmaili, A., P.A. Taylor and J.F. MacGregor, Direct and Two-step Methods for Closed-loop Identification: A Comparison of Asymptotic and Finite Data Set Performance, J. of Process Control, Vol.10, No.6, 2000, pp.525-37

Ewerbring, L.M. and F. T. Luk, Parallel Algorithm for Canonical Variates Computation, American Conference on Decision and Control, Honolulu, Hawaii, Dec. 1990

Favoreel, W., S. Van Huffel, B. De Moor, S. Vasile and M. Verhaegen, Comparative Study between Three Different Subspace Identification Algorithms, Proc. of the European Control Conference, Karlsruhe, Germany, Aug., 1999

Forssell, U. and L. Ljung, Closed-loop Revisited, Automatica, Vol.35, pp.1215-1241, 1999

Gagnon, L. and Macgregor, J.F., State Estimation for Continuous Emulsion Polymerization, Canadian Journal of Chemical Engineering Vol. 69, No. 3, pp. 648-656, 1991

Gevers, M., L. Ljung and P.M.J. Van den Hof, Asymptotic Variance Expressions for Closed-loop Identification and Their Relevance in Identification for Control". Selected Topic in Identification, Modeling and Control, Vol.9, pp.9-15. 1996

Hartnett, M.K., G. Lightbody, and G.W. Irwin, Identification of State Models using Principal Components

Analysis, Chemometrics Intell. Lab. Syst., Vol.46, pp.181-196, 1999

Hjalmarsson, H., M. Gevers and F. De Bruyne, For Model-based Control Design, Closed-loop Identification Gives Better Performance, Automatica, Vol.32, No.12, pp.1659-1673, 1996

Höskuldsson, A., PLS regression methods, J. Chemometrics, Vol.2, pp.211-228, 1988

Höskuldsson, A., Prediction Methods in Science and Technology, Warsaw: Colourscan, 1996

Hurvich, C.M, and V.L. Tsai, Regression and Time Series Model Selection in Small Samples, Biometrika, Vol.76, pp.297-307, 1989

Isermann, R. and Balle, P., Trends in the Application of Model Based Fault Detection and Diagnosis of Technical Process, Proceeding of 13th IFAC, San Francisco, 1996

Jackson, J.E., A User's Guide to Principal Components, John Wiley and Sons, Inc., 1991

Jasson, M. and B. Wahlberg, A linear Regression Approach to State-space Subspace System Identification, Signal Processing, Vol.52, pp.103-129, 1992

Jha, H. and Georgakis, C., closed-loop Identification of State Space Models Using the Subspace Identification Technique, Progress Report No. 22, 2A, Chemical Process Modeling and Control research Center, Lehigh University, 1996

Jha, H. and Georgakis, C., Application of Subspace Identification Algorithm to Identify State-space Models for Fluid Catalytic Cracking Unit, Progress Report No. 22, 2B, Chemical Process Modeling and Control research Center, Lehigh University, 1996

Jørgensen, S. B. and Jay H. Lee, Recent Advances in Control-Oriented Identification and Closed-Loop Identification, Conference on Process Control, AZ, 2001.

Jutan, A., J.D. Wright and J.F. MacGregor, Multivariable Computer control of a Butane Hydrogenolysis Reactor, Part II: Design and on-line Implementation of a Stochastic Controller using an Identified multivariate Noise Model, AIChE J. Vol. 30, No.2, pp.220-226, 1984

Kaspar, M.H. and W. H. Ray, Dynamic PLS Modeling for Process Control, Chem. Eng. Sci., Vol.48, No.20, pp.3447-61, 1993

Kassidas, A., Fault Detection and diagnosis in Dynamic Multivariable Chemical Process Using Speech Recognition Methods, Ph.D., Thesis, McMaster University, 1997

Kemna, A.H., D.E. Seborg, D.A. Mellichamp and N. Sweerus, Dynamic Models of Recovery Boilers from Inpuit/Output Data, TAPPI Internat. Chemical Recovery Conference, June 1992, Seattle, WA

Kourti, T., P. Nomikos and J.F. MacGregor, Analysis, Monitoring and Fault Diagnosis of Batch Process Using Multiblock and Multiway PLS, J. Proc. Cont., Vol.5, No.4, pp.277-84, 1995

Kourti, T. and J.F. MacGregor, Process Analysis, Monitoring and Diagnosis Using Multivariate Projection Methods, Chemometrics Intell. Lab. Syst., Vol.28, pp.3-21, 1995

Kresta, J., J.F. MacGregor and T.E. Marlin, Multivariate Statistical monitoring of Process Operating Performance, Can. J. Chem. Eng., Vol.69, pp.35-47, 1991

Ku, W.F., R.H. Storer and C. Georgakis, Disturbance Detection and Isolation by Dynamic Principal Component Analysis, Chemometrics Intell. Lab. Syst., Vol.30, pp.179-96, 1995

Kozub, D., Enhanced FIR modeling, technical report, 1994

Lakshminarayanan, S., G. Emoto, S. Ebara, K.tomida and S. Shah, Closed Loop Identification and Control Reconfiguration: An Industrial Case Study, J. of Process Control, Vol.11, pp.587-99, 2001

Lakshminarayanan, S., S. Shah and K. Nandakumar, Modeling and control of Multivariable Process: A Dynamic PLS Approach, AIChE J., Vol.43, No.9, 1997

Lakshminarayanan, S., Process Characterization and Control Using Multivariate Statistical Techniques, Ph.D. thesis, Univ. of Alberta, 1997

Larimore, W. E., Canonical Variate Analysis for System Identification, Filtering and Adaptive Control, Proc. 29th IEEE Conference on Decision and Control, Vol.1, Honolulu, Hawaii, 1990

Larimore, W.E., Order-recursive Factorization of the Pseudoinverse of a Covariance Matrix, IEEE Transactions on Automatic Control, Vol.35, No.12, 1990

Larimore, W. E., Statistical Optimality and Canonical Variate Analysis System Identification, Signal Processing, Vol.52, pp.131-44, 1992

Larimore, W. E., Accuracy Confidence Bands Including the Bias of Model Under-Fitting, Proceedings of the American Control Conference, San Francisco, CA, June 1993

Larimore, W. E., The Optimality of Canonical Variate Identification by Example, Proceedings of the 10th IFAC Symposium on System Identification, Copenhagen, July 1994

Larimore, W. E., the Optimal of Canonical Variate Identification by example, Proc. 10th IFAC Symposium on System Identification, Copenhagen, 1994

Larimore, W.E., Predictive inference, Sufficiency, Entropy and an Asymptotic Likelihood Principle, Biometrika, Vol.70, No.1, pp.175-81

Larimore, W. E., System Identification of Feedback and "Causality" structure Using Canonical Variate Analysis, IFAC SYSID'97, Fukuoka, Japan, July 8-11,1997

Larimore, W. E., Optimal Reduced Rank Modeling, Prediction, Monitoring and Control Using Canonical Variate Analysis, Preprints of ADCHEM, Banff, 1997

Larimore, W. E., A Unified View of Reduced Rank Multivariate Prediction Using Generalized Singular Value Decomposition, unpublished.

Larimore, W.E., and J. Baillieul, Identification and Filtering of Nonlinear System using Canonical Variate Analysis, Proc. 29th IEEE conference on Decision and Control, Vol.1, Honolulu, Hawaii, 1990

Larimore, W. E. and R. K. Mehra, The Problem of Overfitting Data, Byte, pp.167-180, Oct. 1985

Li, W. and S.J. Qin, Consistent dynamic PCA Based on Errors-in-variables Subspace Identification, J. of Process Control, Vol.11, No.6, pp.661-78, 2000.

Lindquist A. and G. Picci, Canonical Variate Analysis, Approximate Covariance Extension, and Identification of Stationary Time Series, Automatica, Vol.32, pp.709-33, 1996

Ljung, L., System Identification: Theory for the User, 2nd edition, Prentice-hall Press, 1999

Ljung, L., System Identification Toolbox for Use with MATLAB, User's Guide, Version 4, The Mathworks Inc., 1997

Ljung, L., A Simple Start-up Procedure for Canonical State Space Identification based on Subspace Approximation, Proc. 30th IEEE Conf. on Decision and Control, pp.1333-36, Brighton, U.K., 1991

Ljung, L. and T. McKelvey, Subspace Identification from Closed loop Data, Signal Processing, V52, 1996

Ljung, L. and T. McKelvey, Interpretation of Subspace Methods: Consistency Analysis, IFAC System Identification, Fukuoka, Japan, 1997

MacGregor, J.F., T. Kourti and J.V. Kresta, Multivariate Identification: A Study of Several Methods, IFAC symposium ADCHEM-91, Toulouse, France, 1991

MacGregor, J.F. and A.K.L. Wong, Multivariate Model Identification and Stochastic Control of a Chemical Reactor, Technometrics, Vol.22, No.4, pp.453-64, 1980

Martens, H. and T. Naes, Multivariate Calibration, Wiley, New York, 1991

Muller, K., Understanding Canonical Correlation Through the General Linear Model and Principal Components, The American Statistician, Vol.36, No.4, 1982

Negiz, A. and A. Cinar, PLS, Balanced, and Canonical Variate Realization Techniques for Identifying VARMA Models in State Space, Chemometrics Intell. Lab. Syst., Vol.38, pp.209-21, 1997

Negiz, A. and A. Cinar, Statistical Monitoring of Multivariable Dynamic Process With State-space Model, AIChE J., Vol.43, No.8, 1997

Nomikos, P. and J.F. MacGregor, Monitoring Batch Process Using Multiway Principal Component Analysis, AIChE J., Vol.40, No.8, 1994

Nomikos, P. and J.F. MacGregor, Multivariate SPC Charts for Monitoring Batch Processes, Technometrics, Vol.37, No.1, 1995

Peloubet, R.P.,Jr., R.L. Haller, and R.M. Bolding, On-line Adaptive Control of Unstable Aircraft Wing Flutter, Proc. 29th IEEE Conference on Decision and Control, Vol.1, Honolulu, Hawaii, 1990

Qin, S.J., Subspace Identification Models for Dynamic Sensor Fault Detection and Diagnosis, AIChE Annual Conference, Nov. 2000, Los Angeles, CA

Rencher, A.C., Multivariate Statistical Inference and Applications, John Wiley and Sons, 1998

Scahper C.D., W.E. Larimore, S.E. Seborg and D.A. Mellichamp, Identification of Chemical process using Canonical variate Analysis, Proc. 29th IEEE Conference on Decision and Control, Vol.1, Honolulu, Hawaii, 1990

Shi, R. and J. F. MacGregor, Modeling of Dynamic Systems using Latent Variable and Subspace Methods, J. of Chemometrics, V14, pp.423-39, 2000

Shi, R. and J. F. MacGregor, A Framework for Subspace Identification Methods, Proceedings of the American Control Conference, pp.3678-83, Arlington, VA, June 25-27, 2001

Shi, R. and J. F. MacGregor, Relationship between N4SID Subspace Identification and Reduced-rank Analysis, AIChE Conference, Reno, NV, Nov. 4-9, 2001

Shibata, R., An Optimal Autoregression Spectral Estimate, Ann. Statistics, Vol.9, pp300-06, 1981

Slama, C., Multivariate Statistical Analysis of Data from an Industrial Fluidized Catalytic Cracking Process using PCA and PLS, M. Eng. Thesis, McMaster Univ., Canada, 1992

Söderstrom, T. and P. Stoica, System Identification, Prentice Hall, 1989

Taylor, P.A., Process Identification Notes, Courseware, McMaster University, 1993

User's Guide, MATLAB System Identification Toolbox, the Math Works Inc.

Van Overschee, P. and De Moor, B., Subspace Algorithms for the Stochastic Identification Problem, Automatica, Vol.29, No.3 pp.649-60, 1993

Van Overschee, P. and De Moor, B., N4SID: Subspace Algorithms for the Identification of Combined Deterministic-Stochastic System, Automatica, Vol. 30, No. 1, pp. 75-93, 1994

Van Overschee, P. and De Moor, B., A Unifying Theorem for Three Subspace System, Identification Algorithms, Automatica, Vol.31, No.12, pp.1835-64, 1995

Van Overschee, P., B. De Moor, W. Dehandschutter and J. Swevers, Choice of State-space Basis in Combined Deterministic-Stochastic Subspace Identification, Automatica, Vol. 31. No.12, pp.1877-83, 1995

Van Overschee, P. and De Moor, B., Subspace Identification for Linear Systems: Theory-Implementation-Applications, Kluwer Academic Publishers, 1996

Verhaegen, M., Identification of Deterministic Part of MIMO State Space Models Given in Innovations from Input-output Data, Automatica, Vol.30, pp.61-74, 1994

Verhaegen, M., Application of a Subspace Model Identification Technique to Identify LTI Systems Operating in Closed-loop, Automatica, Vol. 29 No.4, pp.1027-40, 1993

Verhaegen, M. and Dewilde P., Subspace Model Identification, Part 1., The Output-error State-space Model Identification Class of Algorithms, International Journal of Control, V56, pp.1187-1210, 1992

Verhaegen, M. and Dewilde P., Subspace Model Identification, Part 2., Analysis of the Elementary Output-Error State -Space Model Identification Algorithm, International Journal of Control, Vol. 56 No.5, pp.1211-41, 1992

Verhaegen, M. and Dewilde P., Subspace Model Identification, Part 3., Analysis of the Ordinary Output-Error State -Space Model Identification Algorithm, International Journal of Control, Vol. 58 No.3, pp.555-86, 1993

Viberg, M., Subspace-based Methods for the Identification of Linear Time-invariant System, Automatica, V31, No.12, pp. 1835-51, 1995

Vogel, E.F. and J.J. Downs, Industrial Experience with State-Space Model Predictive Control, Preprints of Chemical Process Control-6 (CPC-6), Tucson, AZ, Jan. 7-12, 2001

Wang, J. and S. Qin, A New Subspace Identification Approach based on Principal Component Analysis, AIChE Conference, 2000,

Wang, S., Use of Multivariable Statistical Methods in Process Engineering, Master thesis, McMaster University, 1999

Wang, Y., Seborg, D. E. And Larimore, W. E., Process Monitoring Using Canonical Variate Analysis and Principal Component Analysis, preprints of ADCHEM, Banff, 1997

Wise, B.M. D.J. Veltkamp, N.L. Ricker, B.R. Kowalski, S.M. Barnes and V. Arakali, Application of Multivariate Statistical Process Control (MSPC) to the West Valley Slurry-Red Ceramic Melter Process, Waste Management '91 Proceeding, Tuscon, AZ, 1991

Wold, S., Cross-Validatory Estimation of the Number of Components in Factor and Principal Components Models, Technometrics, Vol.20, No.4, pp.397-405

Wold, S., C. Albano, W.J. Dunn III, U. Edlund, K. Esbensen, P. Geladi, S. Hellberg, E. Johansson, W. Lindberg, M. Sjostrom, Multivariate Data Analysis in Chemistry, 1983, Proceedings of the NATO Advanced Study Institute on Chemometrics – Mathematics and Statistics in Chemistry, Cosenzas, Italy, Sep. 12-23, edited by B. Kowalski, 1984

Wold, S., N. Kettaneh-Wold, B. Skagerberg, Nonlinear PLS Modeling, Chemometrics Intell. Lab. Syst., Vol.7, pp.53-65, 1989