University of Alberta

Multivariate Process and Control Monitoring – Practical Approaches and Algorithms

by

Sien Lu ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**

in

Process Control

Department of Chemical and Materials Engineering

Edmonton, Alberta
Fall 2005

To the memory of my mother and my father...

Abstract

Performance assessment and process monitoring are two active research areas over the last decade. In this thesis, some practical approaches and algorithms are presented.

First, an improved algorithm for calculation of the interactor matrix is developed, the FCOR algorithm is presented and the subspace approach is described. Three Matlab functions are programmed for these three algorithms, respectively. All of them have been tested on simulation examples as well as applied to industrial case studies.

Second, the basic concepts of Markov chains are briefly reviewed. The applications of Markov chains to two industrial plants are elaborated.

Last, a practical process monitoring method is presented. This method incorporates wavelet transform, symbolic representation and Hidden Markov model (HMM) together. Simulation examples and industrial case studies have shown the value of this method. As a future use of this method, an oscillation detection approach is developed.

Acknowledgements

I wish to express my deep gratitude to Dr. Huang, my supervisor, for his constant and kind support, excellent advice and guidance during the whole research work.

I extend my thanks to the members of Huang's research group for their generous help.

Appreciations are also due to the Syncrude Canada Ltd., the National Science and Engineering Research Council of Canada, the Canada Foundation for Innovation and the University of Alberta for research funding.

Thanks my friends at Alberta, Xin Nie, Hailei Jiang & Jia Jia, Bo lu & Wen Sun, Wei Wang, Fang Yang.

Special thanks to Delta Chi fraternity, especially thanks Davin Swenson, Ryan Brule, Jared Goruk and Alfonso Estevez.

Contents

List of Tables

List of Figures

List of Symbols

A, B, P, ...	Capital letters denote matrices
u, y, z, ...	Lowercase letters denote vectors
I_n	$n \times n$ identity matrix
Y^T	Transpose of Y
Y^{-1}	Inverse of Y
$Y^\dagger$	Pseudo inverse of Y
$\lvert Y \rvert$	Determinant of Y
$\hat{Y}$	Estimation of Y
$det(Y)$	Determinant of Y
$diag(Y)$	Diagonal elements of Y
$eig(Y)$	Eigenvalues of Y
$tr(Y)$	Trace of Y

List of Acronyms

CV	Controlled Variable
CWT	Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
FCOR	Filtering and Correlation
GOHTU	Gas Oil Hydrotreater Unit
HMM	Hidden Markov Model
LMIPA	Linear Matrix Inequality Performance Analysis
LQ	Linear Quadratic
MIMO	Multiple Input Multiple Output
MPC	Model Predictive Control
MV	Manipulated Variable
MVC	Minimum Variance Control
OCI	Out of Control Index
PAT	Performance Analytical Toolbox
PI	Performance Index
PSV	Primary Separation Vessel
RMF	Right Matrix Fraction
r.s.p.m.	row shift polynomial matrix
SISO	Single Input Single Output
SMI	Subspace Method of Identification
SNR	Signal to Noise Ratio
SVD	Singular Value Decomposition
TTI	Transition Tendency Index

1 Introduction

With the development of computer technology, distributed control system (DCS) has been widely used. Meanwhile, as a result of improvement of measurement and manufactory technology, the processes are heavily equipped with sensors. With these developments, data is easily collected and advanced control algorithms can be implemented. To gain a competitive edge in the market place, on the other hand, every company is trying to run their plants as safe and economic as possible. Therefore, the adoption of more sophisticated control technologies is in demand and also possible. Among these new technologies, two of them are of more interests. The first one is the performance assessment of controllers, and the second is the process monitoring.

1.1 Performance Assessment

Undoubtedly, the controller is the heart of a control system. There are hundreds of control loops in a typical plant. Unfortunately, it is impossible for process control engineers to routinely evaluate the performance of the controllers one by one. Moreover, the behaviors of processes are changing. Even the parameters of a controller are tuned very well today, it can not be promised that this controller will work well tomorrow. Therefore, a computer-aided method which can assess the performance of the controllers automatically and routinely is needed.

In performance assessment field, the minimum variance control (MVC) benchmark has been widely used since the work of Harris (1989). With this benchmark, the performance index (PI) of a single-input single-output (SISO) system can be calculated as the ratio of the output variance under minimum variance control and the practical output variance. For the multi-input multi-output (MIMO) system, it is the ratio of the traces of output covariance. Since only under minimum variance control does the variance or covariance reach the lowest bound, the value of PI lies between 1 and 0. Higher the value, better the performance.

Although the minimum variance control is hardly applied in practice because of its aggressive activity, as a benchmark, it can provide useful information about the performance of control systems. The economic performance, for example, is usually measured by the deviation from set-point, thus lower variance means better economic performance. If the PI indicates poor performance, the reduction of variance can be achieved by controller parameters tuning or control algorithm re-design. If the PI indicates good performance, on the other hand, but further reduction of variance is still needed, the tuning or re-design would not be helpful anymore. In this case, change of control structure or process, such as, feedforward control or relocation of sensors, may be necessary. Moreover, MVC benchmark is the only performance measure that can be evaluated without complete knowledge of the process model so far (Ko and Edgar 2001).

A comprehensive review paper about performance assessment using MVC as benchmark is presented by Harris *et al.* (1999). That of single-loop feedback control and feedforward control is studied by Harris (1989) and Desborough and Harris (1993). The cascade case is explored by Ko and Edgar (2000). For the multivariable feedback controls systems, Harris *et al.* (1996), Huang *et al.* (1997b) are two important papers. Harris *et al.* (1996) extended the MVC performance benchmark from SISO system to MIMO system by multivariate spectral factorization. Huang *et al.* (1997b) generalized the filtering and correlation (FCOR) algorithm to MIMO system for performance assessment. Ko and Edgar (2001) directly evaluated the performance as a function related to the first few Markov parameters of the plant. Huang and Shah (1998) addressed practical issues in performance assessment of multivariable feedback control systems, such as, nonminimum phase systems. Horch and Isaksson (1999) developed a modified MVC index, which uses either control design guidelines or additionally available process knowledge. Grimble

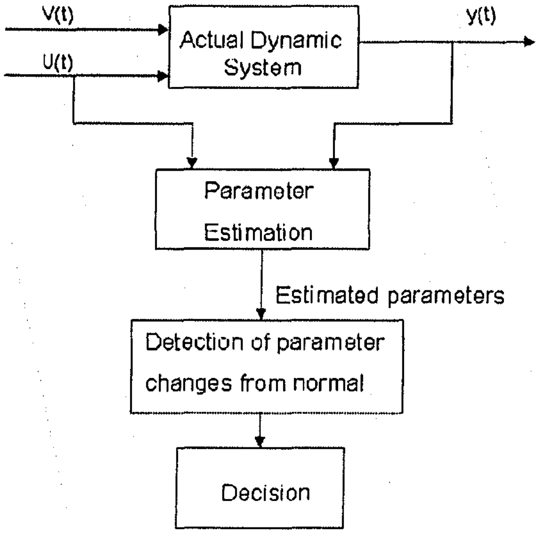


Figure 1.1: Model based process monitoring

(2004) presented an integrated minimum variance criterion to assess the performance of controllers, which takes the integral action of PID controller into account. Thornhill *et al.* (1999) summarized some practical experiences with the control loop performance assessment in a refinery-wide setting. Paulonis and Cox (2003) illustrated a large-scale controller performance assessment system developed by Eastman Chemical Company, which spans over 14,000 PID controllers in 40 plants at 9 sites worldwide.

1.2 Process Monitoring

The principle of model based process monitoring is depicted in Figure 1.1 (Smyth 1994). The first step is parameter estimation of process models through collected input/output data. The next step is to detect whether there is change of parameters. If so, the process probably has abnormal behavior and a warning information should be given.

Basically, process monitoring methods can be divided into three categories: data-driven, analytical and knowledge-based (Chiang *et al.* 2001).

The data-driven approach, for example, principal component analysis (PCA), fisher discriminant analysis (FDA), partial least squares (PLS) and canonical variate analysis (CVA), directly works on input/output process data. This is the reason why these approaches are called data-driven. These approaches transform the high dimensional data into a lower dimension and then capture some important information which is hard to observe in the original space.

The analytical approach is based on mathematical models which are usually constructed from first principles. Thus, it can bring some valuable insights about the process. However, it is often difficult and expensive to obtain the first principle models, especially for large scale and complex processes.

The knowledge-based approach, such as, causal analysis, expert systems and pattern recognition, uses qualitative and semi-quantitative information to monitor the process. The process monitoring method presented in Chapter 4 of this thesis is a pattern recognition method.

At the end of this section, it should be emphasized that all the process monitoring methods have their advantages and disadvantages. Compared with other methods, one method is more sensitive to one kind of fault and less sensitive to another kind of fault. Therefore, the best way is to incorporate several methods together.

1.3 Thesis Outline

The organization of this thesis is as follows. In Chapter 2, an improved algorithm for calculation of the interactor matrix is developed, a MIMO system performance assessment algorithm with the knowledge of interactor matrix is presented and the other without a prior knowledge of interactor matrix is also described. Three Matlab functions are programmed for these three algorithms, respectively. All of them have been tested on simulation examples as well as applied to industrial case studies. In Chapter 3, the theory of Markov chains is briefly reviewed and applied to industrial plants. In Chapter 4, a practical process monitoring method is presented. This method integrates wavelet transform, symbolic representation and Hidden Markov model (HMM) together. Simulation examples and industrial case studies show the value of this method. Based on this process monitoring method, an oscillation detection approach is presented. In Chapter 5, the works of this thesis are summarized and the future extensions are given.

1.4 Thesis Contributions

The contributions of this thesis are summarized as follow:

- An improved algorithm for calculation of the interactor matrix is developed.
- The theory of Markov chain is employed for industrial data analysis. Two indices are defined, one is out of control index (OCI) and the other transition tendency index (TTI).
- A practical process monitoring method is developed. This method converts the basic trend of monitored variable into qualitative and semi-quantitative information and then classifies them into normal and abnormal status by hidden Markov model (HMM). This method has been used to analyze industrial data sets. As an illustrative example of the future use of this method, an oscillation detection method is presented.
- All approaches and algorithms introduced in this thesis have been programmed in Matlab environment and integrated into the Performance Analytical Toolbox (PAT). They have been tested on simulation examples as well as applied to industrial case studies.

2

Algorithms Development for Performance Assessment

Recently, two sets of software package have been developed by Huang and the project team at the University of Alberta: one is named as LMIPA (Linear Matrix Inequality Performance Analysis), which calculates the potential of economic performance improvement for model predictive control (MPC) and also provides the tuning guidelines on how to achieve this potential, and the other named as PAT(Performance Analytical Toolbox), which calculates the dynamic performance of MPC. All the algorithms and functions introduced in this chapter have been integrated into PAT.

2.1 Interactor Matrix

Interactor matrix, D, introduced by Wolovich and Falb (1976), Wolovich and Elliott (1983), as well as Goodwin and Sin (1984), is a generalization of the SISO time delay for the MIMO case. Introduction of the concept of interactor matrix has made it possible to extend many advanced control strategies, for example, adaptive control and minimum variance control, to multivariate systems. It is also a very important prerequisite to performance assessment of multivariate systems based on MVC benchmark.

2.1.1 Algorithms

Nilpotent interactor matrix

Rogozinski *et al.* (1987) introduced the concept of **nilpotent interactor matrix.**

Definition 1. *For every $n \times m$ proper, full rank, rational polynomial transfer function matrix T, any $n \times n$ polynomial matrix D, having the properties*

$$\lim_{q^{-1}\to 0} DT = \lim_{q^{-1}\to 0} \tilde{T} = K \tag{2.1}$$

and

$$|D| = q^r \tag{2.2}$$

will be called a nilpotent interactor matrix for the system defined by T. K is a full rank constant matrix, the integer r is defined as the number of infinite zeros of T, and $\tilde{T}$ is the delay-free transfer function (factor) matrix of T, which contains only finite zeros. The matrix D can be written as

$$D = D_0 q^d + D_1 q^{d-1} + \cdots + D_{d-1} q \tag{2.3}$$

where d, the maximum power of q in D, is denoted as the order of the interactor matrix and is unique for a given transfer function matrix. D_i, $i = 0, \cdots, d-1$, are coefficient matrices.

Rogozinski *et al.* (1987) proposed an algorithm to evaluate a non-unique nilpotent interactor matrix operating on the coefficients of the numerator of the **right matrix fraction** (RMF) description of a system.

Definition 2. *A transfer function matrix $T(q)$ can be (nonuniquely) factored as*

$$T(q) = N(q)E^{-1}(q), \tag{2.4}$$

in which the denominator of the RMF description,

$$E(q) = I + E_1 q^{-1} + \cdots + E_n q^{-n}, \tag{2.5}$$

is a monic polynomial matrix and the numerator is assumed to be a polynomial matrix

$$N(q) = N_0 + N_1 q^{-1} + \cdots + N_n q^{-n}, \tag{2.6}$$

which can be represented by a block coefficients matrix

$$N = \begin{pmatrix} N_0 \\ \vdots \\ N_n \end{pmatrix} \tag{2.7}$$

where n is the degree of the denominator polynomial in the RMF description. Usually, N_i is not a full-rank matrix and, in particular, the leading coefficients $N_0, N_1, \cdots$ can be zero.

Definition 3. *The $n \times n$ first degree polynomial matrix $U(q)$ will be called a row shift polynomial matrix (r.s.p.m.) of order k_i, where*

$$U(q) = U_0 q + U_1 = \begin{pmatrix} 0 & I_r \\ qI_{k_i} & 0 \end{pmatrix}. \tag{2.8}$$

The matrices U_0 and U_1 are defined through the matrix of coefficients:

$$U = \begin{pmatrix} U_0 \\ U_1 \end{pmatrix} = \begin{pmatrix} 0_r \\ I_n \\ 0_{k_i} \end{pmatrix}, n = r + k_i, \tag{2.9}$$

in which, U_0, U_1 are of dimension $n \times n$, I_n is the $n \times n$ identity matrix, and 0_r is a r-row matrix of zeros.

Operating on the numerator of the RMF given in the form of block coefficients matrix N, the algorithm proposed by Rogozinski *et al.* (1987) can estimate a nilpotent interactor matrix consisting of t factors through finite recursive calculation:

$$D(q) = S^{(t)}(q)S^{(t-1)}(q) \cdots S^{(1)}(q), \tag{2.10}$$

where every factor is the product of two matrices:

$$S^{(i)}(q) = U^{(i)}(q)Q^{(i)}, \tag{2.11}$$

in which $U^{(i)}(q)$ is a r.s.p.m. of order k_i and $Q^{(i)}$ is a nonsingular $n \times n$ real matrix.

The algorithm proposed by Rogozinski *et al.* (1987) is introduced as follow.

Algorithm 1.

Initialization : Set $i = 0, N^{(0)}(q) = N$, and $D^0(q) = I_n$.

Iteration : Consider the ith iteration in the evaluation of $D(q)$.

Step 1: If $r_i = \text{rank } N_0^{(i-1)} = \min(n, m)$, the algorithm terminates and the nilpotent interactor matrix is $D(q) = D^{(i-1)}(q)$. If $r_i < \min(n, m)$, factorize $N_0^{(i-1)}$ into

$$N_0^{(i-1)} = (Q^{(i)})^{-1} \begin{pmatrix} 0_i \\ N_{0D}^{(i)} \end{pmatrix}, i.e., Q^{(i)} N_0^{(i-1)} = \begin{pmatrix} 0_i \\ N_{0D}^{(i)} \end{pmatrix} \tag{2.12}$$

where $Q^{(i)}$ is a $n \times n$ nonsingular (usually unitary) matrix, 0_i is a k_i-row zero matrix and $k_i = n - r_i$.

Step 2: Premultiply $N^{(i-1)}(q)$ by matrix $Q^{(i)}$

$$\bar{N}(q) = Q^{(i)} N^{(i-1)}(q). \tag{2.13}$$

Step 3: Premultiply $\bar{N}(q)$ by the r.s.p.m. of order k_i

$$N^i(q) = U^{(i)}(q)\bar{N}(q). \tag{2.14}$$

This multiplication shifts the coefficients matrix of $\bar{N}(q)$ upwards by k_i rows of zeros. Update the matrix

$$D^i(q) = S^{(i)}(q)D^{(i-1)}(q). \tag{2.15}$$

This ends the ith iteration. ■

This algorithm can be applied to any $n \times m$ proper, full rank transfer function matrix T, no matter $n \geq m$ or $n < m$.

Unitary interactor matrix

Peng and Kinnaert (1992) introduced the concept of **unitary interactor matrix.** In fact, a unitary interactor matrix is a special case of nilpotent interactor matrix. The use of unitary interactor matrix will simplify some linear quadratic (LQ) and MVC problems.

Definition 3. *If an interactor matrix satisfies*

$$D^T(q^{-1})D(q) = I, \tag{2.16}$$

then this interactor matrix is denoted as a unitary interactor matrix.

Peng and Kinnaert (1992) proved that in Algorithm 1, at step 1 of each iteration, if the factorization of $N_0^{(i-1)}$ is calculated by the normalized QR factorization, $Q^{(i)}$ is a unitary matrix. Moreover, $U^{(i)}(q)$ is also a unitary polynomial matrix as it satisfies $U^T(q^{-1})U(q) = I_n$. Since the product of unitary matrices must itself be unitary, $D(q)$ is a unitary interactor matrix.

Peng and Kinnaert (1992) limited their algorithm to square transfer function matrices. However, this is not necessary because the algorithm can be extended to non-square transfer function matrices easily without any further assumptions.

Calculate interactor matrix using Markov parameters

No matter nilpotent interactor matrix or unitary interactor matrix, so far, the requirement of *priori* knowledge of interactor matrix is tantamount to almost complete knowledge of the system transfer matrix. Shah *et al.* (1987) suggested factoring an interactor matrix directly from Markov parameters of a process.

Considering a transfer function matrix T using a Markov parameter representation

$$T = \sum_{i=0}^{\infty} G_i q^{-i}, \tag{2.17}$$

it follows from the definition of an interactor matrix that

$$\begin{aligned}&\lim_{q^{-1}\to 0}[D_0 q^d + D_1 q^{d-1} + \cdots + D_{d-1}q][G_0 q^{-1} + G_1 q^{-2} + \cdots]\\ &= D_0 G_{d-1} + D_1 G_{d-2} + \cdots + D_{d-1} G_0 = K.\end{aligned} \tag{2.18}$$

Examination of the above equation results in the following set of linear, algebraic equations for computing $D_0, D_1, \cdots, D_{d-1}$:

$$\begin{aligned}&D_0 G_0 = 0,\\ &D_1 G_0 + D_0 G_1 = 0,\\ &\vdots\\ &D_{d-1} G_0 + \cdots + D_1 G_{d-2} + D_0 G_{d-1} = K.\end{aligned} \tag{2.19}$$

The order of the interactor matrix, d, is the fewest linear combinations of the rows of Markov parameters or matrices, G_i's, required such that the set of simultaneous linear algebraic equations hold.

Following this idea, Huang *et al.* (1997) developed a method for calculation of the interactor matrix. First, the order of the interactor matrix is determined by using singular value decomposition (SVD) to the above set of equations. Second, because Markov parameter is a special case of numerator coefficients matrix of the RMF of the transfer function matrix:

$$T = \sum_{i=0}^{\infty} G_i q^{-i} = N(q)E^{-1}(q) \tag{2.20}$$

where $N(q) = G_0 + G_1 q^{-1} + G_2 q^{-2} + \cdots$, and $E(q) = I$, a unitary interactor matrix can be factorized using the Algorithm 1 operating on the first few (not smaller than interactor matrix order d) Markov parameters instead of the numerator coefficients of the RMF description of the system.

Huang *et al.* (1997) also pointed out that even without the knowledge of the interactor matrix order d, the interactor matrix can be factored provided that enough Markov parameters are given. Thus, based on this result, an improved algorithm is derived, which is simple in the sense of concept.

Algorithm 2

A $n \times m$ proper, full rank transfer function matrix T can be expressed by its Markov parameter representation as $T = G_0 + G_1 q^{-1} + \cdots + G_i q^{-i} + \cdots$. Stack the first i Markov parameters as a block matrix

$$G = \begin{pmatrix} G_0 \\ \vdots \\ G_i \end{pmatrix}. \tag{2.21}$$

Initialization : Since G_0, the first Markov parameter matrix of a causal MIMO system, is always zero, the iteration starts directly from i=1.

Iteration : For every given G, a unitary matrix D can be calculated by Algorithm 1. If this unitary matrix satisfies the first condition of a nilpotent interactor matrix

$$\lim_{q^{-1}\to 0} DT = \lim_{q^{-1}\to 0} \tilde{T} = K, \tag{2.22}$$

it is an interactor matrix for the given transfer function matrix T because the second condition

$$|D| = q^r \tag{2.23}$$

will be satisfied explicitly. Otherwise, increase i to expand the block matrix G, calculate a unitary matrix for this new block matrix and check whether the result satisfies the Equation (2.22). Repeat this iterative process until an interactor matrix is calculated.■

2.1.2 Matlab Function

One Matlab function, *interactor*, is programmed for Algorithm 2. The input of this function can be a discrete or continuous transfer function matrix T or state-space matrices A, B, C, D. The output is a unitary interactor matrix D.

- For a discrete transfer function matrix T, use command *[D] = interactor(T,'dis')*.

- For a continues transfer function matrix T, use command *[D] = interactor(T,'con', Ts, Method).* The meaning of input parameters Ts and $Method$ is the same as that of Matlab function "c2d". The default value of Ts is 1 and the default value of $Method$ is 'zoh' whenever they are omitted.

- For a discrete state-space model, use command *[D] = interactor(A, B, C, D, 'dis').*

- For a continuous state-space model, use command *[D] = interactor(A, B, C, D, 'con', Ts, Method).*

- The 'con' or 'dis' parameter can NOT be omitted.

2.1.3 Examples

The *interactor* function has been tested on lots of examples. Three of them are given in this part to show the correctness of this function.

Example 1. For the example illustrated on Page 21 of Huang and Shah (1999)

$$T = \begin{bmatrix} \frac{q^{-2}}{1+q^{-1}} & \frac{0.5q^{-3}}{1+2q^{-1}} \\ \frac{0.5q^{-1}}{1+3q^{-1}} & \frac{q^{-1}}{1+4q^{-1}} \end{bmatrix},$$

a unitary interactor matrix is evaluated as

$$D = \begin{bmatrix} 0 & q \\ q^2 & 0 \end{bmatrix},$$

which is different from the result presented in Huang and Shah (1999)

$$D' = \begin{bmatrix} q^2 & 0 \\ 0 & q \end{bmatrix}.$$

We can show that like D', D is also a unitary interactor matrix for the given system T because

$$\lim_{q^{-1}\to 0} DT = \begin{bmatrix} \frac{0.5}{1+q^{-1}} & \frac{1}{1+4q^{-1}} \\ \frac{1}{1+q^{-1}} & \frac{0.5q^{-1}}{1+2q^{-1}} \end{bmatrix} = \begin{bmatrix} 0.5 & 1 \\ 1 & 0 \end{bmatrix}$$

is full rank and

$$D^T(q^{-1})D(q) = \begin{bmatrix} 0 & q^{-2} \\ q^{-1} & 0 \end{bmatrix} \begin{bmatrix} 0 & q \\ q^2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

This example shows that, for a given transfer matrix T, the unitary interactor matrix is non-unique.

Example 2. For another example on the same page of Huang and Shah (1999)

$$T = \begin{bmatrix} \frac{q^{-1}}{1+q^{-1}} & \frac{q^{-1}}{1+2q^{-1}} \\ \frac{q^{-1}}{1+3q^{-1}} & \frac{q^{-1}}{1+4q^{-1}} \end{bmatrix},$$

one interactor matrix is calculated as

$$D = \begin{bmatrix} 0.5q^2 + 0.5q & -0.5q^2 + 0.5q \\ 0.5q^3 - 0.5q^2 & -0.5q^3 - 0.5q^2 \end{bmatrix}.$$

Compared with the result in Huang and Shah (1999)

$$D = \begin{bmatrix} 0.5q^2 + 0.5q & -0.5q^2 + 0.5q \\ -0.5q^3 + 0.5q^2 & 0.5q^3 + 0.5q^2 \end{bmatrix},$$

we can see that these two matrices only have difference in the signs of the elements of the second row. This difference does not have any effect on the calculation of an interactor matrix.

Example 3. Gao *et al.* (2003) presented two performance assessment case studies of industrial multivariate MPC based controllers at Mitsubishi chemical complex in Mizushima, Japan. The open-loop process model of a para-xylene distillation unit is

$$T = \begin{bmatrix} q^{-1} & 0 & 0 & 0 & 0 & 0 \\ \frac{-0.00809q^{-1}-0.1408q^{-2}}{1-0.2845q^{-1}-0.6740q^{-2}} & \frac{-0.06038q^{-1}+0.2498q^{-2}}{1-1.69q^{-1}+0.6978q^{-2}} & \frac{-0.02414q^{-1}-0.266q^{-2}}{1-0.8188q^{-1}-0.153q^{-2}} & 0 & \frac{-0.1297q^{-1}+0.4256q^{-2}}{1-1.301q^{-1}+0.3159q^{-2}} & 0 \\ \frac{-0.02498q^{-1}+0.03939q^{-2}}{1-1.868q^{-1}+0.8784q^{-2}} & \frac{0.1092q^{-1}-0.1551q^{-2}}{1-1.873q^{-1}+0.8796q^{-2}} & \frac{0.01148q^{-1}+0.01025q^{-2}}{1-1.876q^{-1}+0.8823q^{-2}} & 0 & \frac{-0.0408q^{-1}-0.008103q^{-2}}{1-1.814q^{-1}+0.8235q^{-2}} & 0 \\ 0 & 0 & 0 & 0 & \frac{-0.1968q^{-1}}{1-0.99q^{-1}} & 0 \\ \frac{-0.1464q^{-2}}{1-0.99q^{-1}} & \frac{0.34q^{-2}}{1-0.99q^{-1}} & 0 & \frac{-1.399q^{-2}}{1-0.99q^{-1}} & \frac{-0.4911q^{-2}}{1-0.99q^{-1}} & \frac{3.766q^{-2}}{1-0.99q^{-1}} \end{bmatrix}$$

The estimated unitary interactor matrix is

$$D = \begin{bmatrix} -0.05925q & -0.7234q & -0.4q & 0.5597q & 0 & 0 \\ 0.006448q & 0.425q & -0.9003q & -0.09344q & 0 & 0 \\ -0.02357q & -0.3104q & -0.09754q & -0.4733q & 0 & 0.8183q \\ 0.03355q & 0.4418q & 0.1388q & 0.6738q & 0 & 0.5748q \\ -0.9974q & 0.06791q & 0.02491q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -q^2 & 0 \end{bmatrix},$$

which is the same as that published in Gao *et al.* (2003) except the difference in the signs of the third row elements.

2.2 FCOR Algorithm

Once the interactor matrix is known, the performance benchmark based on minimum variance control can be extended to multivariable control systems. Harris *et al.* (1996)

developed a method which requires the spectral factorization of the interactor matrix. Compared with it, the FCOR algorithm presented in Huang and Shah (1997b) is simpler. It is truly an extension of SISO FCOR algorithm to MIMO case.

2.2.1 Algorithm

A MIMO process can be modeled as

$$Y_t = TU_t + Na_t, \tag{2.24}$$

where T and N are proper, rational transfer function matrices for the plant and noise, respectively; Y_t is an output vector and U_t an input vector. For stochastic systems, a_t represents a white noise vector with zero mean and covariance matrix Σ_a.

Furthermore, if T is a proper, full rank transfer function matrix, a unitary interactor matrix D can be evaluated and $DT = \tilde{T}$, where $\tilde{T}$ is the delay-free transfer function matrix of T. Therefore, Equation (2.24) can be expressed as

$$Y_t = TU_t + Na_t = D^{-1}\tilde{T}U_t + Na_t. \tag{2.25}$$

Premultiplying both sides of Equation (2.25) by $q^{-d}D$, where d is the order of interactor matrix D as defined before, gives

$$q^{-d}DY_t = q^{-d}\tilde{T}U_t + q^{-d}DNa_t. \tag{2.26}$$

Let $\tilde{Y}_t = q^{-d}DY_t$ and $\tilde{N} = q^{-d}DN$, Equation(2.26) becomes

$$\tilde{Y}_t = q^{-d}\tilde{T}U_t + \tilde{N}a_t. \tag{2.27}$$

Huang and Shah (1997b) showed that since D is a unitary interactor matrix, the minimum variance control law which minimizes the objective function of the interactor-filtered variable $\tilde{Y}_t$, $J_1 = E(\tilde{Y}_t^T\tilde{Y}_t)$, also minimizes the objective function of the original variable Y_t, $J_2 = E(Y_t^TY_t)$, and $J_1 = J_2$, which means that $E(\tilde{Y}_t^T\tilde{Y}_t) = E(Y_t^TY_t)$.

Under feedback control law $U_t = -Q\tilde{Y}_t$, where Q is the transfer function matrix of a controller, the closed-loop transfer function can be expressed as

$$\tilde{Y}_t = -q^{-d}\tilde{T}Q\tilde{Y}_t + \tilde{N}a_t. \tag{2.28}$$

Using Diophantine identity, $\tilde{N}$ can be decomposed into two parts:

$$\tilde{N} = F + q^{-d}R \tag{2.29}$$

where $F = F_0 + F_1q^{-1} + \cdots + F_{d-1}q^{-(d-1)}$ and R is the remaining proper and rational transfer function matrix. Substituting Equation (2.29) into Equation (2.28) yields

$$\tilde{Y}_t = (q^dI + \tilde{T}Q)^{-1}q^d(F + q^{-d}R)a_t. \tag{2.30}$$

Simplifying this equation, finally we get

$$\tilde{Y}_t = Fa_t + La_{t-d}, \tag{2.31}$$

where

$$L = R - \tilde{T}(I + q^{-d}Q\tilde{T})^{-1}Q\tilde{N} \tag{2.32}$$

is a proper rational transfer function matrix.

Since F is independent of the controller Q, the two terms on the right hand side of Equation (2.31) are independent. Therefore, the following two equations hold

$$\begin{aligned} Var(\tilde{Y}_t) &\geq Var(Fa_t) \\ E[\tilde{Y}_t^T\tilde{Y}_t] &\geq tr(Var(Fa_t)) \end{aligned} \tag{2.33}$$

and the equality holds only under minimum variance control law, which lets $L = 0$. As a result, the process output under minimum variance control, Fa_t, is feedback controller-invariant.

Through time series analysis performed on a set of closed-loop routine operating data, we can get the estimation of the noise model N. Filter N by $q^{-d}D$ and express $\tilde{N}$ as

$$\tilde{N} = q^{-d}DN = F_0 + F_1q^{-1} + \cdots + F_{d-1}q^{-(d-1)} + Rq^{-d}, \tag{2.34}$$

then, we can have the estimation values of F: $F_0, F_1, \cdots, F_{d-1}$. Putting these values into Equation (2.33),

$$E[\tilde{Y}_t^T\tilde{Y}_t]|_{mvc} = tr(Var(Fa_t)) = tr(F_0\Sigma_aF_0^T + \cdots + F_{d-1}\Sigma_aF_{d-1}^T). \tag{2.35}$$

Therefore, the performanc $\frac{minimum\quad covariance}{actual\quad covariance} = \frac{E[Y_t^TY_t]|_{mvc}}{E[Y_t^TY_t]}$ expressed as

$$\begin{aligned} \eta = \frac{mini}{act} &= \frac{E[\tilde{Y}_t^T\tilde{Y}_t]|_{mvc}}{E[Y_t^TY_t]} = \frac{tr(Var(Fa_t))}{tr(E[Y_t^TY_t])} \\ &= \frac{E[\tilde{Y}_t^T\tilde{Y}_t]|_{mvc}}{E[Y_t^TY_t]} = \frac{tr(Var(Fa_t))}{tr(E[Y_t^TY_t])} \end{aligned} \tag{2.36}$$

and the performance index of every individual output is defined as

$$\eta_n = \frac{diag(Y_t^T Y_t)|_{mvc}}{diag(Y_t^T Y_t)}, \tag{2.37}$$

where η_n is a $n \times 1$ vector and n is the output number.

Because under minimum control law,

$$diag(Y_t^T Y_t) \neq diag(\tilde{Y}_t^T \tilde{Y}_t) \tag{2.38}$$

even though

$$tr(Y_t^T Y_t) = tr(\tilde{Y}_t^T \tilde{Y}_t), \tag{2.39}$$

we should find a way to compute the value of $diag(Y_t^T Y_t)|_{mvc}$.

Since

$$\tilde{Y}_t = q^{-d} D Y_t \Rightarrow Y_t = q^d D^{-1} \tilde{Y}_t$$

and from the definition of unitary interactor matrix,

$$\begin{aligned} D^T(q^{-1})D(q) = I &\Rightarrow D^{-1}(q) = D^T(q^{-1}) \\ D^{-1} = (D_0 q^d + \cdots + D_{d-1}q)^{-1} &= D_0^T q^{-d} + \cdots + D_{d-1}^T q^{-1} \end{aligned} \tag{2.40}$$

we can get

$$\begin{aligned} Y_t &= q^d (D_0^T q^{-d} + \cdots + D_{d-1}^T q^{-1})(F_0 + \cdots + F_{d-1} q^{-(d-1)}) a_t \\ &\triangleq E a_t \\ &= (E_0 + E_1 q^{-1} + \cdots + E_{d-1} q^{-(d-1)}) a_t \end{aligned} \tag{2.41}$$

$$[E_0, E_1, \cdots E_{d-1}] \triangleq [D_0^T, D_1^T, \cdots, D_{d-1}^T] \begin{bmatrix} F_0 & F_1 & \cdots & F_{d-1} \\ F_1 & F_2 & \cdots & \\ \vdots & \vdots & & \\ \vdots & F_{d-1} & & \\ F_{d-1} & & & \end{bmatrix}. \tag{2.42}$$

Thus,

$$\begin{aligned} E[Y_t^T Y_t]|_{mvc} &= tr(Var(Ea_t)) = tr(E_0 \Sigma_a E_0^T + \cdots + E_{d-1} \Sigma_a E_{d-1}^T) \\ diag(Y_t^T Y_t)|_{mvc} &= diag(Var(Ea_t)) = diag(E_0 \Sigma_a E_0^T + \cdots + E_{d-1} \Sigma_a E_{d-1}^T). \end{aligned} \tag{2.43}$$

Substituting Equation (2.43) into Equation (2.37), finally, the performance index of every individual output is calculated as

$$\eta_n = \frac{diag(Y_t^T Y_t)|_{mvc}}{diag(Y_t^T Y_t)} = \frac{diag(E_0 \Sigma_a E_0^T + \cdots + E_{d-1} \Sigma_a E_{d-1}^T)}{diag(Y_t^T Y_t)}. \tag{2.44}$$

2.2.2 Matlab Function

One Matlab function, *fcor*, is programmed for this algorithm.

$$[Eta, Eta_n] = fcor(OutputData, InteractorMatrix)$$

The input and output parameters are explained as follow:

$OutputData$: A set of routine operating output data used for performance assessment.

$InteractorMatrix$: A unitary interactor matrix of the given system calculated by *interactor* function introduced in Section 2.1 or estimated by other methods.

Eta : Performance index of the overall system.

Eta_n : Performance index of individual output loop, a $n \times 1$ vector.

2.2.3 Simulation Example

One of the simulation examples designed for the test of *fcor* function is illustrated in this part, which is on Page 90 of Huang and Shah (1999).

$$T = \begin{bmatrix} \frac{q^{-1}}{1-0.4q^{-1}} & \frac{Kq^{-2}}{1-0.1q^{-1}} \\ \frac{0.3q^{-1}}{1-0.1q^{-1}} & \frac{q^{-2}}{1-0.8q^{-1}} \end{bmatrix}$$
$$N = \begin{bmatrix} \frac{1}{1-0.5q^{-1}} & \frac{-0.6}{1-0.5q^{-1}} \\ \frac{0.5}{1-0.5q^{-1}} & \frac{1.0}{1-0.5q^{-1}} \end{bmatrix}$$
$$Q = \begin{bmatrix} \frac{0.5-0.20q^{-1}}{1-0.5q^{-1}} & 0 \\ 0 & \frac{0.25-0.200q^{-1}}{(1-0.5q^{-1})(1+0.5q^{-1}} \end{bmatrix}$$

T is a 2×2 MIMO process, Q is the controller and N is the disturbance transfer function matrix. The white noise excitation, a_t, is a two-dimensional normally-distributed white noise sequence with $\Sigma_a = I$.

By the *interactor* function, the interactor matrix of T is calculated as

$$D = \begin{bmatrix} -0.9578q & -0.2873q \\ 0.2873q^2 & 0.9578q^2 \end{bmatrix}.$$

From the interactor matrix D and the noise model N, the theoretical values of the performance index can be calculated. For the detailed calculation procedure, readers are refer to Huang and Shah (1999). The parameter K in the transfer function matrix of T increases from 0 to 10. A set of closed-loop data is simulated for every K. With the

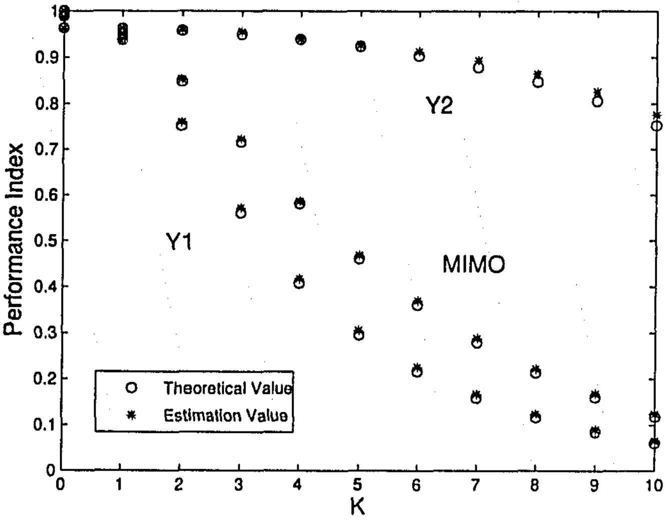


Figure 2.1: Performance assessment result by the FCOR algorithm

interactor matrix and simulated closed-loop data, the performance index of the overall MIMO system and that of two output loops, $Y1$ and $Y2$, are estimated. The performance assessment results are shown in Figure 2.1. From this figure, we can see that the theoretical values and the estimation values match well for different K, which show that both the theory and function of the *fcor* function are correct.

2.3 Subspace Method

So far, the interactor matrix can be estimated from the first few Markov parameters of a process by the algorithm presented in section 2.1. When the interactor matrix is known a *prior*, the performance index can be estimated from a set of closed-loop routine operating data. Although the interactor matrix is a meaningful generation of time delay term in the SISO case to MIMO case, its concept and calculation is an obstacle, especially, for industrial users. Therefore, next challenge for the performance assessment of MIMO system is the calculation of performance index without the knowledge of interactor matrix.

Ko and Edgar (2001) developed a simple method which integrates the calculation of interactor matrix and the estimation of performance index together and simplifies them to an explicit "one-shot" solution. Huang *et al.* (2004) presented a subspace method to calculate the performance index directly from input/output data without the knowledge of interactor matrix.

The subspace method of identification (SMI) has been an active research area since the beginning of 1990s. Compared with the traditional identification method, such as, the prediction error method (PEM), the advantages of SMI are numerical reliability and computational simplicity.

2.3.1 Algorithm

This section is cited from Huang et al. (2004).

If we describe a linear time-invariant system with l-inputs, m-outputs and n-states using the innovations state space representation as

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + Ke_t \\ y_t &= Cx_t + e_t \end{aligned} \tag{2.45}$$

where the dimensions of system state space matrices A, B, C and K are $n \times n, n \times l, m \times n$ and $n \times m$, respectively. K is the Kalman filter gain and e_k is an unknown innovation or white-noise sequence with covariance matrix S.

Stack the input u_t into two block Hankel matrices,

$$U_p = \begin{bmatrix} u_1 & u_2 & \cdots & u_j \\ u_2 & u_3 & \cdots & u_{j+1} \\ \vdots & \vdots & \vdots & \vdots \\ u_N & n_{N+1} & \cdots & u_{N+j-1} \end{bmatrix} \quad \mathit{and} \quad U_f = \begin{bmatrix} u_{N+1} & u_{N+2} & \cdots & u_{N+j} \\ u_{N+2} & u_{N+3} & \cdots & u_{N+j+1} \\ \vdots & \vdots & \vdots & \vdots \\ u_{2N} & u_{2N+1} & \cdots & u_{2N+j-1} \end{bmatrix}$$

where p denotes the past and f denotes the future. Similarly, the output y_t and the white noise e_t are stacked into two block Hankel matrices, respectively,

$$Y_p = \begin{bmatrix} y_1 & y_2 & \cdots & y_j \\ y_2 & y_3 & \cdots & y_{j+1} \\ \vdots & \vdots & \vdots & \vdots \\ y_N & y_{N+1} & \cdots & y_{N+j-1} \end{bmatrix} \quad \mathit{and} \quad Y_f = \begin{bmatrix} y_{N+1} & y_{N+2} & \cdots & y_{N+j} \\ y_{N+2} & y_{N+3} & \cdots & y_{N+j+1} \\ \vdots & \vdots & \vdots & \vdots \\ y_{2N} & y_{2N+1} & \cdots & y_{2N+j-1} \end{bmatrix},$$

$$E_p = \begin{bmatrix} e_1 & e_2 & \cdots & e_j \\ e_2 & e_3 & \cdots & e_{j+1} \\ \vdots & \vdots & \vdots & \vdots \\ e_N & e_{N+1} & \cdots & e_{N+j-1} \end{bmatrix} \quad \mathit{and} \quad E_f = \begin{bmatrix} e_{N+1} & e_{N+2} & \cdots & e_{N+j} \\ e_{N+2} & e_{N+3} & \cdots & e_{N+j+1} \\ \vdots & \vdots & \vdots & \vdots \\ e_{2N} & e_{2N+1} & \cdots & e_{2N+j-1} \end{bmatrix}.$$

The past and future state sequences are defined as

$$X_p = [\ x_1 \quad x_2 \quad \cdots \quad x_j\ ] \quad and \quad X_f = [\ x_{N+1} \quad x_{N+2} \quad \cdots \quad x_{N+j}\ ].$$

Notice that, each block element in the above matrices is a column vector, i.e.

$$u_t = \begin{bmatrix} u_t^1 \\ u_t^2 \\ \vdots \\ u_t^l \end{bmatrix}, y_t = \begin{bmatrix} y_t^1 \\ y_t^2 \\ \vdots \\ y_t^m \end{bmatrix}, e_t = \begin{bmatrix} e_t^1 \\ e_t^2 \\ \vdots \\ e_t^m \end{bmatrix}, x_t = \begin{bmatrix} x_t^1 \\ x_t^2 \\ \vdots \\ x_t^n \end{bmatrix}.$$

Through recursive substitution of Equation (2.45), three important equations in SMI are achieved:

$$\begin{aligned} Y_p &= \Gamma_N X_p + H_N U_p + H_N^s E_p && (2.46) \\ Y_f &= \Gamma_N X_f + H_N U_f + H_N^s E_f && (2.47) \\ X_f &= A^N X_p + \Delta_N U_p + \Delta_N^s E_p && (2.48) \end{aligned}$$

where,

$$\Gamma_N = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{bmatrix}$$

is the extended observability matrix,

$$H_N = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ CB & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ CA^{N-2}B & CA^{N-3}B & \cdots & 0 \end{bmatrix} \quad and \quad H_N^s = \begin{bmatrix} I_m & 0 & \cdots & 0 \\ CK & I_m & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ CA^{N-2}K & CA^{N-3}K & \cdots & I_m \end{bmatrix}$$

are the lower triangular Toeplitz matrices containing the Markov parameters corresponding to the deterministic input u_k and the unknown stochastic input e_k, respectively,

$$\Delta_N = [\ A^{N-1}B \quad A^{N-2}B \quad \cdots \quad B\ ] \quad and \quad \Delta_N^s = [\ A^{N-1}K \quad A^{N-2}K \quad \cdots \quad K\ ]$$

are reversed extended controllability matrices. In subspace identification literature, the following short-hand notation is often used:

$$W_p = \begin{pmatrix} Y_p \\ U_p \end{pmatrix}.$$

Subspace method of identification always has two steps. The first step is the projection of certain subspace generated from the data in order to get the estimation of subspace matrices Γ_N and H_N and/or X_f. The second step is the estimation of system state space matrices A, B, C and K from Γ_N and H_N or X_f. The method introduced in this section only needs the estimation of subspace matrices, without the knowledge of system state space matrices, so it is also a model-free method.

Ko and Edgar (2001) showed that, under minimum variance control, the behavior of the outputs under infinite horizon is the same as that under a finite horizon if the rank of the plant transfer function $T(q^{-1})$ is equal to the number of outputs for almost all q. Moreover, it can be expressed as

$$y_t|_{mvc} = \sum_{i=0}^{N-1} F_i e_{t-i}, \tag{2.49}$$

in which

$$\begin{pmatrix} F_0 \\ \vdots \\ F_{N-1} \end{pmatrix} = (I - H_N H_N^{\dagger}) H_{N,1}^{s}.$$

$H_{N,1}^{s}$ is the first block column of H_N^s. Thus, the MVC benchmark can be written as

$$J_{mvc} = tr(I - H_N H_N^{\dagger}) H_{N,1}^{s} S H_{N,1}^{sT} (I - H_N H_N^{\dagger})^T \tag{2.50}$$

From above equation, we can see that the MVC benchmark can be calculated directly from H_N and H_N^s which are comprised of first few Markov parameters of the plant and noise model, respectively. The algorithm developed by Huang *et al.* (2004) is based on subspace method, by which H_N can be estimated from a set of open loop experimental data and H_N^s can be estimated from a set of closed-loop routine operating data. The detailed procedure is explained as follow.

Estimation of H_N

To get the estimation of H_N, Equation (2.47) can be rewriten as

$$Y_f - \Gamma_N X_f = H_N U_f + H_N^s E_f. \tag{2.51}$$

Multiply above equation with $U_f^T (U_f U_f^T)^{-1}$,

$$(Y_f - \Gamma_N X_f) U_f^T (U_f U_f^T)^{-1} = H_N U_f U_f^T (U_f U_f^T)^{-1} + H_N^s E_f U_f^T (U_f U_f^T)^{-1}. \tag{2.52}$$

Because U_f and E_f are independent under open loop condition, Equation (2.52) can be simplified as

$$H_N = (Y_f - \Gamma_N X_f)U_f^T(U_fU_f^T)^{-1} = (Y_f - \Gamma_N X_f)U_f^{\dagger}, \qquad (2.53)$$

where "†" means the pseudo inverse.

In subspace literature, two projections are frequently used. One of them is oblique projection and the other is orthogonal projection.

Definition 4. *The oblique projection of the row space of $A \in R^{p\times j}$ along the row space of $B \in R^{q\times j}$ on the row space of $C \in R^{r\times j}$ are defined as $A/_BC$ and can be calculated via*

$$A/_BC = A\begin{pmatrix} C \\ B \end{pmatrix}^{\dagger}(:,1:r)C \qquad (2.54)$$

where $(:,1:r)$ denotes the extraction of the matrix product resulted from $A\begin{pmatrix} C \\ B \end{pmatrix}^{\dagger}$ from first column to the rth column.

For the oblique projection, two properties are important:

$$A/_AC = 0 \qquad (2.55)$$

$$A/_BA = A. \qquad (2.56)$$

Definition 5. *The orthogonal projection of the row space of A onto the row space of B is denoted by A/B and can be calculated through $A/B = AB^{\dagger}B$.*

Performing an oblique projection of Equation (2.47) results

$$Y_f/_{U_f}W_p = \Gamma_N X_f/_{U_f}W_p + H_N U_f/_{U_f}W_p + H_N^s E_f/_{U_f}W_p. \qquad (2.57)$$

The second item of right hand side of the above equation is zero due to the property of oblique projection $A/_AC = 0$ and the last item is also zero since E_f, U_f and W_p are independent under open loop. Therefore Equation (2.57) becomes

$$Y_f/_{U_f}W_p = \Gamma_N X_f/_{U_f}W_p. \qquad (2.58)$$

From Equation (2.46), E_p can be express as

$$E_p = (H_N^s)^{-1}Y_p - (H_N^s)^{-1}\Gamma_N X_p - (H_N^s)^{-1}H_N U_p \qquad (2.59)$$

and substituting Equation (2.59) into Equation (2.48) results

$$\begin{aligned} X_f &= A^N X_p + \Delta_N U_p + \Delta_N^s(H_N^s)^{-1}Y_p - \Delta_N^s(H_N^s)^{-1}\Gamma_N X_p - \Delta_N^s(H_N^s)^{-1}H_N U_p \\ &= (A^N - \Delta_N^s(H_N^s)^{-1}\Gamma_N)X_p + [\Delta_N^s(H_N^s)^{-1}|\Delta_N - \Delta_N^s(H_N^s)^{-1}H_N]W_p. \qquad (2.60) \end{aligned}$$

Then, perform an oblique projection of Equation (2.60) on to W_p through U_f,

$$\begin{aligned} X_f/_{U_f}W_p &= (A^N - \Delta_N^s(H_N^s)^{-1}\Gamma_N)X_p/_{U_f}W_p \\ &\quad +[\Delta_N^s(H_N^s)^{-1}|\Delta_N - \Delta_N^s(H_N^s)^{-1}H_N]W_p/_{U_f}W_p \end{aligned} \tag{2.61}$$

where $X_f/_{U_f}W_p$, defined as $\hat{X}_f$, is an optimal prediction of the state by a Kalman filter with $X_p/_{U_f}W_p$ as intial state. Now, Equation (2.58) can be rewritten as

$$Y_f/_{U_f}W_p = \Gamma_N\hat{X}_f \tag{2.62}$$

and substituting into Equation (2.53), finally, the estimation of H_N is

$$\hat{H}_N = (Y_f - \Gamma_N\hat{X}_f)U_f^{\dagger} = (Y_f - Y_f/_{U_f}W_p)U_f^{\dagger}. \tag{2.63}$$

From above equation, we can see that $\hat{H}_N$ can be estimated directly by a set of open loop experiment data.

Estimation of H_N^s

Since the noise model under open loop condition is different from the model in closed-loop condition, we should use the close-loop data to get the estimation of H_N^s. Like the Equation (2.47) for the open loop case, we can write one for the close-loop condition and let $U_f = 0$,

$$Y_f^{cl} = \Gamma_N^{cl}X_f^{cl} + H_N^{cls}E_f^{cl} \tag{2.64}$$

where Γ_N^{cl} and H_N^{cls} are the same as Γ_N and H_N^s in Equation (2.47) but the superscript *cl* stands for close-loop condition.

Performing an orthogonal projection of Equation (2.64) onto Y_p^{cl} results

$$Y_f^{cl}/Y_p^{cl} = \Gamma_N^{cl}X_f^{cl}/Y_p^{cl} + H_N^{cls}E_f^{cl}/Y_p^{cl}. \tag{2.65}$$

Since the future noise E_f^{cl} and past output Y_p^{cl} are independent even under closed-loop condition, the last item of right hand side of above equation equals zero and

$$Y_f^{cl}/Y_p^{cl} = \Gamma_N^{cl}X_f^{cl}/Y_p^{cl} = \Gamma_N^{cl}\hat{X}_f^{cl} = Y_f^{cl}Y_p^{cl\dagger}Y_p^{cl}. \tag{2.66}$$

The Lemma 2 of Huang *et al.* (2004) proved that, under an arbitrary stable control,

$$H_N^{cls} = L_h diag(\Phi^T S^{-\frac{1}{2}}, \cdots, \Phi^T S^{-\frac{1}{2}}) \tag{2.67}$$

where Φ is a unitary matrix and L_h is a lower triangular matrix resulted from a QR decomposition according to the following equation:

$$L_h Q = \frac{1}{\sqrt{j}}(Y_f^{cl} - \Gamma_N^{cl} X_f^{cl}) \tag{2.68}$$

where Q is a unitary matrix.

Substituting Equation (2.66) into Equation (2.68), the estimation of L_h is

$$\hat{L}_h Q = \frac{1}{\sqrt{j}}(Y_f^{cl} - \Gamma_N^{cl} \hat{X}_f^{cl}) = \frac{1}{\sqrt{j}}(Y_f^{cl} - Y_f^{cl} Y_p^{cl\dagger} Y_p^{cl}) \tag{2.69}$$

Estimation of J_{mvc} and performance index

Now, the output variance under minimum variance control can be written as

$$\begin{aligned} J_{mvc} &= tr(I - H_N H_N^{\dagger}) H_{N,1}^{cls} S (H_{N,1}^{cls})^T (I - H_N H_N^{\dagger})^T \\ &= tr(I - H_N H_N^{\dagger}) L_{h,1} \Phi^T S^{-\frac{1}{2}} S S^{-\frac{1}{2}} \Phi L_{h,1}^T (I - H_N H_N^{\dagger})^T \\ &= tr(I - H_N H_N^{\dagger}) L_{h,1} L_{h,1}^T (I - H_N H_N^{\dagger})^T \end{aligned} \tag{2.70}$$

where $H_{N,1}^{cls}$ is the first block column of H_N^{cls} and $L_{h,1}$ is the first block column of L_h. Substituting the estimation of H_N and L_h

$$\hat{H}_N = (Y_f - Y_f / _{U_f} W_p) U_f^{\dagger} \tag{2.71}$$

$$\hat{L}_h = \frac{1}{\sqrt{j}}(Y_f^{cl} - Y_f^{cl} Y_p^{cl\dagger} Y_p^{cl}) \hat{Q}^T \tag{2.72}$$

into Equation (2.70),

$$\hat{J}_{mvc} = tr(I - \hat{H}_N \hat{H}_N^{\dagger}) \hat{L}_{h,1} \hat{L}_{h,1}^T (I - \hat{H}_N \hat{H}_N^{\dagger})^T. \tag{2.73}$$

The overall MIMO performance index using minimum variance control as benchmark can be expressed as

$$\eta = \frac{\hat{J}_{mvc}}{tr(E[Y_t^T Y_t])}. \tag{2.74}$$

For every individual output loop, the performance index can be calculated as

$$\eta_n = \frac{diag[Y_t^T Y_t]_{mvc}}{diag[Y_t^T Y_t]}. \tag{2.75}$$

Since $(I - \hat{H}_N \hat{H}_N^{\dagger}) \hat{L}_{h,1} \hat{L}_{h,1}^T (I - \hat{H}_N \hat{H}_N^{\dagger})^T$ is a $Nm \times Nm$ block matrix in which every block is a $m \times m$ matrix, the $[Y_t^T Y_t]_{mvc}$ can be calculated as the summation of main diagonal block matrices of $(I - \hat{H}_N \hat{H}_N^{\dagger}) \hat{L}_{h,1} \hat{L}_{h,1}^T (I - \hat{H}_N \hat{H}_N^{\dagger})^T$.

2.3.2 Matlab Function

One Matlab function, *pass*, is programmed for this subspace method. *Pass* stands for Performance Assessment by SubSpace method.

The outputs of this function are performance indices of overall system, η, and that of individual output loop, η_n. Three kinds of performance index can be calculated by this function: estimation value, theoretical value and mixed value.

Estimation value

As introduced in last part, the performance index can be estimated directly from a set of open loop experimental data and a set of closed-loop routine operating data by this subspace method. Therefore, this performance index is named as estimation value.

For the estimation value, use command

$$[Eta, Eta_n] = pass(U_{open}, Y_{open}, Y_{close}, \text{`}est\text{'}).$$

The input parameters U_{open} and Y_{open} are a set of open loop experimental input/output data and Y_{close} is a set of closed-loop routine operating output data.

Theoretical value

On the other hand, if the models of plant, controller and noise are available, the theoretical value of H_N and H_N^{cls} are known because they are comprised of Markov parameter of plant and closed-loop system, respectively. Then, the theoretical value of J_{mvc} can be calculated as

$$J_{mvc} = tr(I - H_N H_N^{\dagger}) H_{N,1}^{cls} S (H_{N,1}^{cls})^T (I - H_N H_N^{\dagger})^T. \tag{2.76}$$

For the theoretical value, use command

$$[Eta, Eta_n] = pass(plant, controller, noise, S, \text{`}the\text{'}).$$

The input parameters *plant*, *controller*, *noise* are models of plant, controller and noise, respectively, expressed in transfer function matrix and S is the intensity matrix of square root of covariance matrix of white noise.

Mixed value

If only the plant model is available, "mixed value" of J_{mvc} can be calculated as

$$\hat{J}_{mvc} = tr(I - H_N H_N^{\dagger})\hat{L}_{h,1}\hat{L}_{h,1}^T(I - H_N H_N^{\dagger})^T, \qquad (2.77)$$

where H_N is calculated from plant model and $\hat{L}_h$ is estimated from a set of closed-loop data.

For the mixed value, use command

$$[Eta, Eta_n] = pass(plant, Y_{close}, \text{'}mix\text{'}).$$

Implementation Issues

- After get the estimation of H_N, we need to perform data cleaning. Because H_N consists of Markov parameter of the plant, for a causal system, the elements located in the upper block triangular of $\hat{H}_N$ should be zeros theoretically. However, they are some small numbers due to the effect of noise. The existence of these small numbers will affect the result of this subspace method significantly and sometimes this method may fail. Thus, first we need to set all the upper block triangular elements to zero and calculate statistical data, such as, mean value and confidence intervals, from these elements. Next, those elements located in the lower block triangular of $\hat{H}_N$ are processed based on these statistical data.

- For SMI, one key point is the selection of appropriate N for the data Hankel matrices U_p, U_f, Y_p and Y_f. If N is too small, the accuracy is poor. If N is too large, on the other hand, the computational speed becomes extreme slow. In this algorithm, the rank of $(I - H_N H_N^{\dagger})$ will not change when N increases to a certain number. Examples about this will be shown in the following example part. Using this property, the algorithm can automatically find a suitable N for given data. The initial value of N is set to 2. In the iterative process, for every N, the rank of $(I - H_N H_N^{\dagger})$ is calculated for three consecutive numbers, such as, $N, N+1$ and $N+2$. If these three ranks are same, current N is a appropriate value for the given data. Otherwise, increase N and the incremental step is 5. In the case that the rank of $(I - H_N H_N^{\dagger})$ still changes when N is a quite large number, such as 100, the algorithm will stop automatically and an error massage is given.

2.3.3 Simulation Example

In this part, the example that has been employed to illustrate the FCOR algorithm is used to test this subspace performance assessment method.

$$T = \begin{bmatrix} \frac{q^{-1}}{1-0.4q^{-1}} & \frac{Kq^{-2}}{1-0.1q^{-1}} \\ \frac{0.3q^{-1}}{1-0.1q^{-1}} & \frac{q^{-2}}{1-0.8q^{-1}} \end{bmatrix}$$
$$N = \begin{bmatrix} \frac{1}{1-0.5q^{-1}} & \frac{-0.6}{1-0.5q^{-1}} \\ \frac{0.5}{1-0.5q^{-1}} & \frac{1.0}{1-0.5q^{-1}} \end{bmatrix}$$
$$Q = \begin{bmatrix} \frac{0.5-0.20q^{-1}}{1-0.5q^{-1}} & 0 \\ 0 & \frac{0.25-0.200q^{-1}}{(1-0.5q^{-1})(1+0.5q^{-1}} \end{bmatrix}$$

Four kinds of performance index are shown in Figure 2.2. The theoretical and estimation values of FCOR algorithm are the same as those shown in Figure 2.1. For the subspace method, the theoretical and estimation values are calculated by the function *pass*. As we can see, they all match well. Both theoretical values are exactly same, which shows match of the two methods in theory. Compared with the FCOR algorithm, moreover, this subspace method has obvious advantage. No concept and computation, such as, interactor matrix, is needed. We can directly estimate the performance index from data, a set of open loop experiment data and a set of closed-loop routine operating data. This is the reason that this method is called an "one-shot" data-driven approach.

The next step is to demonstrate the property that the rank of $(I - \hat{H}_N\hat{H}_N^{\dagger})$ will not change when N increase to a certain number. Set $K = 1$ and the dimension of $(I - \hat{H}_N\hat{H}_N^{\dagger})$ is $Nm \times Nm$.

When N equals 2,

$$I - \hat{H}_N\hat{H}_N^{\dagger} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.0957 & -0.2942 \\ 0 & 0 & -0.2942 & 0.9043 \end{bmatrix}$$

$rank(I - \hat{H}_N\hat{H}_N^{\dagger}) = 3.$

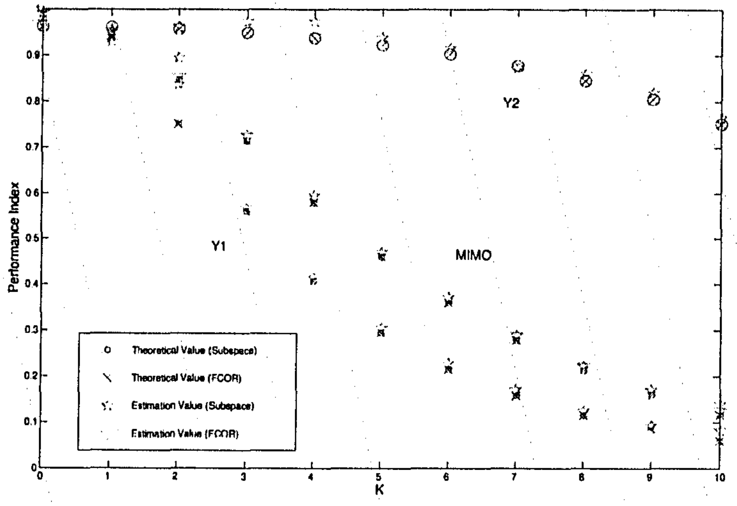


Figure 2.2: Performance assessment results by the FCOR algorithm and subspace method

When N increases to 3,

$$I - \hat{H}_N\hat{H}_N^{\dagger} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0923 & -0.2895 & 0 & 0 \\ 0 & 0 & -0.2895 & 0.9077 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$rank(I - \hat{H}_N\hat{H}_N^{\dagger}) = 3$ and notice that all new elements are zeros.

When N increases to 4,

$$I - \hat{H}_N\hat{H}_N^{\dagger} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0923 & -0.2895 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.2895 & 0.9077 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$rank(I - \hat{H}_N\hat{H}_N^{\dagger}) = 3$.

Therefore, $N = 2$ is a appropriate selection for this process to stack the data Hankel matrices.

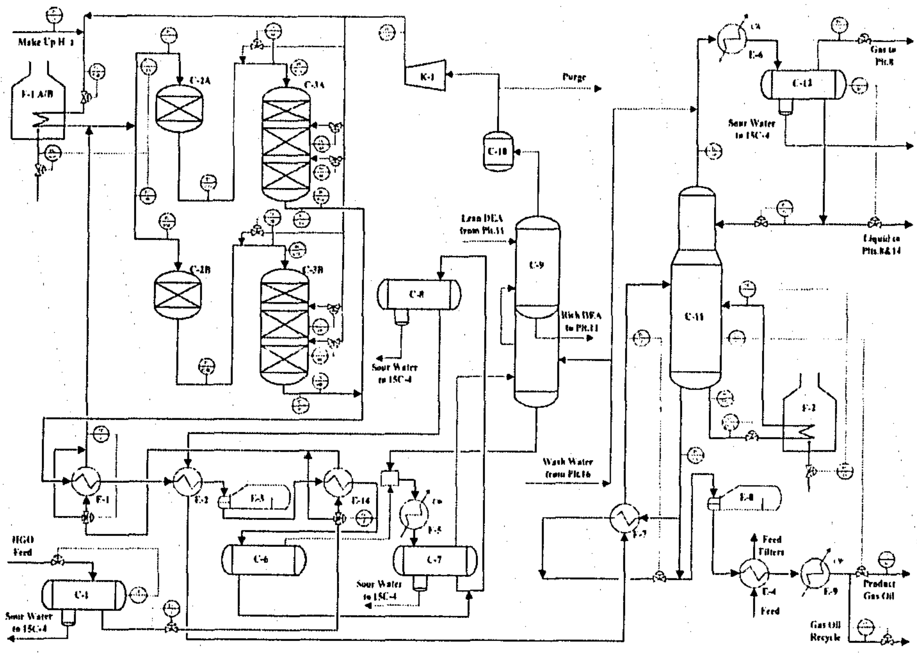


Figure 2.3: Gas oil hydrotreater unit

2.4 Industrial Case Study

The purpose of this section is to apply the performance assessment algorithms introduced in above sections to a real industrial process – the gas oil hydrotreater unit (GOHTU) of Syncrude Canada Ltd. at Ft. McMurray, Alberta. The objective is to use minimum variance control benchmark to assess the performance of process controllers. This will identify the potential for further improvement in terms of variability reduction and this reduction of variability can directly be transferred to economic benefits and environmental impacts.

2.4.1 Process Overview

The product of GOHTU is high quality treated gas oil and the byproduct is partially treated naphtha. The schematic diagram of the GOHTU is shown in Figure 2.3.

The main parts of the GOHTU are feed section, reactor section, reactor effluent section and fractionator section. In the feed section, the raw gas oil is filtered in feed filters to remove particulate matter and flows into the surge drum, from where it is pumped to the reactor section. In the reactor section, the raw gas oil is preheated and combined with treat gas before it is sent to the reactors. The treat gas is composed of recycle gas and fresh makeup hydrogen. The hydrogenation reactions occur in the reactors with catalyst. The reactor effluent is then cooled and separated into a liquid stream and a vapor stream in the hot high pressure separator. The liquid stream is sent to the fractionator section. The recycle gas is compressed after the removal of light oils, NH_3 and H_2S. In the fractionator section, the liquid stream from the reactor section is fractionated into product gas oil (fractionator bottoms) and the partially treated naphtha(fractionator overhead).

The main operating variables in the GOHTU are reactor equivalent isothermal temperatures, hydrogen partial pressure, catalyst activity and fractionation.

Syncrude has applied advanced control technology to optimize hydrogen treating unit operations, including the control, optimization and coordination of the naphtha hydrotreaters, the gas oil hydrotreaters and the light gas oil hydrotreater. The control and optimization of the GOHTU is part of the whole project.

The reactor section has 41 Controlled Variables (CVs) and 15 Manipulated Variables (MVs).

Table 2.1: MV list of the reactor section

MV	DESCRIPTION	MV	DESCRIPTION
1	T1A: A Reactor Bed 1 Inlet Temp	2	T1B: B Reactor Bed 1 Inlet Temp
3	T2A: A Reactor Bed 2 Inlet Temp	4	T2B: B Reactor Bed 2 Inlet Temp
5	T8A: A Reactor Bed 3 Inlet Temp	6	T8B: B Reactor Bed 3 Inlet Temp
7	T9A: A Reactor Bed 4 Inlet Temp	8	T9B: B Reactor Bed 4 Inlet Temp
9	FM5A: A Reactor Treat Gas Ratio	10	FM5B: B Reactor Treat Gas Ratio
11	FR195A: Cell A Air-to-Fuel Ratio	12	FR195B: Cell B Air-to-Fuel Ratio
13	S309: K-1 Compressor Speed	14	FD34: Feed Bias Between A and B Reactor
15	NM13: Recycle Gas Purity		

2.4.2 Case Study Results

In addition to closed-loop routine operating data, the transfer function model of reactor section is also available. It is a 41 × 15 matrix. The interactor matrix for this reactor

Table 2.2: CV list of the reactor section

CV	DESCRIPTION	CV	DESCRIPTION
1	AC910, Gas Oil Nitrogen Content	2	AC911, Gas Oil Sulphur Content
3	PC903, A Reactor H_2 Partial Pres	4	PC904, B Reactor H_2 Partial Pres
5	TC1A, Delta Outlet Temp Between Beds 3 and 4 in A Reactor	6	TC1B, Delta Outlet Temp Between Beds 3 and 4 in B Reactor
7	TC2, Diff Between A Reactor WABT and B Reactor WABT		
8	TK1A, A Reactor Bed 1 Hot Spot	9	TK1B, B Reactor Bed 1 Hot Spot
10	TK2A, A Reactor Bed 2 Hot Spot	11	TK2B, B Reactor Bed 2 Hot Spot
12	TK3A, A Reactor Bed 3 Hot Spot	13	TK3B, B Reactor Bed 3 Hot Spot
14	TK4A, A Reactor Bed 4 Hot Spot	15	TK4B, B Reactor Bed 4 Hot Spot
16	F5A.OP, A Rx Treat Gas Valve Output	17	F5B.OP, B Rx Treat Gas Valve Output
18	T2A.OP, A Reactor Bed 2 Quench	19	T2B.OP, B Reactor Bed 2 Quench
20	T8A.OP, A Reactor Bed 3 Quench	21	T8B.OP, B Reactor Bed 3 Quench
22	T9A.OP, A Reactor Bed 4 Quench	23	T9B.OP, B Reactor Bed 4 Quench
24	PC22A, A Reactor Bed 3 DP	25	PC22B, B Reactor Bed 3 DP
26	PC23A, A Reactor Bed 4 DP	27	PC23B, B Reactor Bed 4 DP
28	PC5A, C2A DP	29	PC5B, C2B DP
30	Y112, Compressor Anti-Surge Valve	31	P190, K-1 Discharge Pressure
32	A23A, F-1A Excess O_2	33	A23B, F-1B Excess O_2
34	T300, F-1A COT	35	T301, F-1B COT
36	P117B, F-1A Fuel Gas Pres	37	P119B, F-1B Fuel Gas Pres
38	TK5A, F-1A Radiant TMT Hi-Sel	39	TK5B, F-1B Radiant TMT Hi-Sel
40	TK6A, F-1A Convection TMTs Hi-Sel	41	TK6B, F-1B Convection TMTs Hi-Sel

section model is calculated by *interactor* function. The result is a 41 × 41 matrix.

$$D = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & z \\ 0 & 0 & 0 & \cdots & z & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & z & \cdots & 0 & 0 \\ 0 & z & 0 & \cdots & 0 & 0 \\ z & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

The reason that this interactor matrix has a simple format is that the second block of plant Markov parameters, G_1, is of full rank. Thus the algorithm terminates only after one time iteration.

Because the reactor section has considerable size, the time series analysis algorithm fails and thus the FCOR algorithm can not proceed the calculation. The noise model can not be estimated correctly from the closed-loop data through time-series analysis because of the high dimension of the process.

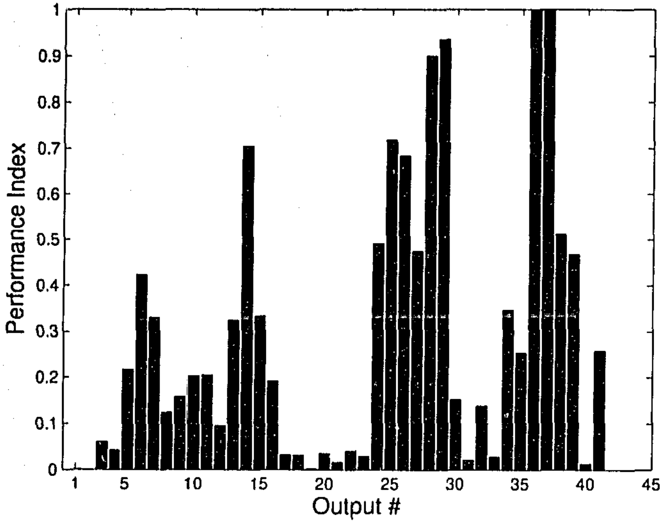


Figure 2.4: Performance assessment result by the subspace method

However, the subspace method works for this process, which shows the robustness of this subspace method. Since we have plant model and closed-loop routine operating data, the mixed value of performance index can be calculated. For one typical data set, the overall performance index is 0.01 and the individual performance index of every output loop is showed in Figure 2.4, from which we can see that this controller has potential to improve.

2.5 Conclusions

In this chapter, an improved algorithm for calculation of the interactor matrix is developed. The FCOR algorithm is described. If the interactor matrix is known, the performance index of a MIMO system can be estimated from a set of closed-loop data by this FCOR algorithm. A subspace method without a prior knowledge of the interactor matrix is also presented. This method simplifies the calculation of performance index and gives an explicit "one-shot" solution. The performance can be assessed from a set of open loop experimental data and a set of closed-loop routine operating data.

Three Matlab functions are programmed for these three algorithms, respectively. These functions have been tested on simulation examples and integrated into the software package

PAT. One industrial case study is presented, which revealed that the real process has potential to improve in terms of variability reduction.

3
Markov Chains

The basic concepts of Markov chains were introduced by A. A. Markov in 1907. They are the simplest mathematical models for random phenomena evolving in time. Their simple structure makes it possible to say a great deal about their behavior. At the same time, the class of Markov chains is rich enough to serve in many applications. This makes Markov chains the first and most important examples of random processes.

Compared with other methods, the advantage of Markov chains analysis is that it can provide considerable insight about the behavior of the control system. However, it does not weaken the merits of other methods. On the contrary, these methods can provide us complementary information about the performance of the control system, which will enable us to achieve better and deeper understanding.

3.1 Basic Concepts of Markov Chains

Markov Chains

If a process has no memory of where it has been in the past, this means that only the current state of the process can influence where it goes next, we called it a Markov process. When it has only a finite or countable set of states, it is usually referred as a Markov chain. (Kemeny and Snell 1976)

A Markov chain is observed to transit from one state to another state frequently.

Suppose a Markov chain has q states, which means there are q possible outcomes at each point in time. The set of q possible outcomes is denoted as S, state space. The ith element of S is $S_i, i = 1, 2, \cdots, q$. We also denote the sequence of n observations of a Markov process by $X_t, t = 0, 1, 2, \cdots, n$.

State-holding time T_i is defined as the average time interval in which the sequence spends in state S_i before transiting to another state. Occupation time C_i is defined as the number of times that the sequence is observed to be in a particular state S_i.

Initial distribution π_0 and transition probability matrix P

The probability of being in any one of these q states at time t can be described by a vector π_t. That means, $(\pi_t)_i = Probability(X_t = S_i)$. The elements of π_t always sum to one,

$$\sum_{i=1}^{q} (\pi_t)_i = 1. \tag{3.1}$$

π_0 is known as the *initial distribution*.

Another important concept about Markov chain is the transition probability matrix P. P_{ij} denotes the transition probability from state i to state j in one step. Obviously, $p_{ij} \geq 0$ for all i and j and each row of P sums to one.

Given an initial distribution π_0 and a transition probability matrix P, a Markov chain can be uniquely determined, since

$$\begin{aligned}
\pi_1^T &= \pi_0^T P, \\
\pi_2^T &= \pi_1^T P, \\
\cdots & \\
\pi_k^T &= \pi_{k-1}^T P = \pi_0^T P^n.
\end{aligned} \tag{3.2}$$

The transition probability matrix P can be estimated by

$$\hat{p}_{ij} = \frac{C_{ij}}{C_i}, \tag{3.3}$$

where C_{ij} is defined as the number of occurrences of transition from state i to state j in the sequence X_t and C_i is the occupation time.

Types of Markov chains

The states of a Markov chain can be divided into transient and ergodic sets. The former, once left, are never again entered; while the latter, once entered, are never again left.

We can divide the Markov chains into two types: ergodic Markov chains and absorbing Markov chains. If a state is the only element of an ergodic set, then it is called an absorbing state. For such a state s_i, the entry p_{ii} must be 1, and hence all other entries in this row of the transition matrix are 0. A chain, all of whose non-transient states are absorbing, is called an absorbing chain.

An ergodic Markov chain is one that has no transient sets, and has a single ergodic set. A Markov chain was ergodic if and only if it is possible to be in any state after some number N of steps, no matter what the starting state. An ergodic Markov chain must be either regular or cyclic.

Equilibrium distribution

If P is a regular transition matrix, then the powers P^n approach a probability matrix A as n tends to infinity. Each row of A is the same probability vector $\alpha = \{a_1, a_2, \cdots, a_n\}$. A is named as limiting matrix and α limiting vector or equilibrium distribution.

For a given transition probability matrix P, the equilibrium distribution α is unique such that $\alpha P = \alpha$. We cannot get an unique solution through above equation since the summation of every row's elements equals to one, P does not have full rank. However, the summation of all elements in α equals to one. Put them together,

$$\begin{aligned} \alpha P &= \alpha \\ a_1 + a_2 + \cdots + a_n &= 1 \end{aligned} \tag{3.4}$$

With this constraint, we can get an unique solution of α.

For any initial distribution vector π_0, $\pi_0 P^n$ also approaches the vector α as n tends to infinity, which means that the initial distribution has no effect on the equilibrium.

Passage Time

For a regular Markov chain, the passage time f_k is a function whose value is the number of steps before entering s_k for the first time after the initial position.

The mean passage time matrix is denoted as M. Entry m_{ij} is the mean passage time from state s_i to s_j. The mean passage matrix M is given by

$$M = (I - Z + EZ_{dg})D, \tag{3.5}$$

where I is an identity matrix, $Z = (I - (P - A))^{-1}$ is called the fundamental matrix for a regular Markov chain, Z_{dg} denotes the diagonal elements of the fundamental matrix, E is a matrix with all entries 1, and D is the diagonal matrix with diagonal elements $d_{ii} = \frac{1}{a_i}$.

Passage details

So far, we can get the mean passage time from state s_i to state s_j. By changing the process from a regular Markov process to an absorbing process, we can get more detailed information, such as the mean numbers of times that it will be in each of the other states before reaching s_j for the first time. To do this, what we need to do is to change state s_j to an absorbing state, then the process will be an absorbing process with a single absorbing state. By using the absorbing Markov chain theory, we can get some information which cannot be observed from the observation about regular Markov chain.

The question is whether it is reasonable to apply the theory of absorbing Markov chain to a regular Markov chain. The answer is positive. What we need to do is just change one or more states to absorbing states, and the behaviors of the original regular process and the observed absorbing process are exactly the same as before reaching the absorbing states. Hence we can translate all of the information about our original chain.

For an absorbing Markov chain, it is convenient to express the transition probability matrix P in the following way

$$P = \begin{pmatrix} S & 0 \\ R & Q \end{pmatrix}, \tag{3.6}$$

where the submatrix S is the transition probability matrix of all absorbing states and Q is that of all transient states. The fundamental matrix of an absorbing Markov chain is denoted as N and every elements of N, n_{ij}, is the number of times that the process is in transient state s_j after it leaves the initial state s_i and before it reaches an absorbing state. N is calculated as

$$N = (I - Q)^{-1}. \tag{3.7}$$

3.2 Industrial Case Studies

Harris and Yu (2003) defined a variable, degree of freedom (DOF), and demonstrated how the Markov chains can be used to analyze the industrial data. In this section, the theory of Markov chains is used to analyze the data collected from two industrial plants.

3.2.1 Case Study 1 – GOHTU

As the first case study, a set of industrial data collected from the Syncrude's GOHTU plant is analyzed. This plant has 41 CVs and 15 MVs. There is detailed information about this plant in Chapter 2. The data sample time is 15 seconds.

Control stability

In addition to the values of these CVs and MVs, this set of data also recorded some parameters of CVs and MVs, such as the control low and high limit. These parameters are set by the process control engineers and used by the advanced control algorithms.

If the value of a CV exceeds its high or low limit, it means that this variable is out of control. Based on it, we define a discrete variable:

Out of Control Index (OCI) = number of out of controlled CVs.

For this data set, the range of this variable is from 1 to 7. 1 means that the process is in the normal state and no CV is out of control. On the other hand, 7 means that the process is in the worst condition of this data set and there are 6 CVs which are out of control simultaneously. Larger the number, more out of control the process.

Transition probability matrix

For this set of data, the estimated transition probability matrix is shown in Table 3.1.

Table 3.1: Transition probability matrix

$\hat{P}$	State 1	State 2	State 3	State 4	State 5	State 6	State 7
State 1	0.86	0.13	0.01	0	0	0	0
State 2	0.26	0.63	0.10	0.01	0	0	0
State 3	0.04	0.28	0.57	0.10	0.01	0	0
State 4	0	0.07	0.36	0.48	0.09	0	0
State 5	0	0.01	0.07	0.27	0.58	0.06	0.01
State 6	0	0	0	0	0.22	0.72	0.06
State 7	0	0	0	0	0.67	0	0.33

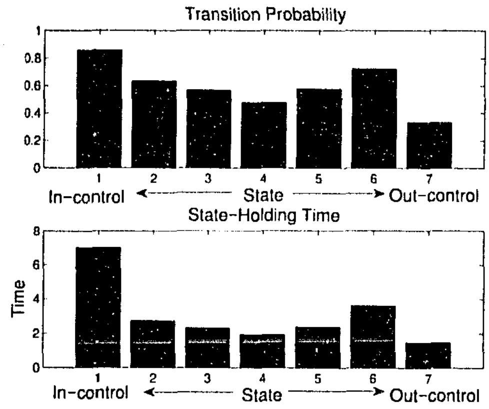


Figure 3.1: Transition probability v.s. state-holding time

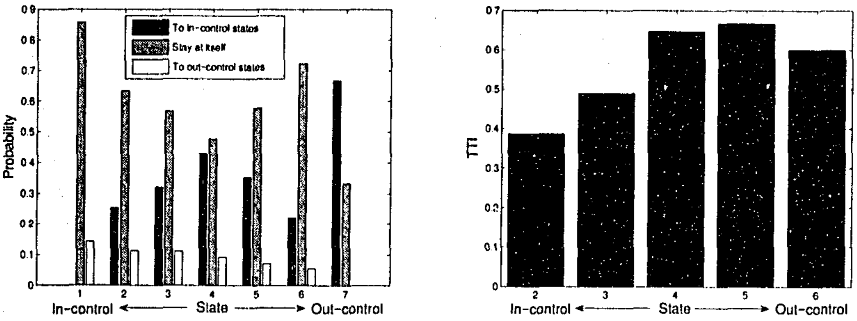
In Table 3.1, the main diagonal elements are much larger than others. It means that every state will stay at itself for certain time before transiting to another state. This is easy to understand because the sample time is fast. Moreover, these diagonal elements are proportional to the state-holding times of these states, as shown in Figure 3.1. The larger the probability is, the longer the state-holding time is.

We notice that $\hat{p}_{i(i-1)} > \hat{p}_{i(i+1)}, i = 2, 3, ..., 6$. This means that every state has higher probability to transit to in-control states than to out-control state. The other feature that we notice is that $\hat{p}_{ij}$ is almost zero when $j > i + 1$ or $j < i - 1$. The interpretation is that when the state changes, it most likely changes to a neighboring state. As a result, we can say that the process has no dramatic changes in CVs.

In Figure 3.2 (a), for every state, the left bar is the summation of probabilities to all previous, more in-control states and the right bar is to all next, more out-control states. Obviously, every state has higher tendency to transfer to more in-control states than out-control state. We can define a transition tendency index (TTI) as

$$TTI_i = \frac{\sum_1^{i-1} p_{ij} - \sum_{i+1}^{n} p_{ij}}{1 - p_{ii}}.$$

The TTI is an index of magnitude $-1 \leq TTI \leq 1$, where 1 indicates the best in terms of tendency toward in-control states, while -1 indicates the worst. Thus it is an index about the transition tendency to normal states. From Figure 3.2 (b), we can see that TTI has an obvious increase with the increase of out of controlled CV numbers. Thus we can



(a) Transition probability matrix

(b) Transition tendency index

Figure 3.2: Transition probability

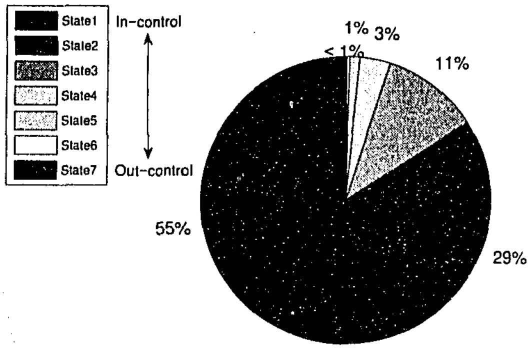


Figure 3.3: Equilibrium distribution

conclude that this process is in stable control.

Equilibrium distribution

Equilibrium distribution provides us with a prediction about the future. For this set of data, the equilibrium distribution is calculated as

$$\alpha = \left[\begin{array}{cccccccc} 0.4537 & 0.3340 & 0.1321 & 0.0477 & 0.0195 & 0.0097 & 0.0028 & 0.0005 \end{array}\right].$$

From Figure 3.3, we can see that at equilibrium process spends more time at less out-control states. In fact, for 55% time, the process is completely within control state and for 29% time, the process just has one CV out of controll. The percent time spent at three worst states, in which the process has more than 3 out of controlled CVs, is less than 5%.

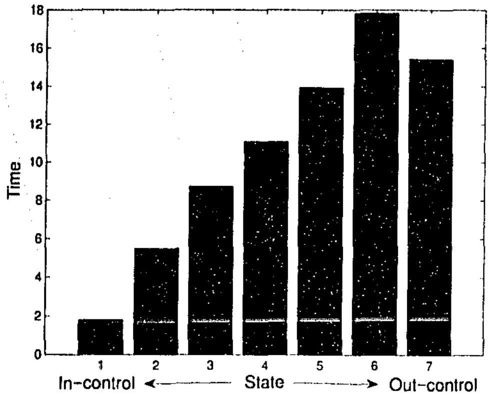


Figure 3.4: Passage time to s_1

As a conclusion, we can say that this process is in stable operation.

Passage time

For our example, the mean passage time matrix M is calculated in Table 3.2.

Table 3.2: Mean passage time

	State 1	State 2	State 3	State 4	State 5	State 6	State 7
State 1	1.81	7.31	27.86	81.62	309	1530	3469
State 2	5.48	3.44	22.25	75.62	303	1525	3463
State 3	8.72	4.95	9.16	59.51	285	1507	3446
State 4	11.11	7.20	6.62	29.50	240	1459	3398
State 5	13.93	9.98	9.45	18.45	89.75	1271	3185
State 6	17.83	13.88	13.35	22.35	3.9	354	2551
State 7	15.43	11.48	10.95	19.95	1.5	1272	2124

The first column data in the Table 3.2 is shown in Figure 3.4. They are the average times the process needed to return to the complete in-control state s_1. If the process starts from s_7, for example, the average time before it returns to s_1 is about 15 sample time. In Table 3.2, we also notice that $M_{ij}(i < j)$ is much larger than M_{ji}. For instance, it will take about 3469 sample time for the process to reach s_7 from s_1; however, just 15 sample time from s_7 to s_1. This means that the process has less tendency to the worse state and more tendency to better in-control state. Again, it is proved that the process is in stable control.

Passage detail

For our example, we are interested in the mean numbers of times that the process is in each of the other states before reaching the two extreme states, the normal state and the worst state.

For the normal state s_1, first we need to change s_1 into an absorbing state. Then the transition probability matrix becomes

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.26 & 0.63 & 0.10 & 0.01 & 0 & 0 & 0 \\ 0.04 & 0.28 & 0.57 & 0.10 & 0.01 & 0 & 0 \\ 0 & 0.07 & 0.36 & 0.48 & 0.09 & 0 & 0 \\ 0 & 0.01 & 0.07 & 0.27 & 0.58 & 0.06 & 0.01 \\ 0 & 0 & 0 & 0 & 0.22 & 0.72 & 0.06 \\ 0 & 0 & 0 & 0 & 0.67 & 0 & 0.33 \end{bmatrix}.$$

For an absorbing Markov chain, it is convenient to express the transition probability matrix P according to Equation 3.6. Thus, in this example,

$$S = [1],$$

$$R = \begin{bmatrix} 0.26 & 0.04 & 0 & 0 & 0 & 0 \end{bmatrix}^T,$$

$$Q = \begin{bmatrix} 0.63 & 0.10 & 0.01 & 0 & 0 & 0 \\ 0.28 & 0.57 & 0.10 & 0.01 & 0 & 0 \\ 0.07 & 0.36 & 0.48 & 0.09 & 0 & 0 \\ 0.01 & 0.07 & 0.27 & 0.58 & 0.06 & 0.01 \\ 0 & 0 & 0 & 0.22 & 0.72 & 0.06 \\ 0 & 0 & 0 & 0.67 & 0 & 0.33 \end{bmatrix}.$$

Table 3.3 shows N calculated according to Equation 3.7.

Table 3.3: Passage details when s_1 as an absorbing state

	State 2	State 3	State 4	State 5	State 6	State 7
State 2	3.72	1.21	0.39	0.13	0.03	0.01
State 3	3.22	3.99	1.05	0.36	0.09	0.01
State 4	3.26	3.53	3.14	0.90	0.23	0.04
State 5	3.27	3.53	2.61	3.60	0.77	0.14
State 6	3.27	3.53	2.61	3.60	4.37	0.44
State 7	3.27	3.53	2.61	3.60	0.77	1.64

From the mean passage time matrix, we have known how much time is needed for the process to reach state s_j from state s_i. For example, it takes about 15 sample times from s_7 to s_1. From the matrix N, we can get more detailed information about what happened

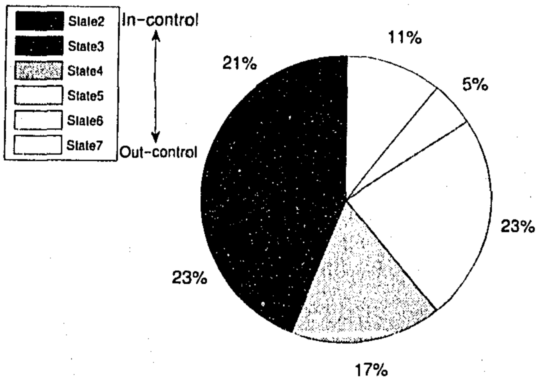


Figure 3.5: Transition process from s_7 to s_1

during this transition. It will stay at s_2 about 3.27 samples times, at s_3 about 3.53 sample times, at s_4 about 2.61 sample times, and so on.

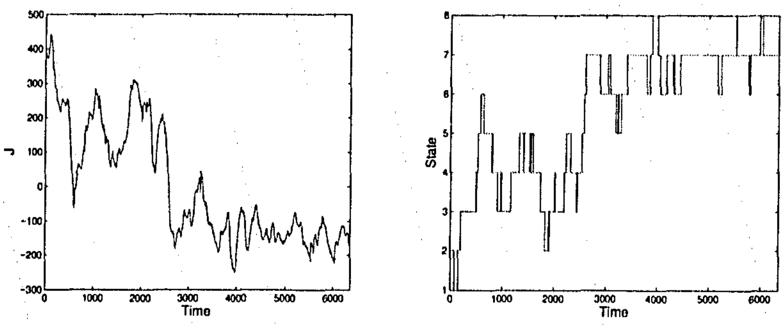
The detailed passage process from state s_7 to s_1 is shown in Figure 3.5, from which we can see that even during the passage process from the most out of controlled state s_7 to normal state s_1, the process spent more time to stay at in-control zone than at out-control zone. This means that the process can quickly and automatically back to in-control status under the control of the current.

In same way, we can change state s_7 to an absorbing state and observe how the process behaves during the passage to state s_7.

Table 3.4: Passage details when s_7 as an absorbing state

	State 1	State 2	State 3	State 4	State 5	State 6
State 1	1923	1010	377	116	35	9.07
State 2	1917	1010	377	116	35	9.07
State 3	1905	1004	377	115	35	9.07
State 4	1878	989	371	116	35	9.07
State 5	1758	926	348	108	35	9.00
State 6	1407	741	278	86	28	10.80

Compared with the data by treating s_1 as an absorbing state, these numbers in Table 3.4 are quite large. The interpretation is that it will take much longer time for every state, especially those more in-control states, to reach state s_7. It verifies our conclusion that



(a) Original data

(b) Discrete sequence

Figure 3.6: Economic index J

this process is in stable control.

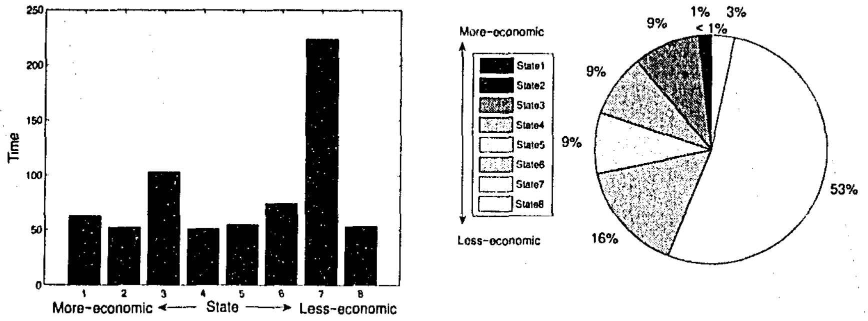
Economic analysis

The economic optimizer has been used to control and optimize the GOHTU. The optimization objective is to maximum CV1, CV2, CV3, CV4, MV1, MV2, MV3, MV4, MV9, MV10 (linear objective) and to minimize CV18, CV19, CV32, CV33, MV13, MV15 (linear objective), CV5, CV6, CV7, MV14 (quadratic objective), all subject to constraints. Therefore, the economic index J is calculated according following optimization objective function for the same data set analyzed above:

$$\begin{aligned} J &= -(-10cv1 - 5cv2 - cv3 - cv4 + (cv5)^2 + (cv6)^2 + (cv7)^2 + 0.5cv18 \\ &\quad +0.5cv19 + 0.5cv32 + 0.5cv33 - 5mv1 - 5mv2 - mv3 - mv4 \\ &\quad -0.6mv9 - 0.6mv10 + 0.4mv13 + 0.05(mv14)^2 + 10mv15). \end{aligned}$$

Then, the economic index is discretized into eight states. State s_1 means the process is in the best economic state and s_8 means that in the worst economic state. The original data and the discrete sequence of J are shown in Figure 3.6.

For this sequence of the economic index, the estimated transition probability matrix is shown in Table 3.5. This matrix is almost an identity matrix, which means that this process is almost an absorbing process and every state is near absorbing state. The reason is that the state-holding time of every state, as shown in Fig 3.7 (a), is pretty large and



(a) State-holding time

(b) Equilibrium distribution

Figure 3.7: Economic analysis

there are few transitions during the whole sample process.

Table 3.5: Transition probability matrix

$\hat{P}$	State 1	State 2	State 3	State 4	State 5	State 6	State 7	State 8
State 1	0.985	0.015	0	0	0	0	0	0
State 2	0.005	0.975	0.02	0	0	0	0	0
State 3	0	0.003	0.989	0.008	0	0	0	0
State 4	0	0	0.008	0.983	0.09	0	0	0
State 5	0	0	0	0.01	0.98	0.001	0	0
State 6	0	0	0	0	0.004	0.986	0.01	0
State 7	0	0	0	0	0	0.003	0.996	0.001
State 8	0	0	0	0	0	0	0.025	0.975

The equilibrium distribution is calculated as

$$\alpha = \begin{bmatrix} 0.01 & 0.03 & 0.15 & 0.12 & 0.10 & 0.14 & 0.43 & 0.02 \end{bmatrix}$$

and shown in Figure 3.7 (b), from which we can see that the process stays at less economic states for most of time.

Table 3.6 shows the mean passage time matrix M. We notice that, compared with other elements, the diagonal elements are quite small. This means that the process has very strong tendency to stay at the original state.

The passage details when treat s_1 and s_8 as an absorbing state are shown in Table 3.7 and Table 3.8, respectively. In these tables, we noticed that, during every transition from one state to s_1 or s_8, the process stays at less economic states for most of time.

Table 3.6: Mean passage time

	State 1	State 2	State 3	State 4	State 5	State 6	State 7	State 8
State 1	97	65	131	292	538	1070	1346	3408
State 2	6243	32	66	227	473	1005	1281	3343
State 3	7754	1511	7	161	407	939	1215	3277
State 4	8269	2026	741	8	246	778	1054	3116
State 5	8602	2359	1267	655	10	532	808	2870
State 6	8939	2696	1893	1475	1045	7	276	2338
State 7	8953	2710	2004	1649	1295	425	2	2062
State 8	8994	2751	2045	1690	1336	466	41	52

Table 3.7: Passage details when s_1 as an absorbing state

	State 2	State 3	State 4	State 5	State 6	State 7	State 8
State 2	200	925	761	650	887	2698	122
State 3	200	1156	951	813	1109	3373	153
State 4	200	1123	1033	882	1204	3662	166
State 5	200	1095	994	942	1285	3909	177
State 6	200	1052	936	869	1407	4281	194
State 7	200	1038	916	845	1350	4405	200
State 8	200	1038	916	845	1350	4405	240

In the control stability analysis, all the results support the conclusion that the process is in stable control; however, the economic analysis states that the process is in less economic status most of time. Therefore, this controller may be tuned to achieve further economic profits.

3.2.2 Case Study 2 – PSV

This part has been removed due to the consideration of proprietary.

3.3 Conclusions

In this chapter, the theory of Markov chain is first reviewed. The basic concepts, such as transition probability matrix, equilibrium distribution, etc. are introduced.

The Markov chain theory has been applied to industrial plants. Two indices are defined, one is out of control index (OCI) and the other transition tendency index (TTI). All the analysis results performed on the GOHTU, including the OCI and TTI indices,

Table 3.8: Passage details when s_8 as an absorbing state

	State 1	State 2	State 3	State 4	State 5	State 6	State 7
State 1	128	193	780	580	433	394	899
State 2	63	193	780	580	433	394	899
State 3	47	143	780	580	433	394	899
State 4	40	122	648	580	433	394	899
State 5	34	103	535	471	433	394	899
State 6	25	76	365	308	271	394	899
State 7	22	67	308	254	217	296	899

support the conclusion that this plant was in stable control. The economic analysis, however, revealed that this plant is not optimized.

4
A Practical Process Monitoring Method

The traditional approach of process monitoring is based on limit sensing(Chiang *et al*2001). High and/or low threshold is predefined for some critical variables. For the most critical variables, extreme high and/or extreme low threshold are also predefined. If the measured value of one variable exceeds its high or low threshold, an alarm is triggered. The process operators need to figure out what has happened to this variable and take some necessary actions, sometimes with the help of process control engineers and technicians. If this variable continues to deviate from normal value and reaches the extreme high or extreme low threshold, the emergency shutdown (ESD) system has to put into effect and part of, even whole, process may shutdown immediately.

The limit sensing approach has two disadvantages. On one hand, it works on the present measured values of the monitored variables, without taking the trends into account. Sometimes, just a short peak of the measured value will result in serious consequences. On the other hand, it neglects the relations between variables. In some cases, the deviation of one variable from normal value does not mean the whole process is in abnormal situation. Therefore, a ideal process monitoring method should not only monitor the trend of a variable, but also consider the relations between trends of several related variables.

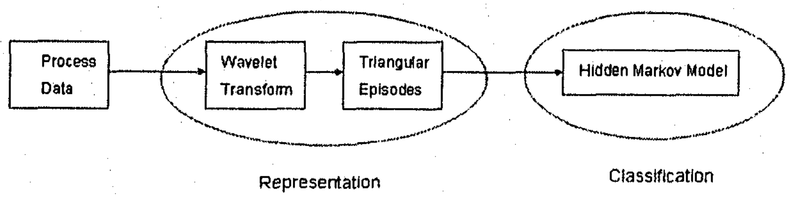


Figure 4.1: Architecture of this process monitoring method

4.1 Methodology

This process monitoring method consists of two parts: representation and classification. In the representation part, the process data is filtered by wavelet transform and then described by a sequence of triangular episodes. By the triangular description, the original data is converted into qualitative and semi-quantitative information. Then this information is input into one or more hidden Markov models (HMM) for classification.

4.1.1 Wavelet Analysis

Since the 90's of the last century, wavelets have been widely used as an important tool for signal analysis. The concept is not new to mathematicians and physicists; however, their applications to signal processing have experienced increasing popularity since the work of J.Morlet *et al.* (1984), Mallat(1989, 1991) and Daubechies (1990).

A family of wavelets is derived from the translations and dilations of a single function, as showed in Figure 4.2 and Figure 4.3.

Definition 6 *If $\psi(x)$ is the starting function, to be called a* **wavelet**, *the members of the family are given by :*

$$\frac{1}{\sqrt{e}}\psi\left(\frac{x-u}{e}\right) \quad for \quad (e,u) \in R^2 \tag{4.1}$$

in which the scale parameter e indicates the dilation and the translation parameter u indicates the shifting of the mother wavelet $\psi(x)$.

The continuous wavelet transform (CWT) is the projection of a function $F(x) \in L^2(r)$ on a wavelet, i.e.

$$W_{eu}F(x) = \int_{-\infty}^{+\infty} F(x)\left\{\frac{1}{\sqrt{e}}\psi\left(\frac{x-u}{e}\right)\right\}dx. \tag{4.2}$$

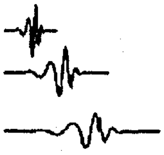


Figure 4.2: Dilation of a function

0
Wavelet function
0
Shifted wavelet function

Figure 4.3: Translation of a function

The original function $F(x)$ can be reconstructed from the wavelet transforms by:

$$F(x) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} W_{eu}F(x) \left\{ \frac{1}{\sqrt{e}} \psi \left(\frac{x-u}{e} \right) \right\} dedu. \tag{4.3}$$

For the discrete wavelet transform (DWT), the scale parameter e, and the translation parameter, u, have to be discretized. Usually, the scale parameter is sampled along the dyadic sequence.

$$e = 2^m, \quad m = 0, 1, 2, \cdots, L \in Z \tag{4.4}$$

The scale $m = 0$ represents the finest scale, which is the original measured data and $m = L$ represents the coarsest scale. As the scale parameter e is increased, high frequencies of signal are removed. Thus, we can think the wavelet as a band-bass filter.

There are two approaches for the discretization of the translation parameter. The first one is to sample u uniformly over dyadic intervals:

$$u = \alpha 2^m k, \quad (m, k) \in Z^2, \tag{4.5}$$

where α is the sampling interval. Another approach involves uniform sampling of the signal at all scales, i.e.

$$u = n \quad such \quad that \quad n = Z \tag{4.6}$$

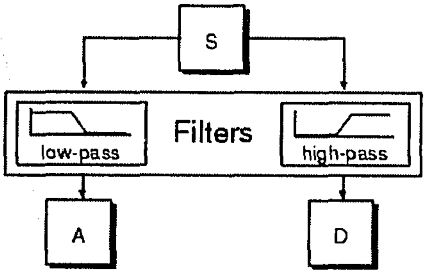


Figure 4.4: DWT decomposition

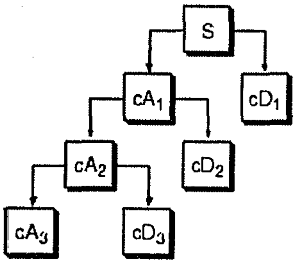


Figure 4.5: DWT multi-decomposition

In the first approach, the sample interval between adjacent wavelets doubles as the scale increases and it does not change with scale in the second approach.

Mallat (1989) developed an efficient way to implement the discrete wavelet transform, which in fact is a vary practical filtering algorithm. As illustrated in Figure 4.4, the original signal, s, passes through two complementary filters and emerges as two signals: A, which stands for the approximation, is the low-frequency component of the signal and D, which stands for the detail, is the high-frequency component. The detail, D, is the wavelet transform of the original signal. It is the difference between the original signal and the approximation. This process is called decomposition in wavelet and can be performed iteratively.

If we implement filtering in this way, unfortunately, the total data number of A and D will be twice as much as that of the original signal. Thus, downsampling has to be integrated into this algorithm to keep the total number of data constant. In the Figure 4.6, cA and cD are the coefficients of the approximation and the detail after downsampling,

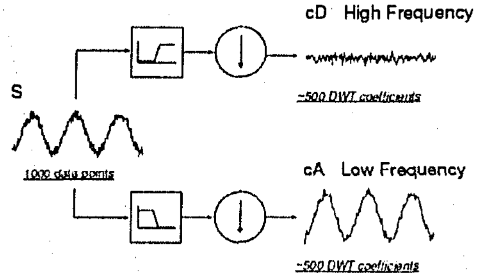


Figure 4.6: Downsampling

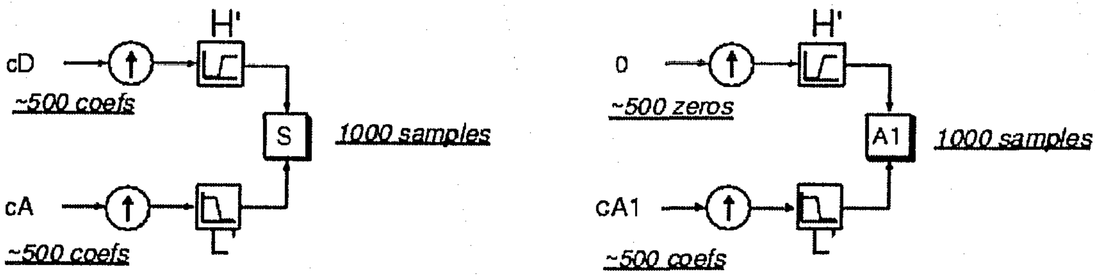


Figure 4.7: DWT reconstruction

respectively.

Reconstruction performs in a reverse way, as shown in Fig 4.7. Combining upsampled approximation and detail coefficients, it is possible to reconstruct the original signal. Using zeros instead of detail coefficients, approximation is reconstructed, which has the same length as the original signal but high-frequency information is filtered out.

Figure 4.8 shows an electrical consumption measured over the course of 3 days. Like most signals collected from the real world, noise is introduced into this signal as the measurements were being made.

The wavelet decomposition of this signal is shown in Figure 4.9. The raw data ($m = 0$) is decomposed into the approximation and the detail at $m = 1$. This process is continued until desired level is reached. The details are the high frequency noisy signals filtered from the upper layer approximation signals. The approximations keep the basic trend of the original data.

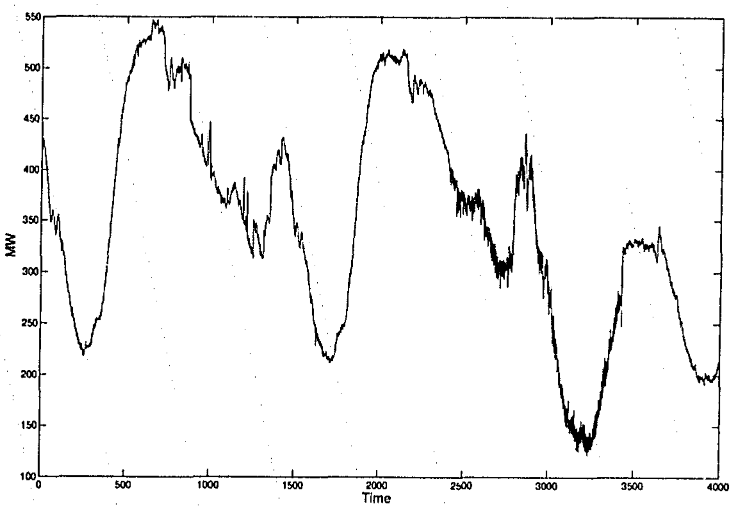


Figure 4.8: Electrical consumption

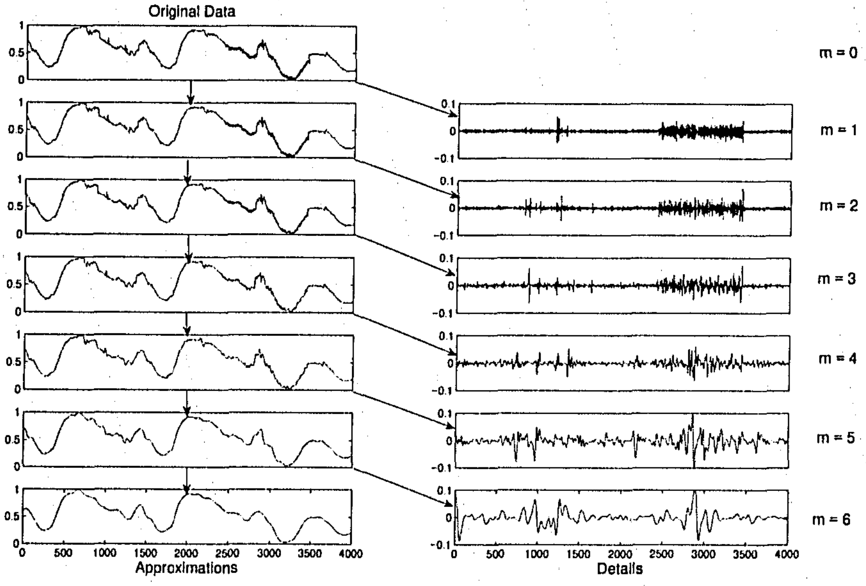


Figure 4.9: Wavelet decomposition of the electrical consumption

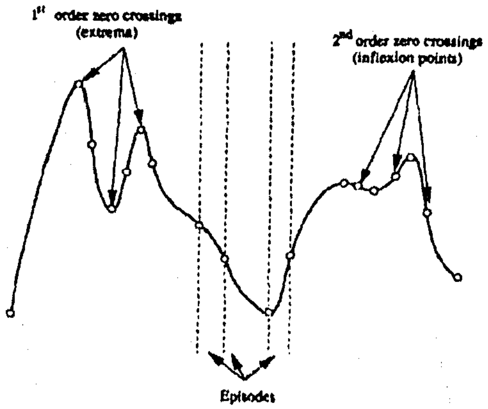

1st order zero crossings (extrema)
2nd order zero crossings (inflexion points)
Episodes

Figure 4.10: Illustration of definitions for describing process trends

4.1.2 Triangular Representation

By the wavelet transform, the basic trend is extracted from the noisy data. The next step is to represent this trend in an efficient way.

A great deal of effort has been focused on using local extrema as descriptive primitives to analyze one or two dimensional signal because these extrema usually have direct semantic interpretations. The extraction of these features can help to characterize a signal and separate it from others. Witkin (1983) suggested the use of zero-crossings because, compared with extrema, zero-crossings are more convenient for implementation.

The nth order zero-crossings in a signal, $F(t)$, are the points that satisfy

$$\frac{\partial^n F}{\partial t^n} = 0, \frac{\partial^{(n+1)} F}{\partial t^{(n+1)}} \neq 0. \tag{4.7}$$

Therefore, an *extremum* is a first order zero-crossing and an *inflexion point* is a second order zero-crossing.

The series of paper by Stephanopoulos (1990a, 1990b, 1994) developed a method – triangular representation – to describe qualitative and quantitative information in a process trend. A signal is first smoothed by wavelet for the extraction of basic trend, and then, according to the extrema and inflexion points, it is divided into *episodes*. An episode consists of an extremum and a neighbored inflexion point. Formally speaking, an episode is a part of signal whose first and second derivative have constant signs. These definitions are illustrated in Figure 4.10.

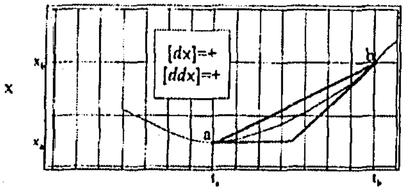


Figure 4.11: Episode described by a triangle

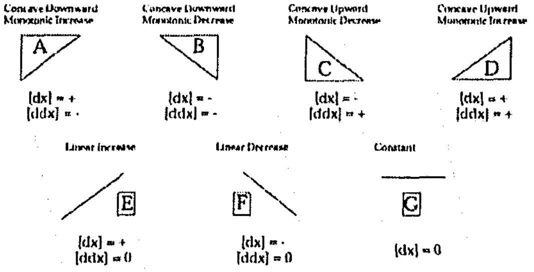


Figure 4.12: Definition of triangles

Using this method, every episode can be described by a triangle. As shown in Figure 4.11, one side of the triangle is constructed by drawing a line between the two end points of the episode. The remaining two sides are drawn by connecting the tangents of these end points, up to the point where the slope intersects.

There are seven kinds of triangle defined in Bakshi and Stephanopoulos (1994), named as A, B, C, D, E, F and G. Their shapes and definitions are showed in Figure 4.12 (Wong *et al.* 1998). In fact, type E, F and G are three kinds of line: linear increasing, linear decreasing and constant. It is hardly to observe these three kinds of line in a smoothed trend. Therefore, they are not adopted in this work. Then, this triangular representation method is simplified to contain four kinds of triangle: A, B, C and D.

The type of triangle determines the qualitative information of an episode. In addition to the type, some semi-quantitative information are also used to characterize and classify an episode, such as duration and magnitude. The duration of a triangle is defined as the time interval between two end points of an episode. The magnitude of a triangle is the vertical difference between these two end points.

There are three kinds of magnitude: large, medium and small. There are three kinds

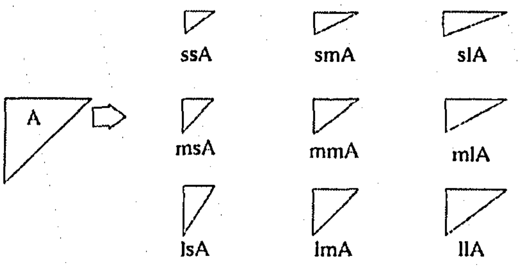


Figure 4.13: Sub-type triangles of type A

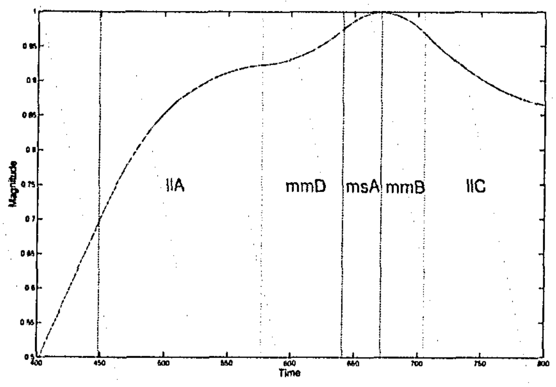


Figure 4.14: Segment of the triangular description of the electrical consumption

of duration: long, middle and short. Then, for every type of triangle, say, "*A*", there are nine possible outcomes, as shown in Figure 4.13. For example, "*lmA*" stands for a large magnitude, middle duration, type "*A*" triangular. The same definition holds for the triangles "*B*" to "*D*". The triangular description of a segment of the basic trend of electrical consumption is shown in Fig 4.14, from which we can see that the basic trend has been accurately divided into episodes according to extrema and inflexion points. Therefore, this triangular representation method gives the qualitative as well as semi-quantitative characteristics of a trend.

Finally, using this triangular representation method, a signal is converted into a sequence of symbols, which has 36 symbolic character alphabet.

As Bakshi and Stephanopoulos (1994) pointed out, this method "*is complete, correct and robust, and allows explicit description of the important information of a trend.*" The advantage of this method is that it converts a signal into a symbolic sequence, which captures the most important qualitative and quantitative information contained in the signal. Compared with filtered process data, this symbolic form is convenient as the input of a following classifying system such as HMM.

The drawback of this method is that it is sensitive to noisy data. Therefore, some filtering procedures, such as the wavelet transform, have to be implemented first.

4.1.3 Hidden Markov Model

Like the Markov chain introduced in the last chapter, neither the theory of hidden Markov model nor its applications is new. It has been successfully used in some applications, such as speech recognition. Smyth (1993) presented a particularly effective method for fault detection, which uses HMM to classify the process status into a normal state, an intermittent state or a "hard-fault" state. This method has been applied to a real-world antenna fault diagnosis system. Based on the work of this paper, Wong *et al.* (1998, 2001) adopted HMM method to classify the process trends, which are first filtered by the wavelet transform and then described by the triangular representation method. Some illustrative examples about HMM can be found in Rabiner (1989) and the Matlab Statistic toolbox manual.

There is an important difference between Markov chain and HMM. In a Markov chain, the output of process is a set of states where each state corresponds to an observable event. In HMM, however, the observation is a probabilistic function of the state and the underlying set of states is not observable, but can only be indirectly observed through the sequence of observations.

A Markov chain can be uniquely determined by an initial distribution, π_0, and a transition probability matrix, P. For HMM, the situation becomes more complicated because it has two stochastic processes, one of which is hidden and the other observable. Thus, in addition to the transition probability matrix, there is another probability matrix – the observation probability matrix, B. Every element of B, B_{ij}, gives the probability of jth observation symbol observed at state i, where $1 \leq i \leq N$ and $1 \leq j \leq M$. N is the number of states and M is the number of distinct observation symbols per state.

Rabiner (1989) summarized that there are three basic problems of interest for an HMM application:

- Given the observation sequence and a model, how do we efficiently compute the probability of this observed sequence?

- Given the observation sequence and the model, how do we choose a corresponding state sequence which is optimal in some meaningful sense?

- How do we adjust the model parameters so as to best account for the observed sequence?

For the method proposed in this chapter, the first and last problems are of interests. The last one is a training problems – given an observation sequence, how do we optimize the model parameters to create the best model? Similar to other model based process monitoring and fault detection methods, this problem is difficult and the solution to this problem is crucial. The solution to the first problem is used for classification. Given a model and any observation sequence, how do we compute the probability that the observed sequence was produced by this model? Another case of classification is that, given several models and one sequence of observation, how do we determine the model which best matches this observation?

The HMM-based classification method introduced in this chapter is implemented using the Matlab Statistics toolbox, which includes five functions designed for HMM analysis. The function *hmmtrain* calculates the maximum likelihood estimate of HMM parameters. The default algorithm of this function uses an iterative algorithm – Baum-Welch method. Detailed explanation about this algorithm can be found in Rabiner (1989).

The function *hmmdecode* calculates the probability of an observation sequence being generated by a model. The algorithm of this function is explained as follow.

Given a hidden Markov model λ, let $O = o_1 o_2 \cdots o_T$ denote an observation sequence, $Q = q_1 q_2 \cdots q_T$ denotes a state sequence. The probability of this state sequence can be calculated as

$$P(Q|\lambda) = \pi_{q_1} P_{q_1 q_2} P_{q_2 q_3} \cdots P_{q_{T-1} q_T} \tag{4.8}$$

in which, π_{q_1} is the probability that the initial state is q_1 and $P_{q_1 q_2}$ is the transition probability from state q_1 to state q_2. The probability that this observation sequence is generated by this state sequence is given by

$$P(O|Q,\lambda) = B_{q_1 o_1} \cdot B_{q_2 o_2} \cdots B_{q_T o_T} \tag{4.9}$$

Put Equation 4.8 and Equation 4.9 together, given model λ and state sequence Q, the joint probability of observation sequence O is

$$\begin{aligned} P(O, Q|\lambda) &= P(O|Q,\lambda)P(Q|\lambda) \\ &= \pi_{q_1} B_{q_1 o_1} P_{q_1 q_2} B_{q_2 o_2} \cdots P_{q_T\ {}_1 q_T} B_{q_T o_T}. \end{aligned} \tag{4.10}$$

Then, the probability that an observation sequence, O, is generated from given model, λ, is obtained by summing the joint probability in Equation (4.10) over all possible state sequences

$$\begin{aligned} P(O|\lambda) &= \sum_{allQ} P(O|Q,\lambda)P(Q|\lambda) \\ &= \sum_{q_1,q_2,\cdots,q_T} \pi_{q_1} B_{q_1 o_1} P_{q_1 q_2} B_{q_2 o_2} \cdots P_{q_{T-1} q_T} B_{q_T o_T}. \end{aligned} \tag{4.11}$$

This calculation method is straightforward, but the complexity is in the order of $O(2T \cdot N^T)$. Even for small values of N and T, the computation load will be very large! A more efficient algorithm is called the forward-backward algorithm.

Following is the forward-backward algorithm introduced in Rabiner (1989): The forward variable $\alpha_t(i)$ is defined as

$$\alpha_t(i) = P(o_1 o_2 \cdots o_t, q_t = S_i|\lambda)$$

which is the probability of the partial observation sequence, $o_1 o_2 \cdots o_t$, (until time t) and state S_i at time t, given the model λ. Inductively, $\alpha_t(i)$ can be solved as 1)Initialization:

$$\alpha_1(i) = \pi_i B_{io_1} \quad 1 \le i \le N. \tag{4.12}$$

2)Induction:

$$\begin{aligned} \alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) P_{ij}\right] B_{jo_{t+1}}, \quad & 1 \le t \le T-1 \\ & 1 \le j \le N. \end{aligned} \tag{4.13}$$

3)Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{4.14}$$

The complexity of this inductive algorithm decreases to $O(N^2 T)$, which is much smaller than the $O(2T \cdot N^T)$ of the straightforward method. The backward procedure is conducted in a similar manner.

It can be seen that in Equation 4.13, $\alpha_t(j)$ will starts to head toward zero exponentially as t becomes large since every item in transition probability matrix P and observation probability matrix O is usually less than 1. For most applications in which t is sufficiently large, such as several hundreds or even thousands, the dynamic range of the $\alpha_t(j)$ computation will exceed the precision range of any computer. Thus, to perform the above forward-backward algorithm, a scaling procedure is needed to incorporate into it.

The basic idea of scaling is to multiply $\alpha_t(j)$ by a scaling coefficient to keep the computation within a reasonable range. The scaling coefficient should just depend on time t not on the state number N.

One choice for scaling coefficient is the summation of all forward variables at the same time t,

$$c_t = \frac{1}{\sum_{j=1}^{N} \alpha_t(j)}. \tag{4.15}$$

Then, the scaled forward variable becomes

$$\hat{\alpha}_t(j) = \frac{\alpha_t(j)}{\sum_{i=1}^{N} \alpha_t(i)}. \tag{4.16}$$

At the Termination step of the forward-backward algorithm, the final probability $P(O|\lambda)$ will have some changes accordingly because all the items $\hat{\alpha}_t(j)$ have been scaled. Finally, the probability $P(O|\lambda)$ is given by

$$log[P(O|\lambda)] = -\sum_{t=1}^{T} logc_t. \tag{4.17}$$

The Matlab function *hmmdecode* is programmed based on the above scaled forward-backward algorithm. One disadvantage of this function is that it does not check whether the value of $\sum_{i=1}^{N} \alpha_t(i)$ is equal to zero before divide it and take log of it. In some cases, for example, where the model is not well trained, $\sum_{i=1}^{N} \alpha_t(i)$ will be zero. Any computation trying to divide or log it will result in an error message. This function is re-programmed for this application and the output parameter *logpseq*, logarithm of probability, will set to zero instead of giving an error message in this case.

4.2 Implementation Issues

4.2.1 Why is Wavelet?

In this application, the wavelet analysis is used to extract significant temporal features contained in a record of measured data. Then, first question is why is wavelet, not the

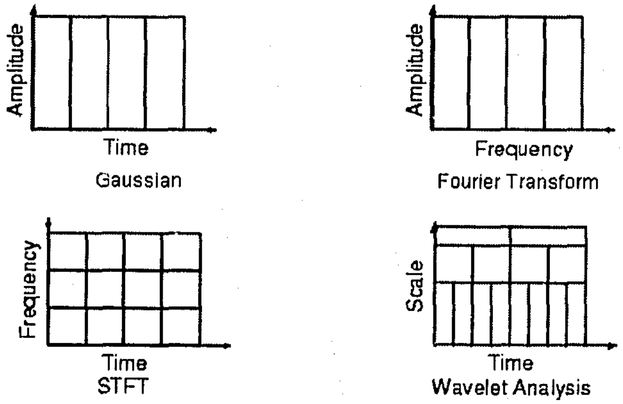


Figure 4.15: Four signal analysis methods

commonly used Gaussian filter or Fourier transform?

Witkin (1983) presented a time-frequency representation of trends via Gaussian filtering. Because the Fourier transform of a Gaussian still looks like a Gaussian, the Gaussian is localized in both the time and frequency domains. This property makes it an optimal filter. The disadvantages of Gaussian filtering are obvious, such as the difficulty of selecting suitable parameters of Gaussian filter, and so on.

The Fourier analysis perhaps is the most well-known method for signal analysis. It transforms a signal from the time-amplitude view to the frequency-amplitude view, and the magnitude in the frequency domain corresponds to the energy of the signal. The serious drawback of Fourier transform is that no time localization information is available, which means it is impossible to determine when a special event occurred.

The windowed Fourier transform, sometimes called Short-Time Fourier Transform (STFT), is an improvement over Fourier transform in order to correct this deficiency. It can provide some temporal location information; however, it still has some drawbacks; for example, the limit precision of the temporal location information determined by the fixed window size.

Compared with Gaussian filtering and Fourier analysis, the wavelet analysis possesses excellent time-frequency localization properties since it uses a time-scale region. Therefore, using wavelet analysis, we can get multi scale description of trends and extract features, which enable us to analyze the data efficiently.

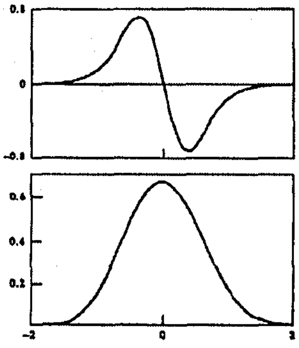


Figure 4.16: Cubic spline wavelet and scaling function

4.2.2 Selection of Wavelet Type

There are different type of wavelet families, such as Haar wavelet and Daubechies wavelets. Depends on the application property, we can select appropriate wavelet.

In this process monitoring method, the wavelet is used to not only smooth the raw data, but also detect the extreme and inflexion points. At the image processing and machine vision field, the Laplacian (second derivative) of a Gaussian is widely used to filter a signal and extract the inflexion point. Along similar lines, Mallat and Zhong (1992) suggested the use of a wavelet which is the first derivative of a scaling function $\phi(x)$. The scaling function has the effect of smoothing. The wavelet, $\psi(x)$ is the first derivative of $\phi(x)$:

$$\psi(x) = \frac{d\phi(x)}{dx}.$$

The wavelet and scaling function used by Mallat and Zhong (1992) is given in Figure 4.16 (Bakshi and Stephanopoulos 1994). This wavelet is a cubic spline of the derivative of the Gaussian function, while the scaling function is the Gaussian function itself. In this application, what we used is *biro*3.1, which is selected from *biorthogonal* wavelets family of Matlab wavelet toolbox. It is similar to the wallet Mallat and Zhong (1992) suggested.

4.2.3 Selection of Wavelet Level

Using wavelet decomposition, we can get approximations and details at different scales. At lower scale, the detail consists of high frequency noise signal and the approximation keeps the dynamic behavior of data. On the other hand, at higher scale, the basic trend of data is extracted and the detail shows the low frequency noise. In this application, the wavelet is used for the noise filtering and the feature extraction; therefore, high scale is more suitable. However, if the selected scale is beyond certain level, the most important information contained in a signal, which can help to differentiate it from others, will lost. Therefore, the selection of an appropriate level is the first and the most crucial step of this process monitoring method.

Wong *et al.* (1998) suggested the use of signal to noise ratio (SNR). The disadvantage of this method is that, in order to calculate the SNR, the noise free signal is needed. For most industrial cases, this condition is impossible to satisfied, thus this method is impractical for industrial applications.

So far, there is no way of automatically selecting an appropriate wavelet level for any given data. All the results presented in this chapter have to base on visualization and experience.

4.2.4 Approximate Derivative Calculation

The extraction of extrema and inflexion points from a trend is based on the calculation of first and second derivative. The simplest way to perform derivative calculation is numerical differentiation:

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}. \tag{4.18}$$

For discrete data, the sample interval usually is even and equal to one, then Equation 4.18 can be simplified to

$$f'(x_i) = f(x_i) - f(x_{i-1}). \tag{4.19}$$

Similarly, the second derivative can be calculated as

$$f''(x_i) = f'(x_i) - f'(x_{i-1}). \tag{4.20}$$

This numerical differentiation method is direct and easy to implement. However, its drawback is also obvious: it is too sensitive to noise and the first derivative data must be smoothed by certain filter before the calculation of the second derivative . An approximate derivative calculation method proposed by Leung (1998) has a major advantage that differentiation and smooth can be carried out in the same calculation.

In this method, the approximate first derivative is calculated as the difference between two scale coefficients at the first resolution level:

$$f'(x) = C_{1,D_{16}} - C_{1,D_2} \tag{4.21}$$

where D_{16} and D_2 denote the sixteenth and second order of Daubechies family wavelets, respectively. Using the first derivative data as the input to the wavelet transform, the second derivative calculation can be achieved.

The disadvantage of this approximate derivative method is that the number of data points gets halved after each derivative order computation (Nie *et al.* 2002). The decreasing of data points makes it difficult to exactly find the location of extrema and inflexion points of smoothed trend. For example, as mentioned before, inflexion points in a signal appear as extrema in first derivative data, and zero-order zero crossings in second derivative data. Because the number of data points becomes halved and quartered, the maximum bias of data location on the smoothed trend will increase to four points. Thus, this method is only recommended to use for cases in which the smoothed data trend still looks noisy.

4.2.5 Selection of HMM Parameters

For any HMM application, the selection of hidden state number N is important. Unfortunately, there is no simple, theoretically precise way of making such choice. In this application, the observation symbol number M is equal to 36. Different state numbers have been tried for simulation examples and industrial case studies. For most cases, state numbers, ranging from 6 to 9, work well and all the results about HMM presented in this chapter are based on the selection that N is equal to 6.

For the estimation of the transition probability matrix P and the observation probability matrix B, the iterative algorithm is a maximum likelihood estimation procedure and will lead to local maxima. Therefore, the initial estimation is crucial. In Smyth (1994), these matrices are estimated by a database of trouble reports, for example, the mean time between failure (MTBF) data, which are routinely collected from a real-world system. For our application, unfortunately, this kind of database is not available because of limited data type and size. Therefore, the initial values of matrices P and B have to been randomly selected and then regulated to stochastic matrices according the following equations:

$$\begin{aligned} &\sum_{j=1}^{N} P_{ij} = 1, \quad 1 \le i \le N \\ &\sum_{j=1}^{M} B_{ij} = 1, \quad 1 \le i \le N \end{aligned} \tag{4.22}$$

4.3 Simulation Examples

In this section, two simulation examples are given to illustrate that this process monitoring method can pick up the different behaviors of a process.

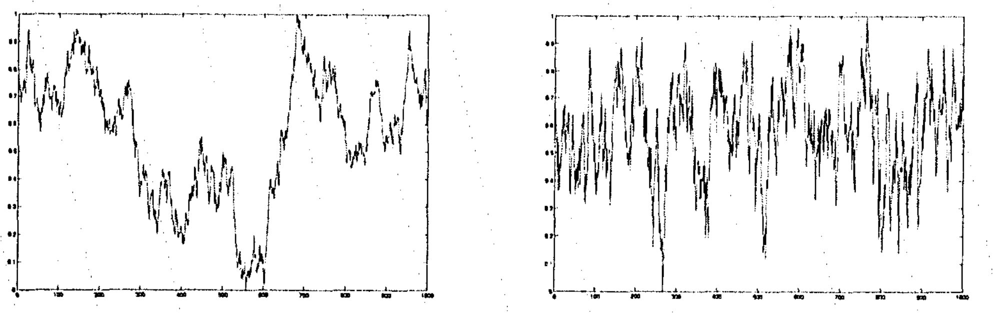
4.3.1 Example 1

In this example, the normal and abnormal behaviors of a process are modelled as follow:

$$\begin{aligned} T_{Normal} &= \frac{0.01}{1 - 0.99q^{-1}} \\ T_{Abnormal} &= \frac{0.1}{1 - 0.9q^{-1}} \end{aligned}$$

Two sets of estimation data are simulated from the normal model and the abnormal model, respectively, as shown in Figure 4.17. The data length is 10,000 points. The obvious difference between the normal and the abnormal behaviors of the process can be seen just at a glance in Figure 4.17. As we introduced in the above sections of this chapter, the original data is filtered by the wavelet transform. Figure 4.18 shows the filtered normal and abnormal data. Comparing Figure 4.18 with Figure 4.17, we can see that the filtered data keeps the basic trend of the original data and high frequency noise has been removed. The next step is the triangular representation, by which the filtered data is represented by triangles. Every episode is described by one type of the triangle: A, B, C or D.

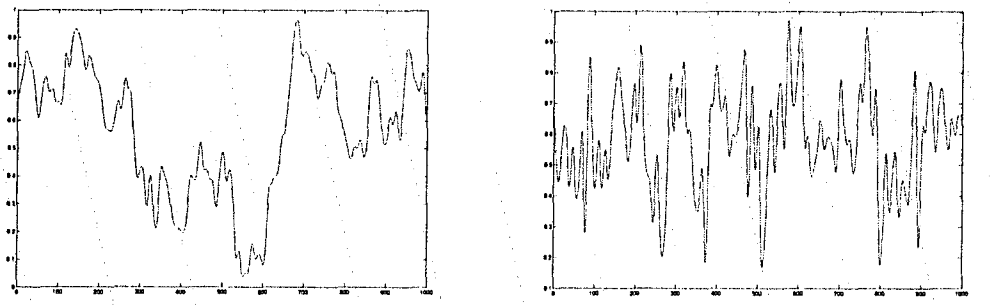
Therefore, 10,000 points of normal and abnormal data are converted into 1,604 and 1,747 triangles, respectively. The magnitudes and durations of these triangles are collected and divided into three zones : *large, middle, small* or *long, medium, short*. Because every episode corresponds to one of 36 types of triangles, the basic trend of data is converted into a discrete sequence, which has 36 states. Part of normal and abnormal sequences are shown in Figure 4.19, from which the difference between the normal and the abnormal



(a) Normal data

(b) Abnormal data

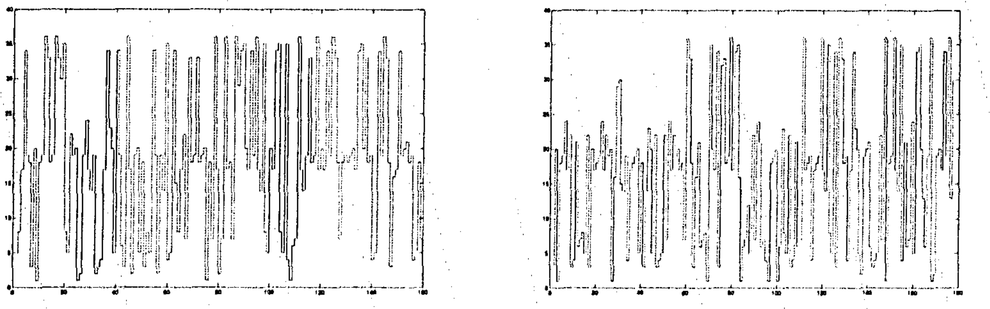
Figure 4.17: Original data



(a) Normal data

(b) Abnormal data

Figure 4.18: Basic trends of the original data



(a) Normal data

(b) Abnormal data

Figure 4.19: State sequences of the original data

behavior can still be observed. As the last step of estimation, a normal HMM, HMM_N and an abnormal HMM, HMM_A, have been estimated from these sequences.

Then, two sets of normal and abnormal data with length of 1,000 points are simulated as validation data. They are filtered, triangully described and finally converted into discrete sequences, same as what have been done for the estimation data. The only difference is that the division of triangles into subtype is based on the magnitude and duration distributions of the estimation data. The validation data of normal behavior, S_N, consists of 165 triangles and that of abnormal, S_A, 177 triangles. Both sequences are input into the normal HMM and the abnormal HMM, individually.

Table 4.1: Simulation result

Triangle #	P($S_N \| HMM_N$)	P($S_A \| HMM_N$)	P($S_N \| HMM_A$)	P($S_A \| HMM_A$)
20	14.82%	0.02%	0%	10.07%
30	14.86%	0.16%	0%	10.58%
40	14.81%	0.47%	0%	12.38%
50	15.51%	0.81%	0%	12.56%
60	14.18%	1.42%	0%	13.97%
70	14.17%	0.31%	0%	12.82%
80	14.40%	0.53%	0%	13.97%

The simulation result is shown in Table 4.1. The data in the first column of table is the length of validation data in triangle numbers. Different length of validation data has been tried to see whether the results are consistent. The second column data are the probabilities that the validation data of normal behavior is generated by the normal HMM. The third is that of validation data of abnormal behavior generated by the normal HMM. The fourth one is the normal data by the abnormal HMM and the last one, the abnormal data by the abnormal HMM.

In Table 4.1, we notice that, compared with the probability that the normal data is generated by the normal HMM, the probability of the abnormal data generated by the normal HMM is much smaller. Similar conclusion holds for the abnormal HMM. This means that either normal HMM or abnormal HMM can distinguish the normal and the abnormal data. We also notice that the results are consistent for different length of validation data.

Therefore, this process monitoring method can detect the behavior change of the process.

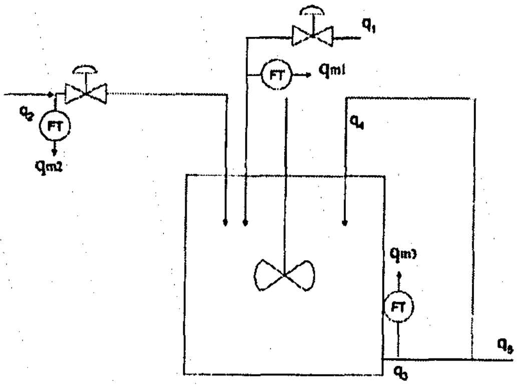


Figure 4.20: Schematic of a blending process

4.3.2 Example 2

The second example is from Huang (2001). It is a blending tank as shown in Figure 4.20 (Huang 2001). The mass balance of this process is

$$q_1 + q_2 = (1 - c)q_3, \tag{4.23}$$

where c is the recycling rate with a nominal value $c_0 = 0.37$. The measurements of q_1, q_2 and q_3 are corrupted by measurement noise and can be written as

$$\begin{aligned} q_{m1} &= k_1 q_1 + v_1 \\ q_{m2} &= k_2 q_2 + v_2 \\ q_{m3} &= k_3 q_3 + v_3 \end{aligned} \tag{4.24}$$

where k_1, k_2 and k_3 are the gains of the flow rate sensors with nominal values 1, v_1, v_2 and v_3 are white noises. In the simulation, q_1 and q_2 are random signals with uniform distribution and magnitude varying between -1 and +1.

Three kinds of fault are simulated in this example. They are 10% increase of the sensor gain of q_1, the sensor gain of q_2 and the recycle rate, respectively, as shown in Table 4.2.

The simulation results are given in Table 4.3 and Table 4.4. In Table 4.3, the standard deviation of white noise is 0.1. For every fault scenario, a set of 10,000 points of normal data is simulated and used for HMM model training. This data set is filtered by wavelet and then described by a series of triangles. The figures in the second column of Table 4.3

Table 4.2: Three types of simulated fault

Source of fault	Sensor 1 gain	Sensor 2 gain	Recycle rate
no fault	1	1	0.37
sensor 1	1.1	1	0.37
sensor 2	1	1.1	0.37
recycle	1	1	0.407

are the triangle numbers. 1,000 points of normal data are simulated and used for model validation. The number of triangles of this validation data set is given in the third column of Table 4.3. The probability that this data set is generated by corresponding HMM is calculated, as shown in the fourth column of Table 4.3. This probability can be thought as a benchmark. Another set of 1,000 points of data is simulated in every fault scenario. The number of triangle and the probability that this fault data is generated by the normal HMM are given in the last two columns of Table 4.3.

Table 4.3: Noise standard deviation of 0.1

Source of fault	Model Estimation Data	Normal Validation Data	P(S_N|HMM_N)	Abnormal Testing Data	P(S_A|HMM_N)
sensor 1	934	87	22.40%	95	0%
sensor 2	905	95	20.29%	104	0%
recycle	920	89	19.78%	104	0%

Compared with the probability that the normal data is generated by the normal HMM, the probability of fault data is quite different. They are zeroes. This means this method can effectively tell the difference between normal and faulty data.

The simulation results with standard deviation of 0.3 are given in Table 4.4, from which we can see that this method does not deteriorate with the increase of noise standard deviation.

Table 4.4: Noise standard deviation of 0.3

Source of fault	Model Estimation Data	Normal Validation Data	P(S_N|HMM_N)	Abnormal Testing Data	P(S_A|HMM_N)
sensor 1	907	90	19.54%	89	0%
sensor 2	931	91	19.14%	98	0%
recycle	897	87	17.46%	97	0%

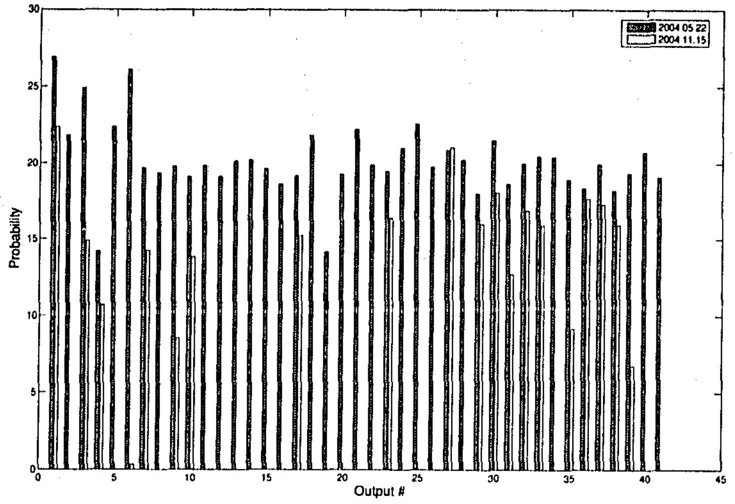


Figure 4.21: GOHTU data analysis result

4.4 Industrial Case Studies

In this section, this process monitoring method is applied to both GOHTU and PSV plant. The useful information achieved show the effectiveness of this method.

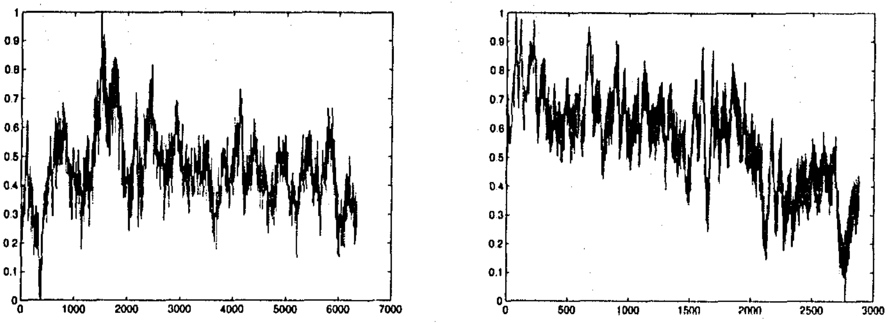
4.4.1 Case Study 1 – GOHTU

As an illustrative example, two sets of real data are analyzed by this method. One of them is collected at May 22th, 2004 and the other at Nov 11, 2004. For every output loop, one HMM is trained by the first data set and the probabilities that these two sets of data are generated by this model are given in Figure 4.21.

What kind of information we can get from this figure? For every output loop, if those two probabilities are close, that means the behaviors of this output loop at these two different days are similar. For example, the 36^{th} output, as shown in Figure 4.22.

If the probabilities are quite different, on the other hand, it warns that the behavior of this output loop has changed. One example , 6^{th} output, is shown in Figure 4.23.

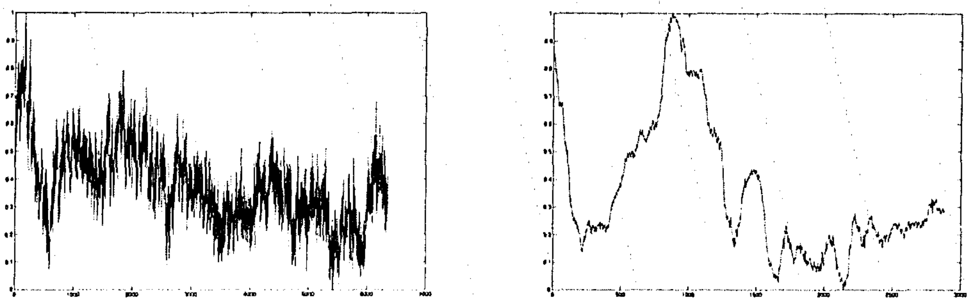
Therefore, this process monitoring method can monitor a process efficiently, especially for a large process like this GOHTU plant which has considerable inputs and outputs.



(a) May 22, 2004

(b) Nov 11, 2004

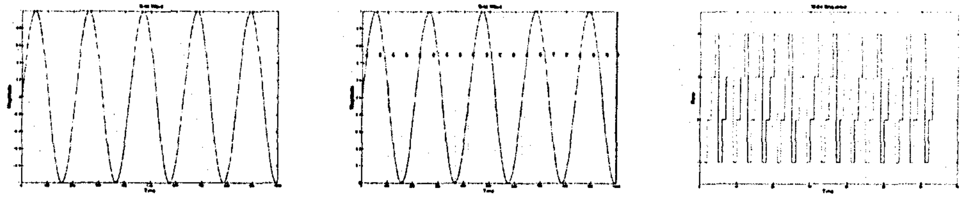
Figure 4.22: The 36^{th} output



(a) May 22, 2004

(b) Nov 11,2004

Figure 4.23: The 6^{th} output



(a) Original signal (b) Triangular presentation (c) State sequence

Figure 4.24: Sine wave

4.4.2 Case Study 2 – PSV

This part has been removed due to the consideration of proprietary.

4.5 Oscillation Detection

With different HMM, this process monitoring method can be used to detect different process behaviors. In this section, as a simple illustrative example, we will introduce how to apply this method for oscillation detection.

4.5.1 Methodology

The most obvious characteristic that differentiate the oscillated signal from other signals is the periodic increase and decrease of the oscillated signal at certain frequency and magnitude. As the sine wave shown in Figure 4.24 (a), every cycle of the oscillation consists of four monotonic parts: concave downward increase, concave downward decrease, concave upward decrease and concave upward increase. As introduced before, each part corresponds to one type of triangle. With the triangular representation, the oscillation can be described by a regularly repeated character string: $A, B, C, D, A, B, C, D, A, B, \cdots$, as shown in Figure 4.24 (b). Because the magnitude and duration of the triangles are neglected, this method can detect the oscillation with changing magnitude and frequency. Every character corresponds to one state, thus the oscillation is converted into a cyclic sequence, which has four states. Figure 4.24 (c) shows the cyclic sequence.

Therefore, the oscillation can be described as a cyclic Markov chain and modeled as a simple HMM, in which there are just four states and every state corresponds to one observation. The transition probability matrix P and observation probability matrix B

are

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

respectively.

From P we can see that the process will certainly transit from state 1 to state 2, from state 2 to state 3, and so on. Thus, it generates a cyclic Markov chain in ascendant order. The matrix B is an identity matrix, which means that at every state, one type of observation will be certainly observed. In this case, the types of triangle A, B, C and D correspond to state 1, 2, 3 and 4, respectively.

In fact, the transition probability matrix and observation probability matrix are estimated as

$$\hat{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.06 & 0 & 0.94 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

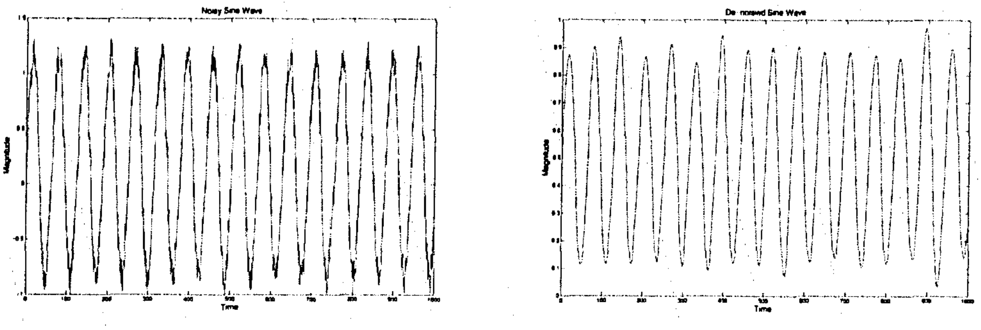
$$\hat{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

through a simulated sine wave.

4.5.2 Simulation Example

This HMM can distinguish the oscillated signal from other kinds of signals. As an example, same length of noisy sine wave and white noise signal are compared. The data length is 1,000 points.

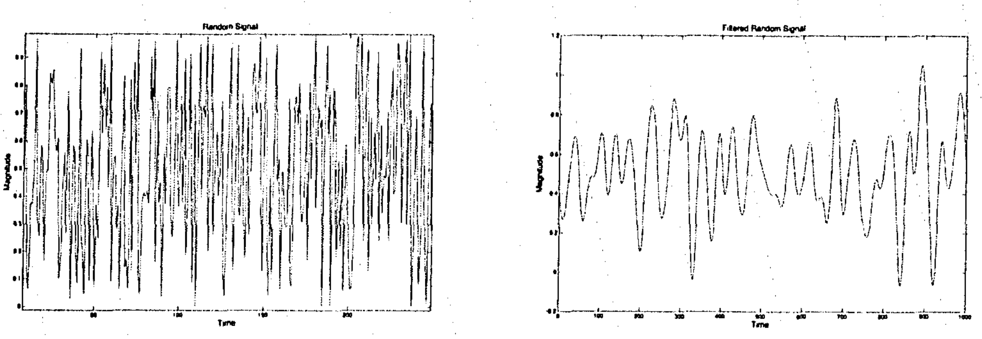
First, a noisy sine wave is simulated. The magnitude and the frequency of this sine wave are 1 and 0.1, individually. This sine wave is corrupted by a white noise signal with standard deviation of 0.1. The noisy sine wave is filtered by the wavelet transform. Figure



(a) Noisy signal

(b) Filtered signal

Figure 4.25: Sine wave



(a) Original signal

(b) Filtered signal

Figure 4.26: White noise with standard deviation 0.1

4.25 shows the noisy and filtered sine wave. Then the filtered sine wave is converted into a discrete state sequence, which is input to HMM. The probability that this noisy sine wave is oscillatory is as high as 94.32%.

Second, the white noise signal that corrupts above sine wave is analyzed in same way. The original and filtered signals are shown in Figure 4.26. The probability that it is an oscillation signal is just 6.45%.

4.5.3 Industrial Case Study

A set of industrial data collected at the Eastman Chemical Company, Kingsport, Tennessee is analyzed by this method. The sampling interval was 20s.

The probabilities of the signals shown in Figure 4.27 are higher than 90%. On the other

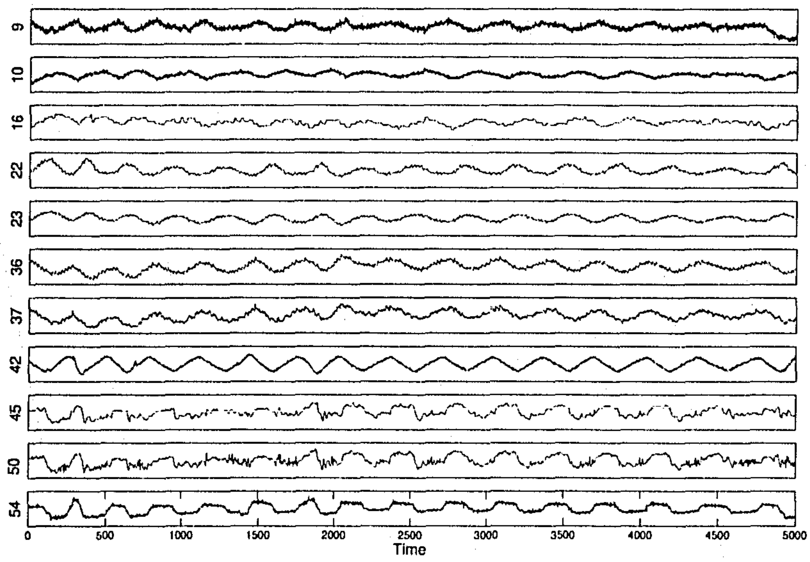


Figure 4.27: Signals detected as oscillatory

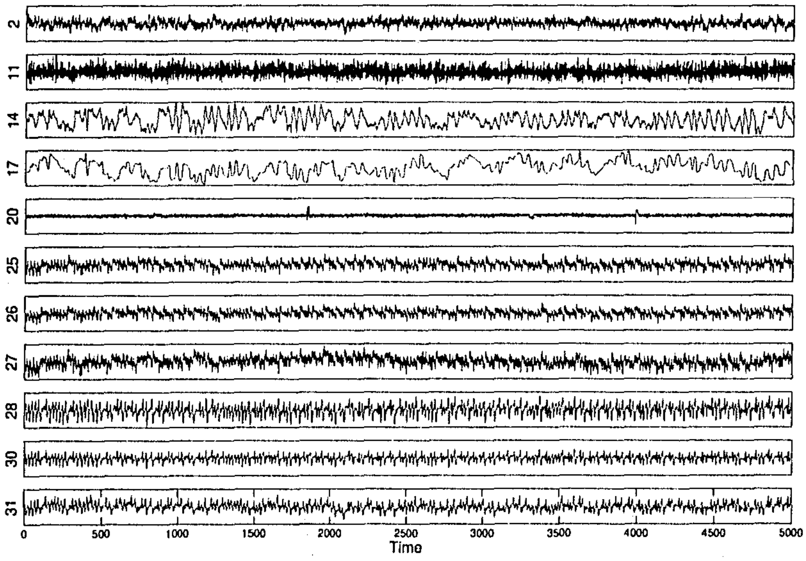


Figure 4.28: Signals detected as having no oscillation

hand, Figure 4.28 shows some signals whose probabilities are almost zero. Compared these two figures, we can say that this method can effectively detect the oscillation.

4.6 Conclusions

In this chapter, a practical process monitoring method is presented. The basic concepts of its three parts: wavelet transform, symbolic representation and HMM are introduced.

This method is verified by two simulation examples, which shows that this method can effectively detect the behavior change of the monitored variable. The potential use of this method is illustrated by industrial case studies. For example, this method can be used as tool for data analysis.

Moreover, an oscillation detection approach is presented as an illustrative example of the application of this process monitoring method. The simulation example and industrial case study show that this method can effective detect the oscillation. It will be useful when we are facing a large amount of data and variables, where it is impossible to visualize the trend of each individual variable.

5

Conclusions and Future Extensions

5.1 Conclusions

In this thesis, new developments for the multivariate process and control monitoring are presented.

- The interactor matrix used to be an obstacle of the performance assessment of MIMO systems. In Chapter 2, an improved algorithm for calculation of the interactor matrix is presented. If the interactor matrix is known, the performance index of a MIMO system can be estimated from a set of closed-loop data by the FCOR algorithm. The subspace method simplifies the calculation of performance index and gives an explicit "one-shot" solution. The performance can be assessed from a set of open loop experimental data and a set of closed-loop routine operating data. No concept and calculation of the interactor matrix is needed any more.

- A mathematical model can bring some invaluable insight about the process. If the behavior of a process can be modeled by different types of models and all these models are meaningful, and it means that we have the opportunity to observe something from different angles.

  The Markov chain introduced in Chapter 3 is another angle from which we observe the behavior of a process. The transition probability matrix reveals how the process evolves; the equilibrium distribution predicts how the process will be in the future; the passage time and passage details elaborate how the process behaves in a transition, etc.

- When we try to evaluate something in the real world, we are not just considering its present behavior. We will search our memory and put some historical records into account because one behavior has different interpretations under different backgrounds. The same principle holds for the process monitoring.

  The process monitoring method presented in Chapter 4 monitors the basic trend of variables, not the isolated, present measured value. Thus it can effective detect the behavior change of a process. The potential use of this method includes the oscillation detection.

- The above algorithms and methods have been programmed in Matlab environment. All of them have been integrated into a performance analytical toolbox. Moreover, the bugs of two Matlab functions have been found and corrected.

- All the methods introduced in this thesis have been applied to industrial applications. These applications helped the engineers and managers to achieve better understanding about the processes, for example, the GOHTU plant. The performance assessment of the advanced controller of this plant identified the potential for further improvement in term of variability. The Markov chain analysis revealed that this plant was in less economic status although it was in stable control at the same time.

  On the other hand, the industrial applications also reveal some limitations of these methods that need further improvement. Such as, the time series analysis part of the FCOR algorithm is not robust enough to a large dimensional process.

5.2 Future Extensions

Although the achievements presented in this thesis are obvious, there are some issues which can be further improved.

- A more robust time series analysis algorithm is needed for the FCOR algorithm.

- A method should be developed, which can automatically select an appropriate wavelet level for any given data.

- The process monitoring method introduced in Chapter 4 can extend to multivariate variables.

- The processes in this thesis are assumed to be first-order Markov chains and homogenous. Anderson and Goodman (1956) discussed the methods which can be used to test the hypotheses that the process is a nth order Markov chain, that several samples are from the same Markov chain of a given order, etc. The test methods of similar hypotheses about HMM can provide useful information, such as the confidence region, for the process monitoring method introduced in Chapter 4.

Bibliography

[1] T. W. Anderson and Leo A. Goodman, Statistical inference about Markov chains, *The Annals of Mathematical Statistics,* Vol.28, No.1, pp.89-110, 1957.

[2] B. R. Bakshi and G. Stephanopoulos, Representation of process trends – III. Multiscale extraction of trends from process data, *Computers and Chemical Engineering,* Vol.18, No.4, pp.267-302, 1994.

[3] J. T-Y. Cheung and G. Stephanopoulos, Representation of process trends – I. A formal representation framework, *Computers and Chemical Engineering,* Vol.14, No.5, pp.495-510, 1990.

[4] J. T-Y. Cheung and G. Stephanopoulos, Representation of process trends – II. The problem of scale and qualitative scaling, *Computers and Chemical Engineering,* Vol.14, No.5, pp.511-539, 1990.

[5] L. H. Chiang, E. L. Russell and R. D. Braatz, Fault detection and diagnosis in industrial systems, *Springer,* 2001.

[6] I. Daubechies, The wavelet transform, time-frequency localiztion and signal analysis, *IEEE Trans. Inform. Theory,* Vol.36, pp.961-1005, 1990.

[7] G. Goodwin and K. Sin, Adaptive Filtering Prediction and Control, *Englewood Cliffs: Prentice-Hall,* 1984.

[8] M. J. Grimble, Integral minimum variance control and benchmarking, *Journal of Process Control,* Vol.14, No.2, pp.177-191, 2004.

[9] T. J. Harris, Assessment of control loop performance, *Canadian Journal of Chemical Engineering,* Vol.67, No.5, pp.856-861, 1989.

[10] L. D. Desborough and T. J. Harris, Performance assessment measures for univariate feedforward/feedback control, *Canadian Journal of Chemical Engineering*, Vol.71, No.4, pp.605-616, 1993.

[11] T. J. Harris, F. Boudreau and J. F. MacGregor, Performance assessment of multivariable feedback controllers, *Automatic,* Vol.32, No.11, pp.1505-1518, 1996.

[12] T. J. Harris, C. T. Seppala and L. D. Desborough, A review of performance monitoring and assessment techniques for univariate and multivariate control systems, *Journal of Process Control,* Vol.9, No.1, pp.1-17, 1999.

[13] T.J. Harris and W. Yu, Analysis of multivariable controllers using degree of freedom data, *International Journal of Adaptive Control and Signal Processing,* Vol.17, pp.569-588, 2003.

[14] Alexander Horch and Alf J. Isaksson, A modified index for control performance assessment, *Journal of Process Control,* Vol.9, No.6, pp.475-483, 1999.

[15] Biao Huang, Sirish L. Shah and Hiroyuki Fujii, The unitary interactor matrix and its estimation using closed-loop data, *Journal of Process Control,* Vol.7, No.3, pp.195-207, 1997.

[16] Biao Huang, S. L. Shah and E. K. Kwok, Good, bad or optimal? Performance assessment of multivariable processes, *Automatic,* Vol 33, No.6, pp 1175-1183, 1997.

[17] Biao Huang and Sirish L. Shah, Practical issues in multivariable feedback control performance assessment, *Journal of Process Control,* Vol 8, No.5-6, pp 421-430, 1998.

[18] Biao Huang and Sirish L. Shah, Performance assessment of control loops. Theory and applications, *Springer*, 1999.

[19] Biao Huang, Detection of abrupt changes of total least squares models and application in fault detection, *IEEE Transaction on Control Systems Technology,* Vol 9, No.2, pp 357-367, 2001.

[20] Biao Huang, Ramesh Kadali and Steven X. Ding, A direct input/output data-driven approach to multivariable feedback control performance assessment without interactor matrices, 2004.

[21] Biao Huang, Steven X. Ding and Nina Thornhill, Practical solutions to multivariable feedback control performance assessment problem: reduced a prior knowledge of interactor matrices, *Journal of Process Control,* Vol 15, No.5, pp 573-583, 2005.

[22] John G. Kemeny and J. Laurie Snell, Finite Markov chains, *Van Nostrand: Princeton,* 1976.

[23] Byung-Su Ko and Thomas F. Edgar, Performance assessment of cascade control loops, *A.I.Ch.E Journal,* Vol.46, pp.281-291, 2000

[24] Byung-Su Ko and Thomas F. Edgar, Performance assessment of multivariable feedback control systems, *Automatic,* Vol.37, No.6, pp.899-905, 2001

[25] Alexander Kai-man Leung, Foo-tim Chau and Jun-bin Gao, Wavelet transform: a method for derivative calculation in analytical chemistry, *Anal. Chem,* Vol.70, pp.5222-5229, 1998.

[26] Stephane Mallat, A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol.11, No.7, pp.674-693, 1989.

[27] Stephane Mallat, Zero-Crossings of a Wavelet Transform, *IEEE Transactions on Information Theory,* Vol.37, No.4, pp.1019-1033, 1991.

[28] Stephane Mallat and Sifen Zhong, Characterization of signals from multiscale edges, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol.14, No.7, pp.710-732, 1992.

[29] R. Di Mascio and G. W. Barton, The economic assessment of process control quality using a Taguchi-based method, *Journal of Process Control,* Vol.11, No.1, pp.81-88, 2001.

[30] P. Goupillaud, A. Grossmann and J. Morlet, Cycle-octave and related transforms in seismic signal analysis, *Geoexploration,* Vol.23, pp.85-102, 1984.

[31] Lei Nie, Shouguo Wu, Xiangqin Lin, Longzhen Zheng and Lei Rui, Approximate derivative calculated by using continous wavelet transform, *J. Chem. Inf. Comput. Sci.,* Vol.42, pp.274-283, 2002.

[32] Michael A. Paulonis and John W. Cox, A practical approach for large-scale controller performance assessment, diagnosis and improvement, *Journal of Process Control,* Vol.13, No.2, pp.155-168, 2003.

[33] Youbin Peng and Michel Kinnaert, Explicit solution to the singular LQ regulation problem, *IEEE Transaction on Automatic Control,* Vol.37, No.5, pp.633-635, 1992.

[34] Lawrence R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of The IEEE,* Vol.77, No.2, pp.257-286, 1989.

[35] Maciej W. Rogozinski, Andrzej P. Paplinski and Michael J. Gibbard, An algorithm for the calculation of a nilpotent interactor matrix for linear multivariable systems, *IEEE Transaction on Automatic Control,* Vol.AC-32, No.3, pp.234-237, 1987.

[36] V. I. Romanvsky, Discrete Markov chains, *Van Nostrand: Princeton,* 1976.

[37] S. L. Shah, C. Mohtadi and D. W. Clarke, Multivariable adaptive control without a prior knowledge of the delay matrix, *Systems and Control Letters,* Vol.9, No.4, pp.295-306, 1987.

[38] Padhraic Smyth, Hidden Markov models for fault detection in dynamic systems, *Pattern Recognition,* Vol.27, No.1, pp. 149-164, 1994.

[39] N. F. Thornhill, M. Oettinger and P. Fedenczuk, Refinery-wide control loop performance assessment, *Journal of Process Control,* Vol.9, No.2, pp.109-124, 1999.

[40] Andrew P. Witkin, Scale space filtering: a new approach to multi-scale description, *Image Understanding(S. Ullman and W. Richards),* pp.79-95, Ablex Norwood, NJ, 1983

[41] W. Wolovich and P. Falb, Invariants and canoncial forms under dynamic compensation, *SIAM J. Control,* Vol.14, pp.996-1008, 1976

[42] W. Wolovich and H. Elliott, Discrete models for linear multivariable systems, *Int. J. Control,* Vol.38, No.2, pp.337-357, 1983

[43] James C. Wong, Karen A. McDonald and Ahmet Palazoglu, Classification of process trends based on fuzzified symbolic representation and hidden Markov models, *Journal of Process Control,* Vol.8, No.5-6, pp.395-408, 1998.

[44] James C. Wong, Classification of process trends based on fuzzified symbolic representation and hidden Markov models, *PhD disertation, University of California, Davis, CA*, 1998.

[45] James C. Wong, Karen A. McDonald and Ahmet Palazoglu, Classification of abnormal plant operation using multiple process variable trends, *Journal of Process Control,* Vol.11, No.4, pp.409-418, 2001.

Appendices

A
Matlab Debugging

Undoubtedly, Matlab is a very successful computing language and commercial software. However, unfortunately and inevitably, there are still some bugs in some Matlab functions. For instance, when compute the product of two transfer functions or transfer function matrices using the Matlab function "*" , sometimes, some coefficients of the result that should be zero will be actually very small numbers.

For instance, for a 2 × 2 MIMO system

$$T = \begin{bmatrix} \frac{q^{-1}}{1+q^{-1}} & \frac{q^{-1}}{1+2q^{-1}} \\ \frac{q^{-1}}{1+3q^{-1}} & \frac{q^{-1}}{1+4q^{-1}} \end{bmatrix},$$

one unitary interactor matrix is

$$D = \begin{bmatrix} 0.5q^2 + 0.5q & -0.5q^2 + 0.5q \\ 0.5q^3 - 0.5q^2 & -0.5q^3 - 0.5q^2 \end{bmatrix}$$

and the product of $D \times T$ should be

$$\begin{bmatrix} \frac{2+2q^{-1}}{1+4q^{-1}+3q^{-2}} & \frac{2+3q^{-1}}{1+6q^{-1}+8q^{-2}} \\ \frac{-2}{1+4q^{-1}+3q^{-2}} & \frac{-3}{1+6q^{-1}+8q^{-2}} \end{bmatrix}.$$

However, if we do this multiplication using Matlab function "*", the result is

$$\begin{bmatrix} \frac{2+2q^{-1}}{1+4q^{-1}+3q^{-2}} & \frac{2+3q^{-1}}{1+6q^{-1}+8q^{-2}} \\ \frac{-2.22e^{-16}q^{2}-2}{1+4q^{-1}+3q^{-2}} & \frac{-3}{1+6q^{-1}+8q^{-2}} \end{bmatrix}.$$

Comparing these two results, we can find a very small and strange number: $-2.22e^{-16}$.

This kind of problem is very common when using a computer to do floating-point computation. For Matlab, we can use the command "eps" to get the floating-point relative accuracy. It is $2.2204e^{-16}$, almost same as that small number found in above example. Usually, this kind of small number can be omitted. In some cases, however, it will result in absolutely different solutions.

At the iteration step of algorithm 2, for example, when calculate the limit of $D \times T$ as $q^{-1} \to 0$, the result for the correct product is

$$\lim_{q^{-1}\to 0} DT = \lim_{q^{-1}\to 0} \begin{bmatrix} \frac{2+2q^{-1}}{1+4q^{-1}+3q^{-2}} & \frac{2+3q^{-1}}{1+6q^{-1}+8q^{-2}} \\ \frac{-2}{1+4q^{-1}+3q^{-2}} & \frac{-3}{1+6q^{-1}+8q^{-2}} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ -2 & -3 \end{bmatrix}$$

and for the second product is

$$\lim_{q^{-1}\to 0} DT = \lim_{q^{-1}\to 0} \begin{bmatrix} \frac{2+2q^{-1}}{1+4q^{-1}+3q^{-2}} & \frac{2+3q^{-1}}{1+6q^{-1}+8q^{-2}} \\ \frac{-2.22e^{-16}q^{2}-2}{1+4q^{-1}+3q^{-2}} & \frac{-3}{1+6q^{-1}+8q^{-2}} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ \infty & -3 \end{bmatrix}.$$

The difference between these two results is that the first one is full-rank constant matrix which means D is a unitary interactor matrix for the given T and the second one, however, is non-causal. This example shows that the correct answer probably will be missed when using Matlab multiplication function "*" in above algorithm.

Moreover, in the case of high order transfer function multiplication, this kind of strange number will be quite large and, sometime, it is difficult to tell them from other normal coefficients. Therefore, this problem will seriously affect the computing accuracy.

Thus, the multiplication of two transfer function (matrices) has to be programmed explicitly in the new algorithm instead of directly using the Matlab multiplication function "*".

First step is to find out where this kind of small, strange numbers come from. The number in above example appeared at the first item of the second row and it is the result of "$D(2,1)*T(1,1)+D(2,2)*T(2,1)$". Just calculate this equation in Matlab environment, the answer is $\frac{-2.22e^{-16}q^{-3}-2q^{-1}}{q^{-1}+4q^{-2}+3q^{-3}}$, which shows that either multiplication of transfer function matrices or that of transfer functions will result in this kind of problem.

Next, let "$a = (0.5q^2 - 0.5q) * (1 + 3q^{-1})$", the answer is "$0.5q^2 + q - 1.5$" and let "$b = (-0.5q^2 - 0.5q) * (1 + q^{-1})$", the answer is "$-0.5q^2 - q - 0.5$". All look fine so far. When adding them together, that strange number is coming.

$$a + b = -2.22e^{-16}q^2 - 2.$$

It should just be -2!

The problem is arose from the addition. The reason is, sometimes, we can not get the exact zero because of the floating-point accuracy. In the above example, for instance, when add $0.5q^2$ and $-0.5q^2$ together, the answer is $-2.22e^{-16}q^2$ instead of 0. Moreover, this kind of small number will be accumulated to a quite large number with the repeat of addition. In order to overcome this problem, we should keep everything happened during computation under our control. The key issues are

- For transfer function computation, just use vector multiplication and vector addition and do not use any computation operating on transfer function or transfer function matrix;

- After every single vector addition operation, check whether all the coefficients are in a reasonable range. Set those coefficients which have same magnitude order as *eps* to zero, so as to keep the result as accurate as possible and protect these numbers from accumulation;

- Do not use the vector division operation, since it will call the addition operation automatically and results in the accumulation of those strange, small coefficients.

For example, the multiplication of two transfer function matrices, D and T, is calculated in following way:

Let $m \times m$ matrix D expressed as

$$\begin{bmatrix} D(1,1) & D(1,2) & \cdots & D(1,m) \\ D(2,1) & D(2,2) & \cdots & D(2,m) \\ \vdots & \vdots & \vdots & \vdots \\ D(m,1) & D(m,2) & \cdots & D(m,m) \end{bmatrix}$$

and $m \times n$ matrix T as

$$\begin{bmatrix} \frac{N(1,1)}{E(1,1)} & \frac{N(1,2)}{E(1,2)} & : & \frac{N(1,n)}{E(1,n)} \\ \frac{N(2,1)}{E(2,1)} & \frac{N(2,2)}{E(2,2)} & : & \frac{N(2,n)}{E(2,n)} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{N(m,1)}{E(m,1)} & \frac{N(m,2)}{E(m,2)} & \cdots & \frac{N(m,n)}{E(m,n)} \end{bmatrix}$$

in which, $D(i,j), N(i,j)$ and $E(i,j)$ are vectors used to express transfer functions. Similarly, the product DT, a $m \times n$ matrix, is expressed as

$$\begin{bmatrix} DT(1,1) & DT(1,2) & \cdots & DT(1,n) \\ DT(2,1) & DT(2,2) & \cdots & DT(2,n) \\ \vdots & \vdots & \vdots & \vdots \\ DT(m,1) & DT(m,2) & \cdots & DT(m,n) \end{bmatrix}.$$

Every element of DT can be calculated as

$$\begin{aligned} DT(i,j) &= D(i,1) \times \frac{N(1,j)}{E(1,j)} + D(i,2) \times \frac{N(2,j)}{E(2,j)} + \cdots + D(i,m) \times \frac{N(m,j)}{E(m,j)} \\ &= \frac{D(i,1)N(1,j)E(2,j)\cdots E(m,j) + D(i,2)N(2,j)E(1,j)E(3,j)\cdots E(m,j) + \cdots + D(i,m)N(m,j)E(1,j)\cdots E(m-1,j)}{E(1,j)E(2,j)\ldots E(m,j)} \end{aligned}$$

By this way, finally, the multiplication of two transfer function matrices is calculated by just using vector multiplication and vector addition. Moreover, after every vector addition, the coefficients of the resulting vector are checked and are set to zero if they have same magnitude order as *eps*. Therefore, the problem that sometimes there are some strange, small coefficients in the product of two transfer function (matrices) is resolved.