ISSUES IN ON-LINE OPTIMISATION

By

Moufid Mansour

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

CITY UNIVERSITY, LONDON

SCHOOL OF ENGINEERING AND MATHEMATICAL SCIENCES
CONTROL ENGINEERING RESEARCH CENTRE

SEPTEMBER, 2004.

To

My dear mother,

My dear father.

My only brother

Mahdi.

My sisters

Karima, Fatiha, Leila, Ouassila, Amal.

My nephews and nieces

Mohamed-amine, Zahra, Sarah, Haroun, Ramzi

The beautiful memory of my grandfather and grandmother.

The beautiful memory of my uncle Tounsi.

The memory of

Mr. Lackson K. Chishimba BEng, MSc, MPhil.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and gratitude to my supervisor, Doctor Joe E. Ellis for his guidance, encouragement, help and patience throughout the course of this research.

I would also like to thank Dr. V. M Becerra who provided many useful suggestions and support which enhanced the quality of this work.

I wish also to add a special note of thanks to the Algerian government for having funded my studies here at City University.

I would like to thank my mother for her patience, support, encouragement and prayers throughout and my father for his valuable guidance and confidence.

My thanks are due to Kamal, Mahdi, Halim, Chris, Yasser and Shagufta for their encouragement.

Last, but by no means least, I would like to express my thanks to my fellow research colleagues in the Control Engineering Research Centre for their help and encouragement. Thanks to Ziad, Daniel, Stavros, Costas, Ali, Ermina, Tanya, Dimitrios, Dia, Ram and Gabriel. My thanks also to Joan Rivellini and Linda Carr in the Electrical Engineering office for their tireless and caring attitude to help at all times.

DECLARATION

The author grants powers of discretion to the University Librarian to allow this thesis to be copied in whole or part without further reference to him. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

PUBLICATIONS

The following is a list of publications based on the work described in this thesis:

1. Comparison of methods for estimating real process derivatives in on-line optimisation, Mansour M., Ellis J.E., *Applied Mathematical Modelling*, Vol. 27, pp. 275-291, 2003.

2. Issues in On-line Optimisation, Mansour M., Ellis J.E., *Technical Report no. 174*, CERC, City University, London, UK, April 2000.

The following papers based on the work described in this thesis, have been submitted for publication:

1. Process Optimisation with the aid of Artificial Neural Networks, Mansour M., Ellis J.E., *Applied Artificial Intelligence*.
2. Methodology of On-line Optimisation Applied to a Chemical Reactor, Mansour M., Ellis J.E., *Applied Mathematical Modelling*.
3. Data Reconciliation and Gross Error Detection in On-line Optimisation, Mansour M., Ellis J.E., *International Journal of Control*.

ABSTRACT

In general, on-line optimisation can be defined as the on-line process of finding the optimum set-points of the system. Several areas might be concerned in this procedure. This thesis evaluates algorithms for on-line Optimisation. Techniques for steady-state detection, static data reconciliation, gross error detection and steady-state optimisation are presented and implemented separately and within an on-line optimisation methodology.

It has been acknowledged for some time now that the estimation of derivative information is probably the major drawback of the steady-state optimisation technique considered here: the ISOPE algorithm. This thesis investigates the requirements of these derivatives, methods proposed to estimate them, and presents some attempts to overcome some related problems. Also a modified version of the dynamic model identification method that uses a nonlinear model representation is proposed, and compared under simulation with other available techniques. In the same context, an alternative method based on Artificial Neural Networks to estimate the derivatives is also implemented and tested.

Often, rigorous steady-state detection is crucial for process performance assessment, simulation, optimisation and control. In general, at steady-state data is collected for safe, beneficial and rational management of processes. A method for automatic detection of steady-state in multivariable processes is implemented and tested. The technique is applied on a dynamic model of a chemical reactor.

The presence of errors in process measurements can invalidate the potential gains obtained from advanced optimisation and control techniques. Data reconciliation and gross error detection methods are used to reduce the inaccuracies of these measurements. The implementation and application of static data reconciliation and gross error detection techniques in this thesis show a noticeable improvement in the operation of the system, and general control system performance.

The various algorithms mentioned above are successfully implemented and tested under simulation. It is illustrated that in some cases, it is possible to use steady-state detection in conjunction with data reconciliation, gross error detection, parameter estimation and optimisation, to form an on-line optimisation methodology. The methodology was tested on a dynamic model of a chemical reactor.

NOMENCLATURE

Symbol	Definition
A_k	Gain Matrix
$A^*(q^{-1})$	Polynomial matrix
a	Vector of measurement adjustments
$B^*(q^{-1})$	Polynomial matrix
b	Vector of bias parameters
$\hat{b}$	Bias estimate
BR	Broydon's update matrix
C	Weighting matrix
D	Weighting matrix
d	Minimum pure time delay
$d_{f,i}^2$	Filtered square difference of successive data
e	Weighting vector
e_i	Residual of i^{th} process measurement
f	Weighting vector
F_*	Directional derivatives
g	Mapping of output dependent inequality constraints
H	Model input-output mapping
H^*	Real process input-output mapping
I	Identity matrix
K	Relaxation gain matrix
L_1, L_2, L_3	Filter factors
N_d	Length of data window
N_u	Model updating period
n_y	Size of measured outputs vector

n_u	Size of manipulated variables vector
p_u	Vector of price coefficients for manipulated variables
p_y	Vector of price coefficients for measured variables
P_α, P_β	Fixed and updated parameter vectors
$P(.)$	Probability distribution function
p	Integer ratio between data window length and model updating period
Q	Objective function
q^{-1}	Backward shift operator
$\tilde{R}$	Auxiliary matrix
R_{crit}	Critical value of R-statistic
r	Convexification factor
SS	Steady-state identification variable
T	Sampling time
u	Vector of decision variables
$u_{\min}, u_{\max}$	Controls lower and upper bounds
u_d	Desired value for the decision variable vector
u_i	Element i of the external input vector
V	Covariance matrix
v	Vector of manipulated variables
$v_{\min}, v_{\max}$	Lower and upper bounds for the vector of manipulated variables
$\tilde{v}(l)$	Vector of net internal activity levels of neurons in layer l
$v^{(l)}$	Vector of function signals of neurons in layer l
x	State vector
X_i	Process measured variable at time i.
$X_{f,i}$	Filtered value of X at time i.
y	Vector of model outputs
y^*	Vector of measured variables

$\hat{y}$	Predicted future outputs
$y_{\min}, y_{\max}$	Bounds on measured variables
y_m	Vector of measured process variables
y_d	Reference value for the measured variable
y_i	Element j of the network output vector
y_{true}	Vector of true process variables
Z^{-1}	Unit delay
$W^{(l)}$	Synaptic weight vector of a neuron in layer l
w	Set of relaxation variables

List of Greek Symbols

α	Vector of free parameters
α_i	Level of significance of variable i
Δ	Increment operator
ε	Vector of random measurement errors
ε_g	Vector of gross errors
ε_i	Standardised measurement error
ε_r	Vector of random errors
Φ	Terminal state weighting matrix
ζ_{av}	Average squared error
λ	Lagrange multiplier
λ_f	Forgetting factor
Θ	Least squares parameter matrix
φ	Least squares regression vector
ρ	Penalty factor
η	Learning-rate parameter
δ	Perturbation signal
δ_j	Maximum allowed value
$\sigma_{\min}$, $\sigma_{\max}$	Minimal and maximal singular values
$\delta^{(l)}$	Vector of local gradients of neurons in layer l
Γ	Auxiliary matrix
γ	Activation function
σ	Measurement noise standard deviation
Λ	Matrix of eigenvalues of H

List of Abbreviations

Abbreviation	Description
ADALINE	ADAptive LINear Element
AI	Artificial Intelligence
AISOPE	Augmented ISOPE
ALMISOPE	Approximate Linear Model ISOPE
ANN	Artificial Neural Networks
ARMA	Auto Regressive Moving Average
ARMAX	Auto Regressive Moving Average with Exogeneous input
BIBO	Bounded Input Bounded Output
CSTR	Continuous Stirred Tank Reactors
DCS	Distributed Control System
DISOPE	Dynamic Integrated System Optimisation and Parameter Estimation
DMI	Dynamic Model Identification Method
DR	Data Reconciliation
EVM	Errors in Variables Measured
FDAM	Finite Difference Approximation Method
GED	Gross Error Detection
GLR	Generalized Likelihood Ratio
ISOPE	Integrated System Optimisation and Parameter Estimation
LMS	Least Mean Squares
M-file	Matlab file
MADALINE	Multiple ADALINE
MIMO	Multi Input Multi Output
NLP	Nonlinear Programming
ODE	Ordinary Differential Equation
PRBS	Pseudo Random Binary Sequence

QP	Quadratic Programming
SDR	Steady-state Data Reconciliation
SISO	Single Input Single Output
SQP	Sequential Quadratic Programming
WLS	Weighted Least-Squares

CHAPTER 1

INTRODUCTION

On-line Optimisation techniques have been in existence for quite some time and yet few industrial applications have been implemented to date. This may stem from the general attitude in many manufacturing environments, where advanced technologies not fully understood are rejected. In addition there are tendencies to approach control problems from the traditional side, especially if the solution works "well enough".

1.1 OPTIMISATION

Optimisation, in the context of this work, may be thought of as the science of determining the 'best' solutions to certain mathematically defined problems, which are often models of physical reality. It involves the study of optimality criteria for problems, the determination of algorithmic methods of solution, the study of the structure of such methods, and computer experimentation with methods both under trial conditions and on real life problems (Fletcher, 1980).

The concept of optimisation is now well rooted as a principle underlying the analysis of many complex decision or allocation problems. It offers a certain degree of philosophical elegance that is hard to dispute, and it often offers an indispensable degree of operational simplicity.

In a mathematical cense, Optimisation may be concerned with finding the minimum (or maximum) of an objective function, where there may exist

restrictions or constraints as to what are permissible values of the independent variables.

Generally, it not possible to fully represent all the complexities of variable interactions, constraints, and appropriate objectives when faced with a complex decision problem, a particular optimisation formulation should be regarded only as an approximation like all quantitative techniques of analysis.

1.2 ON-LINE OPTIMISATION

On-line optimisation is an approach for trying to maintain a plant at its optimum operating conditions by determining the required set-points of the plant. In the majority of cases, the set-points will be made available to the plant's Distributed Control System (DCS), although they could be used by a stand-alone computer system. In most industrial processes, the optimal operating point is continually shifting in response to changing market demands for products, fluctuating costs of raw materials, products and utilities, and changing equipment efficiencies and capacities. In addition, ambient conditions, variations in feed quality and availability, and changes in equipment configuration are additional constraints that can alter the location of the optimal operation point. The time frame over which these various changes can occur ranges from minutes to months. The competitive economic environment requires timely response to these changing factors. This means that the optimisation must be carried out on-line to have the plant operate continually under the best conditions.

On-line optimisation takes advantage of the fact that plants generally operate at steady-state and have transient periods that are relatively short compared to steady-state operations. Therefore, in on-line optimisation, steady-state models are usually able to be used to describe these plants and their behaviour. The basic methodology of on-line optimisation adopted in this thesis is to automatically detect steady-state from the data samples collected from the process itself, reconcile them to remove any random and/ or gross errors, to update parameters in the plant model in order to obtain plant-model matching. Then the current plant

and its model are used to conduct optimisation and to generate a set of optimum set-points. This procedure is able to be run continuously to cope with the possibility of internal conditions (plant parameters and plant configuration) and/or external conditions (economic parameters) changing.

Besides determining the optimum operating condition of the process from the solution of the on-line optimisation problem, a number of other benefits are also apparent. The detail operation information generated from on-line optimisation provides a better understanding of the processes; this can be used to de-bottleneck the process and to improve operating difficulties. Also, abnormal measurement information obtained from gross error detection can help instrument and process engineers to trouble shoot the plant instrument errors. Parameter estimation is very useful for process engineers to evaluate the equipment conditions and to identify the decreasing efficiencies and problem sources. Furthermore, the detailed process simulation from on-line optimisation can be used for process monitoring and serves as a training tool for new operators to obtain first hand operating experience.

There are a number of areas which are central to the work and these are briefly introduced in Sections 1.2.1-1.2.5

1.2.1. Automatic Detection of Steady-State

Process owners analyse processes when they are at steady-state, for this reason, and for the reason that static data reconciliation and process optimisation are steady-state procedures, it is important that the process has to be at steady-state before applying the data reconciliation and optimisation procedures. Identification of the steady-state can prove to be difficult because process variables may be noisy and measurements do not settle. So, steady-state identification requires statistical tests to compensate for the noisy data.

1.2.2. Data Reconciliation

Measured process data inherently contains inaccurate information. This is due to the fact that measurements are obtained using imperfect techniques. Using this inaccurate information to estimate process variables and control the process, results in the state of the system to be misrepresented and the control performance to be poor, leading to sub-optimum and even unsafe process operation. The objective of data reconciliation is to correct measured data variables so they obey natural laws, such as energy and mass balances. Unfortunately, in the presence of biases, all the adjustments can be greatly affected by these types of gross error, and would in general not be reliable indicators of the state of the process.

1.2.3. Gross Error Detection

Raw process data is subject to two types of errors: random errors and gross errors. Random errors are dealt with using data reconciliation techniques, while gross errors need a different type of techniques, namely, gross error detection techniques. Ideally, the aim of a gross error detection technique is to:

1 Detect the existence of the gross error
2 Identify its location
3 Identify its type
4 Determine its size

After the gross errors are identified, two responses are possible and/or desired (Bagajewicz, 2003):

1 Eliminate the measurement with the bias, or
2 Correct the model such as the case of a leak and run the reconciliation again.

1.2.4. Parameter Estimation

Mismatch between models and plants can be due to a number of factors such as: uncertain parameters, unknown state variables, unmeasured disturbances, error in the model structure, and measurement noise. Proper adaptation schemes, where the model parameters are updated on the basis of recent measurements, need to be incorporated into the model-based optimisation control approach to minimise the plant-model mismatch. There exist several approaches to cope with this problem, where all are adaptive in nature but differ in their adaptation schemes.

1.2.5. Process Optimisation

Although there are many different available optimisation techniques, they can be classified into two general categories: direct search and model-based optimisation methods (Garcia and Morari, 1981).

As an on-line optimisation procedure, the Integrated System Optimisation and Parameter Estimation (ISOPE) algorithm (or modified two step in some literature) developed by Roberts in 1979, has some special features which can either be considered as direct or indirect. It is based on a number of features including derivatives calculation, originally estimated by using real process measurements, to update a model used in the model-based optimisation, thus reaching the real optimum of the process in spite of plant-model mismatch. Estimation of the derivatives by means of measurement, which increases geometrically with problem dimensionality, is a major problem of the ISOPE technique.

1.3 OBJECTIVES OF THE THESIS

The main objective of this research is to contribute to the improvement of the current tools in the field of on-line process optimisation. This includes the construction of plant models, the development, evaluation and comparison of

algorithms for process derivatives estimation, conducting and implementing steady-state detection, data reconciliation, gross error detection, parameter estimation and optimisation.

Any improvement in this area should help give more understanding of the way on-line optimisation has to be implemented, and hence lead to more benefits of on-line optimisation.

1.4 THESIS SCOPE

This thesis is concerned with certain on-line optimisation structures and the general ISOPE algorithm which integrates an optimisation scheme together with parameter estimation. Examples of situations using this structure are presented within this thesis, and these examples should help to indicate how practical problems can be treated and structured in this form. The thesis is also concerned with the analysis and comparison of algorithms and techniques for solving both general on-line optimisation problems and some related sub-problems. Problems of steady-state detection, data reconciliation, gross error detection and parameter estimation are also discussed and treated.

1.5 THESIS OUTLINE

This thesis is structured as follows:

Chapter 2 introduces the well known Integrated System Optimisation and Parameter Estimation (ISOPE) algorithm developed by Roberts (1979). The method was developed to overcome the problems of measurements and noise in the direct optimisation approach, and plant-model differences in the indirect one. A brief history is given together with the advantages and disadvantages that the method presents. One major drawback that poses a practical limitation and which will be the basis of some research in the following chapters is the need for real process output derivatives with respect to the set-points to be computed at each iteration of the algorithm.

Chapter 3 presents simulation case study systems. Two systems which will be used throughout the thesis for simulation purposes in order to assess and compare the performance and effectiveness of some of the techniques developed and presented in this thesis.

In chapter 4, a comparison study between some of the established methods for real process output derivatives with respect to set-points when used within the ISOPE algorithm and a new method based on a nonlinear dynamic model is made. These methods try to overcome the limitation caused to the ISOPE algorithm, by the need to perturb the system to obtain these derivatives. The methods are implemented under simulation on one of the two case study systems presented in chapter 3, which is the two Continuous Stirred Tank Reactors (CSTR) system. Results of the simulations are then presented and compared

Chapter 5 presents a method for estimating real process derivatives with respect to set-points. This method is based on Artificial Neural Networks (ANN). At first, a brief history is given on ANN's, together with the main philosophy behind the creation of ANN's. The neural network scheme is then presented and tested under simulation on the two systems presented in chapter 3. Results are compared with those obtained using a method described in chapter 4.

Chapter 6 introduces the area of data reconciliation and gross error detection for the use to correct data measurements by removing both random and gross errors from the data set. After a review of previous work, data classification and a description of the problems that both random and gross errors present on on-line optimisation and a full description of data reconciliation and gross error detection techniques is given. The performance of these techniques is demonstrated in a simulation case study. The case study uses the CSTR system described in chapter 3.

Chapter 7 incorporates the techniques implemented in chapter 7 for the static data reconciliation and gross error detection for the elimination of random noise and biases within the ISOPE algorithm. Again, simulations are carried out using the CSTR system. Results of these simulations are discussed and compared to the case when data reconciliation or gross error detection is not implemented.

Chapter 8 presents the on-line optimisation methodology as adopted in our work, and gives a brief description on how each step of the methodology is carried out. Methods for steady-state detection, data reconciliation, gross error detection, parameter estimation and process optimisation are reviewed. Difficulties and drawbacks of each method are discussed and compared to other methods in the literature. Simulation studies were conducted to test some of the key methods on the two Continuous Stirred Tank Reactors (CSTR) system. Finally, the whole methodology was implemented under simulation on the CSTR system. The implemented methodology procedure includes a steady-state detection module connected to a gross error detection module, which in itself connects to a static data reconciliation module. This latter is directly linked to the ISOPE algorithm module for optimisation of the two CSTR system.

The thesis concludes with a number of suggestions for further research related to the work carried out in this thesis.

1.6 SUMMARY

An introduction to the broad area of optimisation was given in this chapter. Specifically, on-line optimisation, which here is considered to be a multi-step procedure consisting of steady-state detection, data reconciliation, gross error detection, parameter estimation and the actual optimisation procedure. One particular method for system optimisation and parameter estimation (ISOPE) was presented. The scope and a short outline of the thesis were also given.

In the next chapter, the ISOPE algorithm is presented in more detail.

CHAPTER 2

THE ISOPE ALGORITHM

2.1 INTRODUCTION

The On-line optimisation problem, which consists of the determination of controls, or set-points, of a system controller, can be divided into two major categories: the direct and indirect approaches. The direct approach uses measurements taken directly from the real physical system and applies one of the basic optimisation techniques to optimise the process performance objective function. However, in practice this can give rise to some difficulties such as having to contend with measurement noise and having to allow the process to settle sufficiently before measurements are taken.

In the indirect or model-based approach, the optimisation is performed on a mathematical model of the plant instead of the real system itself. When found, the results are then applied to the real system. The use of model-based approaches has several advantages. The measurements contaminated by noise and other process disturbances are largely avoided. Also, there may not be a need to allow the system to settle before taking measurements or to have available all measurements of process variables which appear in the performance index (Ellis et al., 1988). Again, this is unlikely to produce the process optimum, as it is inevitable that model-reality differences exist at least to some extent, in terms of structure and parameter.

To overcome the problems of measurements and noise in the direct approach, and model-reality differences in the indirect one, the ISOPE (Integrated System Optimisation and Parameter Estimation) algorithm was introduced (Roberts, 1979). Possessing features from both approaches, the key feature of the algorithm

is to replace the model-based optimisation problem, after an analysis of first-order optimality conditions (Appendix A) by an equivalent problem which is ultimately decomposed into a parameter estimation problem and a modified model-based optimisation problem (Roberts et al., 1988). In this method, information gained from the real process is used to correct the errors occurring in the model. Hence, reaching the optimum of the real process in spite of model-reality differences.

All ISOPE algorithms designed to date are derived from the basic and well-known *two-step* technique, which consists of two major steps. The first step solves, with the aid of process measurement, a simple model parameter estimation procedure. The updated model is then used in the optimisation problem. The second step obtains the process controls via an optimisation routine (Figure 2-1). The major drawback of the *two-step* method is that it assumes a complete match between the output derivatives with respect to set-points of the real system and its model. This is highly unlikely to happen in reality where the degree of non-linearity is very high and the environments are varying. This problem was addressed by the ISOPE (sometimes referred to as the *modified two-step*) method, by introducing a new modifier variable. This modifier takes into account differences between the real process and model-based output derivatives with respect to the set-points, which ensures satisfaction of the system optimality conditions.

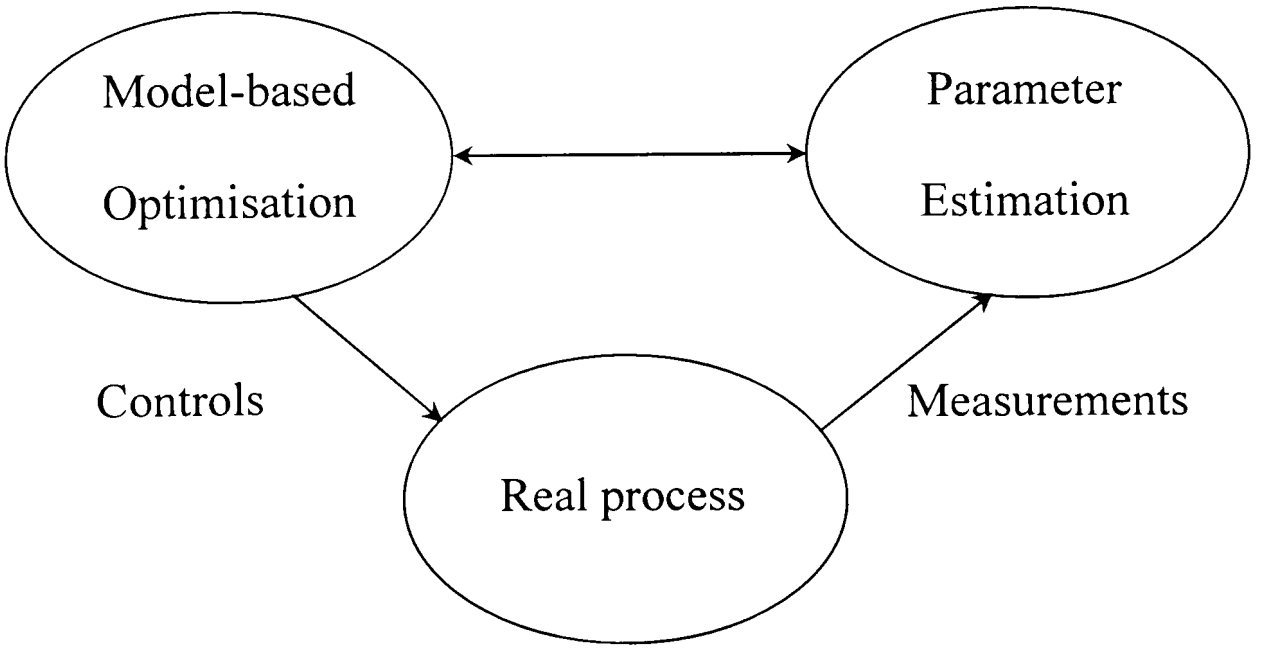

Figure (2-1): The *two-step* Method.

2.2 THE ISOPE ALGORITHM AND ITS DEVELOPMENT

The ISOPE algorithm as initially proposed by Roberts (1979), deals only with unconstrained problems. It was only until 1986 that Brdys et al. (1986) and then later Lin et al. (1988) extended it to include problems with output independent and output dependent inequality constraints. Nevertheless, the algorithm was used successfully in a large variety of cases before that. Indeed, Ellis and Roberts (1981) used the algorithm for on-line optimisation of a chemical reactor. The results were promising, and opened the door for other researchers to investigate the method more deeply. The performances of the algorithm, particularly the stability and convergence properties as well as the effect of real process measurement errors were investigated by Roberts and Williams (1981). Also, a convergence analysis was conducted by Brdys and Roberts (1987). Ellis et al (1988) conducted a comparison study where three methods were applied to a fuel gas mixer process. The methods compared were the Conjugate direction method, a rationalised form of the ISOPE algorithm and an Approximate Linear Model ISOPE (ALMISOPE) method. It was concluded that in some specific cases, the ALMISOPE is more efficient than the other two methods. An algorithm with dual control effect for which the generated control signal satisfies the main control goal as well as providing sufficient information for future identification action was proposed by Brdys and Tatjewski (1994). Roberts (1992) introduced DISOPE, a dynamic extension of the ISOPE algorithm used for solving nonlinear discrete time optimal control problems. Data reconciliation techniques were also used within the ISOPE algorithm to improve static optimisation schemes where data was contaminated by noise and systematic bias (Abu-el-Zeet, 2000). And lately, a comparison study including the most popular techniques for estimating process derivatives needed by the ISOPE algorithm, was conducted by Mansour and Ellis (2003). In the study, it was shown that the optimum operating point is reached with all the different estimating techniques used, but with a difference in speed of convergence; the Dynamic Model Identification technique being superior. Further work was carried out on the ISOPE algorithm, including: Abdullah (1988) for the

Augmented ISOPE (AISOPE) and Becerra et al (1998) in the area of predictive control. A review of the ISOPE algorithm can be seen in Roberts (1995).

2.3 FORMULATION OF THE PROBLEM

Consider the general steady-state optimisation problem of finding the optimum set-points of a system, which the behaviour obeys to the following relationships:

$$y^* = H^*(v) \tag{2.1}$$

$$g(y^*) \leq 0 \tag{2.2}$$

$$v_{\min} \leq v \leq v_{\max} \tag{2.3}$$

where y^* is an n_y vector of measured outputs, v is an n_u vector of manipulated variables, H^* represents the real process input-output mapping and g is a mapping of output dependent inequality constraints.

The performance of the system is measured with the objective function Q, which is assumed to be continuous, and differentiable.

The system optimisation problem is then considered to be:

$$Min \;\; Q(v, y^*) \tag{2.4}$$

Subject to:

$$y^* = H^*(v) \tag{2.5}$$

$$g(y^*) \leq 0 \tag{2.6}$$

$$v_{\min} \leq v \leq v_{\max} \tag{2.7}$$

In general, the above system optimisation problem is converted into a model-based optimisation problem where the following model of the real system is used:

$$y = H(u, \alpha) \tag{2.8}$$

where y is a vector of model outputs, u is a vector of decision variables; H is the model used to approximate the real process mapping and α is a vector of free parameters.

After analysis of the 1st order necessary optimality conditions (Appendix A), the problem (2.4) to (2.7) becomes:

$$\min Q(u, y) \tag{2.9}$$

subject to:

$$y = H(u, \alpha) \tag{2.10}$$

$$H^*(v) = H(u, \alpha) \tag{2.11}$$

$$v = u \tag{2.12}$$

$$g(y) \leq 0 \tag{2.13}$$

$$u_{\min} \leq u \leq u_{\max} \tag{2.14}$$

The free parameters α are chosen so that the model and real process outputs match at the current operating point, the model is then said to be point parametric (Ellis et al., 1988).

The above equations, after applying the necessary optimality conditions, yield the following model-based procedure:

$$\min_{u,w} \; Q(H(u,\alpha),u) - \lambda u + \tfrac{1}{2}\rho w^T w + \tfrac{1}{2} r \,\|\, u - v \,\|^2 \qquad (2.15)$$

subject to:

$$g(H^*(v)) + M(u - v) + w \le 0 \qquad (2.16)$$

$$\bar{u}_{\min} \le u \le \bar{u}_{\max} \qquad (2.17)$$

where

$$\bar{u}_{\min} = \max(u_{\min}, v - \delta) \qquad (2.18)$$

$$\bar{u}_{\max} = \min(u_{\max}, v + \delta) \qquad (2.19)$$

δ_j is the maximum allowed value of $\left|u_j - v_j\right|$, $j = 1, \cdots, n_u$, M is given by:

$$M = \left[\frac{\partial g(y)}{\partial y}\right]_{y=y^*} \times \left[\frac{\partial H^*(\mu)}{\partial \mu}\right]_{\mu=v} \qquad (2.20)$$

λ is computed from:

$$\lambda = \left[\left[\frac{\partial Q(H(\mu,\alpha),\mu)}{\partial \mu}\right]^T_{\mu=v} - \left[\frac{\partial H^*(\mu)}{\partial \mu}\right]^T_{\mu=v}\right] \times \left[\frac{\partial Q(y,v)}{\partial y}\right]^T_{y=y^*} \qquad (2.21)$$

and α can be obtained from:

$$y^* - H(v,\alpha) = 0 \qquad (2.22)$$

Equation (2.16) is equivalent to (2.13), and λ is a Lagrange multiplier, usually referred to as a modifier.

w is a set of relaxation variables and ρ is a penalty factor. The term $\frac{1}{2} r \,\|\, u - v \,\|^2$ is used only for highly non-convex objective functions and is seen to improve the convergence of the algorithm ($r \ge 0$) (Becerra and Roberts, 2000).

The above problem is then treated as a general nonlinear programming problem. When found, the solution of the problem (Roberts, 1979) is then treated within a relaxation scheme to give in an iterative manner the next control (see procedure in section 2.5) as follows:

$$v_{k+1} = v_k + K(u_{k+1} - v_k) \tag{2.23}$$

Where $K \in [0,1)$ is a relaxation gain matrix, and governs the actual changes made to the real process inputs from one iteration to another. Its purpose is to ensure that excessive alterations are not made.

The basic scheme of the algorithm can be seen in Figure (2-2).

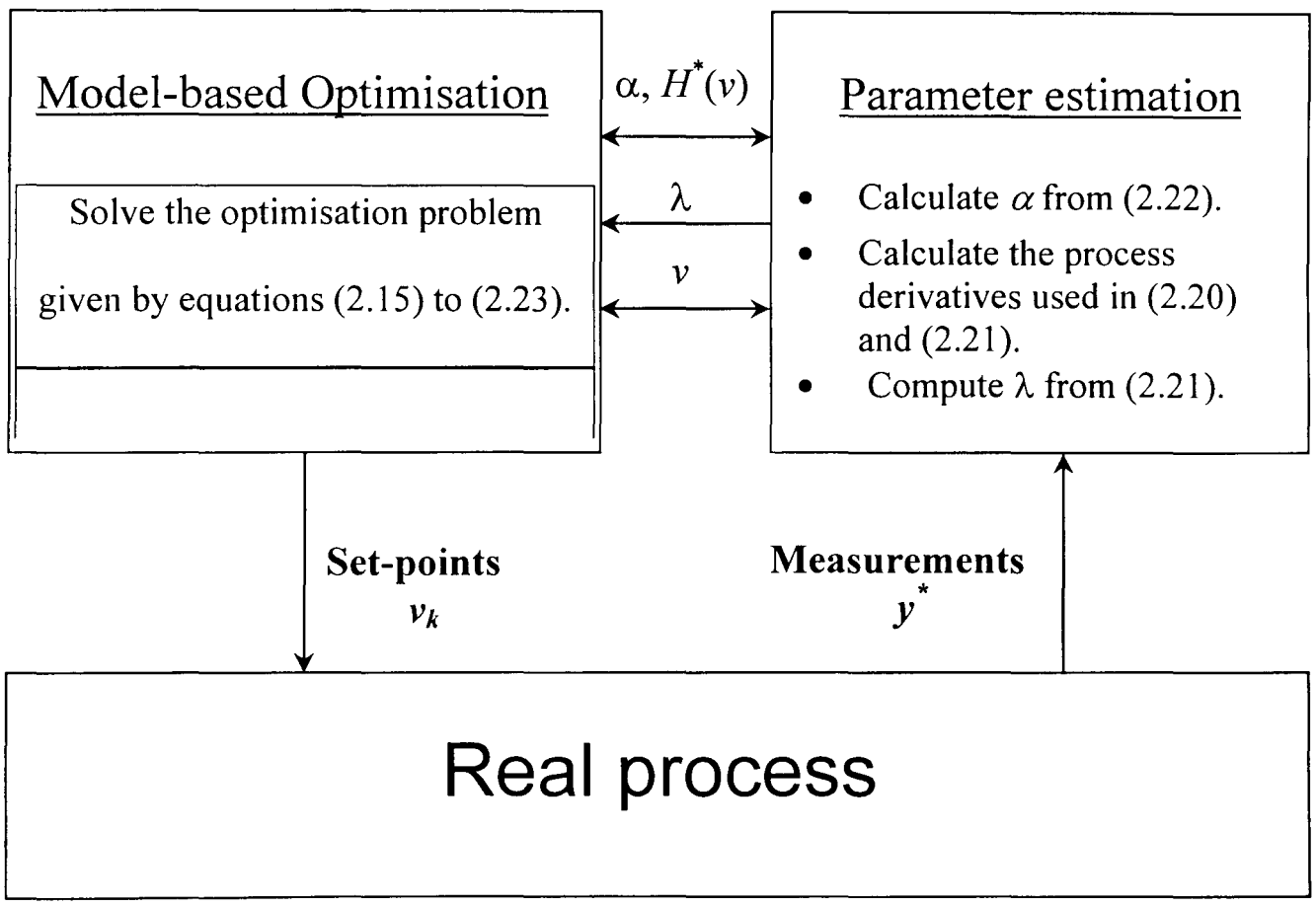

Figure (2-2): The ISOPE algorithm.

From the previous relations (2.20) and (2.21), it can be seen that real process derivatives are needed in order to compute the modifier λ. Various techniques

exist and have been developed and applied to date to estimate these derivatives. Finite Differences Technique using perturbation, based on measurements, was originally suggested with the algorithm by Roberts (1979). Although simple to apply, the technique proved to be inefficient in the case of large, slow and randomly noisy processes. The dynamic model identification method introduced by Zhang and Roberts (1988). The major advantage the technique brought was the reduction of the amount of time taken to estimate the derivatives. However, it encountered some difficulties such as: the huge amount of data needed and the poor inaccurate model it gives at the beginning of the identification.

Broydon's approximation method, based on the well-known Broydon's family of formulas, mainly oriented to the approximation of derivatives was also used and implemented. These techniques are studied in detail in Chapter 4, where an assessment of their efficiency is made through a simulation of a Continued Stirred two Tank reactor (CSTR) system. Other methods have been developed with the aim of totally eliminating the need for the derivative information from within the ISOPE algorithm. However, these techniques have not proven to be highly successful, and therefore have not been included in this work.

2.4 SPECIAL CASE: QUADRATIC OBJECTIVE WITH LINEAR MODEL AND CONSTRAINTS

Although the structure of most dynamical systems is of non-linear form, it is often possible to obtain a good linear approximation to the behaviour of the system around a suitable operating point. Thus, many systems can be described by the following linear representation:

$$y = H(u, \alpha) = Au + \alpha \tag{2.24}$$

$$g(y) = Gy - h \leq 0 \tag{2.25}$$

where A is an $(n_y \times n_u)$ matrix. And g is the output inequality constraints.

In the special case where the objective function Q is quadratic of the form:

$$Q(u,y) = \tfrac{1}{2}(y - y_d)^T C(y - y_d) + \tfrac{1}{2}(u - u_d)^T D(u - u_d) + e^T y + f^T u \qquad (2.26)$$

where C and D are symmetric positive definite weighting matrices, e and f weighting vectors, y_d and u_d are the desired steady-state output and set-point vectors respectively.

The physical interpretation of such a performance index, is that we desire to maintain the output vector y close to a target vector value y_d without using excessive control effort, by keeping u near a given vector value u_d. The weighting matrices C and D (Singh and Titli, 1978) enable us to define the relative importance of keeping the output near the desired target, the expenditure of control effort and the need to ensure that at the final time, the output vector will be very close to the desired target (convergence).

The reasons behind using linear models with quadratic performance indices is to be able to use Quadratic Programming to solve the general non-linear problem of finding the optimum point of a given non-linear system (which can prove to be very difficult and time consuming) by converting it into a simplified quadratic problem. One of the principal properties of quadratic programming problems is that the constraints are linear, so they are convex, and in the case of a convex objective function (which can happen if the weighting matrix is positive definite or positive semidefinite), there is a unique solution to the problem which is the global optimum. Quadratic programming arises in many applications and it forms a basis of some specific algorithms and techniques. As it is usually solved using calculus, many problems which are highly non-linear are converted into quadratic formulation. A quadratic program is greatly simplified, and can be solved in closed form if it contains equality constraints only.

In this case the parameter α will be calculated by:

$$\alpha = y^{*} - Au \tag{2.27}$$

And λ can be found by combining equation (2.21) with (2.24) and (2.26) to be:

$$\lambda = D(v - u_d) + f - A^T \times C(y - y_d) - A^T e \tag{2.28}$$

In this case the modifier λ is found using the above formulation with the help of process information such as measurement (i.e.: matrix A obtained using measurements) and optimiser parameters such as D, C, and e.

The optimisation problem therefore is reduced to the following quadratic programming problem:

$$\min_x \tfrac{1}{2} x^T S x + q^T x \tag{2.29}$$

Subject to:

$$\overline{G}x \le \overline{h} \tag{2.30}$$

$$x_{\min} \le x \le x_{\max} \tag{2.31}$$

where:

$$x = \begin{bmatrix} u \\ w \end{bmatrix} \tag{2.32}$$

$$S = \left[\begin{array}{cc} A^T C A + D + rI_n & | \, 0_{n_y \times l} \\ \hline 0_{l \times n_y} & | \, \rho I_l \end{array}\right] \tag{2.33}$$

$$q = \left[\begin{array}{c} A^T C(\alpha - y_d) - Du_d + f - \lambda + A^T e - rv \\ \hline 0_{l \times 1} \end{array}\right] \tag{2.34}$$

$$\overline{G} = [GA \mid I_l] \tag{2.35}$$

$$\overline{h} = \left[b - Gy^* - GAv\right] \tag{2.36}$$

The above optimisation problem is solved using quadratic programming.

2.5 A SIMPLIFIED VERSION OF THE ISOPE ALGORITHM

A practical version of the ISOPE algorithm presented in this chapter and developed by Becerra and Roberts (2000) is given below. It is worthwhile noting that the convergence of the algorithm for which a summary is given in section 2.6, depends upon several factors. The accuracy of the derivative estimation is one of these factors. The procedure is (Becerra and Roberts, 2000):

Data: $C, D, e, f, y_d, u_d, G, h, r, \rho, K, v_k$ and means for measuring y_k^* and computing A_k. Put $k = 0$ and go to step1.

---

1. Apply the current input v_k to the plant, wait for a steady-state to be reached and measure the process output y_k^*.

2. Update the gain matrix A_k by using one of the available estimation methods presented in chapter 4.

3. Compute α_k using (2.27) and λ_k using (2.28).

4. Solve the optimisation problem given by equations (2.29) to (2.36) using quadratic programming to obtain the next input candidate u_{k+1}.

5. Compute the next process input by using (2.23).

6. Set $k = k+1$, check for convergence and go to step 1.

These steps are repeated until convergence is reached. Convergence occurs when no further improvement is observed. In other words, when the new control is no longer a better candidate than the previous one. Theoretically, convergence is checked in step 6 by testing the equality $v_{k+1} = v_k$. Practically, the previous equality $v_{k+1} = v_k$ is replaced by the following inequality: $\| v_{k+1} - v_k \| < \varepsilon$.
Where $\varepsilon > 0$ is a desired accuracy threshold.

2.6 CONVERGENCE PROPERTIES

The convergence and optimality properties of the ISOPE algorithm for on-line determination of the optimum steady-state operating point of a given process was investigated in detail by Brdys and Roberts (1987). The conclusion was that, under mild assumptions, a suitable gain matrix K exists such that every point generated by the iterative procedure (equation (2.23)) is feasible. In fact, in order to assure feasibility during iterations for a general constrained case, the gain matrix K must be of the form:

$$K = kI$$

Where k is a positive scalar, and I is the identity matrix.
This involves all individual gains k_i to have the same identical numerical values, unlike in the unconstrained case, where the gain matrix K is allowed to have different individual diagonal elements. However, the scalar parameter k in the constrained case, is allowed to change from one iteration to another

Under such conditions, the process performance index is improved at each iteration and each cluster point of the sequence generated by the algorithm satisfies first-order necessary conditions for optimality. Furthermore, every optimal point belongs to the solution set of the algorithm.

Although, in order to guarantee convergence, the composition of the performance index and the process mathematical model should be uniformly convex functional on the set of admissible controls, the real process input-output mapping may be non-linear such that its composition with the performance index is not required to be convex. Hence, the algorithm is applicable to a broad class of real problems (Brdys and Roberts, 1987).

Also, it was found (Kambhampati, 1988) that:

1. The derivative differences given by $\left[\left[\frac{\partial H^*(\mu)}{\partial \mu}\right]^T_{\mu=v} - \left[\frac{\partial H(\mu)}{\partial \mu}\right]^T_{\mu=v}\right]$ constitute the model-reality differences.
2. The modifier λ can be interpreted in either of the following two ways:
   i. A parameter which quantifies the violations of the sufficiency conditions by the models

   or

   ii. A compensator which permits differences in the model based performance index and the system based performance index.

These conclusions helped us understand the model-reality differences and what necessary characteristic the model has to fulfil in order that the performance of the algorithm is efficient. And hence, the smaller the model-reality differences are, the more efficient is the performance of the algorithm.

2.7 SUMMARY

In this chapter, the ISOPE algorithm has been presented and reviewed. An improved version of the algorithm developed by Becerra and Roberts (2000) has also been outlined. The major inconvenience the method possesses which is the need for derivative information to be estimated at each operating point was also addressed. In the algorithm, a special case for quadratic objective and linear model was treated. Finally, the convergence and optimality properties have been outlined.

CHAPTER 3

CASE STUDY SYSTEMS

3.1 INTRODUCTION

In this chapter two examples of systems are introduced. These are to be used in simulations in order to assess and compare the performance and effectiveness of all the techniques presented in this thesis. The first example is a simple SISO (Single Input Single Output) nonlinear discrete time system used as an introduction to illustrate simple algorithmic design aspects. The second is a more realistic system widely used in different situations and which consists of a two Continuous Stirred Tank Reactors (CSTR's) connected in series.

The following subsection introduces the first example with its basic details and system equation. The next subsection gives a detailed description of the second system (CSTR), its functionality, equations and an explanation of all the related constraints and restrictions. The design and implementation issues of both systems are also outlined.

3.2 EXAMPLE 1:

We consider the Single Input Single Output (SISO) non-linear plant represented by the following first order discrete-time input/ output representation:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \qquad (3.1)$$

where $y(k)$ is the plant output at time kT (T is the sampling time) and $u(k)$ is the input.

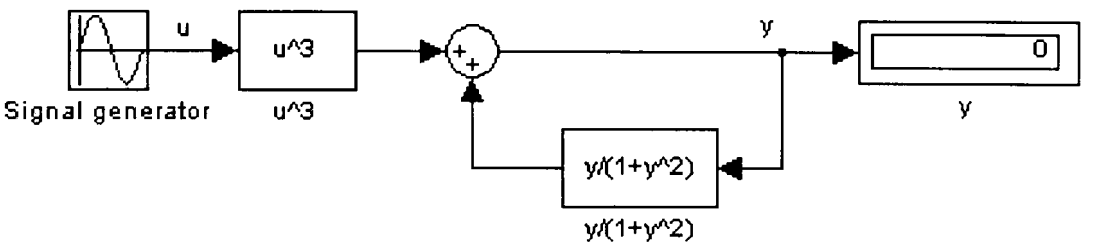

Figure (3-1): SIMULINK implementation example of the SISO non-linear plant.

This example, presented in Narendra and Parthasarathy (1990) and Kambhampati et al. (2000), is an introductory example only. It is used to illustrate simple algorithmic design and applicability aspects. In Chapter 5 it is used under simulation to assess the effectiveness of the Neural Networks model structure used in identification, after training the model with real input/ output data candidates (taken from the real system described above).

3.3 THE TWO CONTINUOUS STIRRED TANK REACTORS (CSTR'S)

This example presented by Garcia and Morari (1981) used by Jang et al. (1987) and later treated by Becerra and Roberts (1995), consists of two Continuous Stirred Tank Reactors (CSTR's) connected in cascade in which an exothermic autocatalytic reaction takes place (Figure 3-2). The components interact in both

directions due to the recycle of 50% fraction of the product stream into the first reactor. Regulatory controllers are used to control the temperature in both reactors.

The reaction is described by the basic reaction equation:

$$A + B \underset{k_-}{\overset{k_+}{\Leftrightarrow}} 2B \tag{3.2}$$

where A and B are two chemical components.

The real process is represented by the following relations:

$$\frac{dC_{a1}}{dt} = \frac{0.5}{\tau_1}(C_{a0} + C_{a2}) - \frac{C_{a1}}{\tau_1} - (k_{1+}C_{a1}C_{b1} - k_{1-}C_{b1}^2) \tag{3.3}$$

$$\frac{dC_{b1}}{dt} = \frac{0.5}{\tau_1}C_{b2} - \frac{C_{b1}}{\tau_1} + (k_{1+}C_{a1}C_{b1} - k_{1-}C_{b1}^2) \tag{3.4}$$

$$\frac{dC_{a2}}{dt} = \frac{C_{a1}}{\tau_2} - \frac{C_{a2}}{\tau_2} - (k_{2+}C_{a2}C_{b2} - k_{2-}C_{b2}^2) \tag{3.5}$$

$$\frac{dC_{b2}}{dt} = \frac{C_{b1}}{\tau_2} - \frac{C_{b2}}{\tau_2} + (k_{2+}C_{a2}C_{b2} - k_{2-}C_{b2}^2) \tag{3.6}$$

Where C_{xi} is the concentration of component x in tank i, $\tau_1 = 30\,\text{min}$ is the mean residence time of reactor 1, $\tau_2 = 25\,\text{min}$ is the mean residence time of reactor 2, which result in an overall time constant of approximately 40 min (Garcia and Morari, 1981). $k_{i\pm} = A_{\pm}\exp(-E_{\pm}/RT_i)$ are the reaction rates, $E_+/R = 17786K$, $E_-/R = 23523K$, $A_+ = 9.73\times10^{22}\,m^3/kmols$, $A_- = 3.1\times10^{30}\,m^3/kmols$, $C_{a0} = 0.1$ is the feed concentration of component A, T_1 is the temperature in tank 1, T_2 is the temperature in tank 2.

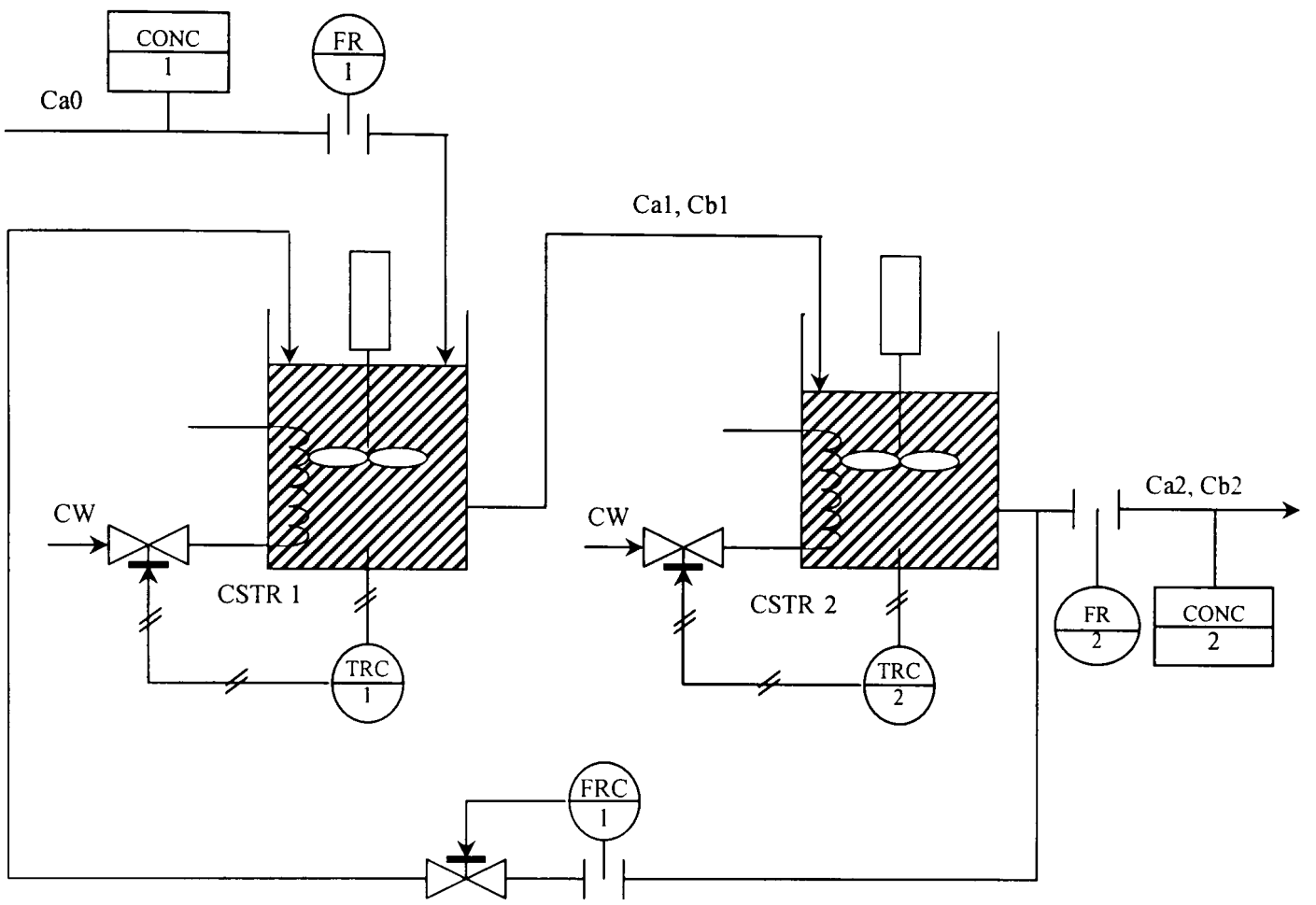

Figure (3-2): The two Continuous Stirred Tank Reactors system.

These equations, together with the regulatory control loops, the measurement transducers and the valve actuators provide the real process description. In our case, the dynamics associated with the regulatory controllers were neglected as well as the measurement transducers and actuators (which were originally modelled as first order lags), as the real system process is a very slow process and its dominant time constant is very large compared to those of the instrumentation. Therefore, the above equations represent the mapping H^* of the real system.

It has to be mentioned that when using the ISOPE algorithm (Chapter 2), an incorrect and simplified model is used as a mapping H to represent the system. This mapping is different from the one given above.

The two CSTR plant has 4 outputs which are the concentrations of the two components A and B in both tanks. Hence, the output vector can be written as:

$$y_* = (C_{a1}, C_{b1}, C_{a2}, C_{b2})^T \tag{3.7}$$

The manipulated variables which are the set points of the temperature controllers in both reactors are given by,

$$v = (T_1, T_2)^T \tag{3.8}$$

These are bounded between upper and lower levels: $300 \leq T_1 \leq 312\ K, 300 \leq T_2 \leq 312\ K$ and are assumed to be known noise free.

3.4 IMPLEMENTATION ISSUES

The implementation of the CSTR and the simple SISO systems presented in this chapter was performed using a MATLAB®/SIMULINK software platform.

MATLAB is a high-performance language for technical computing. It integrates computation, visualisation, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Mathematics and computation
- Algorithm development
- Modelling, simulation, and prototyping
- Data analysis, exploration, and visualisation
- Scientific and engineering graphics
- Application development, including Graphical User Interface (GUI) building.

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning.

The name MATLAB stands for *matrix laboratory*. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB features a family of application-specific solutions called *toolboxes*. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems.

SIMULINK, a companion program to MATLAB, is an interactive system for simulating both linear and nonlinear dynamic systems. It is a graphical mouse-driven program that allows a system to be modelled by drawing a block diagram on the screen and manipulating it dynamically. It can work with linear, non-linear, continuous-time, discrete-time, multivariable and multirate system (The MathWorks, 1996).

The implementation of both systems was achieved by creating a SIMULINK model architecture which is able to interact with the MATLAB environment via calling a subroutine containing the appropriate identification and optimisation algorithms stored in an M-file. The subroutine acquires the information data under the form of measurements from the SIMULINK model of the plant (figure 3-3). The subroutine is executed at every time sample given in the SIMULINK model parameters, which makes the whole procedure recursive. Major consideration and extra care have to be taken when choosing the simulation parameters. For example, we mention that the time step in a SIMULINK model is not the real time step which means if a measured variable is plotted against time; it would be the internal SIMULINK time not real time. Therefore the time, ODE and the other parameters are to be tuned first before simulation starts. These parameters are chosen following some specific criteria so that the whole system (SIMULINK model and optimisation routine) works in a perfect state.

It is worth noting that the simulation times which appear in the results in subsequent chapters relate to the real plant. The simulations would typically run at speeds of between 10 to 100 times faster depending on the computational load on the algorithm; and they are run for suitable time durations, giving time to the

appropriate system to settle down for a steady-state position and for the appropriate algorithms to perform their tasks. All the results are then stored in the workspace to be analysed and plotted.

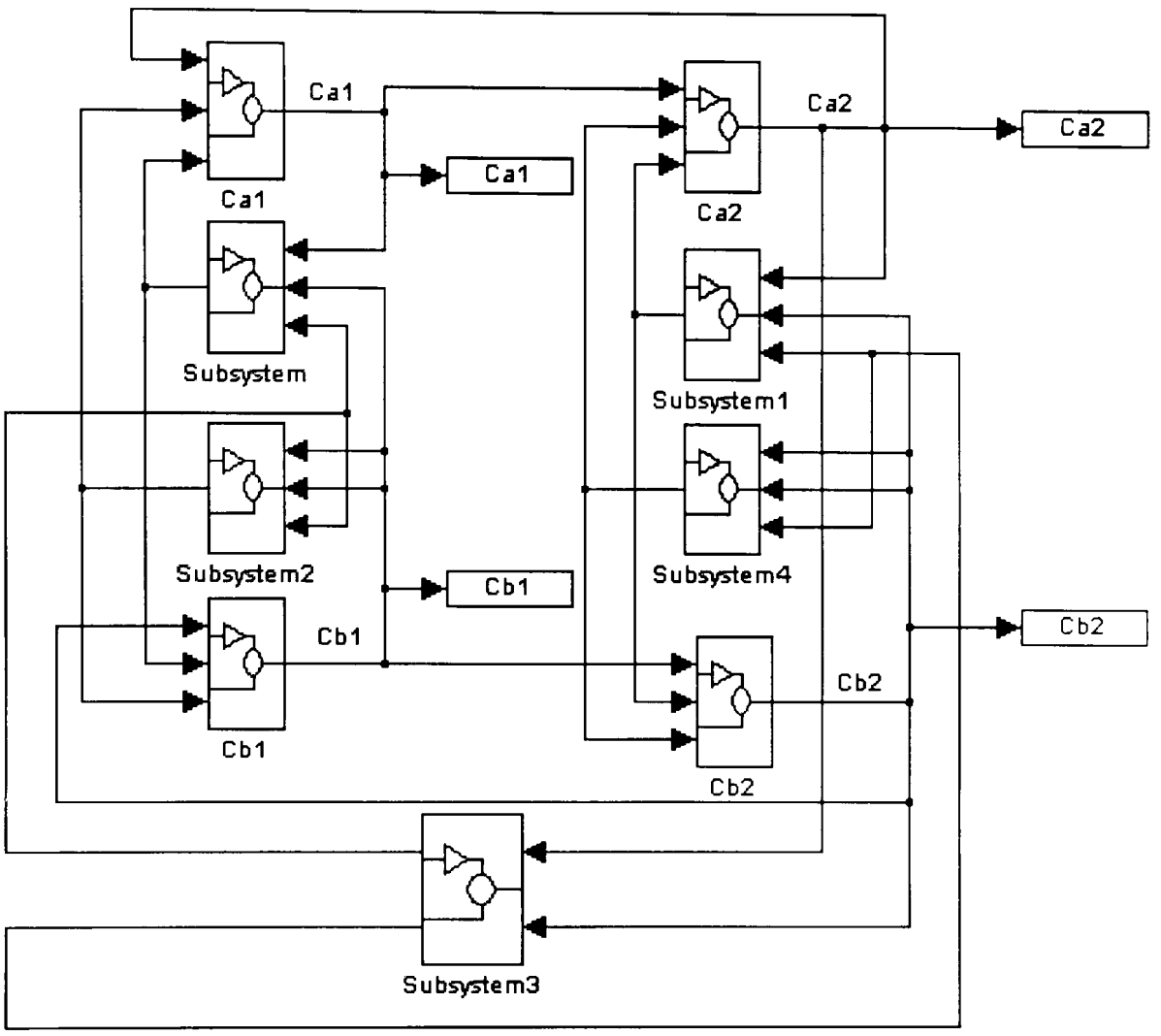

Figure (3-3): SIMULINK implementation example of the CSTR system.

3.5 SUMMARY

Two examples of systems, which are going to be used in case studies throughout this thesis, were presented in this chapter. The first example is a simple introductory system, which consists of a nonlinear discrete time plant. The second one is a two Continuous Stirred Tank Reactors (CSTR's) connected in series. The two plants equations and characteristics together with different implementation aspects are presented in a way to describe the functionality of both systems.

CHAPTER 4

TECHNIQUES FOR THE ESTIMATION OF THE DERIVATIVE INFORMATION

4.1 INTRODUCTION

The model-reality differences problem in the general on line optimisation problem is usually overcome by using adaptive models which can be updated regularly while seeking to reach the solution of the optimisation problem. The Integrated System Optimisation and Parameter Estimation (ISOPE) algorithm uses such models. The major drawback the method possesses is the need for derivative information to be estimated at each operating point. These derivatives are needed by the algorithm in order to satisfy necessary optimality conditions (Appendix A).

This chapter investigates methods and techniques developed for the purpose of estimating the process derivatives. Methods of Finite Difference Approximation, Dual Control Optimisation, Broydon's method and a Dynamic Identification Method, with a Linear and Non-linear models, are presented, implemented and tested, under simulation, on the cascade Continuous Stirred Tank Reactor (CSTR) system presented in Chapter 3.

4.2 FINITE DIFFERENCE APPROXIMATION METHOD (FDAM)

This method was the first employed for estimation of derivatives (Roberts, 1979) and use is made of process measurements. If the process is subject to noise, then the derivative estimates can suffer large errors giving problems in obtaining the correct final solution. Other difficulties which might arise, apart from those concerned with noise, are the obtaining of actual measurements and, for slow dynamic processes, having to wait for the process to settle sufficiently before steady-state measurements are taken.

In most practical situations, the process mapping is not given by a specific formula or is difficult to find; rather it is a combination of experimental and computational procedures. Thus, the output derivative matrix with respect to the set-points needed by the ISOPE algorithm is usually unavailable.

In one dimensional case, the derivative of a certain function $f(x)$ can be replaced by the secant line that goes through f at x_c and at some nearby point $x_c + h_c$ (Dennis and Robert, 1983). The most obvious formulation of that line slope is:

$$a_c = \frac{f(x_c + h_c) - f(x_c)}{h_c} \tag{4.1}$$

Therefore, the output derivative function with respect to the set-point of a given process can be replaced by the following estimation:

$$\frac{\partial f}{\partial x} = \frac{f(x+h) - f(x)}{h} \tag{4.2}$$

However, will the above formulation be a faithful approximation to the derivative function of f?

The answer comes from the fact that as h goes to zero, a_c converges to $f'(x)$. Therefore, h has to be chosen conveniently small so that the estimation (4.2) can be true, and a_c can be called *finite-difference approximation* to $f'(x)$.

In the multidimensional case, it is reasonable to use the same idea to approximate the $(i,j)^{th}$ component of the derivative matrix A by the following *forward difference* approximation

$$a_{ij} = \frac{f_i(x + he_j) - f_i(x)}{h} \tag{4.3}$$

where e_j denotes the j^{th} unit vector. The above formula is equivalent to approximating the j^{th} column of A by

$$A_j = \frac{F(x + he_j) - F(x)}{h} \tag{4.4}$$

where A is the derivative matrix of the multidimensional function F.

Again, the matrix A converges to the true derivative matrix only if h is chosen to be sufficiently small.

In practice, and in MIMO (Multi Input Multi Output) systems, the output derivative matrix with respect to the set-points is similarly given by:

$$D_k = \frac{\partial y}{\partial v} \approx \frac{y(v_k + \delta) - y(v_k)}{\delta} \tag{4.5}$$

where δ is a small perturbation signal applied to the system in order to estimate the derivative matrix and y and v are the output and manipulated (set-point) variables respectively. The perturbation signal δ is chosen to provide enough excitement needed by the system, and at the same time ensures greater accuracy of the derivative estimates. In practice, δ is usually left as an adjustable parameter.

Being the basic method used in the original ISOPE algorithm (Roberts, 1979), this technique can give sufficient accuracy of the derivatives in an acceptable time span, for the case of small and noise-free processes which have reasonably rapid dynamics. However, it has been shown to perform very inconveniently for large and slow processes because of the huge amount of time it takes for the estimation. Indeed, as demonstrated by Ellis et al. (1988) and Mansour and Ellis (2003), a large number of set-point changes are required for problems with large number of inputs and outputs (Roberts, 1995). Furthermore, the inaccuracy of the measurements for noise-contaminated processes might make the robustness of the algorithm against disturbances very poor.

For these reasons, alternatives had to be found in order to overcome these problems. A number of ISOPE techniques have been developed and applied in different situations since the algorithm was first proposed. Below, some of these techniques are listed. For a review of the different ISOPE techniques and their applications, see Roberts (1995).

4.3 METHOD FOR DUAL CONTROL OPTIMISATION

This was the first algorithm based on steady-state measurements that does not require additional set-point changes for the derivative approximation purpose (Brdy's and Tajewski, 1994). It generates a control signal in such a manner that it fulfils the main control goal, and produces an output signal which caries sufficient information for future identification purposes. Below is a brief description of the algorithm:

The algorithm assumes the existence of a collection of $n+1$ points v^i, v^{i-1},..., v^{i-n} such that all vectors

$$\Delta v^{ik} \stackrel{def}{=} v^i - v^{i-k} \tag{4.6}$$

are linearly independent, i.e.

$$\det(S^i \overset{def}{=} [\Delta v^{i1} \quad \Delta v^{i2} \cdots \quad \Delta v^{in}]^T) \neq 0. \tag{4.7}$$

Directional derivatives F_{*_j} of the j^{th} plant output $y \in \Re^{n_y}$ at a point v^i and in a direction $s^{ik} \overset{def}{=} v^i - v^{i-k}$, can be computed as:

$$DF_{*_j}(v^i; s^{ik}) = (s^{ik})^T \nabla F_{*_j}(v^i) \tag{4.8}$$

for each $k = 1, \ldots, n$, $j = 1, \ldots, m$, with m is the number of outputs. Therefore

$$S^i \nabla F_{*_j}(v^i) = \begin{bmatrix} DF_{*_j}(v^i; s^{i1}) \\ \vdots \\ DF_{*_j}(v^i; s^{in}) \end{bmatrix}, \tag{4.9}$$

If the points v^{i-j} are close enough to v^i then for every $j = 1, \ldots, m$,

$$\nabla F_{*_j}(v^i) \cong (S^i)^{-1} \begin{bmatrix} F_{*_j}(v^i) - F_{*_j}(v^{i-1}) \\ \vdots \\ F_{*_j}(v^i) - F_{*_j}(v^{i-n}) \end{bmatrix}. \tag{4.10}$$

This formulation assures generation of consecutive set-points v^i in such a way that the efficient estimation of the plant output derivatives using equation (4.10) can be applied. However, this estimation can not be applied successfully except if the matrix S^i is non-singular and sufficiently well conditioned. This can be fulfilled only if the consecutive set-points v^i are appropriately located in their space (Brdy's and Tatjewski, 1994).

In order to achieve this goal, a new inequality constraint has to be introduced to the modified model-based optimisation problem. This new added constraint is based on a function d connected with non-singularity of the matrix S^i. Brdy's and

Tatjewski (1994) proposed two formulations of the function d. The most practical of them is:

$$d(v^i, v^{i-1}, \cdots, v^{i-n}) = \frac{\sigma_{\min}(S^i)}{\sigma_{\max}(S^i)} \tag{4.11}$$

where $\sigma_{\min}(S^i)$ and $\sigma_{\max}(S^i)$ are the minimal and maximal singular values of S^i. This means that (4.11) is the reciprocal of the condition number of the matrix S^i (in 2-norm). This method is then implemented within the ISOPE algorithm in order to estimate the output derivative matrix with respect to the set-points of a given process. As shown in section 4.6, the performance and ability of this method are demonstrated, and also compared to some other techniques which will be discussed below.

For more details of the method and its practical implementation, see Brdy's and Tatjewski (1994).

4.4 Broydon's Method

One way to avoid estimating derivatives (since in practice the derivatives may not be conveniently available) is the so-called Broydon family of algorithms (Fletcher, 1980).

Proposed by C. Broydon, it is considered to be the most successful secant approximation to the Jacobian. Broydon's approximation or as it is usually referred to as Broydon's update, is used to solve systems of nonlinear equations. The key feature of the method is that it updates the matrix A_k at each iteration so that the next approximation A_{k+1} is given by the equation:

$$A_{k+1} = A_k + \frac{(y_k - A_k S_k) S_k^T}{S_k^T S_k} \tag{4.12}$$

where

$$y_k = f(x_{k+1}) - f(x) \tag{4.13}$$

$$S_k = x_{k+1} - x_k \tag{4.14}$$

Broydon's update, belonging as it does to the class of Newton methods, may need to be supplemented by some techniques to converge from a starting point. This starting point has to be chosen conveniently in order to ensure convergence. In practice, the initial approximation A_0 is computed by using finite differences in order to get a good start. This also makes the minimum-change characteristic of Broydon's update more appealing.

In practice, equation (4.12) is used to estimate the output derivative matrix with respect to the set-points of a given process. The derivative matrix is needed by the ISOPE algorithm in order to calculate the modifier λ (chapter 2).

Broydon's update gives:

$$BR_k = BR_{k-1} + \frac{[y_k - y_{k-1} - (BR_{k-1} * (u_k - u_{k-1}))] * (u_k - u_{k-1})^T}{(u_k - u_{k-1})^T * (u_k - u_{k-1})} \tag{4.15}$$

Where BR_k and BR_{k-1} are respectively the present and previous estimates of the output derivative matrix (also known as Broydon's matrix), y_k and y_{k-1} are the present and previous values of the measured output vector, while u_k and u_{k-1} are the present and previous values of the manipulated variables respectively. The *BR* matrix is updated periodically using present and previous measurements of the output and manipulated variables and needs to be initialised at the start up.

Some remarks have to be made here regarding some obstacles that the algorithm might encounter:

1. The first observation is that the algorithm needs an initial value of the matrix *BR* to start with. Usually and as mentioned above, the finite differences method (FDAM) is used to calculate it. However, in practice and as explained in section 4.2, the FDAM has some disadvantages when used on line. For instance, measurements might be contaminated by noise,

or the system might have slow dynamics leading to a very slow estimation process.

2. The second observation is that close to the optimum point, the current control u_k and the previous one u_{k-1} are very close to each other (i.e.: $u_k \approx u_{k-1}$), which may cause the approximation formula (4.15), to reach some prohibited values and therefore the algorithm to fail in convergence.

More details about Broydon's family of algorithms can be found in Fletcher (1980).

4.5 Dynamic Model Identification Method (DMI)

This method is based on the identification of a dynamic model that is used to approximate the real process locally at each working point for the purpose of estimating steady-state derivatives. It was first introduced into the field of optimisation by Bamberger and Isermann (1978), where it was shown to be an efficient tool for identification, especially in the case of slow processes.

The key feature of the method is to approximate the real process by a dynamic model during the transient using real process information (Figure 4-1). In this case the waiting time for the steady-state to be reached in order to estimate the derivatives is avoided; these derivatives are calculated directly from the steady-state model derived from the identified dynamic model. The structure of the dynamic model to be identified is pre-specified and is updated on-line (Forbes, 1994). In many cases a linear structure is assumed (Garcia and Morari, 1981; Becerra et al., 1998). However this is not always the case as general non-linear forms can also be used (Bamberger and Isermann, 1978; Mansour and Ellis, 2003).

In this work, two different structures of models are used: a linear representation with a non-iterative technique developed by Becerra et al. (1998) and a non-linear model based on a 2^{nd} order Hammerstein Model presented in Bamberger and

Isermann, (1978). The two techniques, based on the two different model representations, follow.

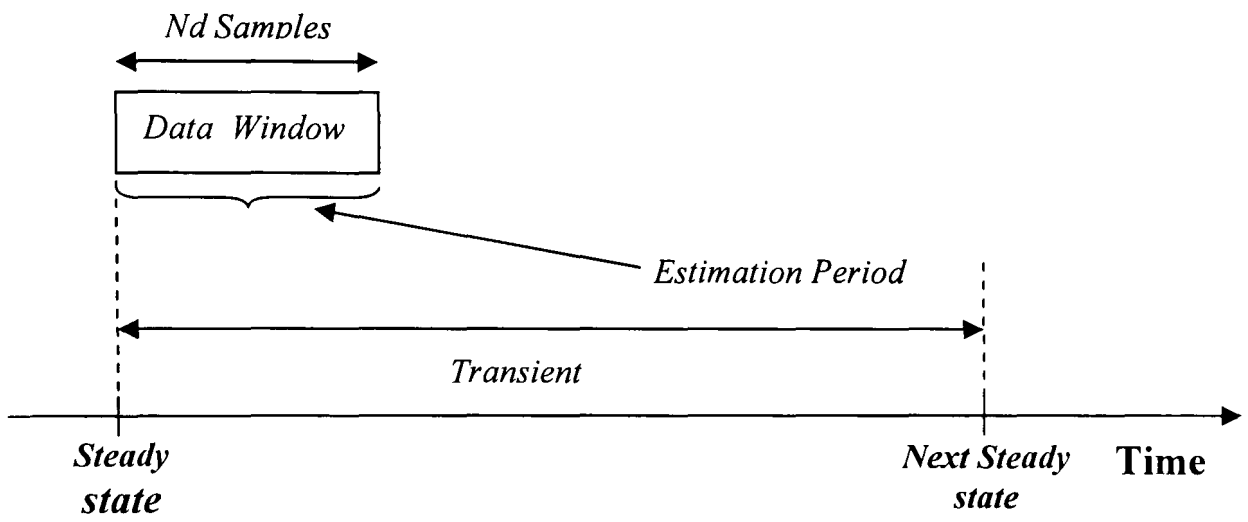

Figure (4-1): The DMI notion aspect.

4.5.1. DMI with Linear Model Representation

A multivariable ARMAX (Auto-Regressive Moving-Average with eXogeneous inputs) model is employed (Becerra et al., 1998) to estimate the linear dynamic model, based on the least squares method.

A general state-space model representation which has the following form is used:

$$\Delta x(k+1)=A\Delta x(k)+B\Delta u(k) \tag{4.16}$$

$$\Delta y(k)=C\Delta x(k) \tag{4.17}$$

where k is an integer index, $x \in \Re^{n_y}$ is a state vector, $u \in \Re^{n_u}$ is a set of independent inputs, $y \in \Re^{n_y}$ is a vector of measured outputs and A, B, C are matrices of the appropriate dimensions. This technique is based on the moving- horizon concept (Figure 4-2), but it exploits the displacement structure of the data window, so that its computational load is reduced. For models with multiple outputs and a large number of parameters it may provide a lower computational load than that of the standard recursive least squares algorithm, as described below. Moreover, using a

non-minimal realization, the state space basis is invariant even when the model matrices are updated periodically.

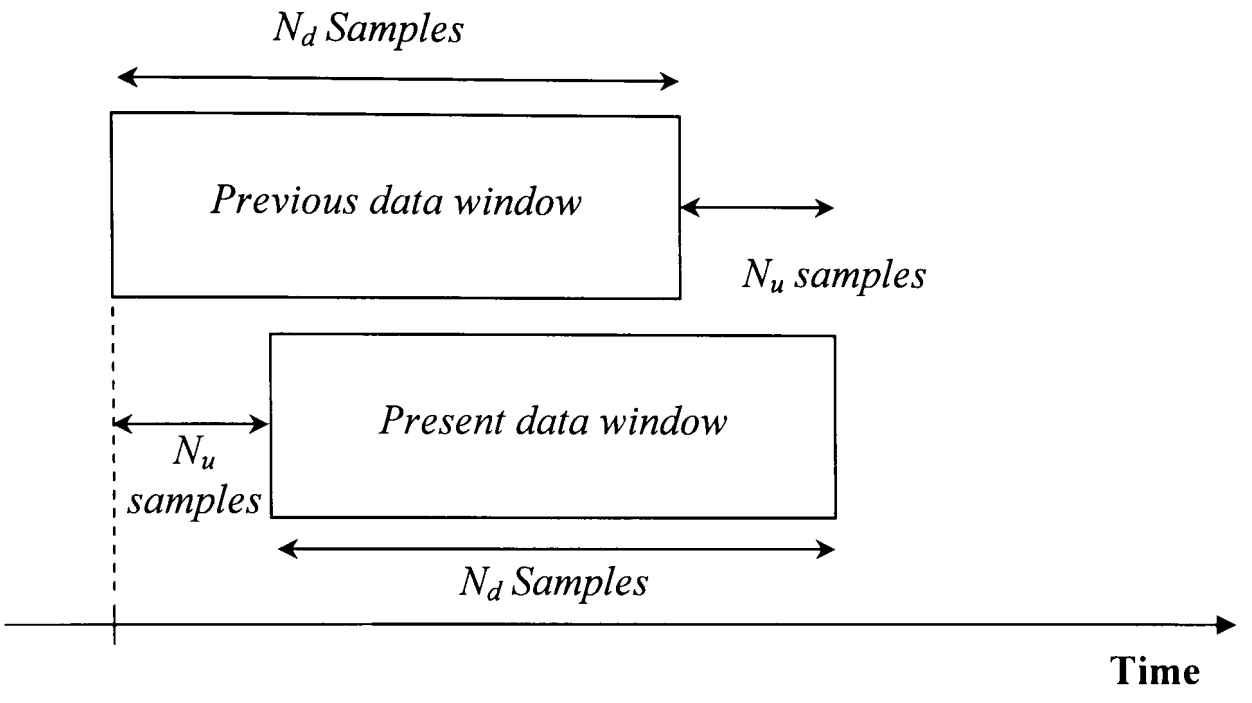

Figure (4-2): The moving Horizon aspect.

Assume that the output of the system at discrete time k is denoted as $y \in \Re^{n_y}$, and the input variable at time k is given by $u(k) \in \Re^{n_u}$. An ARMAX model of the system can be written as (Bamberger and Isermann, 1978):

$$A^*(q^{-1})y(k) = B^*(q^{-1})u(k+1-d) + C^*(q^{-1})\varepsilon(k) + b \tag{4.18}$$

where

$$A^*(q^{-1}) = I + A_1 q + \ldots + A_{n_a} q^{-n_a} \tag{4.19}$$

$$B^*(q^{-1}) = B_1 q^{-1} + B_2 q^{-2} + \ldots + B_{n_b} q^{-n_b} \tag{4.20}$$

$$C^*(q^{-1}) = I + C_1 q^{-1} + \ldots + B_{n_c} q^{-n_c} \tag{4.21}$$

are matrix polynomials of the degrees n_a, n_b and n_c respectively, in the backward shift operator q^{-1}, d is the minimum pure time delay in samples from inputs to

outputs, the sequence $\varepsilon(k) \in \Re^{n_y}$ is assumed to be zero mean discrete white noise, and $b \in \Re^{n_y}$ is an off-set parameter vector introduced to take into account non-zero levels in the signals involved. It has to be mentioned that in an ARMAX model the dynamics of the process are incorporated in the model by the lags in the polynomial arrays A, B and C.

An equivalent non-minimal state-space realization of the deterministic part of the ARMAX model (4.18) is as follows:

$$x(k+1)=Ax(k)+B_u u(k)+c \tag{4.22}$$

$$y(k)=Cx(k) \tag{4.23}$$

where

$$\begin{aligned} x(k) = [&y(k)^T \; y(k-1)^T \; \dots \; y(k-n_a+1)^T \\ &u(k-1)^T \dots u(k-d)^T \dots \; u(k-d-n_b+2)^T]^T \end{aligned} \tag{4.24}$$

is a state vector which contains present and past data values of the output at time k, and past values of the input variables, dim $x = n = n_y n_a + n_u (n_b + d\text{-}2)$, A and B_u are matrices of the appropriate dimensions which are formed in terms of the ARMAX model polynomial coefficients, $c \in \Re^n$ is an off-set vector.

For instance, for the case when $d = 1$, matrices A, B_u and C are given as:

$$A = \begin{pmatrix} -A_1 & -A_2 & \dots & -A_{n_a} & B_2 & \dots & B_{n_b} \\ I_{n_y} & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots I_{n_y} & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots I_{n_u} & 0 \end{pmatrix} \tag{4.25}$$

where A is of dimension: $(n_y n_a + n_u(n_b - 1)) \times (n_y n_a + n_u(n_b - 1))$.

$$B_u = \begin{bmatrix} B_1 \\ 0 \\ \vdots \\ 0 \\ I_{n_y} \\ \vdots \\ 0 \end{bmatrix}_{(n_y n_a + n_u(n_b-1)) \times n_u} \tag{4.26}$$

$$C = \begin{bmatrix} I_{n_y} & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}_{n_y \times (n_y n_a + n_u(n_b-1))} \tag{4.27}$$

Multiplying equations (4.22) and (4.23) by the difference operator $\Delta = 1 - q^{-1}$, the following incremental state-space model is obtained:

$$\Delta x(k+1) = A\Delta x(k) + B_u \Delta u(k) \tag{4.28}$$

$$\Delta y(k) = C\Delta x(k) \tag{4.29}$$

This model is a locally valid linear state space model in the form used in the special case defined above.

The ARMAX model given by equation (4.18) may be written as a regression:

$$y(k) = \Theta^T \varphi(k) + \varepsilon(k) \tag{4.30}$$

where

$$\Theta^T = [A_1 \ \dots \ A_{n_a} \ B_1 \ \dots \ B_{n_b} \ C_1 \ \dots \ C_{n_c} \ b]_{n_y \times (n_a + n_c) n_y + n_b n_u + 1} \tag{4.31}$$

$$\varphi(k) = [-y(k-1)^T \ldots - y(k-n_a)^T \quad u(k-d)^T \ldots u(k-n_b-d+1)^T e(k-1)^T \ldots e(k-n_c)^T 1]^T \tag{4.32}$$

where the residuals $e \in \Re^{n_y}$ may be defined as:

$$e(k) = y(k) - \Theta^T \varphi(k) \tag{4.33}$$

Given N_d distinct data samples $\{ y(i),\ \varphi(i)\}$, i=1,..., N_d, the least square estimate of the parameter matrix Θ is given by the solution of the following linear system:

$$\tilde{R}\Theta = \Gamma \tag{4.34}$$

where

$$\tilde{R} = \sum_{i=1}^{N_d} \varphi(i)\varphi(i)^T \tag{4.35}$$

$$\Gamma = \sum_{i=1}^{N_d} \varphi(i) y(i)^T \tag{4.36}$$

It is assumed that the input sequence u (k), $k \in [1, N_d]$ is such that matrix $\tilde{R}$ is non-singular, which occurs if the input sequence is a sufficiently exciting signal. It is intended to use this formulation in a moving horizon fashion (Figure (4.2)). The length of the data window being N_d, the parameter matrix is updated every N_u data samples, where $N_u < N_d$ and $p = N_d / N_u$ is an integer ratio.

A forgetting factor is generally introduced (Becerra et al., 1998) to enhance the model adaptation to changes in the dynamics, by giving less importance to older data within the data window.

Therefore, the matrices $\tilde{R}$ and Γ may be written as follows:

$$\tilde{R} = \sum_{s=1}^{p} \lambda_f^{p-s} \tilde{R}_s \tag{4.37}$$

$$\Gamma = \sum_{s=1}^{p} \lambda_f^{p-s} \Gamma_s \tag{4.38}$$

where λ_f is a scalar parameter $0 < \lambda_f \leq 1$ known as the forgetting factor, $\tilde{R}_s$ and Γ_s are given as:

$$\tilde{R}_s = \sum_{t=(s-1)N_u+1}^{sN_d} \varphi(t)\varphi(t)^T \tag{4.39}$$

$$\Gamma_s = \sum_{t=(s-1)N_u+1}^{sN_d} \varphi(t)y(t)^T \tag{4.40}$$

where $s = 1, \ldots, p$, $\dim \tilde{R}_s = n_y(n_a + n_c) + n_u n_b + 1 \times (n_a + n_c) + n_u n_b + 1$, and $\dim \Gamma_s = n_y(n_a + n_c) + n_u n_b + 1 \times n_y$.

It is important to note that the algorithm avoids the same sections of data between consecutive parameter updates by exploiting the displacement structure of the data window (Figure 4-2). Also, because the matrices $\tilde{R}_s$ and Γ_s are updated at every sampling instant, there is no need to store N_d pairs of measurements. The algorithm is recursive if $n_c > 0$, since the previous value of the parameter matrix Θ^{j-1} affects its current estimate Θ^j, because the residuals $e(k-1) \ldots e(k-n_c)$ are computed using Θ^{j-1}.

The full algorithm is given below (Becerra et al., 1998):

Data	$N_d, N_u, n_a, n_b, n_c, d, and\ \lambda_f$

Step 0 Set k=0, j=0, *flag*=0, $\Gamma_s = 0, R_s = 0, s = 1,\ldots, p.$

Step 1 Obtain a new measurement vector $y(k)$ and input vector $u(k)$.

Step 2 If $k \geq 1$ do the following sub-steps:

2.1 Form the regression vector $\varphi(k)$, using the latest value of Θ to compute the residuals $e(k-1),\ldots,e(k-n_c)$.

2.2 If *flag* = 0 then set L=N_a, else L= N_u.

2.3 If $k < L$ then do

2.3.1 If *flag* = *1* then set $i = j$, else set

$$i = \text{int}\left[(k-1)/N_u\right]+1$$

2.3.2 Update $R_i = R_i + \varphi(k)\varphi(k)^T$

2.3.3 Update $\Gamma_i = \Gamma_i + \varphi(k)y(k)^T$

else do

2.3.4 If *flag* = 0 then set *flag* = *1*, and compute the following summations:

$$\tilde{R} = \sum_{r=1}^{p} \lambda^{p-s(r)}\, \tilde{R}_r$$

$$\Gamma = \sum_{r=1}^{p} \lambda^{p-s(r)} \Gamma_r$$

where $s(r)$ is an integer function mapping the corresponding segment of the data window from the indexed matrix set index r.

2.3.5 Compute the parameter matrix Θ by solving the linear system (4.34).

2.3.6 If $j = p$ then set $j = 0$

2.3.7 Set $j = j+1$, $R_j = 0$ and $\Gamma_j = 0$

2.3.8 set $k = 0$

2.4 Set $k = k+1$ and go to step 1.

4.5.2. DMI with a Non-linear model representation

Most dynamical systems can be better represented by non-linear models, which are able to describe the global behaviour of the system over a wide operating range, rather than by linear ones that are only able to approximate the system around a given operating point. Hence the use of a general non-linear model (2^{nd} order Hammerstein Model for simplicity) for the identification, in order to extract the derivative matrix from it. A Hammerstein Model is a series combination of a memoryless nonlinearity and linear dynamics. It is used to identify systems of high nonlinearities.

This is the first time it is used and implemented within the ISOPE algorithm. The work is inspired from that of Bamberger and Isermann, (1978).

It is assumed that the process is stable and can be approximated by a non-linear lumped parameter model. The process model to be identified considers n inputs, $u^T = [u_1, u_2, ..., u_n]$ and p outputs, $y^T = [y_1, y_{2,...}, y_p]$. A generalised second order Hammerstein model is used (Bamberger and Isermann, 1978):

$$\begin{aligned} y_1(k) = b_{00} + B_{l10}(q^{-1})u_1(k-d) + \ldots + B_{l11}(q^{-1})u_1^2(k-d) + \ldots + \\ B_{lv\mu}(q^{-1})u_v(k-d)u_\mu(k-d) + \cdots + B_{lnn}(q^{-1})u_n^2(k-d) - A_{1l}^T(q^{-1})y_l(k) \end{aligned} \tag{4.41}$$

where:

$$\begin{aligned} A_{1l}(q^{-1}) &= 1 + a_{1l1}q^{-1} + \ldots + a_{1lm}q^{-m} \\ &= 1 + A_{1l}^T(q^{-1}) \end{aligned} \tag{4.42}$$

$$B_{lv\mu}(q^{-1}) = b_{lv\mu1}q^{-1} + \ldots + b_{lv\mu m}q^{-m} \tag{4.43}$$

are polynomials of order l in the backward shift operator q^{-1}, and T denotes Transpose.

Thus, equation (3.41) can now be written in the form:

$$y_1(k) = \Omega^T(k)\Theta(k) \tag{4.44}$$

where:

$$\begin{aligned} \Omega(k) = [1\ u_1(k-d) \ldots u_1(k-d-m)\ u_1^2(k-d) \ldots u_1^2(k-d-m) \ldots \\ u_v(k-d)u_\mu(k-d) \ldots u_v(k-d-m)u_\mu(k-d-m) \ldots \\ u_n^2(k-d) \ldots u_n^2(k-d-m) \ldots y_1(k-1) \ldots y_l(k-m)] \end{aligned} \tag{4.45}$$

and

$$\begin{aligned} \Theta^T = [b_{00}\ b_{l101} \ldots b_{l10m}\ b_{l111} \ldots b_{l11m}\ b_{lv\mu1} \ldots \\ b_{lv\mu m} \ldots b_{lnn1} \ldots b_{lnnm}\ -a_{1l1} \ldots -a_{1lm}] \end{aligned} \tag{4.46}$$

The calculation of all the parameters is made via a recursive least-square algorithm:

$$\hat{\Theta}(k+1) = \hat{\Theta}(k) + \rho(k)[y_1(k+1) - \Omega^T(k+1)\hat{\Theta}(k)] \qquad (4.47)$$

and

$$\rho(k) = \frac{1}{\Omega^T(k+1)P(k)\Omega(k+1) + \lambda} P(k)\Omega(k+1) \qquad (4.48)$$

$$P(k+1) = [I - \rho(k)\Omega^T(k+1)]p(k)/\lambda \qquad (4.49)$$

with,

$$P(0) = \omega I, \omega >> 1000 \qquad (4.50)$$

$$\hat{\Omega}(0) = 0 \qquad (4.51)$$

with $0.95 \leq \lambda \leq 0.98$.

As in our case the derivative information is needed, only the steady-state model of the system is required. This is obtained by simply setting q=1 (final value for z-transform) in equation (4.41).

Therefore:

$$y_1 = \frac{b_{00}}{A_{1l}(1)} + \frac{B_{l10}(1)}{A_{1l}(1)}u_1 + \frac{B_{l11}(1)}{A_{1l}(1)}u_1^2 + \ldots + \frac{B_{lv\mu}(1)}{A_{1l}(1)}u_v u_\mu + \ldots + \frac{B_{lnn}(1)}{A_{1l}(1)}u_n^2 \qquad (4.52)$$

The coefficients b_{00}, B_{lij} and A_{il} are the result of the least-square identification process of the non-linear model.

By comparison, one can easily extract the output derivative matrix with respect to the set-points of any system that can be represented by equations (4.16) and (4.17).

In the case of very noisy processes, identification with a special correlation technique gives better performance, without any parameter estimation (Bamberger and Isermann, 1978).

Only the steady-state model obtained during the transient phase is used in the ISOPE algorithm in order to optimise the performance index. To start the procedure, a test signal v_s which has to fulfil certain conditions, is used to accelerate the process identification. After the initial crude model is obtained, the optimisation starts providing additional changes of the input v. These changes in the input are to improve the continuing process identification so that the amplitudes of the test signals can be reduced.

4.6 Simulation Case study

A set of simulations is carried out on the two Continuous Stirred Tank Reactors (CSTR's) connected in cascade presented in chapter 3. These simulations were created in order to assess the methods and techniques presented in this chapter. A comparison is made between these methods in terms of convergence, stability and speed.

4.6.1. Optimisation objectives and goals

The objective function for all the simulations using this system was chosen to be linear of the measured variable C_{b2} and reflects the desire of maximising the amount of component B in tank 2. Thus the form of the objective function is as follow:

$$L(y,v) = -C_{b2} \tag{4.53}$$

It has to be noted that this objective function is linear of the measured variable C_{b2}, but is a nonlinear function of the manipulated variables T_1 and T_2. This is due to the nonlinearity of the system equations (chapter 3).

However, the model-based optimisation is performed on the unfaithful model chosen to be linear of the form:

$$y = \begin{bmatrix} Ca_2 \\ Cb_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{4.54}$$

where α_1 and α_2 are the free parameters to be estimated and $a_{ij}\{i = 1, 2, j = 1, 2\}$ are model parameters updated periodically using measurements (for more details, the reader is referred to Ellis et al, 1993).

A SIMULINK model of the real process was created to enable periodic calls to the ISOPE algorithm saved in an M-file. All the simulations were started from the same starting point which is the initial steady-state condition given by: $T_1 = 307K$ and $T_2 = 302K$, which yields the following steady-state outputs: C_{a2}=0.041361 [$kmol/m^3$] and C_{b2}=0.058638 [$kmol/m^3$].

During the simulations, sufficient time was allowed for the system to settle down to a new steady-state condition before measurements were taken. The only exception was for the DMI method where the identification was carried out during the transient and then the updated model was used in the optimisation routine to update the set-points.

When using the dynamic model identification method to approximate the real system output derivatives with respect to the set-points, a pseudo random binary sequence (PRBS) of magnitude $\pm 0.5K$ was needed to excite the system in order to get an accurate enough model, for which the identifier parameters were tuned

as given in Table 4.1. The tuning of these parameters purely depends on practical issues.

Three phases in the DMI procedure can be distinguished: dynamic model identification (or updating), steady-state model updating and model-based optimisation. The model based optimisation is incorporated in the ISOPE algorithm as explained in Chapter 2. The identification is performed during the transient using input/ output data gained from the real process (measurements). When found the steady-state model is extracted and the derivative information is therefore found and used in the model-based optimisation procedure.

The relaxation gain matrix choice is dependent on the method being used. The value of the gain matrix was practically chosen to suit the algorithm convergence and stability.

The final converged results of the simulations for the various techniques using this example are shown in Tables (4.2) to (4.3) and Figures (4-3) to (4-9).

We notice that all the methods converge to the correct process optimum point given by $T_1=312\ K$ and $T_2=310.2K$, with the optimum objective function value of -0.0725. This is to be expected, as all techniques satisfy the necessary system optimality conditions. Table (4.3) shows that the method using dynamic model identification scheme converges faster than the other methods used in the simulations. It is also seen from the same table, that the method using finite differences to estimate the derivatives takes much more time to converge (in terms of number of set-point changes), while it only needs a few iterations. This is in total agreement with what was stated in the previous sections, because in the dynamic model method the derivatives are estimated during the transient using real system measurements, while the original method using finite differences is steady-state and needs n times more the number of iteration (n being the number of set-points); which could be prohibitive for large systems with a large number of inputs and outputs and also for slow processes.

Figures (4-3) to (4-7) show the trajectories taken by the manipulated variables (set-points) and process outputs. It is seen how the changes in the set-points affect the measured outputs and how they derive their values from the initial steady-state condition given by $C_{a2}(0)=0.041361\ [kmol/m^3]$, $C_{b2}(0)=0.058638\ [kmol/m^3]$ to the final converged solution ($C_{a2}=0.0275\ [kmol/m^3]$, $C_{b2}=0.0725\ [kmol/m^3]$).

Figure (4-8) illustrates the results of noise-contaminated case simulations for the dynamic model identification method. The results show that this method is noise-insensitive because even in the presence of noise the final optimum solution was reached, yet it made the algorithm slower taking more time to converge than in the noise-free case.

Table (4.1): Tuning the identifier parameters.

	Linear model	Nonlinear model
Length of data window	$N_d = 120$	$N_d = 60$
Model orders	$n_a = 2, n_b = 5, n_c = 1, d = 1$	$d = 1$
Identifier sampling time	$T_s = 60s$	$T_s = 60s$
Relaxation gain	$K = 0.03I$	$K = 0.1I$

Table (4.2): ISOPE algorithm with the different estimation techniques.

	FDAM	Broydon's method	Dual control method	Linear dynamic model	Nonlinear dynamic model
Function value	-0.0725	-0.0725	-0.0725	-0.0725	-0.0725
Number of Set-point changes	22	12	14	10	12

4.6.2. Results and Discussion

Techniques for estimation of real process derivatives to be used within the ISOPE algorithm have been presented and applied on a cascade process consisting of two Continuous Stirred Tank Reactors.

All methods, due to the satisfaction of optimality conditions, do achieve the real process optimum provided they can be implemented in a stable manner after a suitable choice of relaxation gains. The speed of convergence and the sensitivity to noise are the criteria for algorithm selection.

It is well documented that the FDAM is not a good choice in the case of high order, slow and noisy processes. Each time a process derivative is requested, a set-point perturbation needs to be applied and a measurement time must to be observed to allow the process to settle before the derivatives are calculated. Additional difficulties are observed when noise is present on the output measurement. This set-point perturbation, and the subsequent measurement time, is where the majority of time is spent in the algorithm so this is a major consideration in assessing the algorithm. As can be seen from the simulation of the CSTR's system (Table 4.2), the FDAM, approaches twice the number of set-point changes of the various following methods and would seem not to be the perfect choice of algorithm.

The dual control method takes 14 set-point changes (Table 4.2) to achieve the optimum in the CSTR's simulation. This is still more than the rest of the methods but the ability of the algorithm to estimate the derivatives without any excess in the set-point changes makes it a good choice. However, the applicability of this algorithm is quite limited due to the need of an additional inequality constraint in order to obtain a smooth trajectory of successive set-points in their space, which is not always reachable (Brdy's and Tatjewski, 1994).

Both Broydon's method and DMI with non-linear model take 12 set-point changes in the simulation. Even though they converge to the correct optimum, the first method encountered a major drawback near the optimum (equation 4.15), as it could lead to an infinite estimate of the derivative matrix. While the second method, which is the DMI with non-linear model proved to be suitable only for low-order non-linear systems. This is contrary to DMI with the linear dynamic model, which has a wide applicability range.

The most suitable used method in this example is DMI with the linear dynamic model as only 10 set-point changes are needed. However, this is likely to be because the process performance has a fairly smooth nature. In other situations, where the process performance is more erratic, DMI with the non-linear model may be more appropriate. The DMI with the linear model method is seen to be the fastest to converge and moreover noise insensitive as the least square estimator used in the algorithm plays a filter role. However, the huge amount of data needed for the estimation and the poor model estimates it gives at the beginning of the identification are its major drawbacks.

Table (4.3): Derivatives Comparison table.

Method	Optimum set-points	Estimates of the Derivatives at the optimum
ISOPE with **FDAM**	T_1=312 K T_2=310.2K	[-0.0094 -0.0071 0.0094 0.0071]
ISOPE with Broydon's method	T_1=312 K T_2=310.2K	[0.0081 0.0086 -0.0081 -0.0089]
ISOPE with dual control method	T_1=312 K T_2=310.2K	[-0.0007 -0.0008 0.0007 0.0008]
ISOPE with linear dynamic model	T_1=312 K T_2=310.2K	[-0.0096 0.0071 0.0096 -0.0071]
ISOPE with Nonlinear dynamic model	T_1=312 K T_2=310.2K	[-0.0092 -0.0070 0.0092 0.0070]

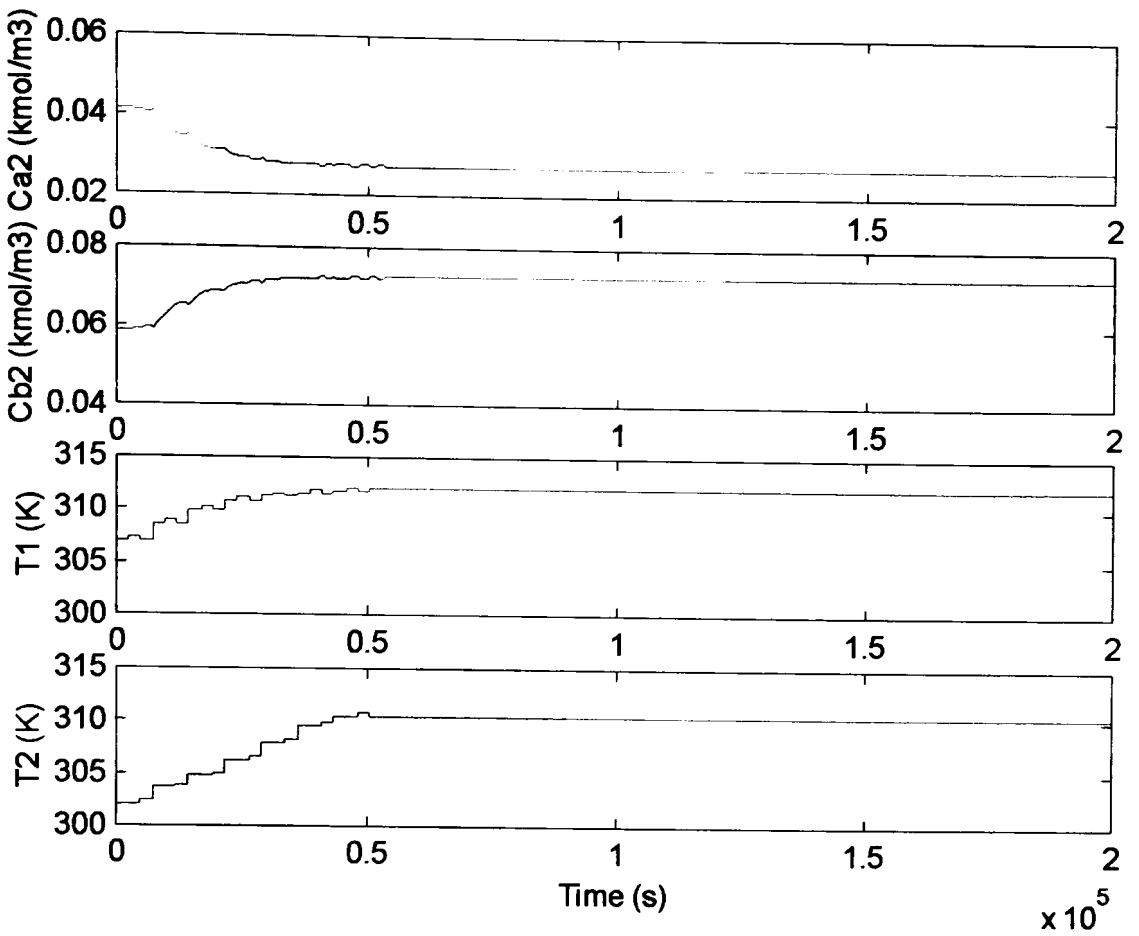

Figure (4-3): FDAM method.

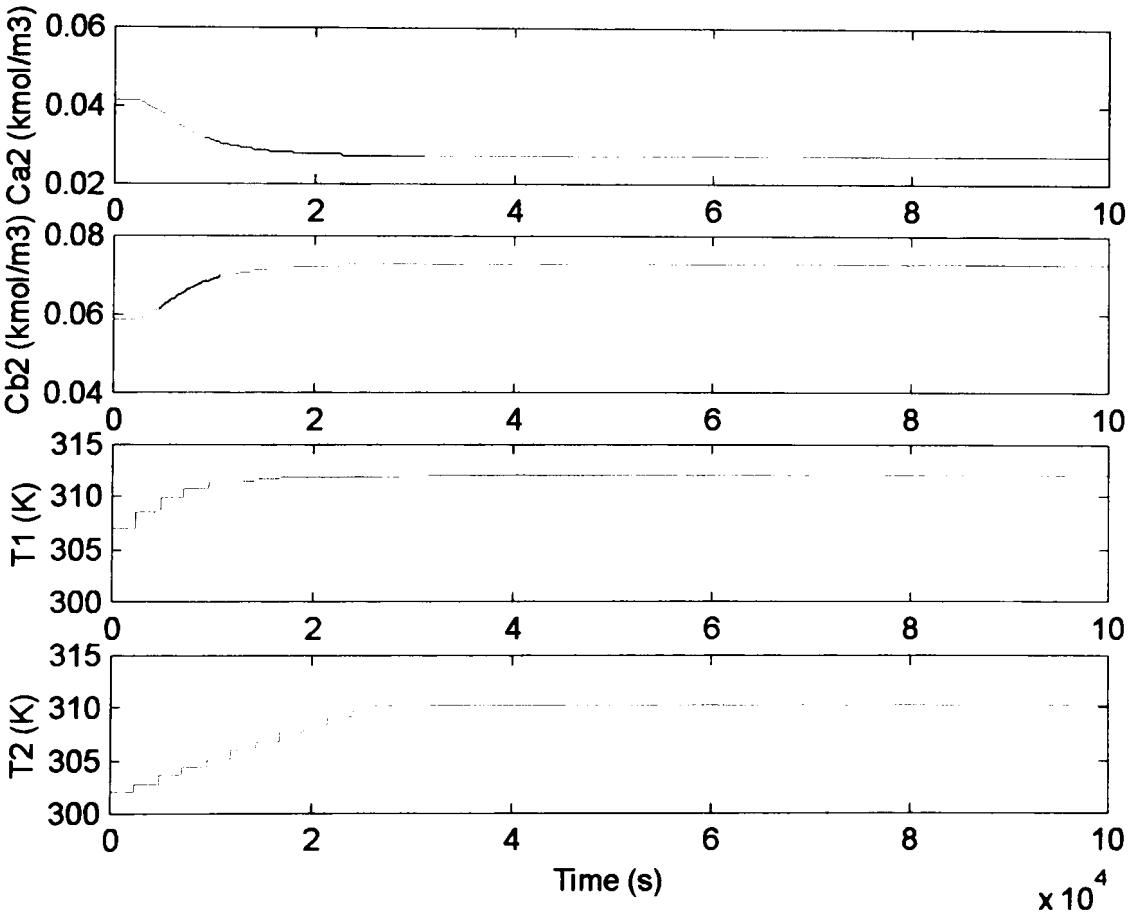

Figure (4-4): Broydon's method.

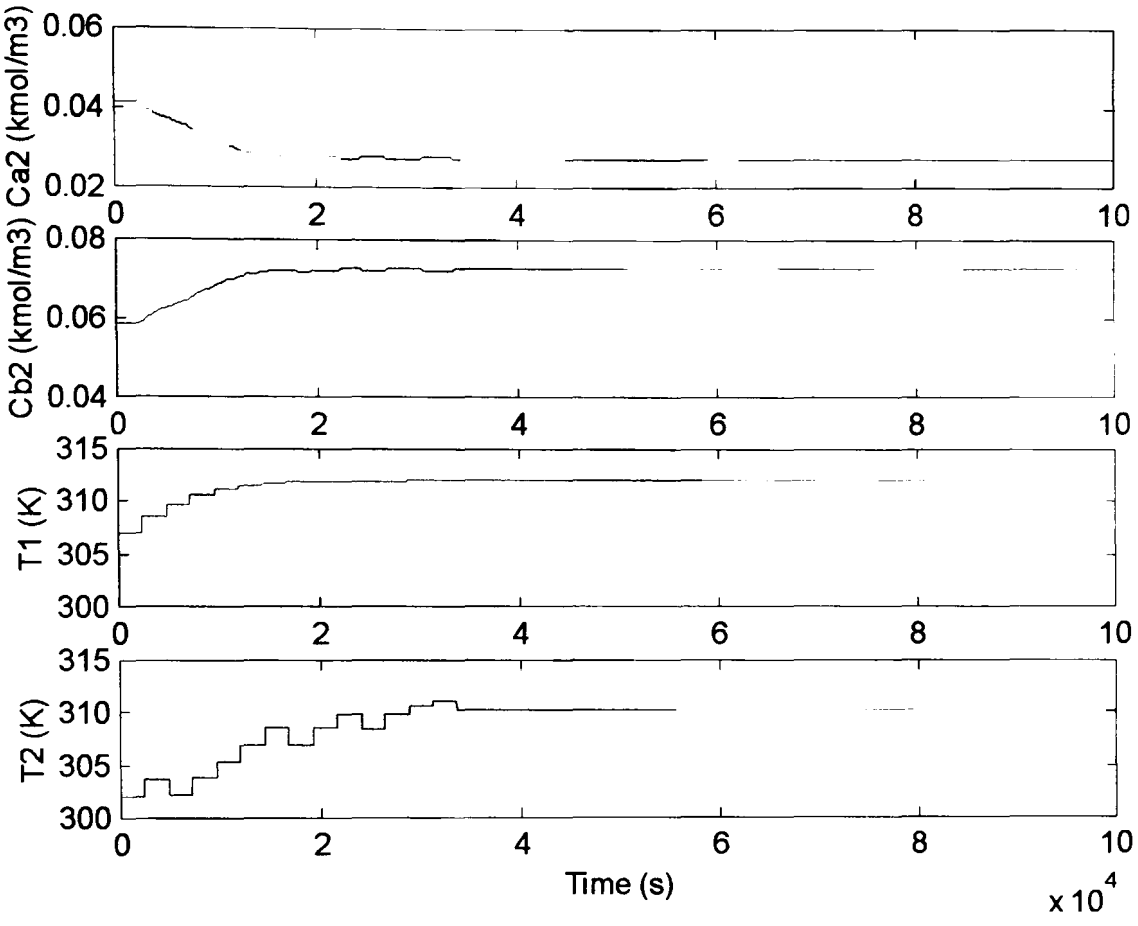

Figure (4-5): Dual control method.

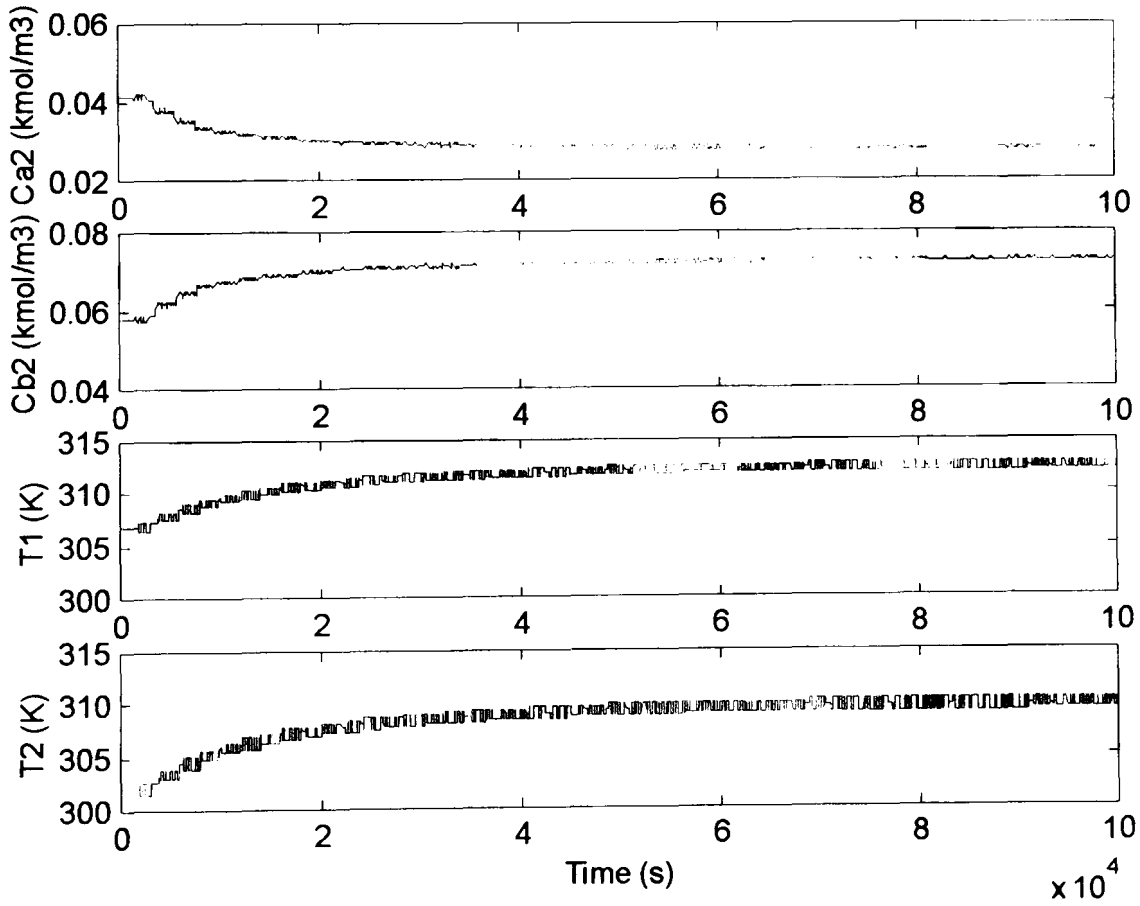

Figure (4-6): DMI with linear model method.

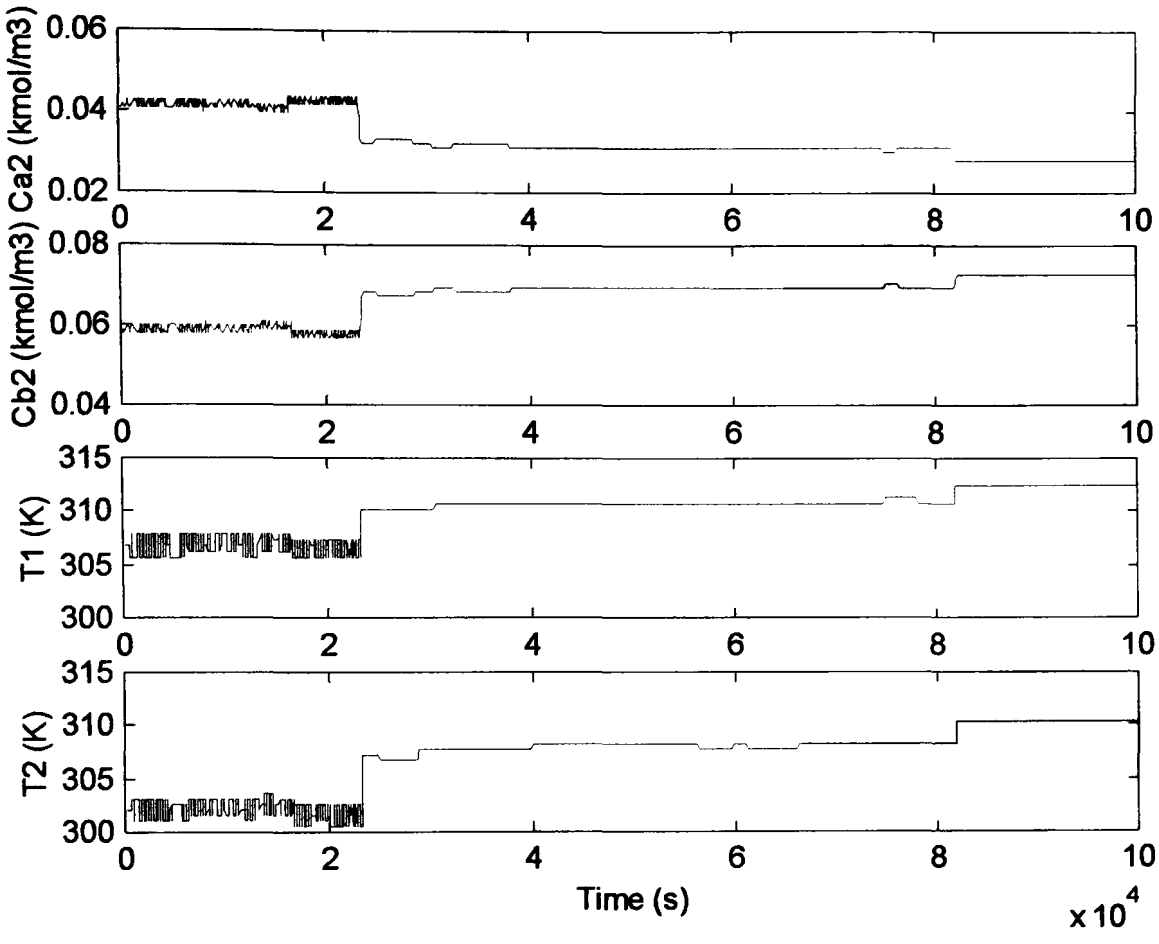

Figure (4-7): DMI with nonlinear model method.

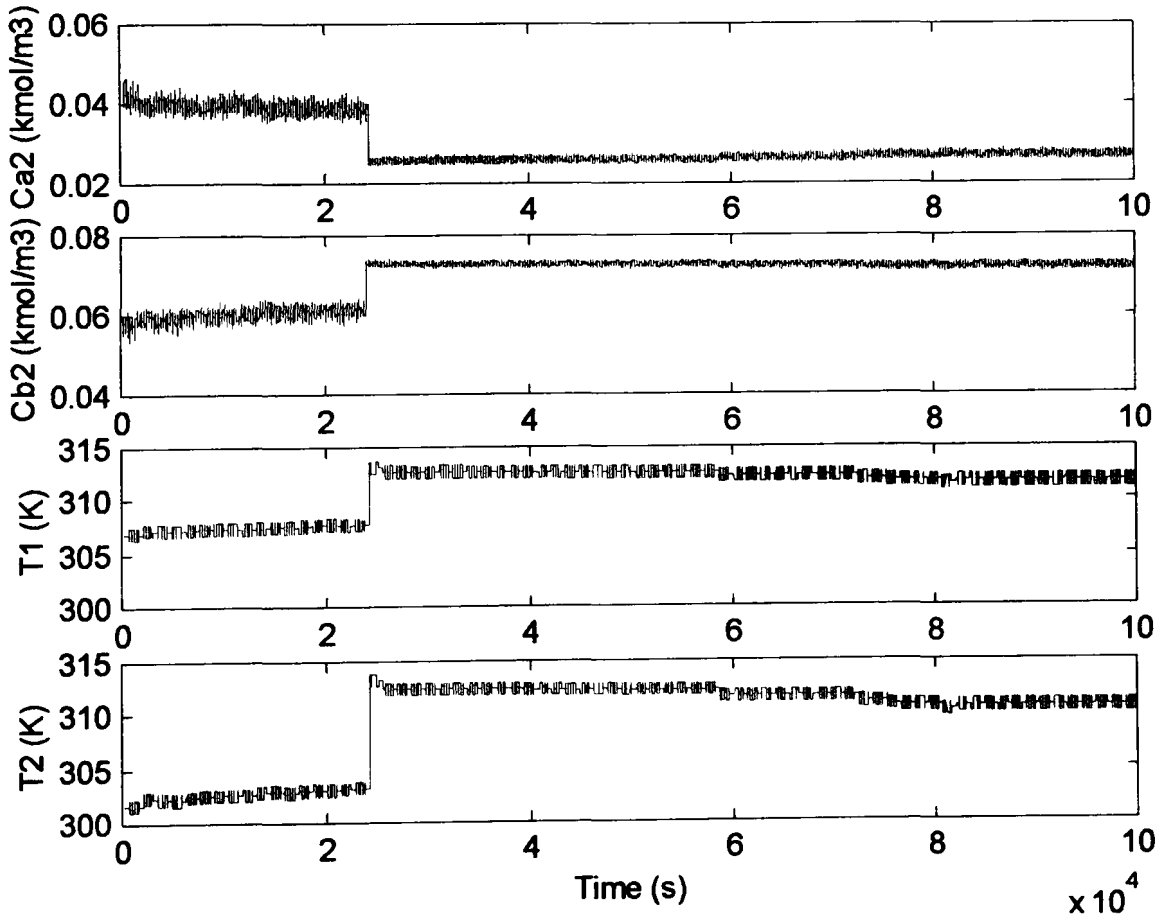

Figure (4-8): DMI with linear model in noise contaminated case.

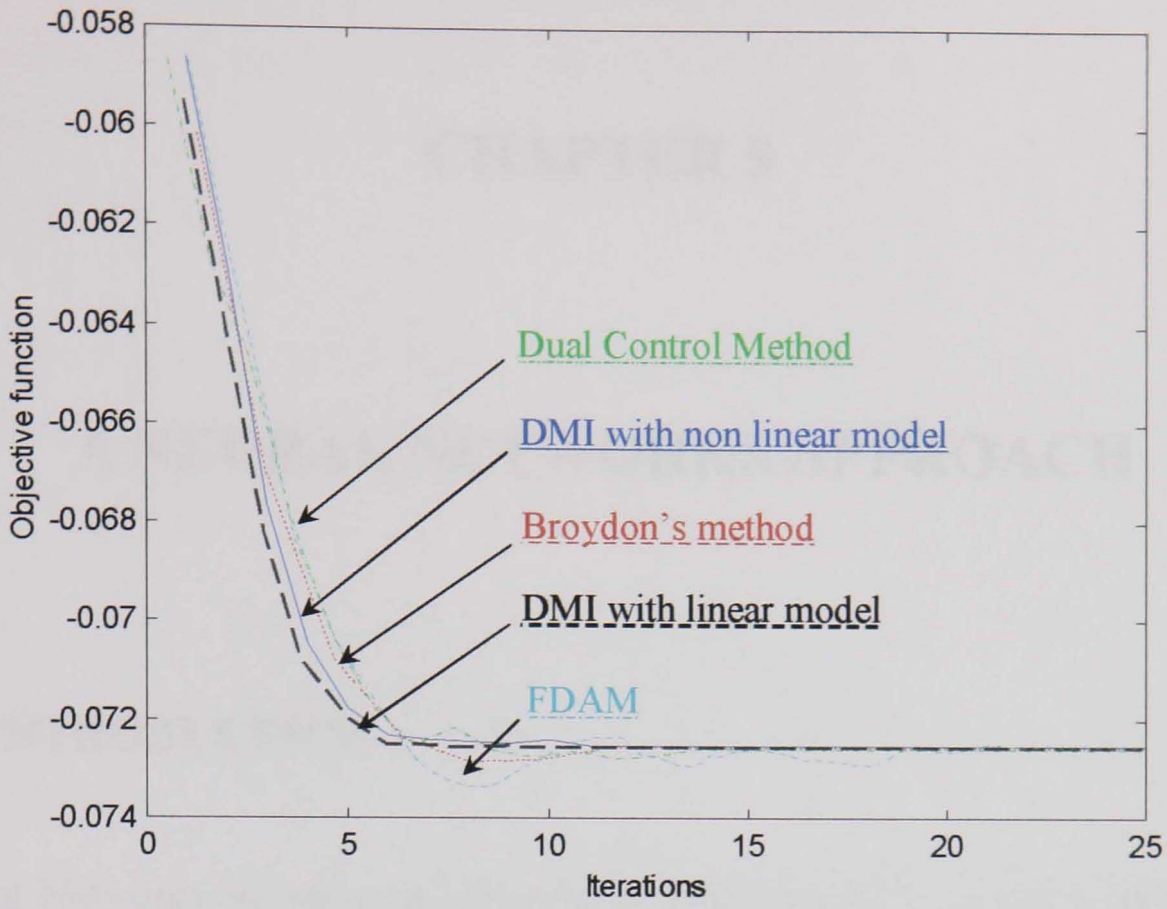

Figure (4-9): Objective function graph.

4.7 Summary

In this chapter, algorithms for estimating real process derivatives were presented. These derivatives are needed by the ISOPE algorithm in order to satisfy necessary optimality conditions. The techniques presented here are well known and most of them have been successfully used in real situations. Comparison simulations were carried out on a Two Continuous Stirred Tank Reactors system connected in cascade. Results showed the superiority of the dynamic model identification method.

In the next chapter, a neural network method for estimating the process derivatives to be used in the ISOPE algorithm will be presented.

CHAPTER 5

A NEURAL NETWORKS APPROACH

5.1 INTRODUCTION

Neural Networks represent an emerging technology rooted in many disciplines. They are endowed with some unique attributes: universal approximation (of functions), the ability to learn from and adapt to their environment, and the ability to invoke weak assumptions about the underlying physical phenomena responsible for the generation of the input data.

This ability of learning from the environment and producing accurate approximation of functions make neural networks an effective tool (Narendra and Parthasarathy, 1990) to be used in identification and control of nonlinear dynamical systems. In fact, the development and design of a neural network which can learn and quickly adapt from its environment, the physical system, is shown in this chapter to give good results when used within the ISOPE algorithm in terms of convergence properties involving such factors as: speed, precision, stability, etc.

This chapter presents an attempt to use Artificial Neural Networks (ANN) to estimate real process derivatives to be used within the ISOPE algorithm. A general but brief introduction to Neural networks is given first, with all the related details and background related to our work. Then, two types of ANN architectures namely Multilayer and recurrent networks are described together with static back-propagation algorithm used to train the network and adjust its parameters. The

performance of the neural network scheme presented in this chapter is tested under simulation in two case studies employing the systems presented in chapter 3. The results are compared with those obtained by the FDAM method described in the previous chapter.

Artificial neural networks (ANNs) can be considered as collections of very simple "computational units" which can take a numerical input and transform it into an output. It resembles the brain in two aspects:

1. Knowledge is acquired by the network through a learning process.
2. Interneuron connection strengths known as synaptic weights are used to store the knowledge.

The procedure used to perform the learning process is called a *learning algorithm*, the function of which is to modify the synaptic weights of the network in an orderly fashion so as to attain a desired design objective (Haykin, 1995).
The principle of supervised learning in ANNs is that the ANNs take numerical inputs (the training data) and transform them into "desired" (known, predetermined) outputs. The input and output nodes may be connected to the "external world" and to other nodes within the network. The way in which each node transforms its input depends on the so-called "connection weights" and "bias" of the node, which are modifiable. The output of each node to another node or the external world then depends on both its weight strength and bias and on the weighted sum of all its inputs, which are then transformed by a normally nonlinear, weighting function referred to as its activation function. The great power of neural networks stems from the fact that it is possible to "train" them. Training is achieved by continually presenting the networks with the "known" inputs and outputs and modifying the connection weights between the individual nodes and the biases, typically according to some kind of back-propagation algorithm (Rumelhart et al., 1986), until the output nodes of the network match the desired outputs to a stated degree of accuracy. If the outputs from the

previously unknown inputs are accurate, the trained ANN is said to be generalised.

Neural networks are characterized by two major capabilities. The first is their parallel distribution structure and the second, is their ability to learn and generalize. These two capabilities make neural networks an attractive tool that can find application in many disciplines.

The use of neural networks offers the following useful properties and capabilities:

1. Nonlinearity: As most physical activities happening around us are mainly nonlinear, neural networks provide a useful tool in dealing with such phenomena because of its nonlinearity capabilities.
2. Input-Output mapping: Similar to the nonparametric statistical inference, neural networks have the capability of performing input-output mappings using the so-called *supervised learning*. In fact, this involves training the network for a given set of data for which the synaptic weights are modified accordingly using an appropriate optimisation criterion.
3. Adaptivity: The ability to adapt to any changes in the surrounding environment added to the natural architecture of a neural network make it an ideal tool in adaptive pattern classification and adaptive control. Indeed, whenever changes occur in the system or its environment, the network is retrained for the new set of data, and the synaptic weights are adapted to their new values.
4. Uniformity of Analysis and Design: Neural networks enjoy universality as information processors. This feature manifests itself in different ways (Haykin, 1994):

    a. Neurons, in one form or another, represent an ingredient common to all neural networks.
    b. This commonality makes it possible to share theories and learning algorithms in different applications of neural networks.
    c. Modular networks can be built through a seamless integration of modules.

Many other properties and capabilities can be offered by neural networks. For further details, see Haykin (1995).

5.1.1 History and development of Neural Networks

The first step toward artificial neural networks came in 1943 when Warren McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper (McCulloch and Pitts, 1943) on how neurons might work. They modelled a simple neural network with electrical circuits.

Donald Hebb (1949) reinforced this concept of neurons and how they work in his book *Organization of Behaviour* published in 1949. In the 1950's more research was carried out in the area of artificial neural networks. Indeed, after many failed attempts, researchers finally succeeded in simulating a neural network. In 1956 Uttley (1956) demonstrated that a neural network with modifiable synapses may learn to classify simple sets of binary patterns into corresponding classes. In the same year, the Dartmouth Summer Research Project on Artificial Intelligence (AI, as it is known in industry) provided a boost to both AI and neural networks, by stimulating research in both the intelligent side, AI, and the much lower level neural processing part of the brain.

The following years saw the introduction of a new approach to the pattern recognition problem by Rosenblatt in his work on the *perceptron*. In the beginning of the sixties, Widrow and Hoff (1960) introduced the least mean square (LMS) algorithm and used it to develop their ADALINE (ADAptive LINear Element) and MADALINE (Multiple ADALINE) models (Widrow, 1962). MADALINE was the first neural network to be used in a real world problem, and is still in commercial use. An important disadvantage was encountered in the design of multilayer perceptrons which is the *credit assignment problem*. This problem was first observed by Minsky (1961). However, the solutions to this problem did not emerge until the 1980's. The reasons behind this lag of over 10 years are multiple, but they are mainly caused by the halt of

funding and the more or less dampening of interest in neural networks in the 1970s, where many researchers deserted the field. In 1982 several events caused a renewed interest. John Hopfield made a transformation in the field of neural networks by introducing a new approach to understanding the computation performed by *recurrent neural networks* with symmetric synaptic connections (Hopfield, 1982). At the same time the US-Japan joint conference on cooperative/ competitive neural networks resulted in a flowing of funding once again. 1986 saw the publication of a two volume book by Rumelhart and McClelland. The book has made considerable contribution in the use of the *back propagation* learning algorithm which is considered to be the most popular learning algorithm for the training of multilayer perceptrons. In the 1990s, neural networks attracted more interest from researchers in different disciplines because of its versatile application and use. Today, neural networks are developing fast and their promise seems to be very bright as nature itself is the proof that such things do work. A full and detailed review of neural networks and its applications can be found in Haykin, (1994).

5.2 MULTILAYER AND RECURRENT NETWORKS

In general, four different classes of neural networks architectures can be distinguished (Haykin, 1994):

1. Single layer Feedforward networks
2. Multilayer Feedforward networks
3. Recurrent Networks
4. Lattice Structures

The difference between each type of architecture is the manner in which the neurons of the neural network are structured and organised within the actual network. Also, we have to mention that the architecture of an ANN is intimately linked with the learning algorithm used to train the network (Haykin, 1994).

In this section, two different classes of architectures namely the multilayer feedforward and recurrent networks are described. These two classes have received considerable attention in the area of ANN. Multilayer networks have proved to be successful in pattern recognition, while recurrent networks have been used in associative memories as well as for the solution of optimisation problems. Theoretically, multilayer networks represent static nonlinear maps of systems. However, recurrent networks are represented by nonlinear dynamic feedback systems (Narendra and Parthasarathy, 1990).

5.2.1. Multilayer Feedforward networks

One distinguished class of neural networks architecture is the multilayer feedforward network (Figure 5-1). It is characterised by the presence of one or more hidden layers; each layer can have one or more hidden neurons. The function of the hidden neurons is to intervene between the external input and the network output. By adding one or more hidden layers, the network is enabled to extract higher-order statistics, for the network acquires a global perspective despite its local connectivity by virtue of the extra set of synaptic connections and the extra dimension of neural interactions. The neural network presented in Figure (5-1) is said to be *fully connected* as every node in each layer is connected to every other node in the adjacent forward layer. However, if some of the synaptic weights are missing, the network is said to be *partially connected.* A simplified block diagram representation of the multilayer neural network of Figure (5-1) is given in Figure (5-2), where each layer of the network is represented by the following:

$$N_i[u] = \Gamma[W^i u] \tag{5.1}$$

The matrices W^i's are weighting matrices tuned as described in section 5.3, Γ is a diagonal nonlinear operator referred to as the activation function.

In this case the input/output mapping of the multilayer network presented in Figure (5-2) is given by:

$$y = N[u] = \Gamma\left[W^2\Gamma\left[W^1\Gamma\right]\right] = N_2N_1[u] \tag{5.2}$$

The choice of the activation function Γ depends on the user and sometimes of the type of the application (Haykin, 1994). The most commonly used activation function is the sigmoidal function which the elements γ are of the form (Figure 5-3):

$$\gamma(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{5.3}$$

One reason that makes ANN's with feed-forward architecture so attractive is that it has been shown mathematically (Hornik et al., 1990; White, 1990) that a neural network consisting of only one hidden layer, with an arbitrarily large number of nodes, can learn any arbitrary, and hence nonlinear, continuous function to an arbitrary degree of accuracy. In addition, ANNs are widely considered to be relatively robust to noisy data (Haykin, 1994).

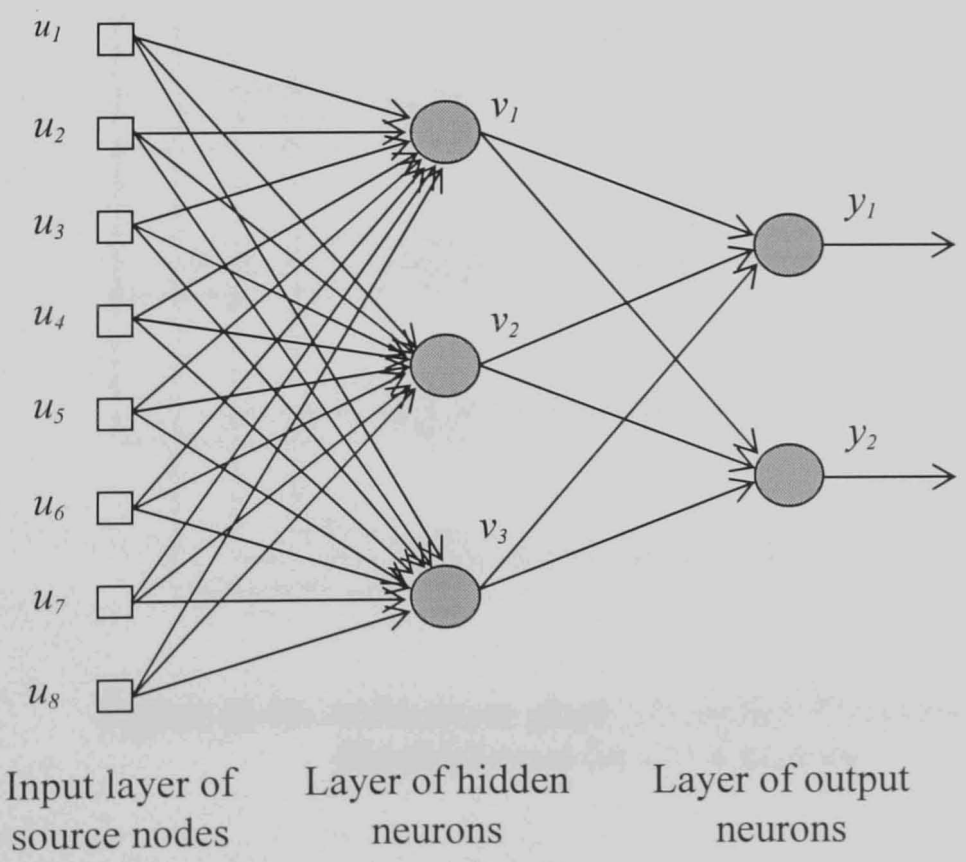

Figure (5-1): A multilayer feedforward neural network with 8 input nodes, 3 hidden and 2 output neurons.

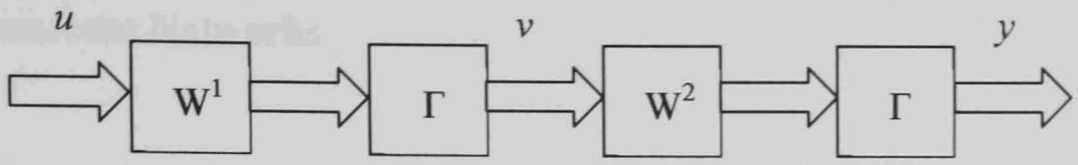

Figure (5-2): Bloc diagram representation of a two layer network.

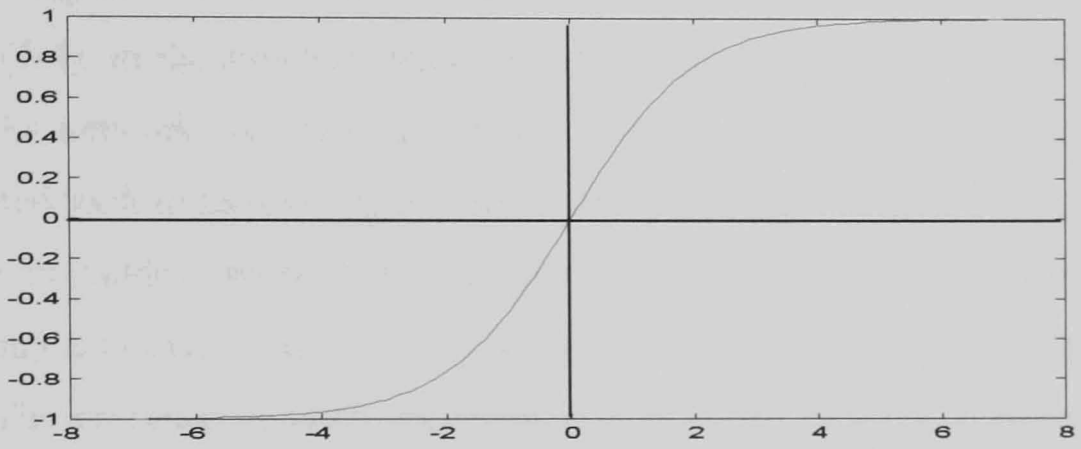

Figure (5-3): Plot of the sigmoidal function.

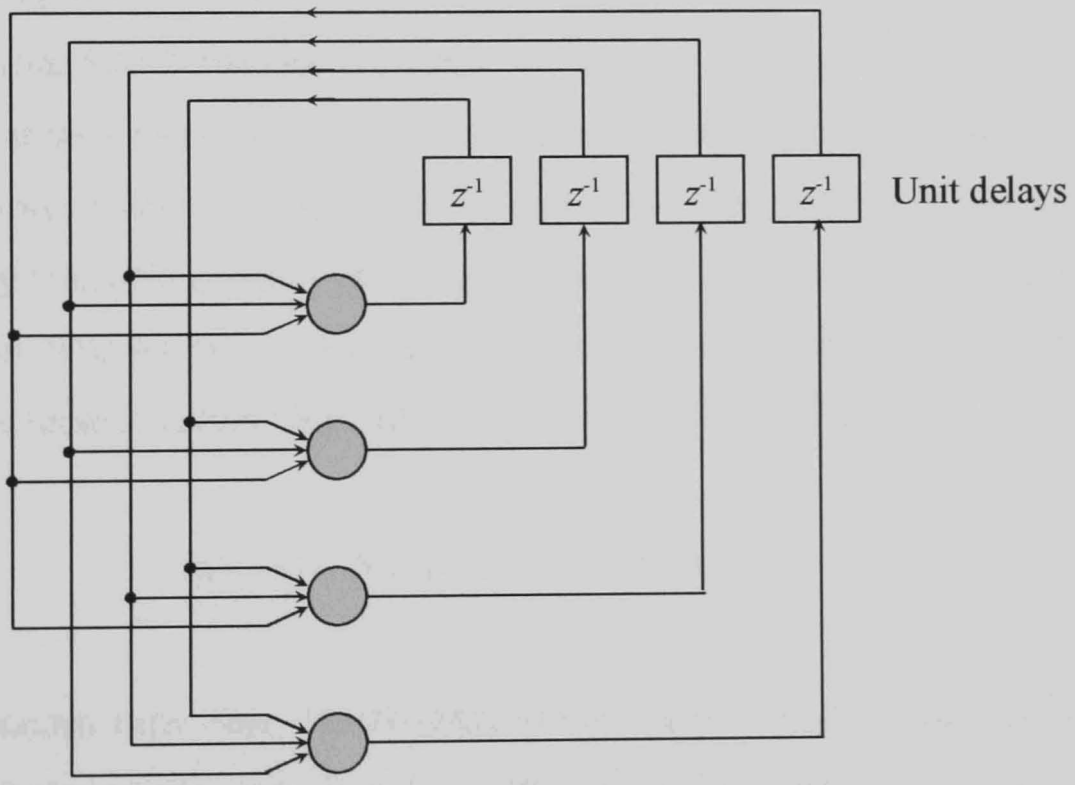

Figure (5-4): Architecture graph of a typical Recurrent network (Hopfield type) for $N = 4$ neurons.

5.2.2. Recurrent Networks

A recurrent neural network distinguishes itself from a feedforward neural network in that it has at least one feedback loop. For example, a recurrent network may consist of a single layer of neurons with each neuron feeding its output signal back to the inputs of all the other neurons, as illustrated in the architectural graph of Figure (5-4). In the structure depicted in this Figure there are no self-feedback loops in the network; self-feedback refers to a situation where the output of a neuron is fed back to its own input. The recurrent network illustrated in Figure (5-4) also has no hidden neurons. Other structures of recurrent networks with hidden neurons may also exist. Figure (5-5) shows an example of a network with hidden neurons. The presence of feedback loops in a recurrent network structure has a profound impact on the learning capability of the network and on its performance (Haykin, 1994). Moreover, the feedback loops involve the use of particular branches composed of unit-delay elements (denoted by z^{-1}), which result in a nonlinear dynamical behaviour by virtue of the nonlinear nature of the neurons. The most common recurrent network architecture is the Hopfield network shown in the example systems in Figures (5-4) and (5-6). One version of the network suggested by Hopfield consists of a single layer network N_1, included in feedback configuration, with a time delay (Figure 5-6). It can be described by the following discrete time representation (Narendra and Parthasarathy, 1990):

$$x(k+1) = N_1[x(k)], \qquad x(0) = x_0 \tag{5.4}$$

In the continuous time case, the dynamic system in the feedback path (z^{-1}) has a diagonal transfer matrix with identical elements of the form: $1/(s+\alpha)$ along the diagonal. The system can then be represented by the following equation:

$$\dot{x} = -\alpha x + N_1[x] + I \tag{5.5}$$

where $x(t) \in \Re^n$ is the state of the system at time t, and the constant vector $I \in \Re^n$ is the input.

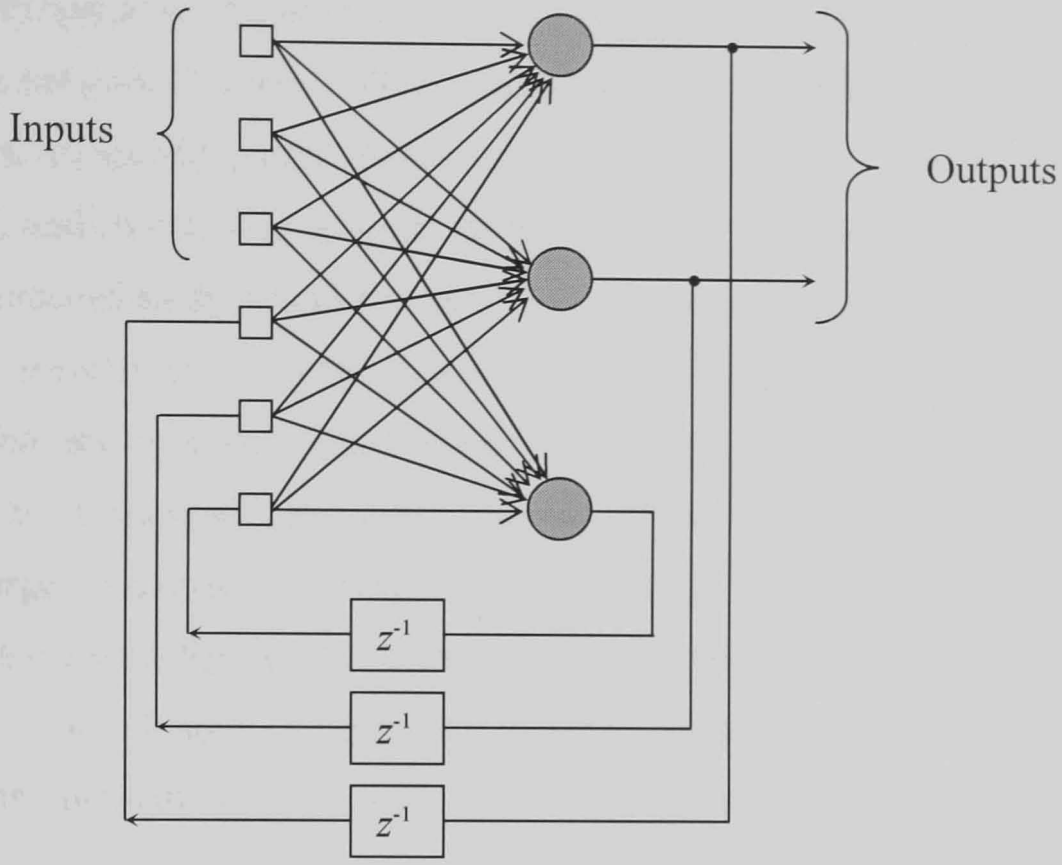

Figure (5-5): Recurrent network with hidden neurons.

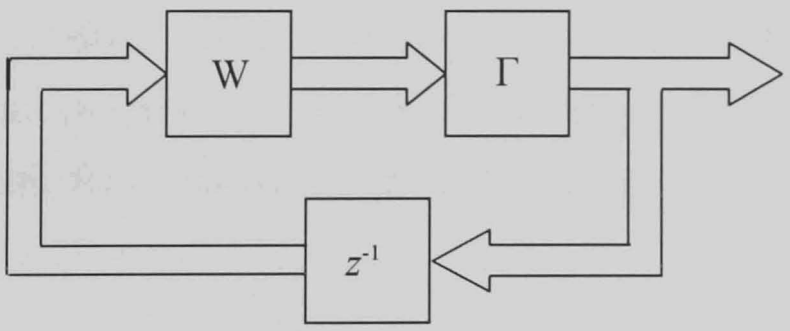

Figure (5-6): Bloc diagram representation of a typical Hopfield network.

5.3 THE BACK PROPAGATION ALGORITHM

The 'Back Propagation Algorithm' or as it is referred to in some literature "The Error Back-Propagation Algorithm", consists of two passes through the different layers of the network (Haykin, 1994): a forward pass and a backward pass. In the forward pass, an activity pattern (input vector) is applied to the sensory nodes of the network, and its effect propagates through it, layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the network are all fixed. During the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with the error-correction rule. Specifically, the actual response of the network is subtracted from a desired target response to produce an error signal. This error signal is then propagated backward through the network, against the direction of the synaptic connections. The synaptic weights are adjusted as so as to make the actual response of the network move closer to the desired response.

The corresponding architecture for back propagation learning algorithm of the architecture layout of the multilayer network of Figure (5-1) is presented in Figure (5-7). The top part of the Figure accounts for the forward phase where the layer index l extends from the input layer ($l = 0$) to the output layer ($l = L$). In Figure (5-7) we have $L = 2$, where L is referred to as the depth of the network. The lower part of the Figure accounts for the backward phase, which is referred to as a sensitivity network for l computing the local gradients in the back-propagation algorithm.

While the network of Figure (5-1) is merely an architectural layout of the back-propagation algorithm, it is found to have substantial advantages in dynamic situations where the algorithmic representation becomes cumbersome (Narendra and Parthasarathy, 1990).

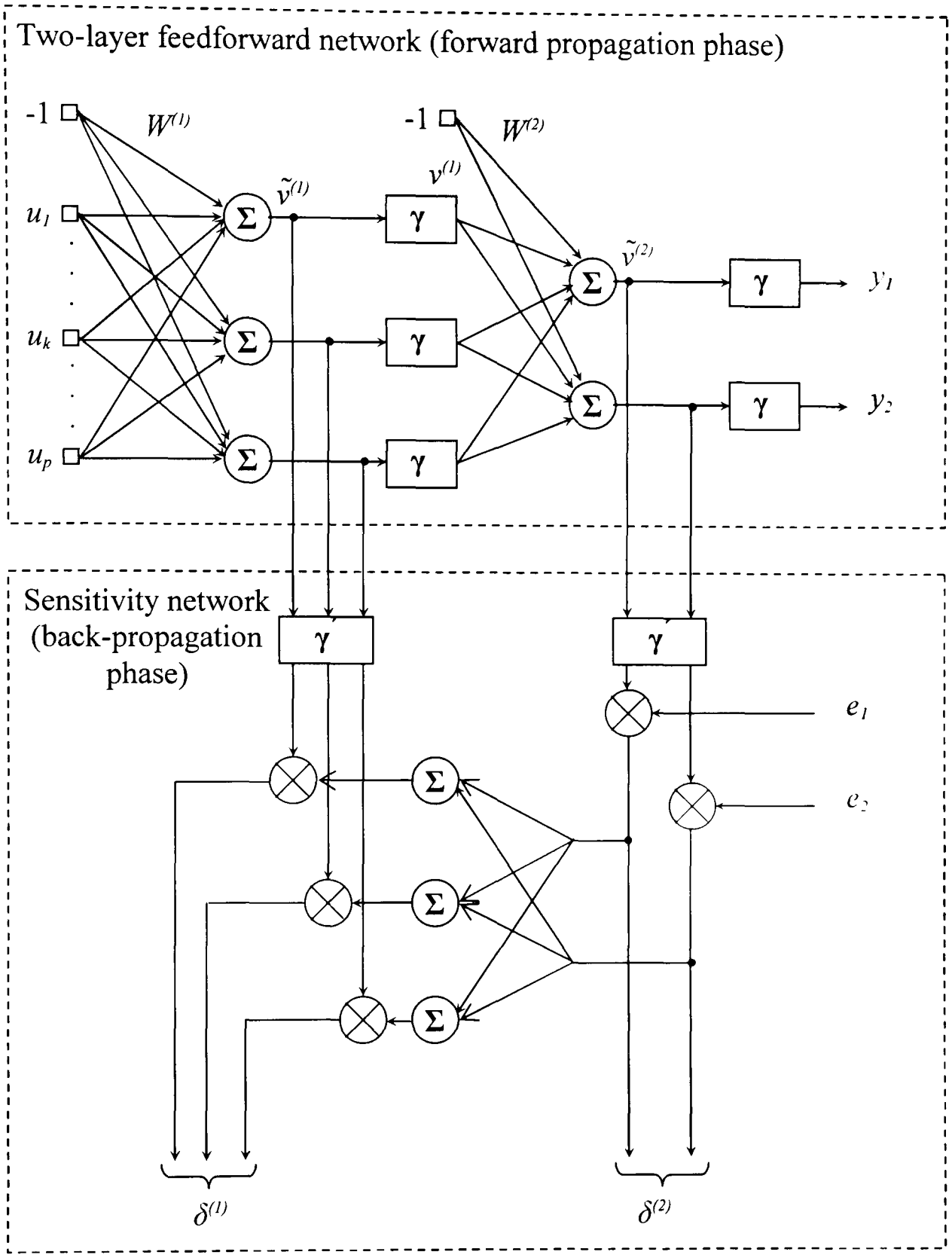

Figure (5-7): Architecture of two-layer feedforward network and its associated back-propagation signal error.

u_i	=	Element i of the external input vector.
$W^{(l)}$	=	Synaptic weight vector of a neuron in layer l.
$v(l)$	=	Vector of net internal activity levels of neurons in layer l.
$y^{(l)}$	=	Vector of function signals of neurons in layer l.
y_j	=	Element j of the network output vector.
$\delta^{(l)}$	=	Vector of local gradients of neurons in layer l.
e	=	Error vector.

It has to be mentioned that the pattern-by-pattern updating of weights is the preferred method for on-line implementation of the back-propagation algorithm. For this mode of operation, the algorithm cycles through the training data $\{[x(n),d(n)];\ n = 1, 2, ..., N\}$ as follows (Haykin, 1994).

1. *Initialisation.* Start with a reasonable network configuration, and set all the synaptic weights and threshold levels of the network to small random numbers that are uniformly distributed.

2. *Presentations of Training Examples.* Present the network with an epoch of training examples. For each example in the set ordered in some fashion, perform the following sequence of forward and backward computations under points 3 and 4, respectively.

3. *Forward Computation.* Let a training example in the epoch be denoted by $[x(n),d(n)]$, with the input vector $x(n)$ applied to the input layer of sensory nodes and the desired response vector $d(n)$ presented to the output layer of computation nodes. Compute the activation potentials and function signals of the network by proceeding forward through the network, layer by layer. The net internal activity level $\tilde{v}_j^{(l)}(n)$ for neuron j in layer l is:

$$\tilde{v}_j^{(l)}(n) = \sum_{i=0}^{p} w_{ji}^{(l)}(n) v_i^{(l-1)}(n) \tag{5.6}$$

where $v_i^{(l-1)}(n)$ is the function signal of neuron i in the previous layer l-1 at iteration n and $w_{ji}^{(l)}(n)$ is the synaptic weight of neuron j in layer l that is fed from neuron i in layer l-1. For $i = 0$, we have $\tilde{v}_0^{(l-1)}(n) = -1$ and $w_{j0}^{(l)}(n) = \theta_j^{(l)}(n)$, where $\theta_j^{(l)}(n)$ is the threshold applied to neuron j in layer l. Assuming the use of a logistic function for the sigmoidal nonlinearity, the function of neuron j in layer l is:

$$v_j^{(l)}(n) = \frac{1}{1+\exp(-\tilde{v}_j^{(l)}(n))} \tag{5.7}$$

If neuron j is in the first hidden layer (i.e., l=1), set

$$v_j^{(0)}(n) = u_j(n) \tag{5.8}$$

where $u_j(n)$ is the j^{th} element of the input vector $u(n)$. If neuron j is in the output layer (i.e., l=L), set

$$v_j^{(L)}(n) = y_j(n) \tag{5.9}$$

and the error signal is computed as:

$$e_j(n) = d_j(n) - y_j(n) \tag{5.10}$$

where $d_j(n)$ is the j^{th} element of the desired response vector $d(n)$ obtained from the real plant.

4. *Backward computation.* Compute the δ's (i.e., the local gradients) of the network by proceeding backward, layer by layer:

$\delta_j^{(L)}(n) = e_j^{(L)}(n) y_j(n)\left[1 - y_j(n)\right]$ for neuron j in output layer L

$\delta_j^{(l)}(n) = v_j^{(l)}(n)\left[1 - v_j^{(l)}(n)\right]\sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n)$ for neuron j in hidden layer l

Thus the adjustment of the synaptic weights of the network in layer l is obtained by applying the following:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1)\right] + \eta\delta_j^{(l)}(n) v_i^{(l-1)}(n) \tag{5.11}$$

where η is the learning-rate parameter and α is the momentum constant.

5. *Iteration.* Iterate the computation by presenting new epochs of training examples to the network until the free parameters of the network stabilise their values and the average squared error ζ_{av} computed over the entire training set is at a minimum or acceptably small value. The order of presentation of training examples should be randomised from epoch to epoch. The momentum and the learning-rate parameter are typically adjusted (and usually decreased) as the number of training iterations increases.

5.4 THE CONTROL PROBLEM AND THE NEURAL NETWORK SCHEME

As discussed in the preceding chapter, one of the problem areas of the ISOPE algorithm is the required estimation of real process derivatives.

In this section, a method based on neural networks for estimating real process output derivatives with respect to the set-points for the general optimisation problem of nonlinear processes is presented. The method is used within the ISOPE algorithm presented in chapter 2.

5.4.1. The Optimisation problem and the ISOPE algorithm

Although many different process optimisation techniques exist, they can be classified into two general categories: direct search and indirect or model-based optimisation methods (Garcia and Morari, 1981). In the direct method, measurements are taken directly from the real process as it is moving from one operating point to another, and a suitable optimisation technique is then applied to optimise the process performance objective function. In the indirect approach, the optimisation is performed on a model of the system instead of the physical system itself, and when found the results are applied to the real process.

As described in chapter 2, the Integrated System Optimisation and Parameter Estimation (ISOPE) technique (Roberts, 1979) has some special features from both approaches: direct and model-based. It is based on derivatives calculation provided by real process measurements to update an unfaithful or deliberately simplified model used in the model-based optimisation, thus reaching the real optimum of the process in spite of model-reality differences. However, it is established now that the need to evaluate real process derivatives at each iteration by the ISOPE algorithm in order to satisfy necessary optimality conditions is probably its major drawback. A neural networks approach has been developed and is presented here as an attempt to overcome this need. The technique reaches successful results in terms of estimating real process derivatives, which makes the ISOPE algorithm, converge to the exact optimum point without having to wait to settle for steady-state or applying repetitive disturbances on the set-points.

5.4.2. The Neural Network scheme

The technique is based on training a neural network to learn from the physical process itself. Once the training is finished, a steady-state neural network model, which imitates the static behaviour of the dynamical system, is reached. This model is used to find the system outputs to a given set-points. In this case, accurate enough model outputs and their derivatives are available to the ISOPE algorithm and prohibitive waiting times are avoided as in the traditional way when computing the output derivatives with respect to the set-points. In the case where system parameters change, the algorithm is set to adapt to it. In other words, the algorithm will retrain the neural network for a suitable time, and provide an accurate updated model of the modified system as illustrated in Figure (5-8).

It has to be mentioned that during training, switches k_2 and k_3 are closed, k_4 is open and k_1 is in position 1. This enables the algorithm to collect input/output data candidates required for the training in order to generate the identification neural network model. The states of these switches are reversed otherwise.

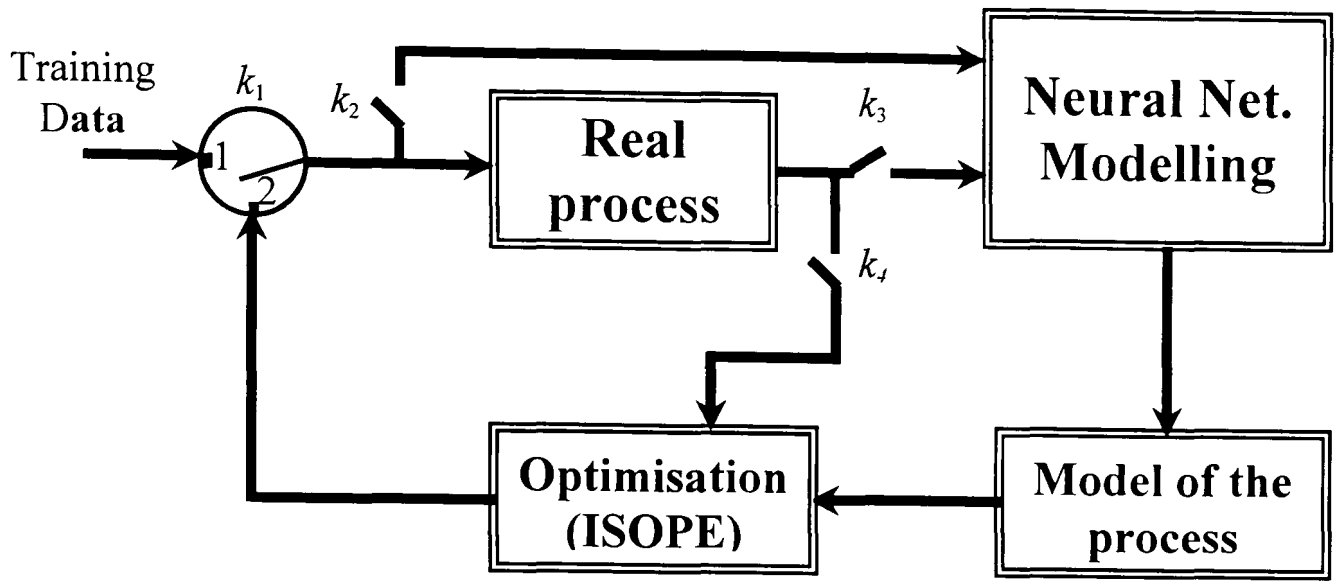

Figure (5-8): Neural Networks scheme used within the ISOPE algorithm.

5.4.3. Identification

As mentioned earlier, the ability of learning from the environment and producing accurate approximation of input-output mappings of systems make neural networks a prime candidate for use in dynamic models for the representation of non linear plants. Therefore, the identification problem consists of setting up a suitably parameterised network model and adjusting the parameters of the model to optimise a performance index based on the error between the plant and the identification model outputs. Every neural network model is composed of a series of weight vectors, which form what we call weight matrices. These matrices are updated each time the network is trained for another input/ output data sample until no further improvement is hoped. Hence, the procedure consists in adjusting the parameters of the neural network in the model using a suitable training algorithm. In our case, we chose the back-propagation algorithm presented in section 5.3 based on the error between the plant and the identification model outputs. However, other types of model networks and training algorithms can also be used. The training of the network is performed once only. This takes place at the beginning of the optimisation procedure. Once a performance goal is

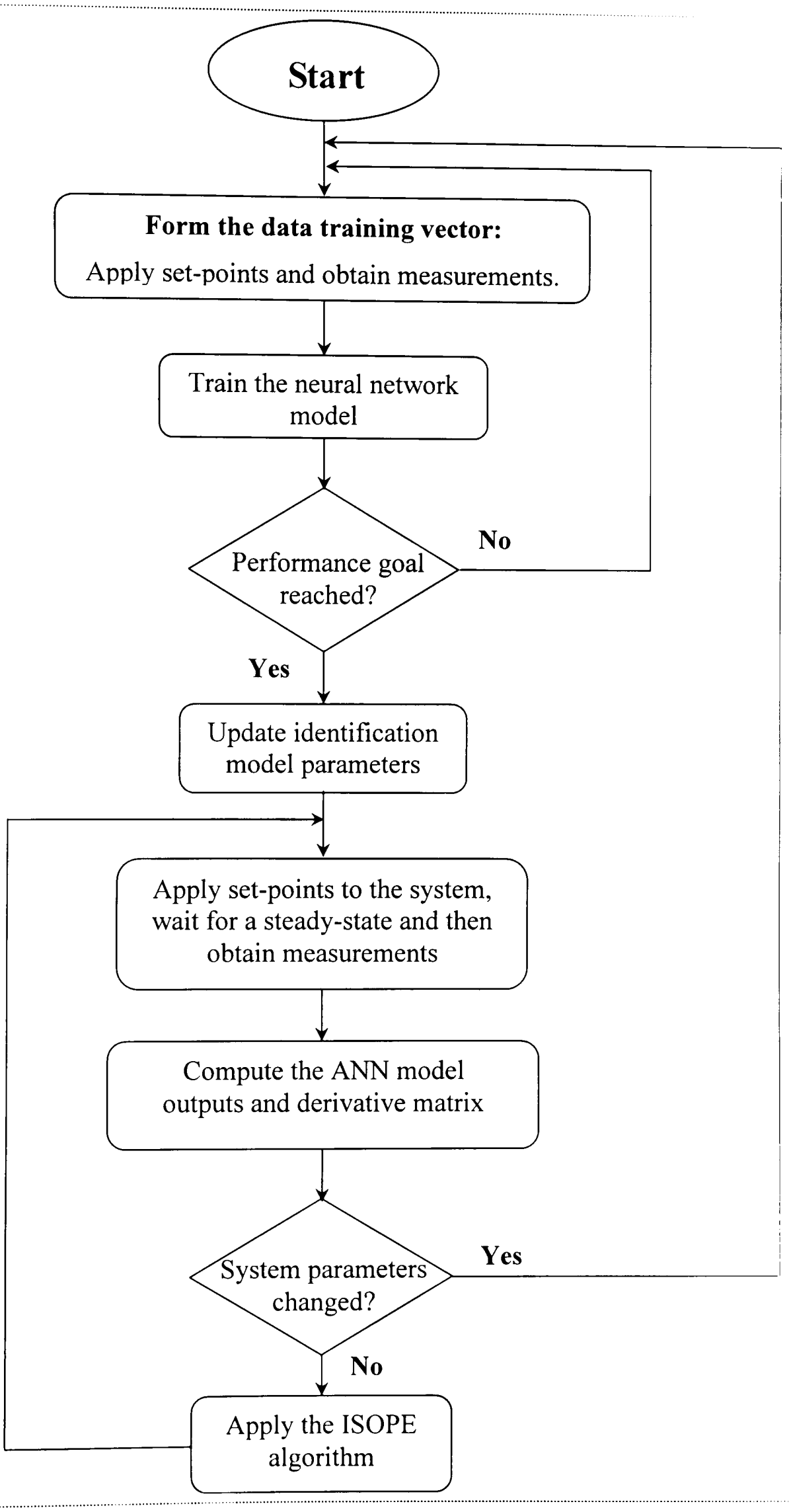

Figure (5-9): Flow chart diagram representation of the neural network scheme.

reached, training stops, the model and its parameters are saved to be used in the optimisation procedure. In case one or more of the system parameters change, the neural network model has to be retrained. In this case, a suitable time is given to the algorithm to perform identification and produce a new model. Once training is finished, the model parameters are updated, saved and passed to the optimisation routine (Figure 5-9). In practice, to cover against system parameter changes, retraining may be carried out at periodic intervals.

5.5 SIMULATION CASE STUDIES

In order to evaluate the performance of the neural network scheme presented in this chapter, two sets of simulations were carried out using the systems presented in chapter 3. The first set uses a simple single input-single output non linear system. This set of simulations was used to assess the accuracy, and adaptability of the neural network scheme. While the second set was carried out on a two Continuous Stirred Tank Reactors (CSTR's) connected in cascade, and was employed with the aim of demonstrating the characteristics of the same neural network scheme on a higher scale, when incorporated within the ISOPE algorithm. A comparison is made between this method and an older method presented in the previous chapter.

5.5.1. Case study 1

Consider the single input-single output non linear plant driven by the following input/ output relationship:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{5.12}$$

where y is the output and u is the input.

A nonlinear feedforward back propagation network model of the above system is created in order to imitate the behaviour of the actual system.

This model has three nodes in the input layer, one hidden layer with five nodes, and one output node. This choice of node numbers was entirely practical, and depended on which option resulted in the best approximation of the real system behaviour by the identification model. The structure of the input vector to the model was therefore chosen to be:

$$X(k) = [u(k), u(k-1), y(k)] \tag{5.13}$$

where $u(k)$ and $u(k-1)$ are the present and previous inputs to the real system respectively, $y(k)$ is the present output of the real system resulting from the application of $u(k)$ to its input.

At first, the network was trained for 10000 different input/output data samples measured from the real system. The input was a random input whose amplitude was uniformly distributed in the interval [-2, 2]. This results in the neural network model approximates the behaviour of this plant over this interval only (which is enough for our application). This in turn results in the variation of y over the interval [-10, 10]. After training, a model of the system is developed. The internal architecture of this model is not known, however its behaviour follows exactly that of the real system.

The results of the simulations for various input signals are shown in Figures (5-10) to (5-12).

Figure (5-10) shows the trajectory taken by the real plant and neural network identification model output signals when applied a sinusoidal function of the

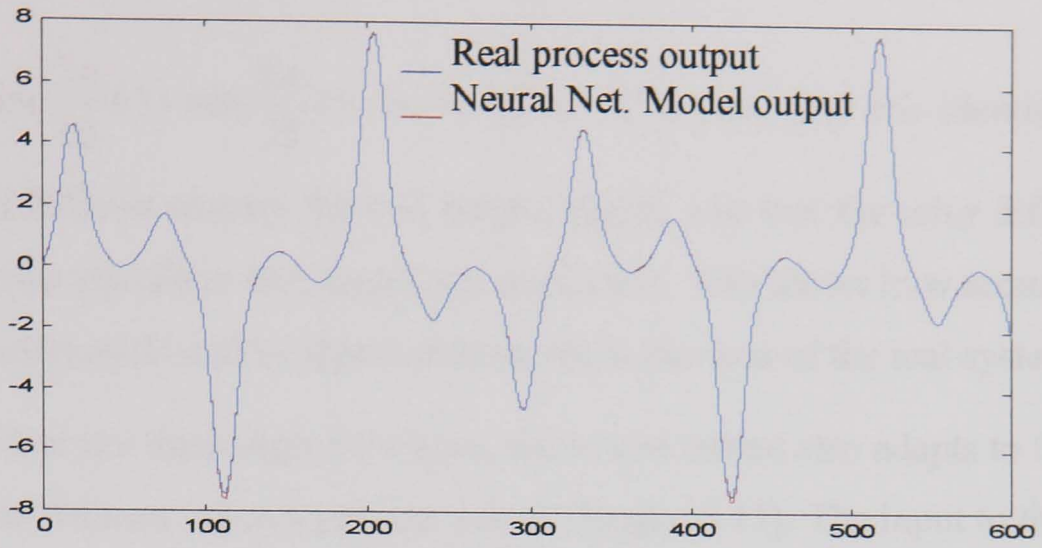

Figure (5-10): Outputs of the real plant and the neural net. identification model.

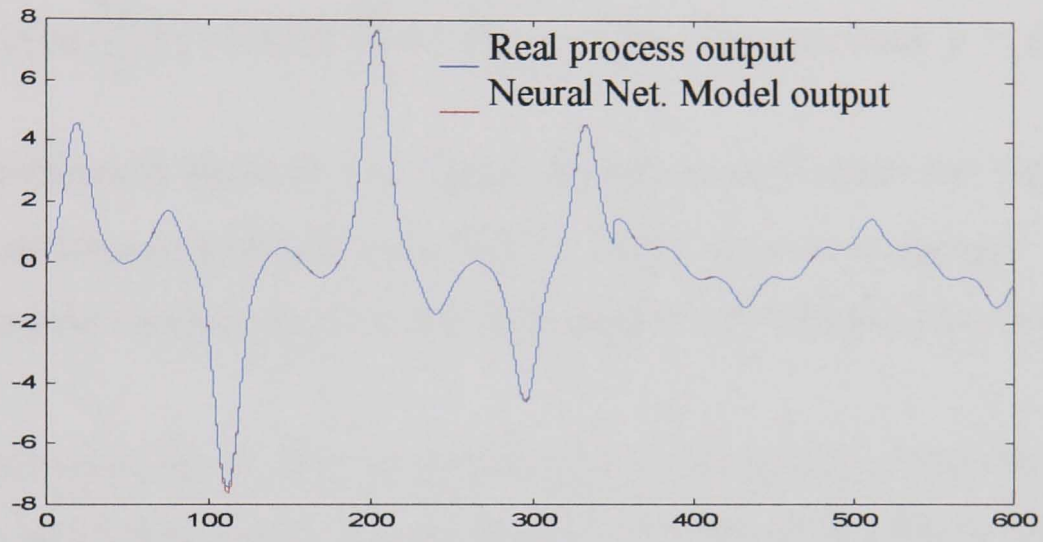

Figure (5-11): Outputs of the plant and the neural net. identification model when input changes.

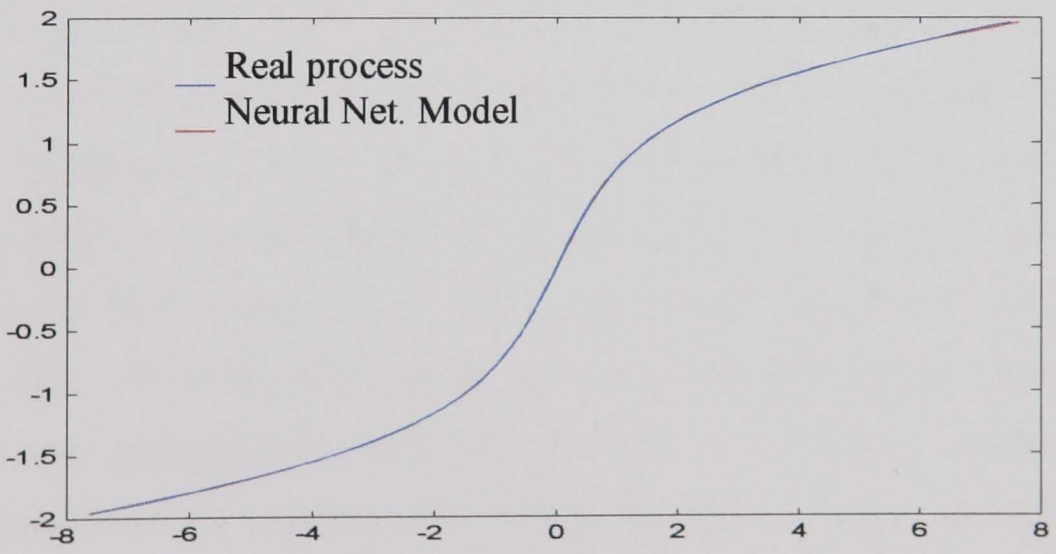

Figure (5-12): Plots of $y = f(u)$ and $\hat{y} = N[u]$ for real process and identification model.

form: u(k)=$\sin(\frac{2\pi}{10}k)+\sin(\frac{2\pi}{25}k)$ in its input. It is clear that the identification model output follows exactly the real output signal, and that the error difference between the two signals is very small and negligible. This shows how accurate the neural network model used in approximating the behaviour of the real system.

In the case when the input signal changes, the model output also adapts to the new set of data and follows the real process output (Figure 5-11). The input to the plant and identification model was given by: u(k)=$\sin(\frac{2\pi}{10}k)+\sin(\frac{2\pi}{25}k)$ for $k \le 350$ and u(k)=$0.2\sin(\frac{2\pi}{5}k)+0.8\sin(\frac{2\pi}{25}k)$ for $k > 350$. The functions $y = f(u)$ and $\hat{y} = N[u]$ are shown in figure (5-12). Again, the difference between the two output signals can not be distinguished, even after the input signal was changed. This is in total agreement with the results found in Narendra and Parthasarathy, (1990).

A second set of simulations this time employing a Radial Basis Function (RBF) network, has been performed on the same system. The results are shown in figures (5-13) to (5-15). From the figures, it is clear that the model output signal trajectory matches the real system output signal trajectory with great precision. Compared to the results found with the back propagation network, this type of network gives more accurate approximation of the system mapping. This is to show the fine ability of an RBF network to approximate nonlinear functions. In fact, RBF networks are differentiated from back propagation networks by the fact they learn much faster especially if the number of input variables is not too high like in our case. However, the required number of neurons in the single hidden layer increases geometrically with the number of the input variables. This becomes prohibitive for systems with a large numbers of input variables. RBF networks are also known to work best when many training vectors are available, which means when more time is spent collecting the input-output candidates for training the network.

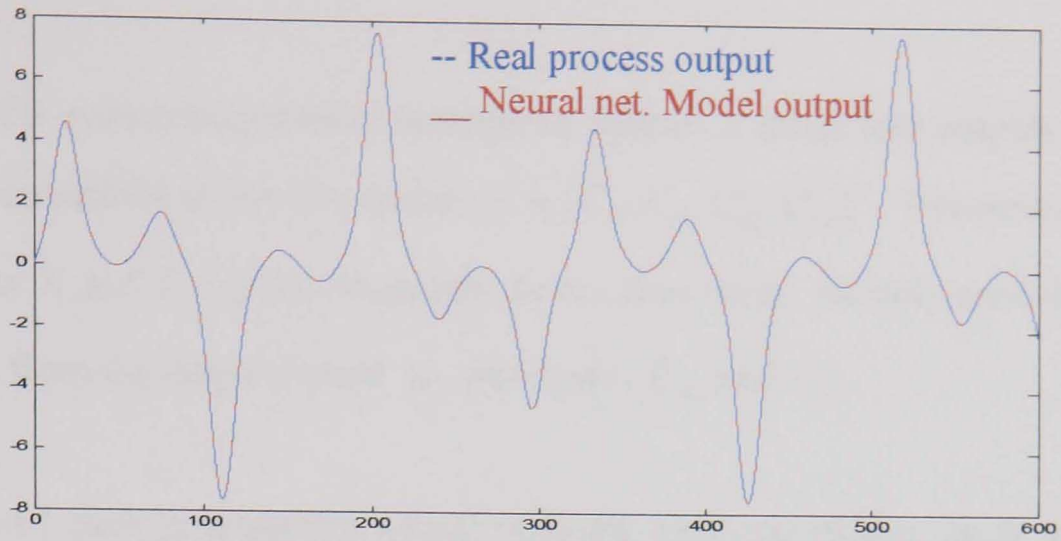

Figure (5-13): Outputs of the real plant and the RBF network identification model.

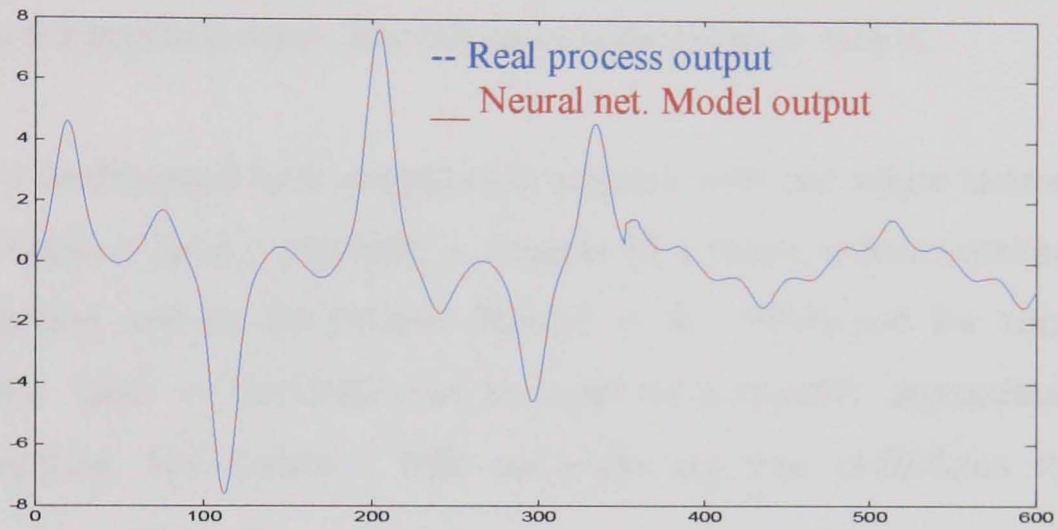

Figure (5-14): Outputs of the plant and the RBF network identification model when input changes.

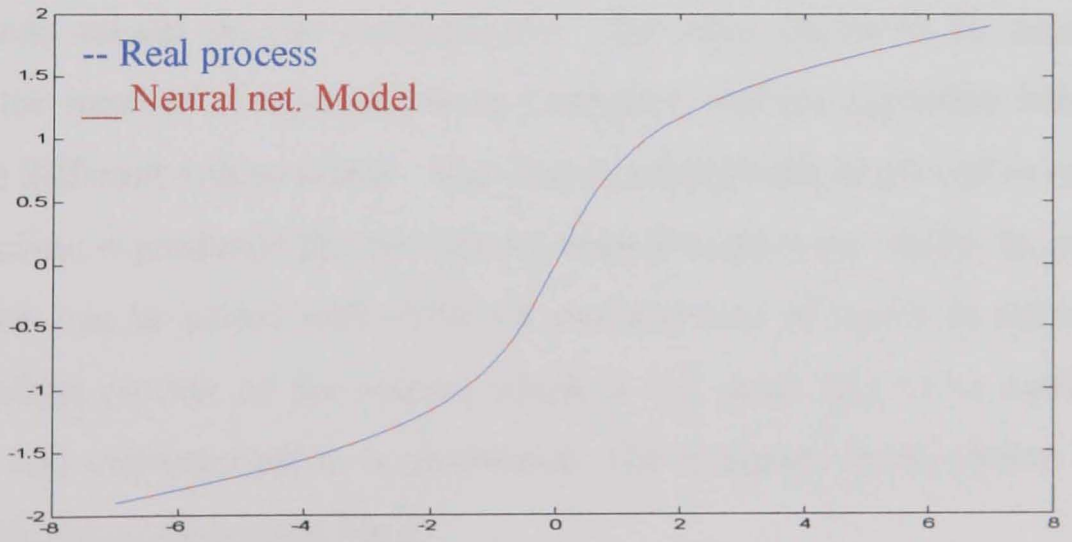

Figure (5-15): Plots of $y = f(u)$ and $\hat{y} = N[u]$ for real process and RBF network identification model.

5.5.2. Case study 2

The two CSTR system is described in detail in chapter 3. It has four outputs which are the concentrations in the two tanks: $y_* = (C_{a1}, C_{b1}, C_{a2}, C_{b2})^T$. Temperatures in the two tanks T_1 and T_2 are the set-points. In our case study, we only consider two components from the output vector y_*, which are: C_{a2} and C_{b2}.

A feedforward back propagation neural network with one hidden layer and six output neurons or units was used in simulation. In a feedforward network, the first layer has weights coming from the input. Each subsequent layer has a weight coming from the previous layer. The last layer is the network output.

In our case, a feedforward back propagation network with one single hidden layer was chosen because such a structure is capable of accurate approximation of an arbitrary function and its derivatives (Hornik et al., 1990) and for simplicity reasons. Other types of networks can be used to accurately approximate the system's mapping. For instance, RBF networks are best candidates for this purpose.

It has to be mentioned that the choice of number of neurons in the single hidden layer depends totally on the experimenter. The main factor to be taken into account is the number of input and output samples, and the algorithm behaviour towards the different values tested. The choice adopted above proved to be more suitable because it produced the best results among those many tested. In practice, the algorithm can be tested with different combinations of layers in simulations based on robust models of the system which is the usual step to be carried out before any real implementation is performed. The optimum (best) choice is then applied on the physical system itself.

The training of the network was performed using the back-propagation algorithm. Training takes place only once (at the beginning, unless the system's parameter change), when a suitable performance goal is reached, training is stopped, model parameters are updated and then passed to the ISOPE algorithm.

At first, the network was trained for 100000 different input/output data samples measured from the real system. Since the real system is a BIBO (Bounded Input Bounded Output) system, the training input set used to train the neural network was chosen to be a random input in the interval [295, 320]. This is due to the fact that we want to cover all the input interval range that the real system inequality constraints satisfy. These are given by the set-points of temperature controllers upper and lower bounds: $300 \le T_1 \le 312\ K, 300 \le T_2 \le 312\ K$.

After training, a model of the system is developed. The internal architecture of this model is not known, but as with the first case study, its behaviour follows that of the real system. During the simulations, sufficient time was allowed for the system to settle down to a new steady-state condition before measurements were taken or new set-points were applied.

The optimisation was performed on a linear objective function of the measured variable C_{b2} and reflects the desire of maximising the amount of component B in tank 2.

$$L(y,v) = -C_{b2} \tag{5.14}$$

As mentioned in the previous chapter, this objective function is linear of the measured variable C_{b2}, but is a nonlinear function of the manipulated variables T_1 and T_2. This is due to the nonlinearity of the system equations (chapter 3).

A SIMULINK model of the process was created to enable periodic calls to the ISOPE algorithm saved in a MATLAB file. The simulations were started from the same starting point which is the initial steady-state condition given by: $T_1 = 307K$ and $T_2 = 302K$, which yields the following steady-state outputs: C_{a2}=0.041361 $[kmol/m^3]$ and C_{b2}=0.058638 $[kmol/m^3]$. The relaxation gain matrix was chosen to be: $K = 0.1I$. The choice of the relaxation gain is crucial and was adopted after a number of trials, and is the most appropriate among several tested. This gain allows the system to remain stable by generating small but suitable changes to the set-points as the system moves towards the optimum as quickly as possible.

5.5.2.1 Simulation results:

The results of the various simulations applied on this system are shown in Table (5.1) and Figures (5-16) to (5-17).

In table (5.1), a comparison of the final set-points and derivative matrix values is given. The comparison was set between the neural network scheme presented in this chapter and the Finite Differences Approximation method (FDAM) presented in chapter 4 for approximating process derivatives. It is clear that the new method based on neural network model performs a good estimation of the real process derivatives; moreover, it is fast and does not need a waiting period for settling down neither it needs major disturbances on the real system inputs. This is demonstrated by the small number of set-point changes the neural network method takes to converge compared to that of the FDAM. It is known that the FDAM take at least (n+1) times set-point changes more as it attempts to compute the process derivatives (n being the number of set-points).

Figure (5-16) shows the trajectory taken by the real system outputs and set-points, while attempting to find the optimum operating point when using the Finite Differences Approximation method (FDAM) (chapter 4) to compute the output

derivative matrix with respect to the set-points. From the figure, we see that time is mostly consumed when the derivatives are computed by applying small perturbations and waiting for the system to settle down before taking measurements. Figure (5-17) shows the trajectories taken by the real system outputs, identification model outputs and set-points. Unmistakably the identification model outputs follow exactly those of the physical system with a big precision driving it from the initial steady-state position given by $C_{a2}(0)=0.04136\ [kmol/m^3]$, $C_{b2}(0)=0.05864\ [kmol/m^3]$ to the final converged solution ($C_{a2}=0.0275\ [kmol/m^3], C_{b2}=0.0725\ [kmol/m^3]$). It is clear that in this case, time is not wasted waiting for the system to settle down for a steady-state position to compute the derivative matrix as this latter is found using the identification (neural network) model instead. This model provides the model outputs that match the real process outputs for a given set-point as well as an approximation of the real process derivatives. This leads to the optimum operating point being reached quickly without applying any perturbations on the system.

It has to be said that for large scale systems, the training takes much more time. This is due to the fact that almost all the situations that the system can be subjected to have to be considered. For instance, in case of no parameter change in the system, we can say that the algorithm performs well and is more advantageous than using earlier methods like the FDAM, as it allows us to gain the time taken to compute the real process derivatives with respect to the set-points which results in a faster convergence. It also provides an accurate estimation of the process outputs if the training was performed well. Moreover, it avoids applying unnecessary perturbations to the actual system when on-line optimisation (ISOPE) is performed.

Table (5.1): Derivatives Comparison table.

METHOD	OPTIMUM SET-POINTS	ESTIMATES OF THE DERIVATIVES AT THE OPTIMUM		NUMBER OF SET-POINTS CHANGES
ISOPE with FDAM	T_1=312 K T_2=310.2K	[-0.0094 0.0094	-0.0071 0.0071]	22
ISOPE with Neural Network scheme	T_1=312 K T_2=310.2K	[-0.0094 0.0094	-0.0071 0.0071]	7

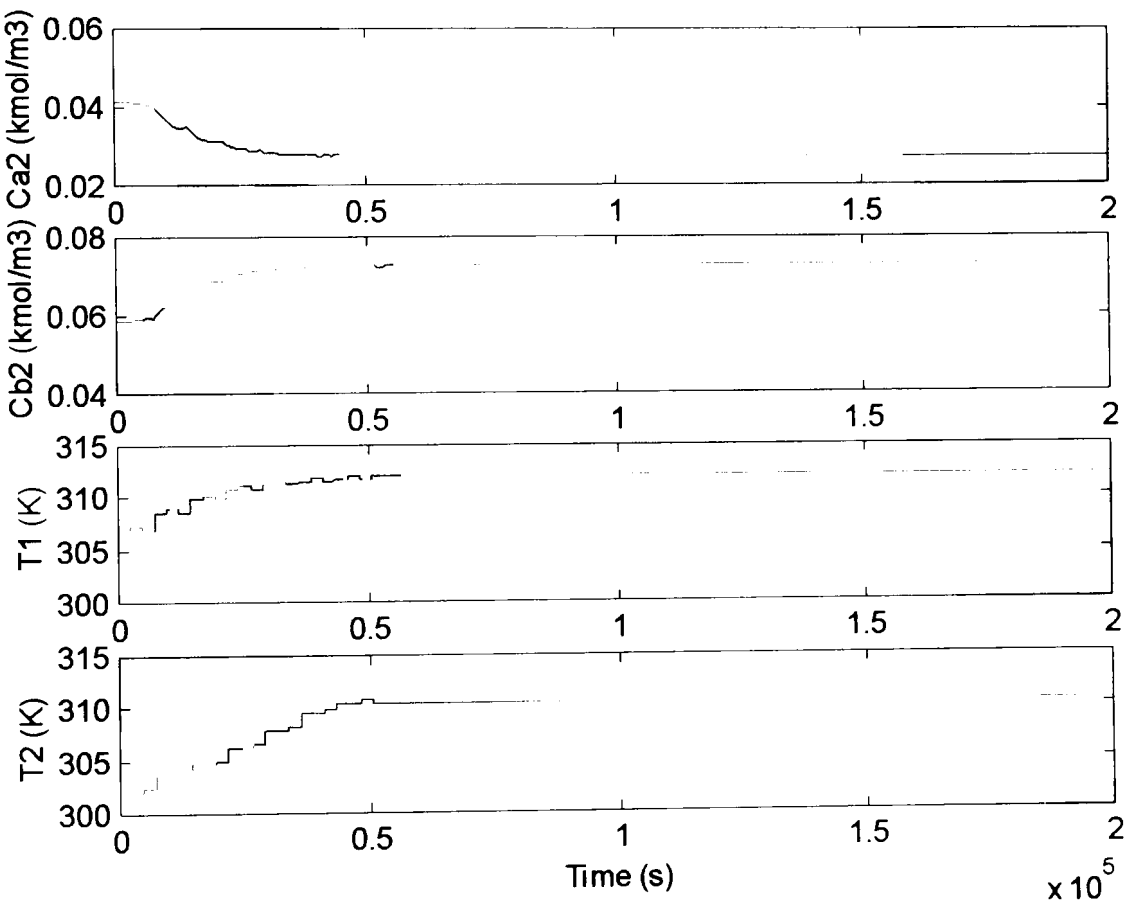

Figure (5-16): Set-points changes and outputs trajectories for the FDAM method.

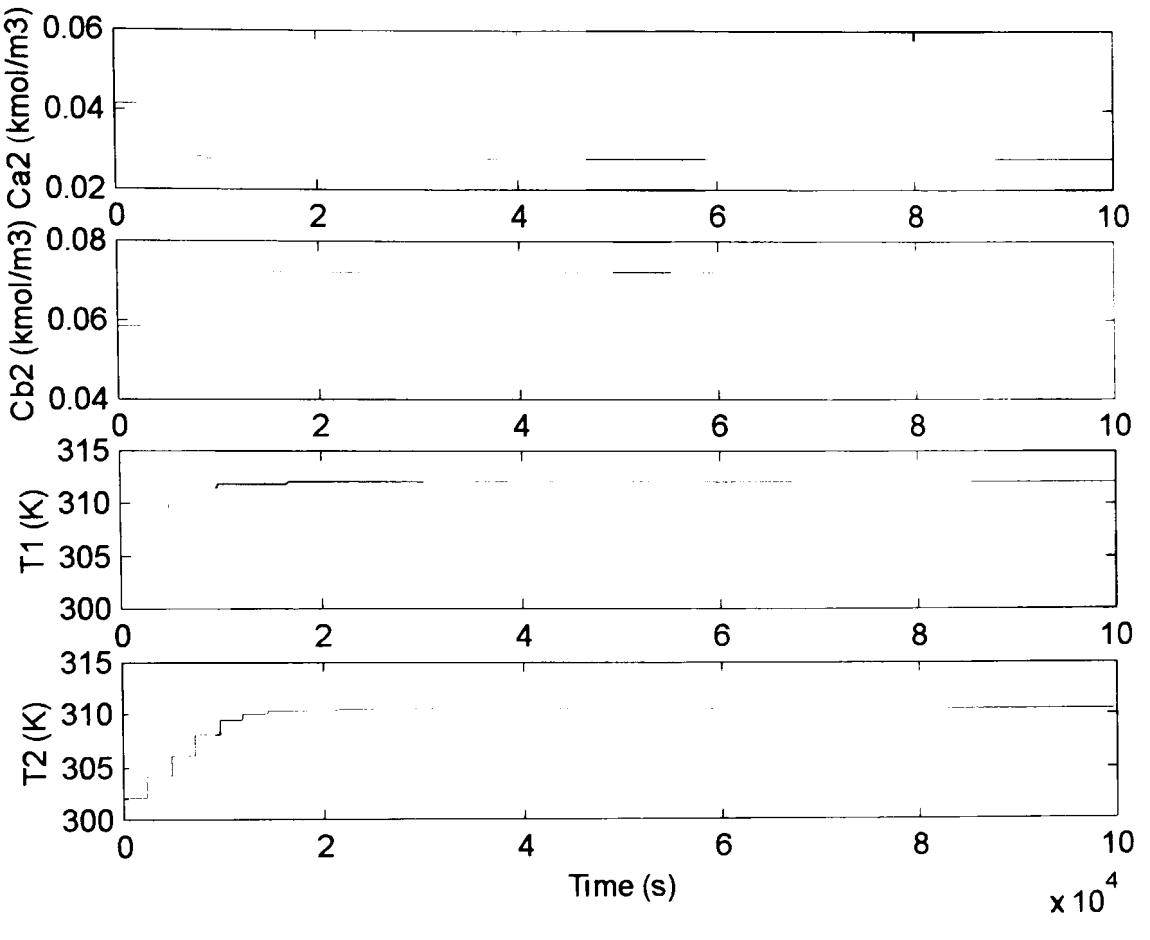

Figure (5-17): Set-points changes and outputs trajectories for the neural network scheme.

The simulation case studies presented in this chapter do not include the case where the system was subjected to an additive noise or gross error. This issue is addressed in chapter 7 where data reconciliation and gross error detection techniques will be employed.

5.6 SUMMARY

A Neural Network technique for estimation of real process output derivatives with respect to the set-points for general nonlinear systems to be used within the ISOPE algorithm has been presented, implemented and applied under simulation on two examples of systems. The first system was a simple single input-single output non linear plant, while the second was a cascade process consisting of two Continuous Stirred Tank Reactors. The method converged to the correct optimum

point even when a change in the real process parameters occurred. The only drawback of the method encounters is that as the process parameters change, the algorithm needs some time to collect data and retrain the neural network to adapt to the new changes. In practice, to cover against system parameter changes, retraining may be carried out at periodic intervals. However, since the generated neural network identification model is a steady-state model, waiting periods for calculating derivatives is avoided, hence, the ISOPE algorithm converges faster. Simulations that include noise contaminated processes are treated in chapter 7 together with the data reconciliation and gross error detection techniques for detecting, locating, estimating and eliminating random and gross errors.

CHAPTER 6

DATA RECONCILIATION AND GROSS ERROR DETECTION

6.1 INTRODUCTION

Process data and measurements are the basis for monitoring, evaluating process performance, and for process models that are used to optimise and control processes. The reliability of measured data is of great importance. However, measurements are usually subject to random errors and/or gross errors, and also not all variables are available for measurement because of cost consideration or technical unfeasibility. The presence of such errors causes the violation of the mass, energy and other physical constraints of the process. When information gained from flawed measurement is used for state estimation and process control, the state of the system is misrepresented and the resulting control performance may be poor and can lead to suboptimal and even unsafe process operation. The objective of data reconciliation and gross error detection techniques is to correct the measured variables by removing both the random and gross errors from the data set, and to estimate the values of the unmeasured variables, so that we obtain an estimate of the true state of the plant. Hence reaching better results when applied in optimisation and control. In other words, using data reconciliation and gross error detection within the ISOPE algorithm to remove errors from measured variables, can improve the parameter estimation, and enhance the quality of the derivative estimation resulting in more efficient operation of the system.

In this chapter, static data reconciliation and gross error detection methods for the estimation of random and gross errors respectively are presented. These are used in case data measurements are corrupted with random and/ or gross errors in order

to detect and eliminate them. The performance of these methods is demonstrated in a simulation case study. The case study uses a two Continuous Stirred Tank Reactors (CSTR) connected in series system. This system was described in chapter 3. In the simulations, biases and random errors were added to the measurements, to investigate whether static data reconciliation and gross error detection are able to detect, estimate and eliminate them. Comparisons are made between simulation results when measurements are affected by noise and/ or bias, with and without data reconciliation and gross error detection. Section 6.5.4 of this Chapter provides a discussion of the results, and Section 6.6 summarises it.

6.2 DATA RECONCILIATION

Data reconciliation, also called validation, allows state estimation and measurement correction problems to be addressed in a global way. The aim of validation is to remove errors from available measurements, and to yield consistent and complete estimates of all the process state variables as well as unmeasured process parameters. Data reconciliation is based on measurement redundancy (Arora et al., 2003). A *redundant* measurement is a measurement which the value can be calculated based on other measurements. There are two types of measurement redundancy (Liebman et al., 1992): *spatial* and *temporal*. A measurement is said to be *spatially redundant* if there are more than enough data to completely define the process model at any instant in time, in other words the system is overdetermined. Whereas a *temporally redundant* measurement is defined as a measurement which past values are available and can be used for estimation purposes. Data reconciliation uses measurement redundancy that arise from the fact that at least some information about the process is known and relates the measurements to each other (Liebman et al., 1992). These are used to correct measurements and convert them into accurate and reliable knowledge. As a result, the reconciled values exhibit a lower variance compared to original raw measurements; this allows process operation closer to limits (when this results in improved economy).

One main aspect of the data reconciliation problem is incomplete measurement sets. Usually, not all process variables are available (or convenient) for measurement because of cost considerations or technical unfeasibility. Therefore. *Coaptation* (Mah et al., 1976), which is the process of estimating the value of some unmeasured variables through mass, energy and component balances, is used.

In general, the data reconciliation problem is stated as follows (Bagajewicz, 2003):

Given a set of measurement values of a subset state variables, it is desired to obtain the best estimators of these measured state variables and as many unmeasured variables as possible.

6.2.1. Types of errors

Raw process data is subject to two types of errors, random errors and gross errors. Gross errors are caused by non-random events such as process leaks, biases in instrument measurements, malfunction of instruments, inadequate accounting of departures from steady-state operations and/or inaccurate process models. The random errors come from the randomness of measurements, such a process noise, and they are normally distributed.

6.2.2. Measurement data processing

Figure (6-1) illustrates the concept of the three basic steps for processing measurement data (Liebman et al., 1992).

Step 1 concerns variable classification. It embroils organising variables into specific categories. Multiple algorithms have been designed to date to deal with this issue (Stanley and Mah, 1981, Crowe, 1986, and Mah, 1990). Variables are classified as observable or unobservable and redundant or undetermined. A variable is said to be unobservable if it is possible to make a feasible change (without violating the conservation constraints) for a variable without being detected by the instruments. In other words, a measured variable is always

observable, but an unmeasured variable may or may not be observable (Figure 6-2). The test for redundancy is as stated above.

The second step concerns gross error detection and identification. In this step, any gross error is detected, identified, its value estimated and subsequently eliminated. Many gross error detection techniques exist and have been developed to date. This subject will be addressed in detail in section 6.3.

The third and final step is the coaptation and data reconciliation step which is as described earlier. The mathematical formulation of the steady-state data reconciliation problem is given in section 6.2.6.

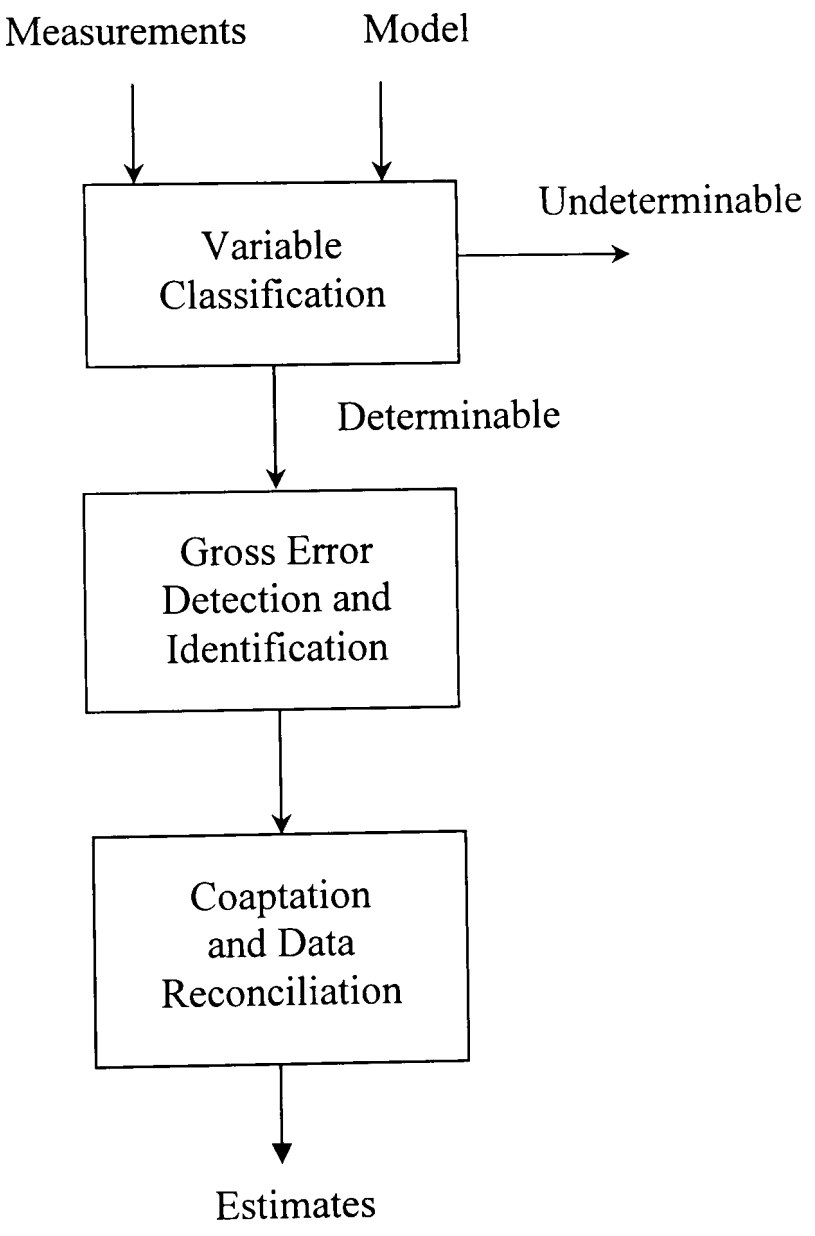

Figure (6-1): Three steps for processing measurement data (Liebman et al., 1992).

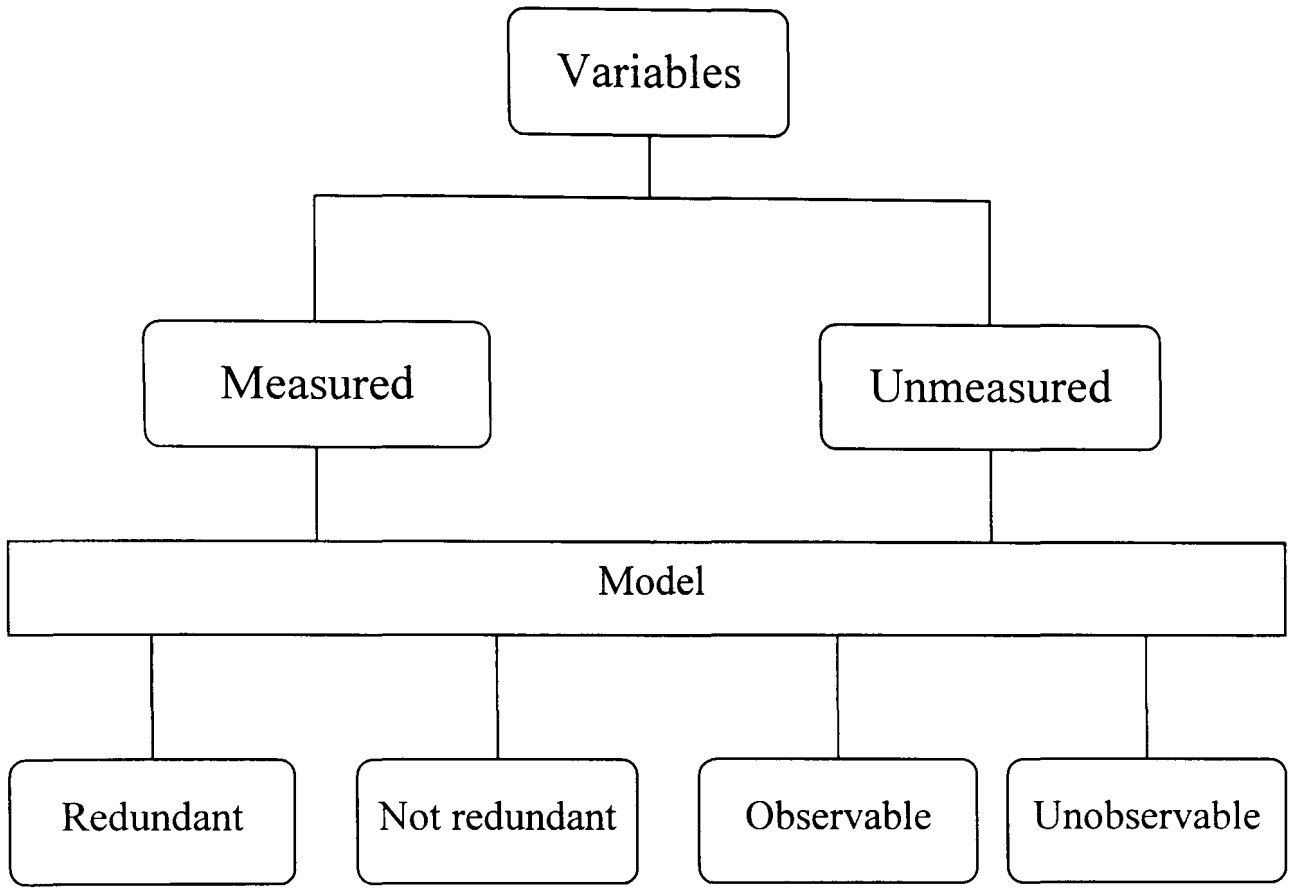

Figure 6-2: Variable classification

The common assumptions underlying steady-state data reconciliation methods and software implementations are as follows (Kim et al., 1997):

1. A stationary process: The system is at steady-state.
2. Measurement error is Gaussian with zero mean value, and known variances (usually diagonal covariance is assumed). This signifies that the measurements are not affected by any gross error.

6.2.3. History

Data reconciliation has been used for several years as means of obtaining accurate and reliable data in process plants. The earliest work reported in the literature is probably that of Kuehn and Davison (1961). The authors presented a formulation of the data reconciliation problem and a method based on Lagrange Multipliers in order to solve the steady-state data reconciliation problem. In dynamic cases, Gelb (1974) used Kalman Filtering successfully to recursively smooth measurement data and estimate parameters. However, both concepts (steady-state and dynamic)

were developed for linear systems only. Therefore, modifications had to be made in order to handle nonlinear systems. Knepper and Gorman (1980) proposed a method based on successive linearisation of the system's nonlinear equations (constraints). The method was based on the application of the analytical solution for the linearly constrained data reconciliation problem. In a comparison study, Jang et al. (1986) came to a conclusion that better results in terms of response to changes in parameters and robustness in the presence of modelling errors and strong nonlinearities can be achieved when nonlinear programming is used. So much so, Liebman and Edgar (1988) illustrated that nonlinear programming gives improved reconciliation estimates compared to successive linearisation. Crowe et al. (1983) proposed a method based on matrix projection to reconcile process flows. In this method, a Chi-square test based on the inverse of the reduced Hessian was used. This method was later reviewed by Crowe (1986). Narasimhan and Mah (1987) introduced their Generalised Likelihood Ratio (GLR) method for gross error detection. This method based on the likelihood ratio statistical test is capable of detecting, identifying, estimating and eliminating a wide variety of gross errors. Also, a strategy for identifying multiple gross errors namely the Serial Compensation strategy was proposed in the same paper. Based on the Chi-square test, a linear combination technique that identifies equivalent gross errors was derived by Rollins et al. (1996). A review of important results for gross error detection is available in Crowe (1996), for steady-state systems and Albuquerque and Kramer (1995) for dynamic systems. Based on a bivariate distribution function constructed using the maximum likelihood principle, Tjoa and Biegler (1991) presented a method for combined data reconciliation and gross error detection applied to steady-state processes.

One of the problems researchers were faced with is the detection of the steady-state. The fact that processes are never in a steady-state, which is the assumption that all data reconciliation algorithms are based on, means not only random errors but also process variations are averaged with good measurements. This issue was addressed in many publications (Narasimhan, 1984, Holly et al., 1989, and Abu-el-zeet et al., 2000).

Most recently, Narasimhan and Jordache (2000) published a book, which provides a systematic and comprehensive treatment of data reconciliation and gross error detection techniques.

The literature is rich with excellent review papers: Mah (1982); Tamhane and Mah (1985); Mah (1990); Madron (1992); and Crowe (1996).

6.2.4. Benefits

The benefits derived from data reconciliation in chemical and process industry are many. They include (Arora et al., 2003):

- Improvement of measurement layout.
- Fewer routine analyses.
- Reduced frequency of sensor calibration (only faulty sensors need to be calibrated).
- Removal of systematic measurement errors.
- Systematic improvement of process data.
- A clear picture of plant operating condition.
- Reduced measurement noise for key variables.

Moreover, monitoring through data reconciliation leads to early detection of sensor deviation and equipment performance degradation, actual plant balances for accounting and performance follow-up, safe operation closer to the process limits and improved quality and performance at the process level.

6.2.5. Recent developments and software packages

People both from academia and industry, are being attracted to the area of data reconciliation. Hundreds of articles have been published, few books have been written and a couple of industrial software packages exist at the present moment (Bagajewicz, 2003).

Recent developments in the field aim at combining online data acquisition with data reconciliation, where reconciled data are displayed in control rooms in

parallel with raw measurements. Departure between reconciled and measured data can trigger alarms and analysis of time variation of those corrections can draw attention to drifting sensors that need recalibration (Arora et al., 2003). Amongst the software packages developed to date, we note: PRECISE, from Ok-Solutions, VALI, which is a data reconciliation and data validation software available from BELSIM s.a, we also note RAGE, which is a software for data reconciliation and gross error detection developed by the Chemical Engineering Department (IIT Madras).

The next subsection presents the mathematical structure of the steady-state data reconciliation problem. A comprehensive case study that illustrates the use of this structure is provided in Section 6.5. The case study also motivates the treatment of gross errors using gross error detection techniques, and uses the two CSTR system presented in chapter 3.

6.2.6. Mathematical structure of the data reconciliation problem

The underlying idea in Data Reconciliation is to formulate the process model as a set of constraints. Here, this will involve mass and energy balance and some constitutive equations. All measurements are corrected in such a way that reconciled values do not violate the constraints. Corrections are minimised in the least-squares sense, and the measurement accuracy is taken into account by using the measurement variance-covariance matrix as a weight for the measurement corrections.

The data reconciliation problem is formulated from data collected at sampling instants i. If we assume these data sets to be independent of each other, and that no gross error is present, and the process is at steady-state, the measurement vector (y_m) can be written as:

$$y_m = y_{true} + \varepsilon \tag{6.1}$$

where y_{true} is the vector of the true values of the variables. ε is a vector of random measurement errors. These errors are assumed to be normally distributed with zero mean, and a covariance matrix V.

The general data reconciliation problem can be stated as the following nonlinear programming (NLP) problem:

$$\begin{aligned} &\underset{y_{true}}{\text{Min}} \quad F(y_m, y_{true}) \\ &\text{Subject to: } h(y_{true}) = 0 \end{aligned} \tag{6.2}$$

where $F(y_m, y_{true})$ is some objective function that depends on a difference between the measurements and their reconciled values, and h is a set of algebraic equality constraint equations. In (6.2), we assume that all variables are identified with a particular data set and the problem is an *errors in variables measured* (EVM) problem. On the other hand, one can also have multiple measurements of each variable, such as in problems with *moving horizons*.

Problem (6.2) is usually formed with objective functions derived from the maximum likelihood (Arora et al., 2003). Here a number of specialisations can be made for data reconciliation. In particular, if we assume data snapshots i are independent and all data have errors from similar sources, we can simplify the error structure. For most applications the objective function in (6.2) is simply a Weighted Least-Squares (WLS):

$$F(y_m, y_{true}) = \frac{1}{2} (y_m - y_{true})^T V^{-1} (y_m - y_{true}) \tag{6.3}$$

V, the variance-covariance matrix which each element V_{ii} is σ_i^2, is assumed to be the same for all data sets. In addition, if we assume that the elements of each data vector are independent of each other, then the off diagonal elements of the variance-covariance matrix can be assumed to be zero. In other words (6.3) can be written as:

$$F(y_m, y_{true}) = \frac{1}{2} \sum_j ((y_{mij} - y_{trueij})/\sigma_j)^2 \tag{6.4}$$

Hence the steady-state data reconciliation problem (6.2) can be simplified to:

$$\min \left\{ \frac{1}{2} (y_m - y_{true})^T V^{-1} (y_m - y_{true}) \right\} \quad \text{subject to: } h(y_{true}) = 0 \tag{6.5}$$

Once formulated, problem (6.5) can be solved with a number of efficient approaches.

6.2.6.1 Nonlinear Programming (NLP)

For instance, any Nonlinear Programming (NLP) solver can solve problem (6.5). Often Sequential Quadratic Programming (SQP) is the method of choice as it requires the fewest function evaluations. In this case, it is simple to add upper and lower bounds on the measured variables, so problem (6.5) can be more generalised. These upper and lower bounds are considered as an extra inequality constraint, and can be formulated as:

$$y_{true,l,i} \le y_{true,i} \le y_{true,u,i} \qquad \forall i, \tag{6.6}$$

where $y_{true,l,i}$ and $y_{true,u,i}$ refer to the lower and upper constraints on variable $y_{true,i}$.

6.2.6.2 Quadratic Programming (QP)

In case the equality constraint equations are linear, or linearised if they are almost linear, problem (6.5) can be reduced to an unconstrained Quadratic Programming problem (QP) that can be solved analytically. In this case,

$$h(y_{true}) = Ay_{true} = 0 \tag{6.7}$$

where A is the Jacobian of the constraint equations, and the solution is obtained by the use of Lagrange multipliers and is given by (Abu-el-zeet, 2000):

$$y_{true} = y_m - VA^T(AVA^T)^{-1}\delta \tag{6.8}$$

where δ is the residual of the unsatisfied balances and is given by:

$$\delta = A\varepsilon = Ay_m \tag{6.9}$$

6.2.6.3 Successive linearisation

A shortcoming of the linear solution is that the solution does not necessarily satisfy the non-linear constraints. In successive linearisation, the linear problem is iterated until an optimal point is obtained satisfying the non-linear constraints. As in the linear solution method, the advantage of successive linearisation is its relative simplicity and fast calculation.

Before solving the NLP problem, equation (6.5), some variable classification and pre-analysis is needed to identify unobservable variables and parameters, and non-redundant measurements. As stated in section 6.2.2, Stanley and Mah (1981), and later Crowe (1986), proposed observability and redundancy tests for steady-state data reconciliation. Albuquerque and Biegler (1996) extended these to dynamic systems and applied a sparse LU decomposition rather than a QR factorization. The reconciliation algorithm will correct only redundant variables. The preliminary analysis should also detect *overspecified variables* (particularly those set to constants) and *trivial redundancy*, where the measured variable does not depend at all upon its measured value but is inferred directly from the model. Finally, it should also identify model equations that do not influence the reconciliation, but are merely used to calculate some unmeasured variables. Such preliminary tests are extremely important, especially when the data reconciliation runs as an automated process.

In the previous section we indicated the usefulness of data reconciliation for obtaining accurate states, assessing the sensitivity of measurements and their uncertainties on estimated parameters, and in providing a tool for fault detection. In the following section we focus on efficient strategies for handling gross errors in data reconciliation.

6.3 GROSS ERROR DETECTION

The least squares objective function in equation (6.3) can, in certain situations, be severely biased leading to incorrect reconciliation and estimation. A common procedure can identify the measurements that suffer from gross errors and eliminate them in a sequential procedure. This procedure is called gross error detection. Early papers on the subject describe tests based on Chi-square statistics as the criteria for identifying outliers (Crowe, 1996). Madron (1985) and (1992) proposed a Chi-square test based on the squared studentized residuals following a non-central Chi-square distribution. They also provided methods for gross error detection and the concept of measurement credibility. Kao et al. (1992) proposed a Chi-square test for gross error detection in serially correlated process data. They also compared this with three other tests for outlier detection.

Two central issues are of concern when dealing with a gross error detection problem: proper location of the gross errors (instrument biases and leaks) and estimation of their sizes. Thus, the main task is to (Bagajewicz, 2003):

- Identify the existence of gross errors
- Identify the gross errors location
- Identify the gross error type
- Determine the size of the gross error.

After the gross errors are identified, two responses are possible and/or desired:

- Eliminate the measurement with the bias, or
- Correct the model (case of a leak) and run the reconciliation again.

The first alternative is the one implemented in commercial software, which only considers biases.

One of the most recognised methods for gross error detection is the Generalized Likelihood Ratio (GLR) method of Narasimhan and Mah (1987).

6.3.1. Formulation of the GLR method for gross error detection

As described in Narasimhan and Mah (1987), the Generalised Likelihood Ratio (GLR) method is used for the detection, identification and estimation of gross errors in steady-state processes.

The method is based on the GLR method developed by Willsky and Jones (1974) to identify abrupt failures in dynamic systems. It assumes the knowledge of a mathematical model describing the effect of a leak and / or bias on the process. Although suited for single gross error identification, a serial compensation strategy is often adopted in combination with the GLR method when dealing with multiple gross errors (Narasimhan and Mah, 1987).

The method is described below:

i. Process model:

Consider a steady-state model of a chemical process described by (6.1) and (6.7). In other words:

$$y_m = y_{true} + \varepsilon \tag{6.10}$$

$$h(y_{true}) = Ay_{true} = 0 \tag{6.11}$$

where all variables are same as before. To mention that equation (6.11) represents the linear or linearised mass and energy conservation constraints.

For a system affected by a gross error of unknown magnitude b, the model of the system is given by: $y_m = y_{true} + \varepsilon + be_i$ for a measurement bias and

$h(y_{true}) = Ay_{true} - bm_j = 0$ for a process leak, as a leak affects the balance constraints only.

For the general problem of detecting the presence of a gross error, identifying its source and estimating its value, we first consider the case where only one gross error is present at most.

Consider r to be the residuals of the material balances

$$r = Ay_m \tag{6.12}$$

Since r is a linear transformation of y_m, it has a multivariate normal distribution. Moreover, in the absence of gross error, the expected value of $E(r) = 0$ (where $E(r)$ is the statistical mean of r), and the covariance matrix $\text{cov}(r) = H = AVA'$, where V is the variance-covariance matrix of the random measurement errors, are assumed to be known.

If a gross error due to a bias of magnitude b is present in a measurement i, then $E(r) = bAe_i \neq 0$. So much so, if a gross error due to a process leak of magnitude b is present in a certain node j (in the system), then $E(r) = bm_j \neq 0$.

In general, in the presence of any type of a gross error, we can write:

$$E(r) = bf_i \tag{6.13}$$

where

$$f_i = \begin{cases} Ae_i & \text{for a bias in measurement } i \\ m_j & \text{for a process leak in node } j \end{cases} \tag{6.14}$$

Therefore, the hypotheses for gross error detection can be formulated as follows:

$$\begin{aligned} H_0 &: \mu = 0 \\ H_1 &: \mu = bf_i \end{aligned} \tag{6.15}$$

where μ is the unknown expected value of r, H_0 is the null hypothesis that there is no gross error and H_1 is the alternative hypothesis that either a process leak or a measurement bias is present. H_1 has got two unknown parameters, b and f_i. The value of the gross error magnitude b can be any real number, and f_i (gross error vectors (Narasimhan and Mah, 1987)) can be any vector from the set F given as:

$$F = \{Ae_i, m_j : i = 1 \ldots n, j = 1 \ldots m\} \tag{6.16}$$

The likelihood ratio test statistic in our case is given by:

$$\lambda = \sup \frac{\Pr\{r \mid H_1 \mid\}}{\Pr\{r \mid H_0 \mid\}} \tag{6.17}$$

where Pr is the normal probability density function for r and the supremum in equation (6.17) is computed over all possible values of the parameters present in the hypotheses.

Using the normal probability density function for r, equation (6.17) can be written as:

$$\lambda = \sup_{b, f_i} \frac{\exp\{-0.5(r - bf_i)' H^{-1} (r - bf_i)}{\exp\{-0.5 r' H^{-1} r\}} \tag{6.18}$$

Equation (6.18) can be simplified to the following (as it is always positive):

$$T = 2 \ln \lambda = \sup_{b, f_i} r' H^{-1} r - (r - bf_i)' H^{-1} (r - bf_i) \tag{6.19}$$

The computation of the test statistic T is very important in the process of detecting, identifying and estimating the value of the gross error of magnitude b. The procedure to compute T is given as follows.

First, we calculate the estimate $\hat{b}$ of b for which equation (6.19) is satisfied. b is called the maximum likelihood estimate, and is given by:

$$\hat{b} = (f_i^{'} H^{-1} f_i)^{-1} (f_i^{'} H^{-1} r) \tag{6.20}$$

Substituting this value of $\hat{b}$ in equation (6.19) we obtain:

$$T_i = {d_i^2}/{C_i} \tag{6.21}$$

Where

$$d_i = f_i^{'} H^{-1} r \tag{6.22}$$

$$C_i = f_i^{'} H^{-1} f_i \tag{6.23}$$

After every T_i (for every f_i in the set F) is computed, the test statistic T is obtained as:

$$T = \sup_i T_i \tag{6.24}$$

ii. The test:

At this stage, a comparison test is performed on the value of the test statistic T. If T is greater than a certain threshold C, a gross error associated with the vector f^* is detected, identified and then its value estimated using equation (6.20). f^* being the vector that leads to the supremum in equation (6.24).

The above algorithm is summarised in Figure (6-3).

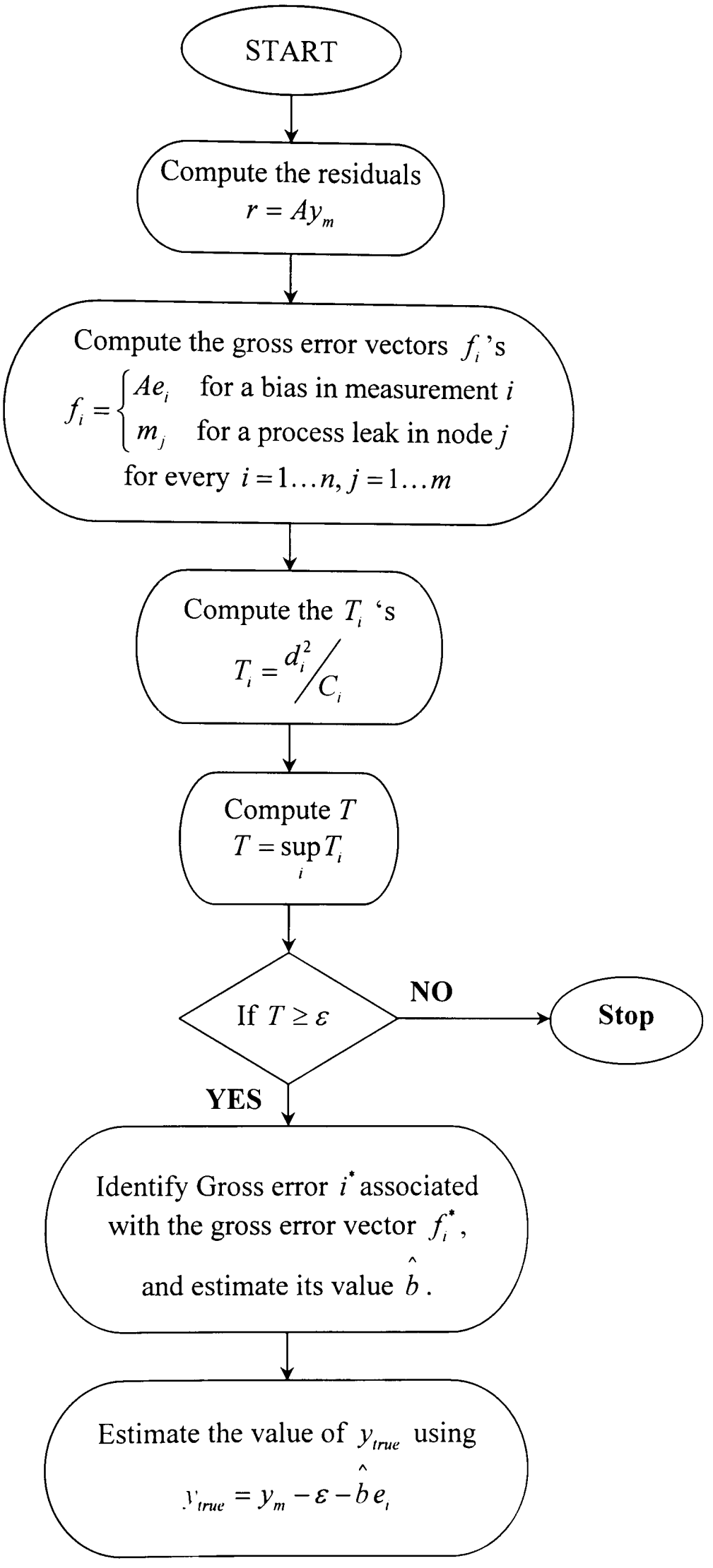

Figure (6-3): Bloc diagram representation of the GLR algorithm.

In the above formulation, additional constraints can be added as to oblige process leak values to be positive. This can be done by testing the computed value of b for nonnegativity. If the value of b is negative, we discard that process leak as a possible source of gross error. So much so, and in a similar way, one can include upper and lower bound constraints on the magnitude of process leaks (Narasimhan and Mah, 1987).

The above formulation of the GLR algorithm is applicable only when a single gross error is present. In the case of multiple gross errors, Narasimhan and Mah (1987) proposed a strategy based on serial compensation of gross errors. This strategy is described in section 6.4.

6.3.2. Bias estimation

In the special case where the locations of the biased variables are known a priori, bias can be estimated as a parameter (McBrayer and Edgar, 1995).

The procedure is to solve the following NLP problem:

$$\text{Min } J(\bar{y},\ \hat{b}) \tag{6.25}$$

subject to:

$$\begin{aligned} & f(\bar{y}) = 0. \\ & \bar{y}_{l,i} \le \bar{y}_i \le \bar{y}_{u,i} \qquad \forall i, \\ & \hat{b}_{l,i} \le \hat{b}_i \le \hat{b}_{u,i} \qquad \forall i, \end{aligned} \tag{6.26}$$

where

$$J(\bar{y},\ \hat{b}) = \left(\frac{\bar{y}_1 - \left(y_{m1} - \hat{b}_1\right)}{\sigma_1}\right)^2 + \left(\frac{\bar{y}_2 - \left(y_{m2} - \hat{b}_2\right)}{\sigma_2}\right)^2 + \cdots + \left(\frac{\bar{y}_i - \left(y_{mi} - \hat{b}_i\right)}{\sigma_i}\right)^2 \tag{6.27}$$

where y_{mi} is the i[th] measured variable, $\bar{y}_i$ is the i[th] estimate, σ_i is the measurement noise standard deviation of the i[th] measured variable and $\hat{b}_i$ is the estimate of bias on the i[th] measured variable. Note that $\hat{b}_i$ is also included in the inequality constraints. This allows for physical limits on the range of admissible biases.

6.4 STRATEGY FOR MULTIPLE GROSS ERROR DETECTION

The GLR method presented in the previous section is only applicable for single gross error detection and identification. However, when more than one gross error is present, strategies are needed to identify them. Many strategies exist and have been developed for this purpose. The serial elimination strategy (Ripps, 1965) is one of the first proposed strategies. It is based on applying the test recursively on the different variables and the elimination of the current measurement. Commercial versions of this procedure (Datacon, Sigmafine) eliminate one measurement at a time and use the measurement test or similar. The serial compensation strategy which is based on identifying one gross error at the time using the GLR test was developed by Narasimhan and Mah (1987). This strategy can be used to identify multiple gross errors of any type. This method will be presented in detail in the next subsection. Another strategy known as serial collective compensation method, exists (Bagajewicz, 2003), and it is based on applying the test recursively, to determine the sizes of all errors and adjust the measurements.

6.4.1. The Serial Compensation Strategy

The serial compensation strategy proposed by Narasimhan and Mah (1987), as opposed to its predecessors, can be used to identify multiple gross errors of any type. In this technique one gross error is identified at a time, by applying the GLR test as presented earlier in this chapter. After estimating its magnitude, this value

is used to compensate for the error before moving to other gross errors. This process is repeated until no further gross errors are detected.

The serial compensation strategy is described below:

Let us denote the compensated measurements and residuals after k applications of the GLR test as y_{mk} and r_k, respectively. Let f_k^* and b_k^* be the gross error vector and estimated magnitude, respectively, of the gross error being identified in application k of the GLR test. Also, let the following matrices E_k^*, M_k^* and G_k^* be defined as follows:

$$E_k^* = [e_1^*, e_2^* \ldots e_k^*] \tag{6.28}$$

where

$$e_i^* = \begin{cases} 0 & \text{if a bias is not identified in application } i \\ e_j & \text{if a bias in sensor } j \text{ is identified in application } i \end{cases} \tag{6.29}$$

$$M_k^* = [m_1^*, m_2^* \ldots m_k^*] \tag{6.30}$$

where

$$m_i^* = \begin{cases} 0 & \text{if a leak is not identified in application } i \\ m_j & \text{if a leak in node } j \text{ is identified in application } i \end{cases} \tag{6.31}$$

From the above formulations of E_k^*, M_k^* and G_k^*, and definition of the gross error vector f_i, we can put into effect that:

$$G_k^* = AE_k^* + M_k^* \tag{6.32}$$

We can then deduce that the compensated measurements and constraints at the end of application k are:

$$y_{mk} = y_m - E_k^* b_k^* \tag{6.33}$$

$$Ay_{true} - M_k^* b_k^* = 0 \tag{6.34}$$

where

$$b_k^* = [b_1^*, b_2^* \ldots b_k^*]' \tag{6.35}$$

If we define the compensated residuals r_k as

$$r_k = Ay_{mk} - M_k^* b_k^* \tag{6.36}$$

Then, by using equations (6.31), (6.32) and (6.12) we can see that

$$r_k = r - G_k^* b_k^* \tag{6.37}$$

Therefore the hypotheses for application k+1 of the GLR test is formulated as

$$\begin{aligned} H_{0.k} &: E[r_k] = 0 \\ H_{1.k} &: E[r_k] = bf_i; f_i \in F_k \end{aligned} \tag{6.38}$$

where F_k is the set of gross error vectors corresponding to the gross errors that are not identified in the first k applications of the GLR test. If we assume that the gross errors identified in the first k applications are actually present in the data and that their actual magnitudes are equal to the estimated magnitudes, then under $H_{0.k}$ the true constraint model is given by equation (6.34) and that true measurement model of the system is given by:

$$y_m = y_{true} + E_k^* b_k^* + v \tag{6.39}$$

Then, by using equations (6.12), (6.32), (6.34), (6.37) and (6.39), we arrive to the conclusion that

$$r_k \sim N(0, H) \text{ under } H_{0,k} \tag{6.40}$$

The test statistic for each gross error vector in F_k can therefore be achieved through equations (6.21), (6.22) and (6.23) by using r_k for r. We draw the attention here that the test statistics for application k+1 are conditional test statistics of the previous tests. In other words, if a gross error is not detected in, say application n of the GLR, then no test is carried out for the application n+1, and the serial compensation strategy is brought to an end. The compensated residuals r_{n-1} are used in data reconciliation to estimate all the variables.

Also, it should be noted that the serial compensation strategy described above should be applied only if the detection of a gross error is not affected by the presence of any other gross error. In case this is not satisfied, then a more correct procedure is to apply the test to all postulated combinations of gross errors (Narasimhan and Mah, 1987).

6.5 SIMULATION CASE STUDY

In order to assess the Steady-state Data Reconciliation (SDR) and Gross Error Detection (GED) schemes presented in this chapter, a set of simulations was carried out on the two Continuous Stirred Tank Reactors (CSTR's) system (chapter 3). In the simulations, biases and random errors were added to the measurements, to test whether the above schemes are able to detect, estimate and eliminate them. Comparisons are made between simulation results when

measurements are affected by noise and or bias, with and without the SDR and GED scheme.

6.5.1. The system

The two CSTR system is described in detail in chapter 3. It has four outputs which are the concentrations of the two components A and B in the two tanks: $y = (C_{a1}, C_{b1}, C_{a2}, C_{b2})^T$. In our example, only two concentrations are considered C_{b1} and C_{b2}.

Temperatures in the two tanks T_1 and T_2 are the set points. These are bounded between upper and lower levels: $300 \le T_1 \le 312\ K, 300 \le T_2 \le 312\ K$ and are assumed to be known noise free.

6.5.2. The simulations

All simulations were started from the same initial operating point given by $T_1 = 307K$ and $T_2 = 302K$, yielding the following steady-state output values of the concentration of product B in Tanks 1 and 2, $C_{b1}(0) = 0.05165$ [kmol/m^3] and $C_{b2}(0)$=0.058638 [$kmol/m^3$]. The set-points of the temperature controllers were not changed during the simulations. Sufficient time was allowed to the system to settle down for a steady-state condition before measurements were taken. This time was chosen to be $T = 60$ min which is enough for the system to settle down given the system's open loop time constant $T_o = 40$ min.

The simulations carried out on the above system were to assess the ability of the SDR and GED scheme in eliminating the effect of noise and detecting, estimating and eliminating biases. These noise and biases were deliberately added to the output measurements in order to simulate a real life situation where measurements might be contaminated by random and/ or gross errors.

The biases added to the measurements are given as follows.

In the case where only C_{b1} was biased, the added bias was of a value of +25% of the nominal value. In the case where only C_{b2} was biased, the value of the bias added to the output measurement C_{b2} was +30% of the nominal value.

In the case where both measurements were biased, the value of the bias added to C_{b1} was -20% of the nominal value, and +20% of the nominal value for C_{b2}.

The noise present on the measurements is considered to be normally distributed and of zero mean. The value of the variance-covariance matrix is:

$$V = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \tag{6.41}$$

where σ_1 is the standard deviation for the variable C_{b1} and was chosen to be 5% of the nominal value, σ_2 is the standard deviation for C_{b2} and was of a value of 5%.

The implementation of the CSTR system together with the SDR and GED scheme presented in this chapter was performed using a MATLAB®/SIMULINK software platform.

A SIMULINK model of the CSTR process (saved in an .mdl file) was used to enable periodic calls to the SDR algorithm saved in an M-file. As the SDR and GED schemes are steady-state procedures, the CSTR system was led to a steady-state position and left static. Therefore, there was no need to wait for the system to settle down afterwards. The algorithm was called periodically to reconcile the noisy and biased measurements C_{b1} and/ or C_{b2}, until the algorithm converged, and the correct values of C_{b1} and/ or C_{b2} were found. Once convergence is reached, the values of the biases (if they exist) on the measurements are estimated, and compared to the values simulated, and later eliminated to provide a more accurate state of the current measurement.

6.5.3. The results

Simulation results for this case study are shown in Table (6.1) and Figures (6-4) to (6-7).

Table (6.1) summarises the results obtained when the SDR and GED scheme was applied to reconciliate measurements affected by noise and bias of different values. The table gives important details of the values of the real, noisy and reconciled output vector. The level of bias added to the measurement C_{b1} for this case was in the range -40% to +40% of the nominal value, whereas the level of bias added to C_{b2} was in the range -20% to +30% of the nominal value. It is clear that data reconciliation was successfully implemented and conducted, and that noise and biases were eliminated.

Figures (6-5), (6-6) and (6-7) show the trajectories taken by the outputs of the plant when a bias of a given magnitude was present on measurements C_{b1} or C_{b2} and C_{b1} and C_{b2} together respectively. The figures show the values of the outputs before and after the SDR and GED scheme was applied. It is clear that the reconciled estimates of the plant outputs and the real system outputs are superimposed, and there is no difference between them. Also, it is shown how the data reconciliation performed on the output measurements does converge in short time whether one measurement or two were biased. This time is relatively small compared to the system's settling time constant, which makes it desirable to be used in steady-state optimisation and control especially for slow processes, when most time is spent waiting for steady-state to be reached.

Table (6.1): Bias values and their estimates.

Bias Added to C_{b1}	Bias Added to C_{b2}	Associated Figure	Reconciled Value	Bias estimates
No	No	6-4	$C_{b1} = 0.05165$ $C_{b2} = 0.05863$	No
-0.0206	No	-	$C_{b1} = 0.05165$ $C_{b2} = 0.05863$	$\hat{b}_1$ =-0.0206
0.0129	No	6-5	$C_{b1} = 0.05165$ $C_{b2} = 0.05863$	$\hat{b}_1$ =0.0129
No	-0.0117	-	$C_{b1} = 0.05165$ $C_{b2} = 0.05863$	$\hat{b}_2$ =-0.0117
No	0.0175	6-6	$C_{b1} = 0.05165$ $C_{b2} = 0.05863$	$\hat{b}_2$ =0.0175
-0.0103	0.0117	6-7	$C_{b1} = 0.05165$ $C_{b2} = 0.05863$	$\hat{b}$ =[-0.0103; 0.0117]
0.0129	0.0175	-	$C_{b1} = 0.05165$ $C_{b2} = 0.05863$	$\hat{b}$ =[-0.0129; 0.0175]

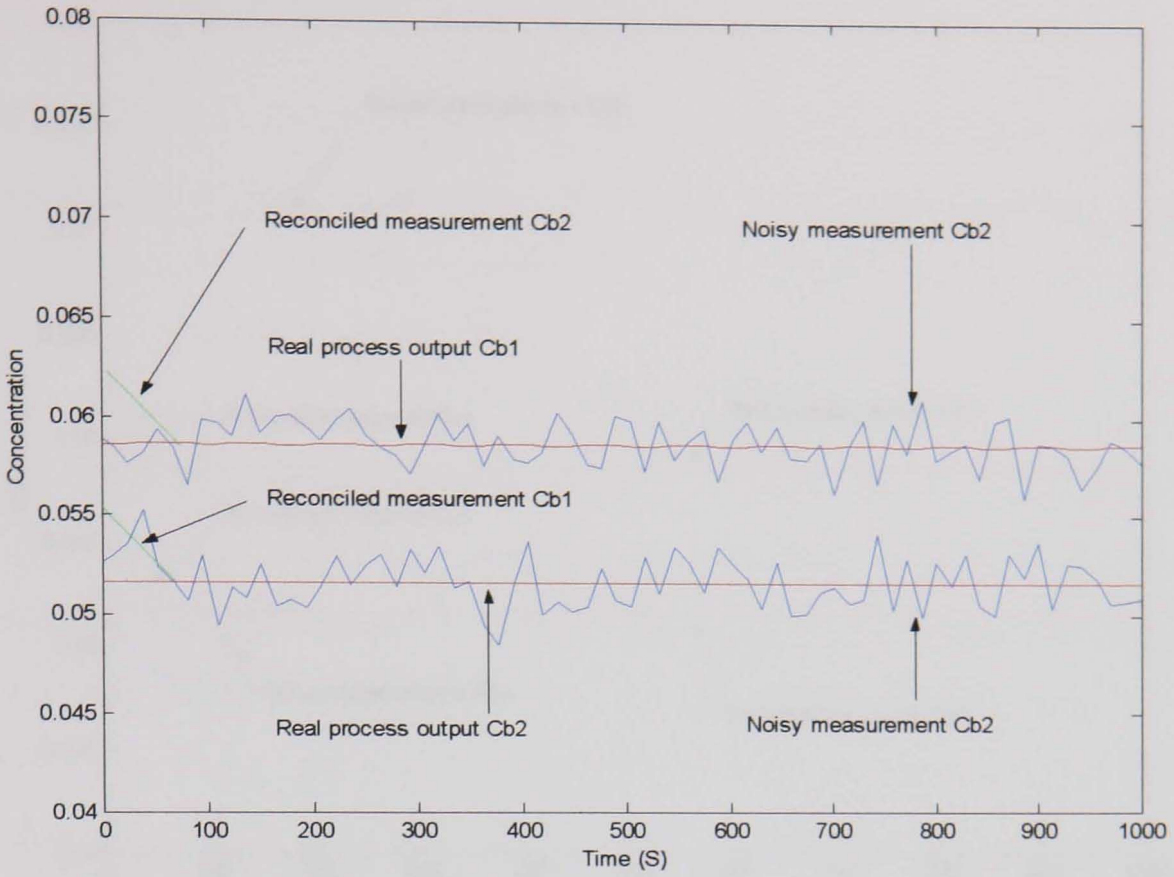

Figure (6-4): Reconciliation when measurement were subject to noise but without bias.

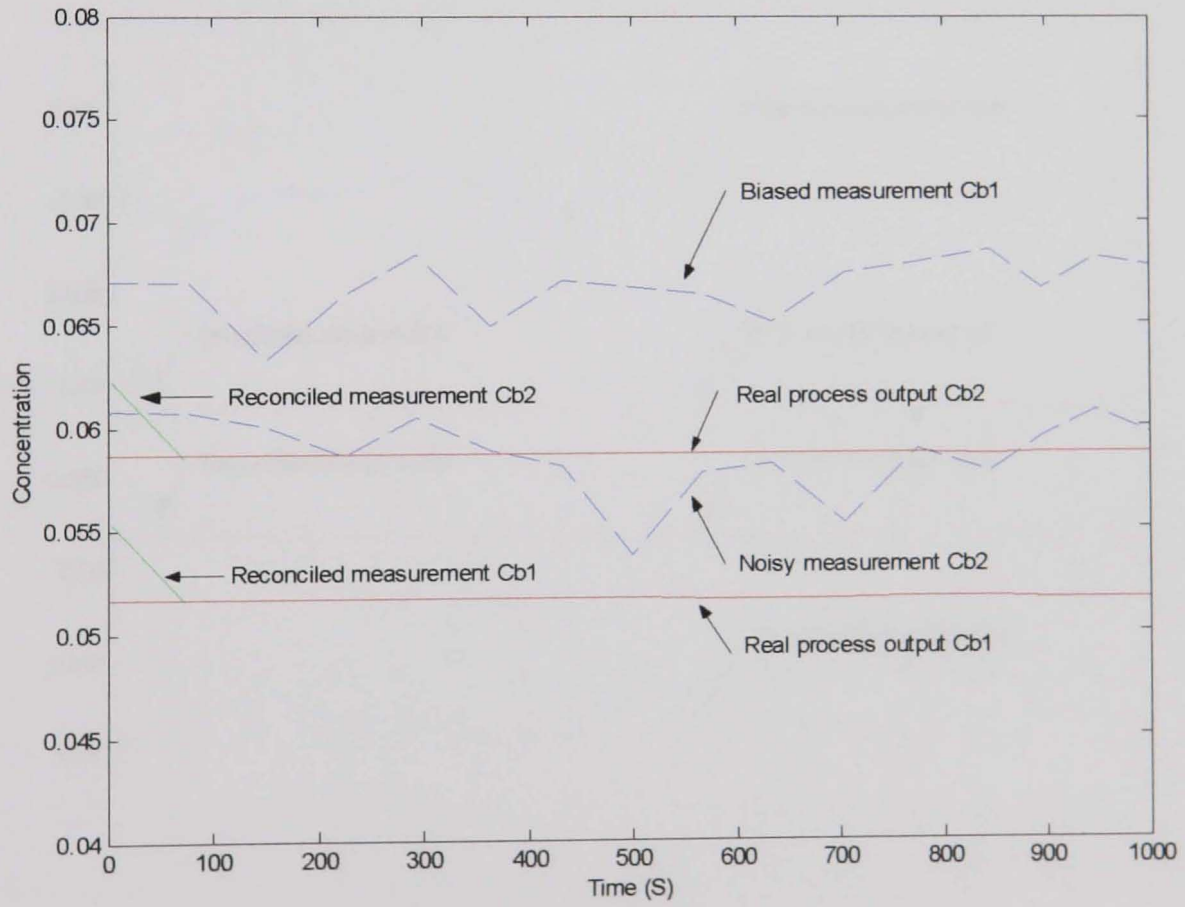

Figure (6-5): Reconciliation when only one measurement C_{b1} was biased.

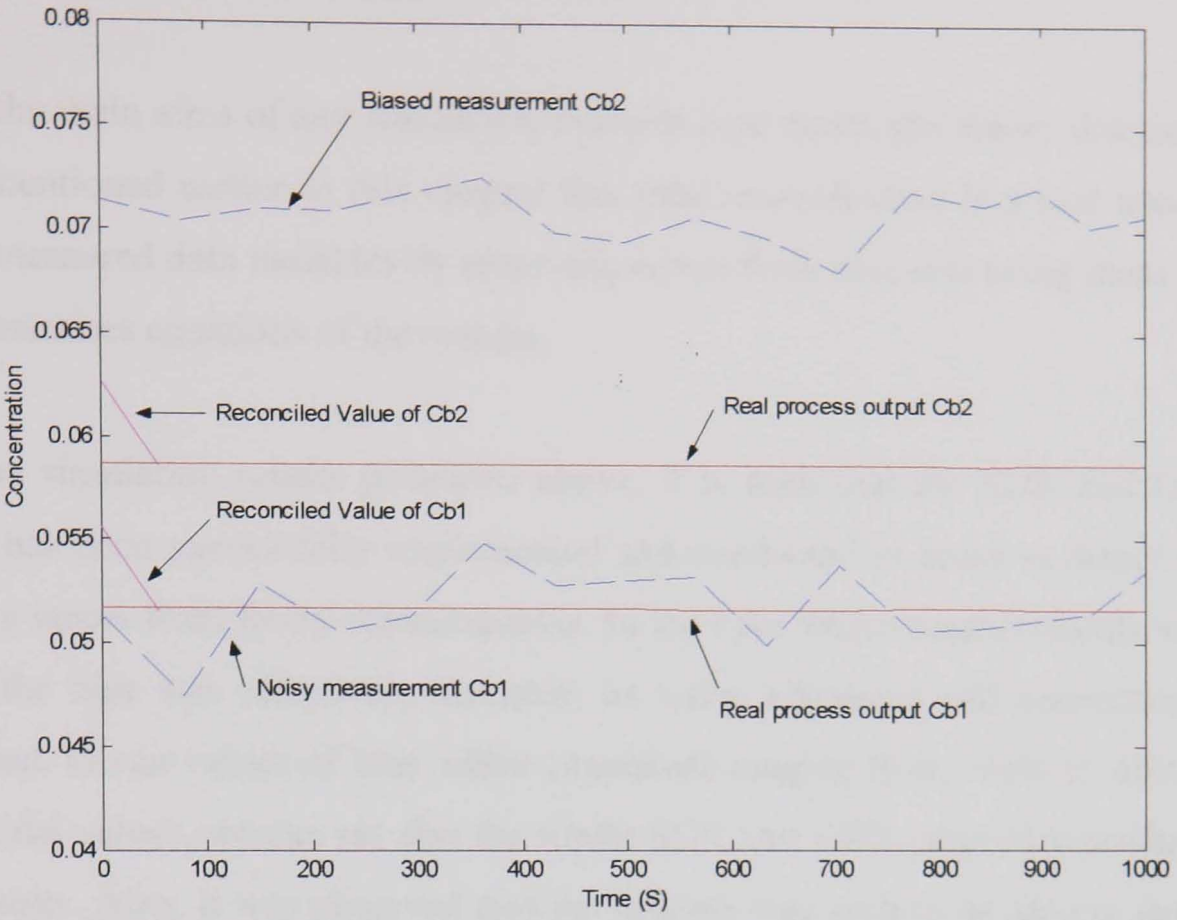

Figure (6-6): Reconciliation when only one measurement C_{b2} was biased.

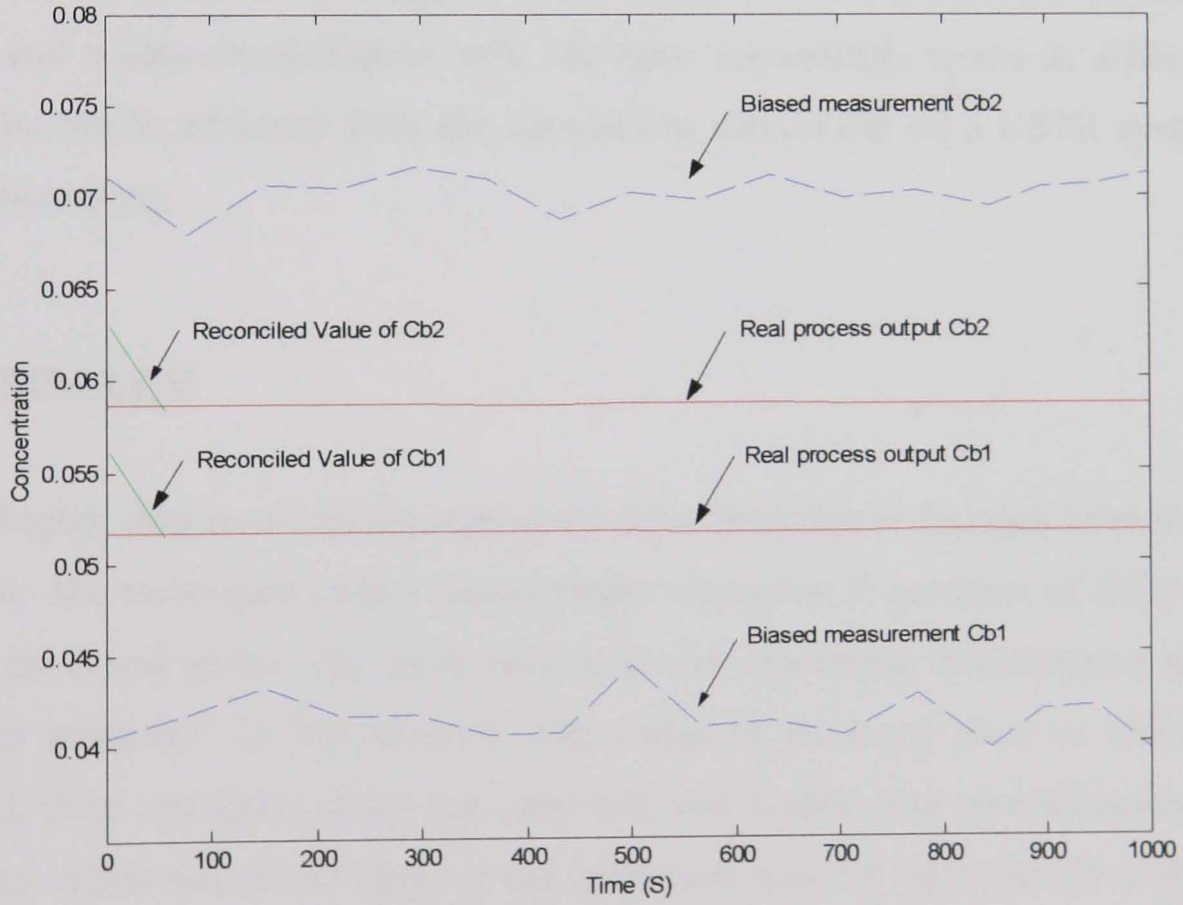

Figure (6-7): Reconciliation when both measurements C_{b1} and C_{b2} were biased.

6.5.4. Discussion of the results

One of the main aims of any simulation exercise is to verify the theory discussed. It was mentioned earlier in this chapter that data reconciliation is a tool used to correct measured data variables by removing errors from data sets using mass and energy balances equations of the system.

From the simulation results presented above, it is seen that the SDR and GED scheme has been successfully implemented and conducted in order to detect and eliminate errors from faulty measurements. In the case where measurements were biased, the bias was effectively detected, its value estimated and consequently eliminated. Given values of bias added (imposed) ranging from -40% to 40% of the nominal values, we can say that the whole SDR and GED procedure produced good results. Also, it was observed that the scheme was seen to be able to detect affected measurements and correct them in most cases. In overall, the SDR and GED scheme studied in this chapter, which comprises of a gross error detection module and a data reconciliation unit, has been successfully tested in different cases. The results obtained from the simulations carried out on a CSTR system were encouraging.

6.6 SUMMARY

In this chapter, data reconciliation and gross error detection techniques have been presented. The techniques ensure measurement correction in presense of different sorts of noise and errors. The three basic steps for processing measurement data were also presented. In this concept, data collected from any plant is initially classified, then any gross errors are removed, and finally data reconciliation is applied to adjust the set of data so the quantities derived from the data obey natural laws, such as material and energy balances. The techniques were tested under simulation on a cascade process consisting of two Continuous Stirred Tank Reactors (CSTR). The simulation results showed that the data reconciliation and gross error detection techniques presented in this chapter were successfully

implemented and applied. This was tested for a large variety of cases using the two CSTR system. In the next chapter, these techniques will be implemented within the ISOPE algorithm to verify if they can improve optimisation.

CHAPTER 7

GROSS ERROR DETECTION AND DATA RECONCILIATION IN ON-LINE OPTIMISATION

7.1 INTRODUCTION

Data reconciliation and parameter estimation are important components of model fitting, validation, and real time optimisation in the chemical and process industries. In its most general form, data reconciliation is a minimisation of measurement errors subject to satisfying the constraints of the process model. The most commonly used formulation of both problems is to minimise the sum of squares of the measurement corrections subject to model constraints and bounds. This formulation is based on the assumption that measurements have normally distributed random errors, in which case least-squares is the maximum likelihood estimator. However, the data reconciliation problem is compounded when gross errors are present in the data, as these can lead to incorrect estimates and severely biased reconciliation of the other measurements. Therefore, gross errors have to be removed from the measurements before data reconciliation can be applied. Gross error detection techniques as seen in the previous chapter are based on hypothesis testing. Combined techniques for gross error detection and data reconciliation exist. They are based on the distribution function of measurement errors. The measurement test method using a normal distribution and robust statistical method using robust functions are the two algorithms used.

In this chapter, data reconciliation and gross error detection methods presented in chapter 6 are applied within the on-line optimisation scheme (ISOPE algorithm) introduced in chapter 2. The effectiveness of this scheme and issues related to it

are demonstrated under simulation on the two CSTR system, described in chapter 3. Simulation results are also compared with results obtained from previous chapters.

7.2 Data Reconciliation

Process measurements from a plant are never error free. Typically, these measurements contain both random and gross errors. Data reconciliation is a necessary operation for obtaining accurate and consistent data in process plants by forcing them to obey constraining mass, component, or energy balances.

Results of research on data reconciliation have been reported for both steady-state and dynamic and linear and nonlinear processes. Chapter 6 provides a short review of previous work.

Generally speaking, the data reconciliation problem can be formulated as a constrained optimisation problem. That is a least-squares estimation problem if the measurements contain random errors only.

If we consider ε to be a vector of random measurement errors:

$$\varepsilon = y_m - y_{true} \tag{7.1}$$

where y_m is the vector of measured process variables, and y_{true} denotes the vector of true values of measured variables.

If these errors are normally distributed (which is assumed in almost the majority of the cases) with zero mean, and a covariance matrix V, the data reconciliation problem can be easily defined as a least-squares estimation problem as follows:

$$\underset{y_{true}}{\text{Minimise:}}\ F(y_m, y_{true}) = \frac{1}{2}\,(y_m - y_{true})^T\, V^{-1}\, (y_m - y_{true}) \qquad \text{subject to: } h(y_{true}) = 0 \tag{7.2}$$

where h is a set of algebraic equality constraint equations.

If the equality constraints are linear, or linearised if they are almost linear, then the above optimisation problem can be reduced to an unconstrained Quadratic Programming Problem (QP) that can be solved analytically (Mah and Tamhane, 1982).

7.3 THE ON-LINE OPTIMISATION PROBLEM AND THE ISOPE ALGORITHM

As has been described in Chapter 2, many different process optimisation techniques exist. They all fall into two major categories: direct search and model-based optimisation methods (Garcia and Morari, 1981). The direct approach uses measurements taken directly from the real system itself and applies one of the basic optimisation techniques to optimise the process performance objective function. While in the model-based approach, the optimisation is performed on a mathematical model of the system, when found, the results are then applied to the real system. However, the two approaches present some major drawbacks as in practice, measurements can be contaminated by noise or all sort of gross errors, and it is inevitable that model-reality differences exist, at least to some extent, in terms of structure and parameters.

As stated in Chapter 2, the Integrated System Optimisation and Parameter Estimation (ISOPE) technique (Roberts, 1979) was developed to overcome such problems as model-reality differences and is an indirect method. It is based on derivatives calculation provided by real process measurements to update an unfaithful or deliberately simplified model used in the model-based optimisation, thus achieving the real optimum of the process in spite of model-reality differences. All ISOPE algorithms designed to date are derived from the basic and well-known *two-step* technique, which consists of two major steps. The first step solves, with the aid of process measurements, a simple model parameter estimation procedure. The updated model is then used in the optimisation problem. The second step obtains the process controls via an optimisation routine (Figure 7-1).

Various versions of the ISOPE algorithm exist such as Approximate Linear Model ISOPE, ALMISOPE (Ellis et al, 1988) and Augmented ISOPE (Abdullah et al, 1988).

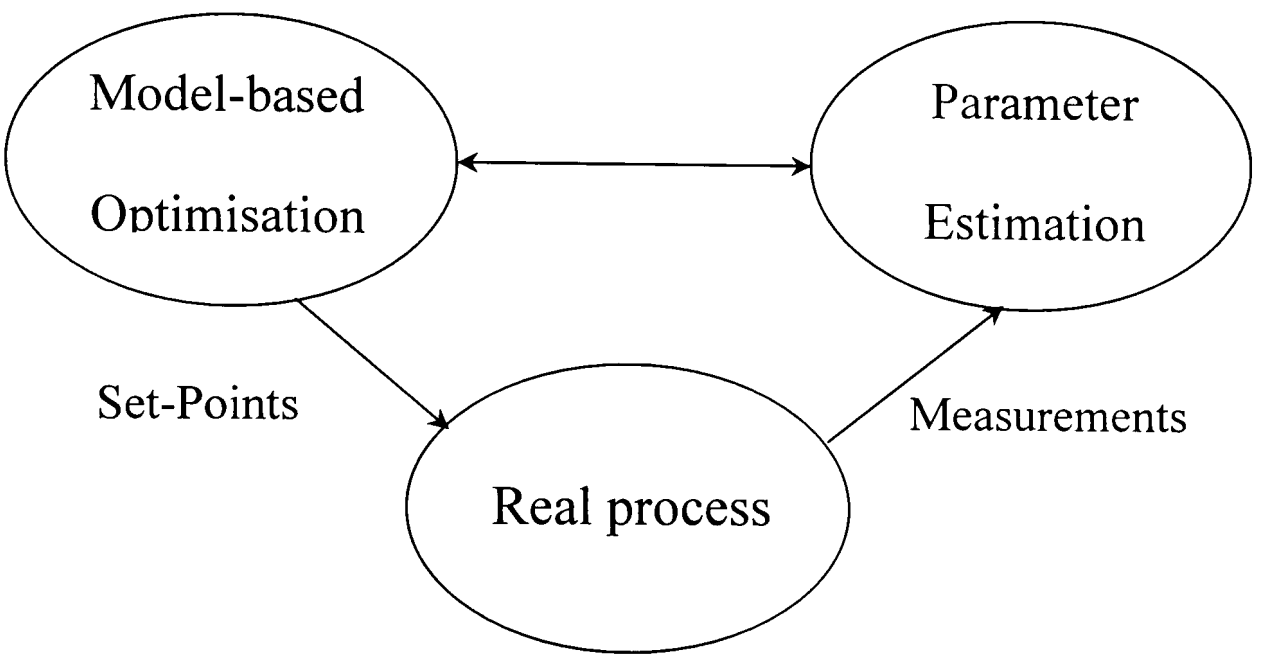

Figure (7-1): The *two-step* Method.

The solution of the ISOPE algorithm problem, as given in Chapter 2, is usually converted from the following general nonlinear programming problem (with equality and inequality constraints):

$$Min \;\; Q(v, y^*) \tag{7.3}$$

subject to:

$$y^* = H^*(v) \tag{7.4}$$

$$g(y^*) \leq 0 \tag{7.5}$$

$$v_{\min} \leq v \leq v_{\max} \tag{7.6}$$

to a simple quadratic programming problem in the case of a quadratic objective function and linear constraints. Quadratic programming problems are easy to solve because the theory is rich and the computation is less.

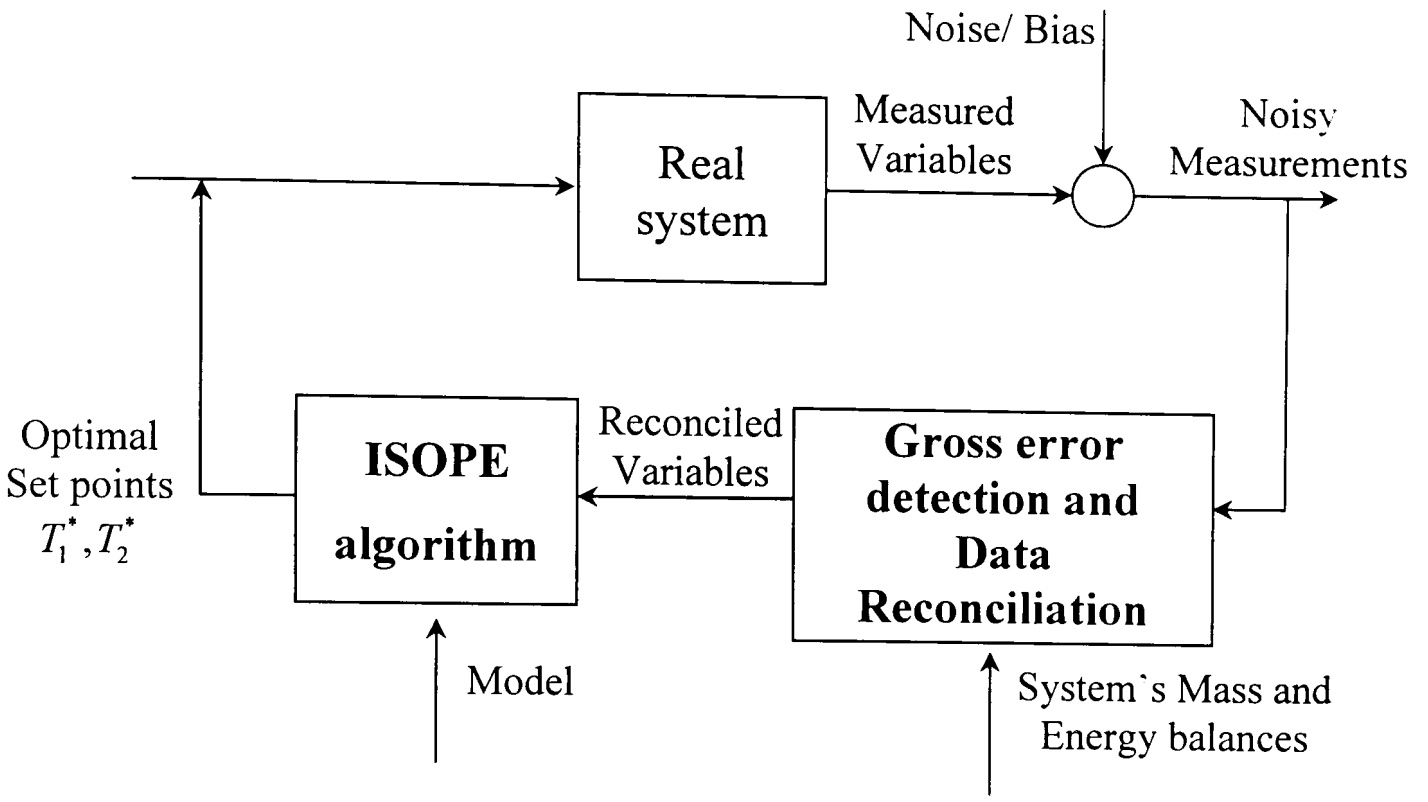

Figure (7-2): Schematic representation of the SDR and GED scheme when implemented in steady state optimisation.

7.4 SIMULATION CASE STUDY

In this case study, a group of simulations was carried out in order to assess the Steady-state Data Reconciliation (SDR) and Gross Error Detection (GED) schemes presented in the previous chapter when applied in steady-state optimisation (using the ISOPE algorithm, figure 7-2). The simulations use a two Continuous Stirred Tank Reactors (CSTR's) connected in cascade (Chapter 3). A comparison is made between simulations when measurements are affected by noise and or bias, with and without the SDR and GED scheme.

The two CSTR system has four outputs which are the concentrations in the two tanks: $y_* = (C_{a1}, C_{b1}, C_{a2}, C_{b2})^T$. In our example here, only two concentrations are

assumed to be available for measurement C_{b1} and C_{b2}. Temperatures in the two tanks T_1 and T_2 are the set-points. In other words:

$$v = (T_1, T_2) \tag{7-7}$$

All simulations were started from the same initial operating point given by $T_1 = 307K$ and $T_2 = 302K$, yielding the following steady-state output values of the concentration of product B in Tank 1 and 2 $C_{b1}(0) = 0.05165$ [kmol/m^3] and $C_{b2}(0)$=0.058638 [$kmol/m^3$]. Sufficient time was allowed to the system to settle down for a steady-state condition before measurements were taken. This time was chosen to be $T = 60\,\text{min}$. Once the system is at steady-state, the data reconciliation and gross error detection take place in order to detect and eliminate random and/ or gross errors.

The added noise was simulated as normally distributed with zero mean. The value of the variance-covariance matrix was chosen to be:

$$V = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \tag{7.8}$$

where σ_1 is the standard deviation for the variable C_{b1} and was chosen to be 5% of the nominal value, σ_2 is the standard deviation for C_{b2} and was of a value of 5%. These values were chosen as they represent typical values in many realistic situations.

The optimisation was performed on a linear objective function of the measured variable C_{b2}. This choice of the objective function manifests a desire to maximise the amount of component B in tank 2. Therefore, the mathematical form of this function is given as:

$$L(y, v) = -C_{b2} \tag{7.9}$$

The whole procedure is carried out as follows (Figure 7-3):

1. The set-points are applied to the CSTR system.
2. Measurements are taken after a suitable period giving time for the system to settle down.
3. The measurements are fed to the SDR and GED module which has the role of eliminating random noise and detecting, identifying and estimating gross errors if any are present.
4. The noisy data is corrected (reconciled). Hence a more accurate process output measurements (and derivatives with respect to the set-points needed by the ISOPE algorithm) are found.
5. Reconciled measurements are fed into the ISOPE algorithm box. One iteration of the optimisation algorithm is performed and the results which are the set-points of the temperature controllers T_1 and T_2 are found.
6. The set-points values of T_1 and T_2 are applied to the real system.

The whole procedure is repeated until convergence is reached. Convergence occurs when no further improvement is observed. In other words, when the new set-points are no longer a better candidate than the previous one.

The whole process was implemented using a MATLAB®/SIMULINK software platform. The CSTR system was modelled under SIMULINK, while the data reconciliation and gross error detection and ISOPE algorithms were implemented under MATLAB in a separate module. Because of the interaction capability between MATLAB and SIMULINK that the software offers, a SIMULINK model of the system was run for a suitable time, during which periodical calls to the data reconciliation and ISOPE algorithms module saved in an M-file were made.
Simulation results for the case study outlined above are shown in Table (7.1) and Figures (7-4) to (7-13).

In the first simulation, both measurements were subjected to 5% additive noise, with no data reconciliation. The ISOPE algorithm failed to converge (figure 7-4).

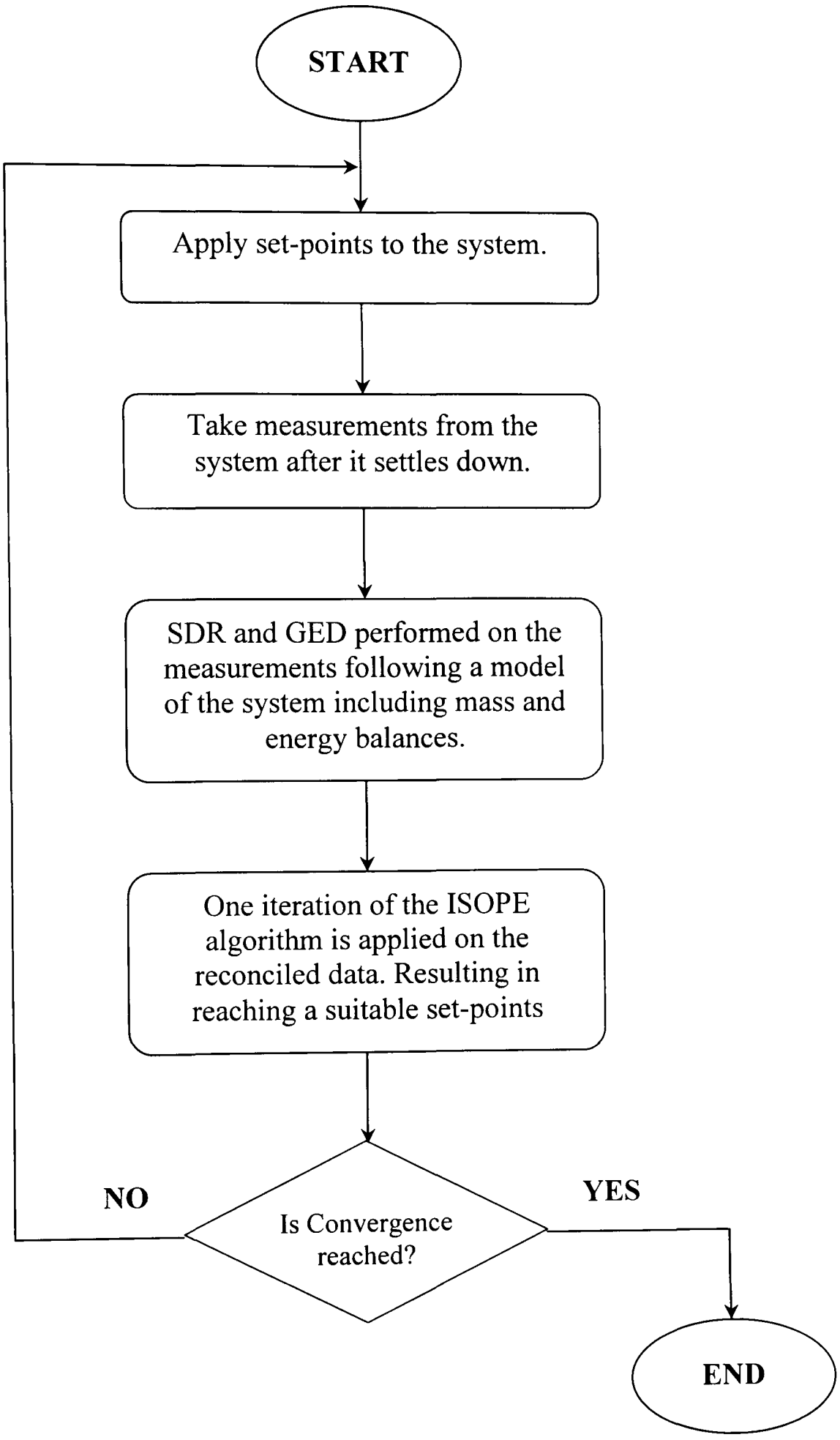

Figure (7-3): Bloc diagram representation of the application of the SDR and GED scheme within the ISOPE algorithm.

In the second simulation however, data reconciliation was applied on the output measurements and the reconciled measurements fed into the ISOPE algorithm. It is clear from figure (7-6) and (7-7) that the reconciled measurements follow exactly the real process outputs (which are the values of the measurements without the noise). Moreover, the outputs converge to the real optimum point given by: $C_{b1}=0.0644$ $[kmol/m^3]$ and $C_{b2}=0.0725$ $[kmol/m^3]$ corresponding to the set-point values of $T_1=312\ K$ and $T_2=310.2K$.

In the following simulations, biases of different values were added to either one of the measurement or both at the time. Figure (7-8) shows the real outputs and reconciled measurements trajectories when bias of a value –20% of the nominal value was added to the measurement of C_{b1}. Figures (7-10) and (7-11) present the results of the overall scheme including SDR, GED and steady-state optimisation when C_{b2} was added a bias of 25% of the nominal value. In the presence of multiple biases, both measurements C_{b1} and C_{b2} were added biases of different values. For instance, the simulation was carried out with a bias of –20% of the nominal value added to C_{b1} and 20% of the nominal value added to C_{b2}. The results of this simulation are shown in figures (7-12) and (7-13).

The last simulation was performed to highlight the contribution of SDR to the enhancement of data collection and use of Artificial Neural Networks (ANN). In fact, the neural network scheme presented in chapter 5 was tested in presence of noise and bias without data reconciliation. The results were not promising (table (7.1)). However, when SDR and GED techniques were applied together with the neural network scheme for estimating process derivatives, the results were very encouraging. Table (7.1) gives a comparison between the two schemes with and without SDR and GED. From the table, it is clear that the real process optimum was reached even in presence of noise and bias in both measurements when data reconciliation was applied with the application of a neural network model based on these measurements.

These results together with those found earlier show that whenever a noise and or bias are present on one or both measurements, the data reconciliation and gross error detection algorithm detects and eliminates them. This also proves that the application and use of data reconciliation and gross error detection on corrupted data measurements within the ISOPE algorithm improves optimisation. This is mainly due to the improved parameter estimation, and derivative estimation as well. Resulting in the ISOPE algorithm performing well and converging to the optimum point even in presence random errors and biases.

Table (7.1): ISOPE with neural network scheme when data reconciliation is applied.

Method	Bias Added to C_{b1}	Bias Added to C_{b2}	Convergence	Final measured outputs
ISOPE without SDR and GED	-0.0103	0.0117	No	No converged values
ISOPE with SDR and GED using ANN	-0.0103	0.0117	Yes	$C_{b1}=0.0644\ [kmol/m^3]$ $C_{b2}=0.0725\ [kmol/m^3]$

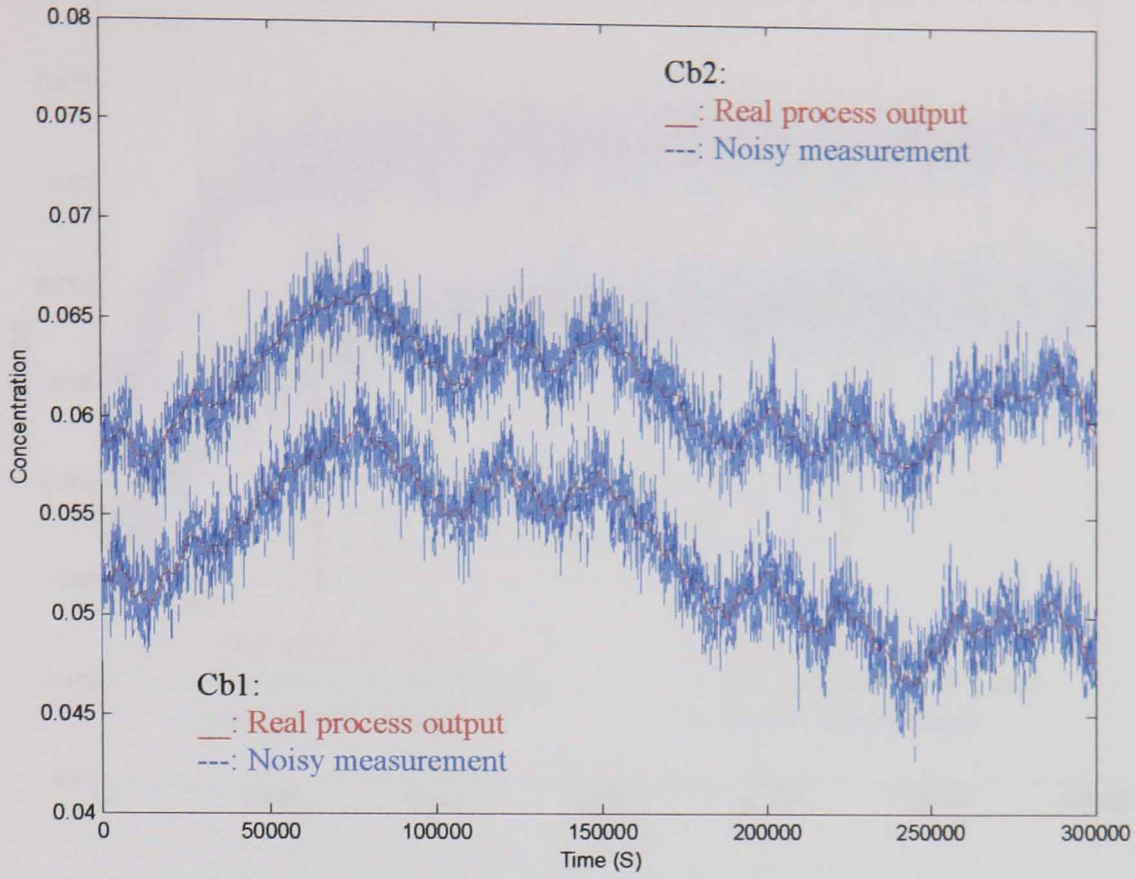

Figure (7-4): Real process output and noisy measurements trajectories, case when optimisation was applied when both measurements were affected by noise without data reconciliation

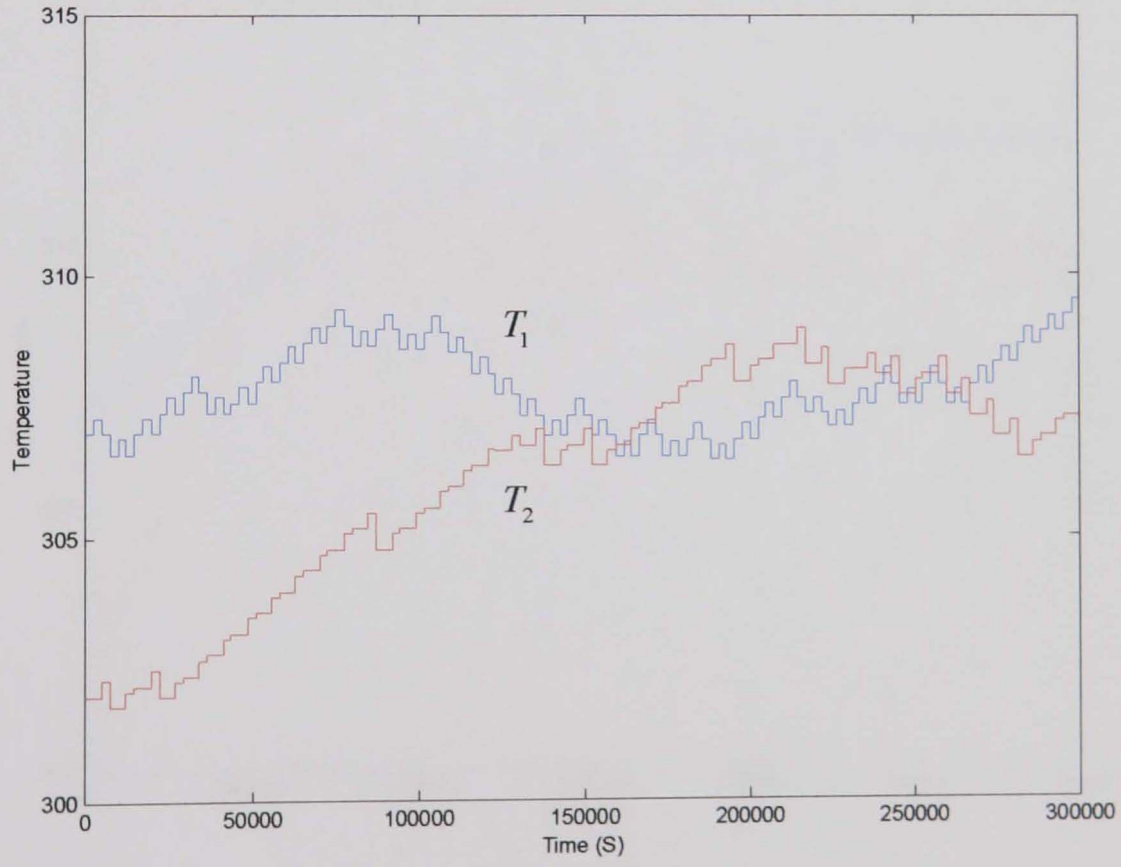

Figure (7-5): Set-points trajectories, case when optimisation was applied when both measurements were affected by noise without data reconciliation.

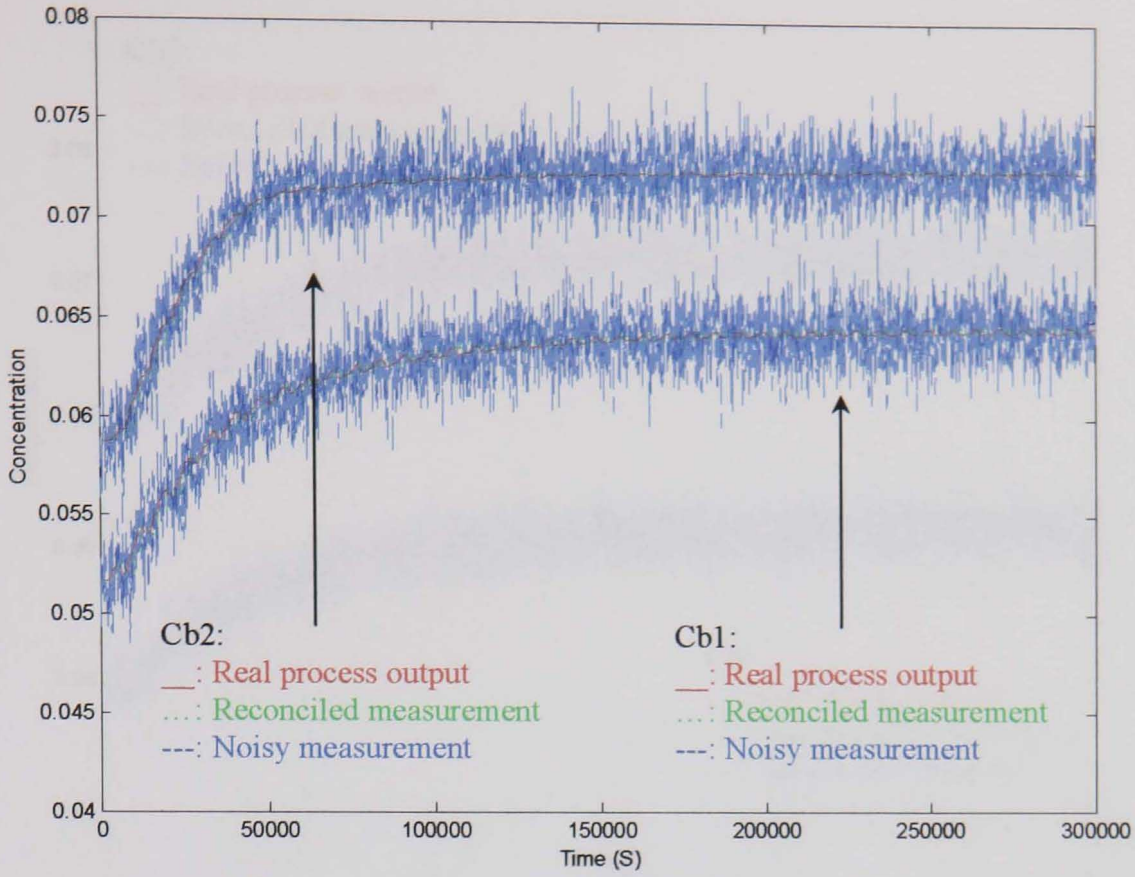

Figure (7-6): Real process output and noisy measurements trajectories, case when optimisation was applied when both measurements are affected by noise with data reconciliation.

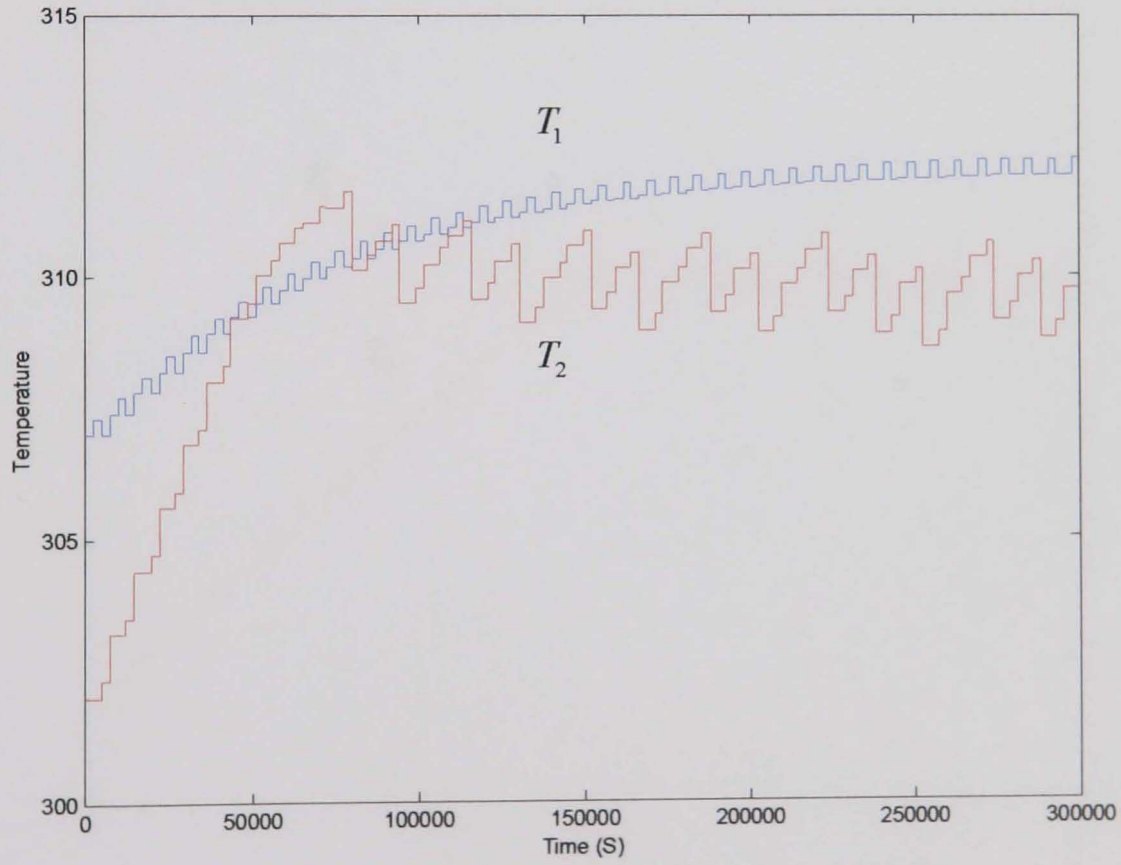

Figure (7-7): Set-points trajectories, case when optimisation was applied when both measurements are affected by noise with data reconciliation.

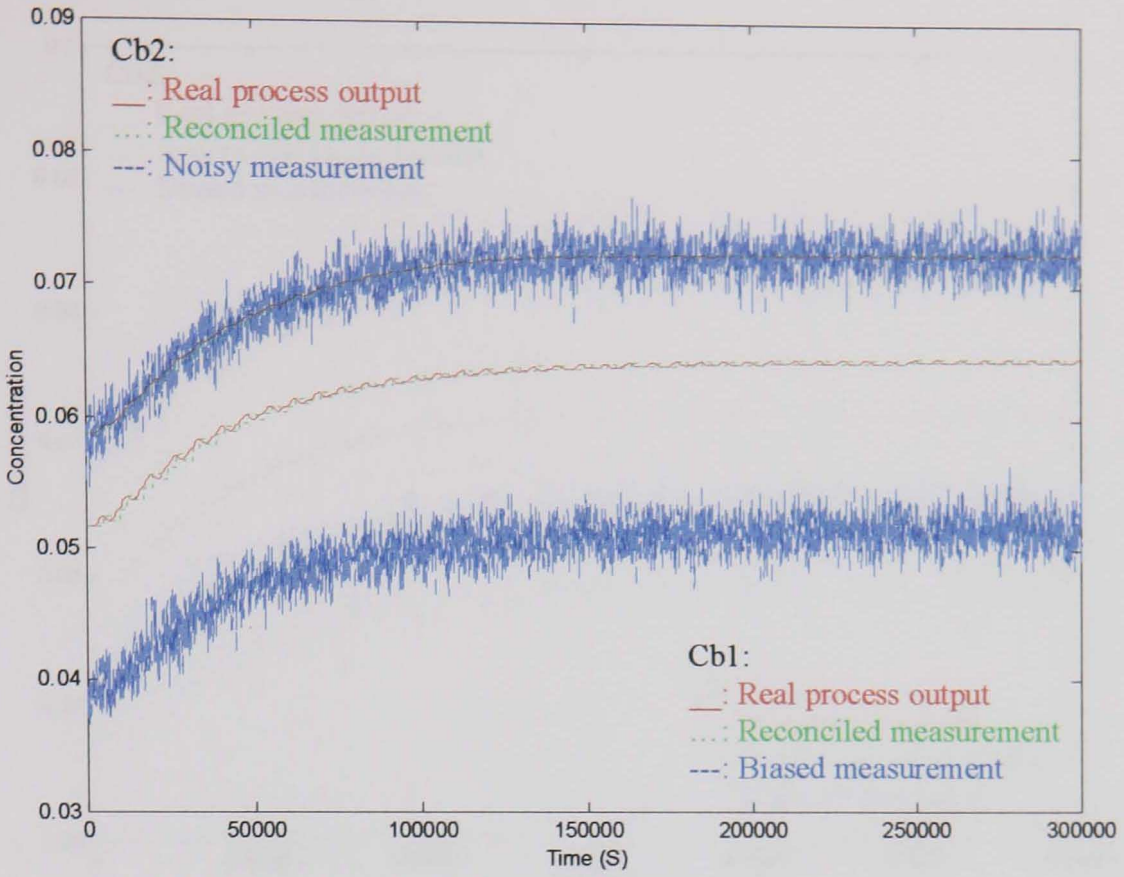

Figure (7-8): Real process output and noisy measurements trajectories, case when optimisation was applied when C_{b1} was biased with data reconciliation.

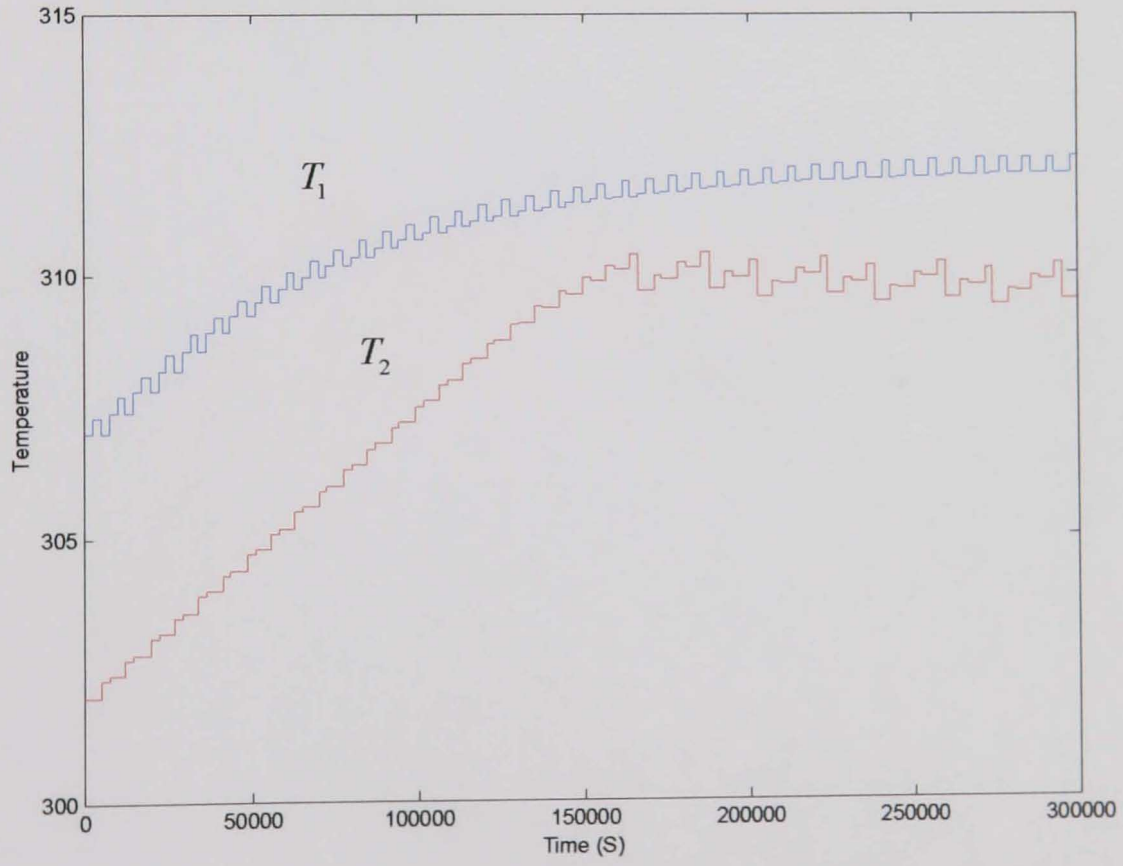

Figure (7-9): Set-points trajectories, case when optimisation was applied when C_{b1} was biased with data reconciliation.

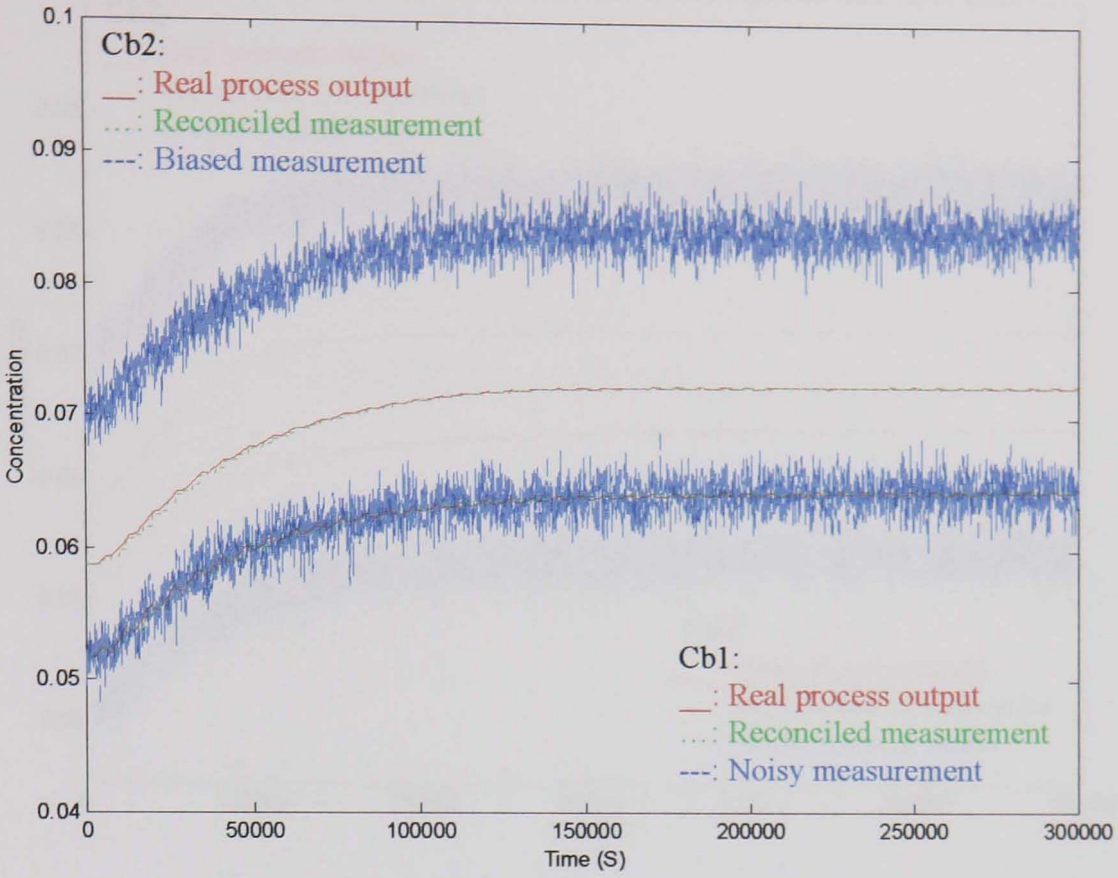

Figure (7-10): Real process output and noisy measurements trajectories, case when optimisation was applied when C_{b2} was biased with data reconciliation.

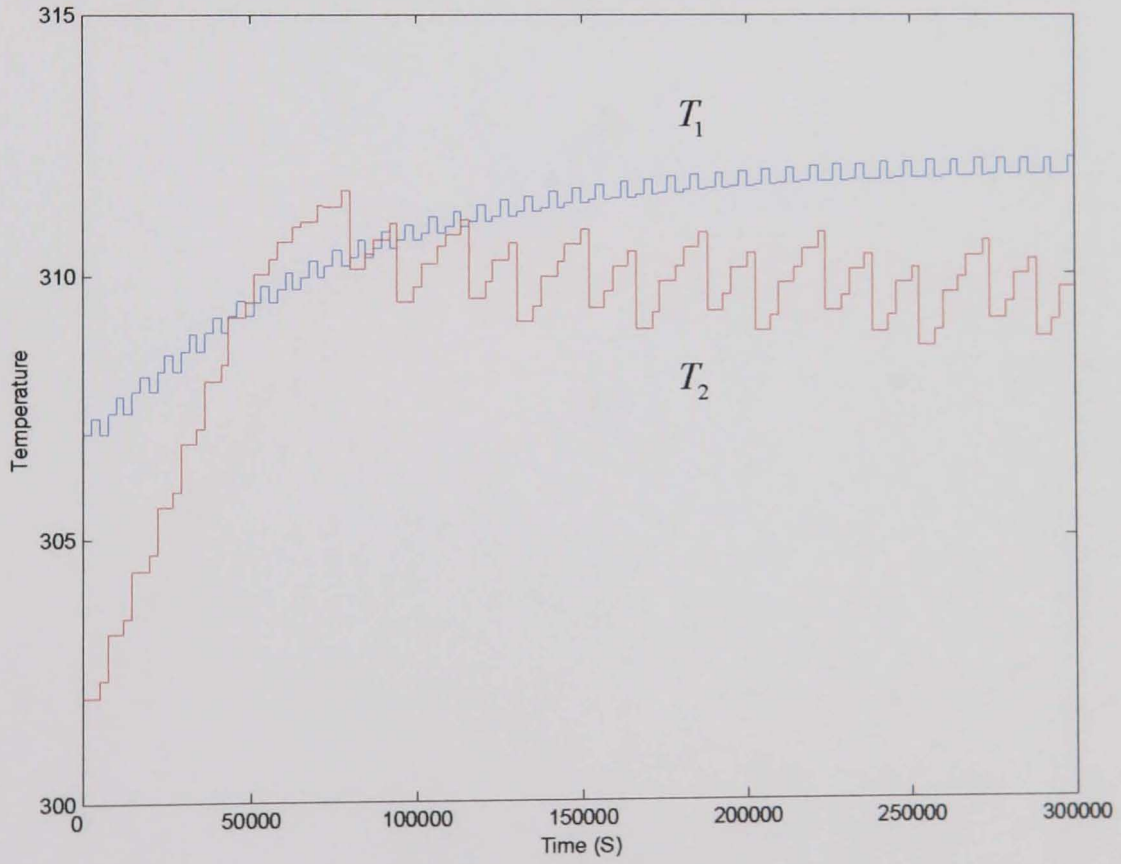

Figure (7-11): Set-points trajectories, case when optimisation was applied when C_{b2} was biased with data reconciliation.

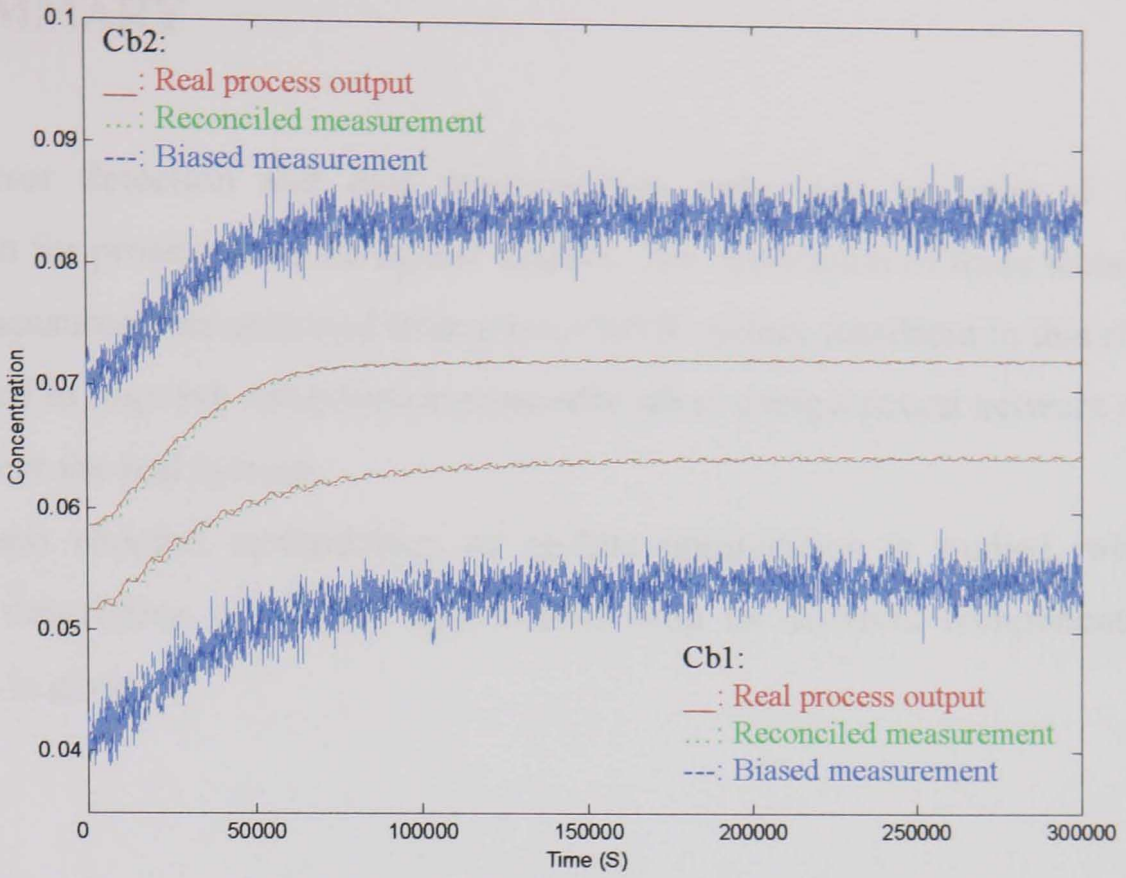

Figure (7-12): Real process output and noisy measurements trajectories, case when optimisation was applied when C_{b1} and C_{b2} were biased with data reconciliation.

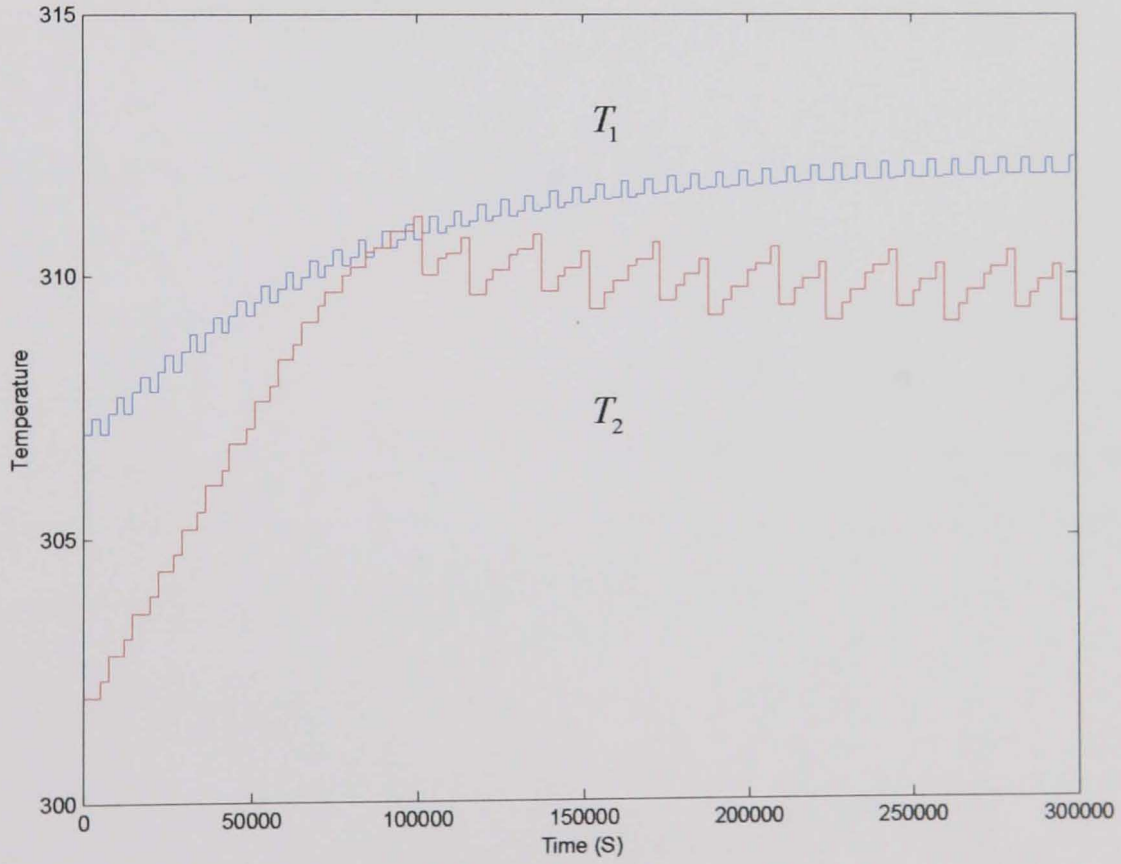

Figure (7-13): Set-points trajectories, case when optimisation was applied when C_{b1} and C_{b2} were biased with data reconciliation.

7.5 SUMMARY

Gross error detection and data reconciliation techniques are part of model validation for process monitoring and control. The application of these techniques to data measurements collected from a two CSTR system simulated in this chapter has proved to improve optimisation especially when using a neural network model to represent the real system.

In the next chapter, methodology of on-line optimisation is studied, where a detailed description of on-line optimisation with its different components and structure is given.

CHAPTER 8

THE METHODOLOGY OF ON-LINE OPTIMISATION

In this chapter, all the steps studied separately in the previous chapters, are grouped together, in order to form a methodology of on-line optimisation. The methodology would, typically, be implemented within a Distributed Control System (DCS) as such systems are new common place within the process industries. For this reason, the description here will focus on the DCS implementation of the methodology. However, it should be borne in mind that the methodology is not restricted to DCS implementation as, especially with the computing power available at the present time, the elements of the methodology could exist safely within a stand-alone computer system. The various steps of Steady-state Detection, Gross Error Detection, Data Reconciliation, Parameter Estimation and Optimisation which are part of the methodology are performed sequentially in a modular way in order for the on-line optimisation procedure to be completed successfully. The application of this methodology is generally seen to be more beneficial than when on-line optimisation is applied in the traditional way. The advantages and disadvantages of using each step of the methodology are given. Also, simulation case studies are performed throughout to assess these schemes when incorporated within the ISOPE procedure.

8.1 INTRODUCTION

As shown in figure (8-1), on-line optimisation involves three steps: Data Validation, Parameter Estimation, and Optimisation. Data sampled from the process and, typically, held within the Distributed Control System (DCS) is first validated. This procedure involves steady-state detection, gross errors removal, and data reconciliation to be consistent with material and energy balances of the

process. This data is next used in parameter estimation to update the plant parameters. This in turn updates the process model to enable plant-model matching. When finished, the updated model is then used in the optimisation to determine the optimal operating point of the plant.

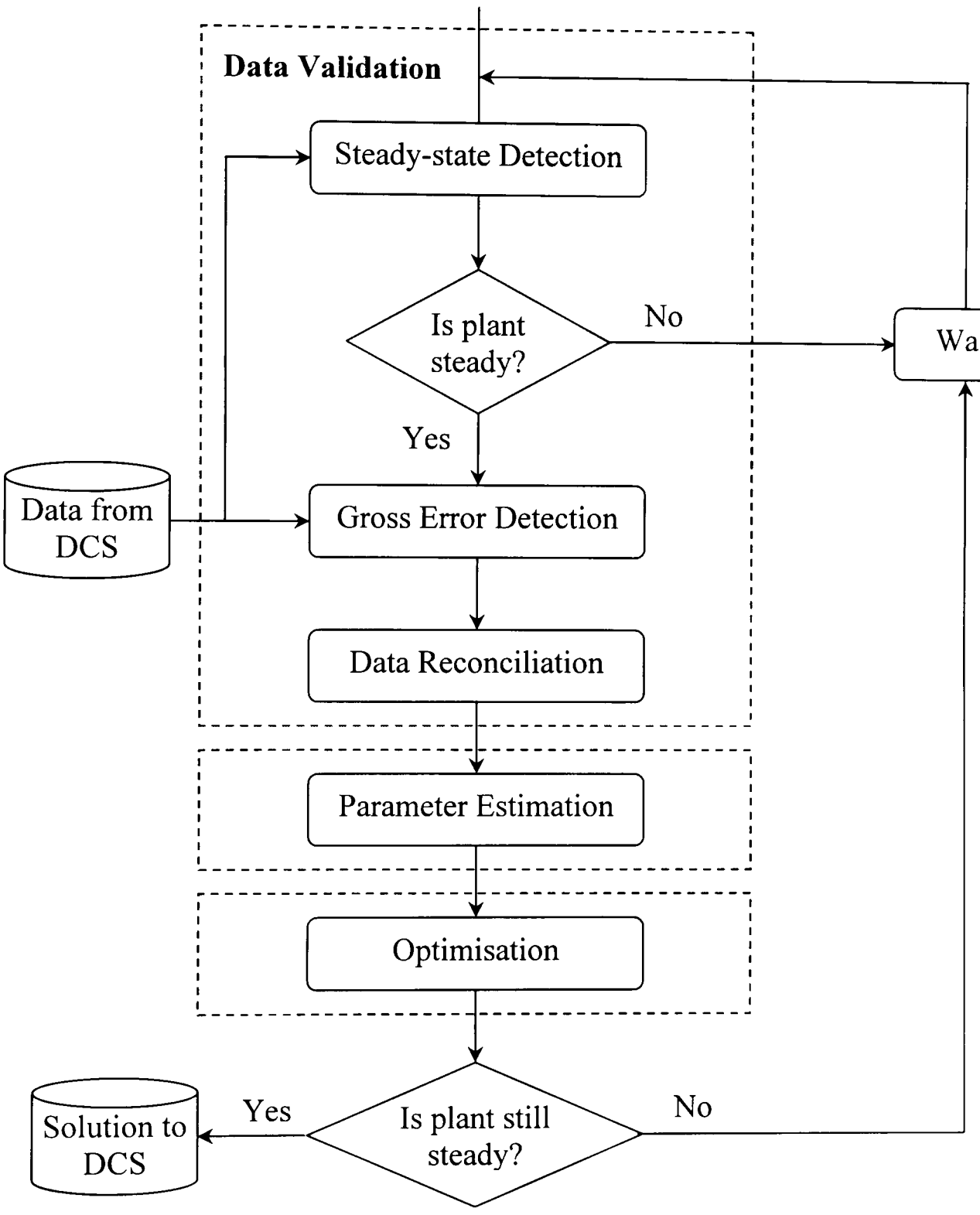

Figure (8-1): Schematic representation of on-line optimisation.

8.1.1. Methodology of On-line Optimisation

Industry practitioners have reported that after four decades there has been an increase in the application of on-line optimisation, but the same initial weaknesses or more generally speaking some common causes of poor performance still remain. These issues are related with the different steps of steady-state detection, data reconciliation and the optimisation itself.

On-line optimisers are directly linked to the plant instrumentation through the DCS. The DCS gathers real time data measurements from the process. This data is used to update and refine the plant model on a continuous basis.
On-line optimisation can be used in two ways:

1. **Open-loop:** In this case, the computations are carried out off-line, but the results are applied or given to the operator to apply on the system.
2. **Closed-loop:** Automatically implementing optimal set-points via the plant's DCS.

Closed-loop optimisers run continuously; responding to changes, ameliorating upsets and exploiting opportunities to create more profit.

The general on-line optimisation problem is to find those optimum operating points for which the system operates most efficiently. This involves the solving of three Non Linear Programming (NLP) problems: one for combined gross error detection and data reconciliation, one for simultaneous data reconciliation and parameter estimation and one for the actual optimisation.

Each of the three NLPs has the following form:

Optimise: Objective function $f(x)$
Subject to: Constraints from plant model

The objective function to be optimised, can be a joint distribution function for data reconciliation, a least squares for parameter estimation or a profit function for plant economic optimisation.

The constraints arise from a variety of causes. They are almost referred to as maximum allowable stresses or displacements according to normative and material capabilities. They can be material and energy balances, chemical reaction rate equations, thermodynamic equilibrium relations, capacities of process units. demand for product, availability of raw materials, and so on. It is usual to express constraints as inequalities; nevertheless inequalities can be converted into equality expressions with the help of slack variables.

The above NLP problem can be solved using many of the methods and techniques that have been developed during the years.

Any one of the above three optimisation problems: gross error detection and data reconciliation; parameter estimation; or optimisation can be solved separately.

As stated in the previous section, the key elements of on-line optimisation are:

- Steady-state detection
- Gross Error Detection
- Data Reconciliation
- Parameter Estimation
- Economic optimisation

These steps are carried out in real time as the process is moving from one operating point to another (Figure 8-1), and are described in detail in the following sections of this chapter.

8.2 AUTOMATIC DETECTION OF STEADY-STATE

Often, in the area of process control, rigorous steady-state detection is crucial for process performance assessment, simulation, optimisation and control. In general, at steady-state data is collected for safe, beneficial and rational management of processes.

However, identifying steady-state can prove to be a difficult task. This may be due to the process variables being noisy and measurements do not settle down at one value (Brown and Rhinehart, 2000).

Steady-state can be defined as an acceptable constancy of the mean values of measurements over a given period of time. Statistical methods based on the constancy of these variables are generally used to test for steady-state identification.

The issue of steady-state detection has been addressed by a number of researchers in the field. In what could be considered to be the first method developed for this purpose the crow et al. (1955) method which uses an F-test. This test is based on the ratio of two variances as measured by two different methods on the same set of data. The first variance is calculated as the mean-squared-deviation from the average of the most recent window data. While the second one is computed from the mean squared differences of successive data. If the process is at steady-state, the ratio of the two variances is unity, as the two methods produce unbiased estimates of the process variance. In practice, however, the ratio of the variances will not be exactly unity, due to limited sampling and random noise but will have a value near unity. If the process is not at steady-state, the ratio will be unusually large. The major drawbacks to this method include the considerable quantity of on-line data handling, as well as user expertise choices of data window length.

Narasimhan et al. (1986) presented a two-stage composite statistical test to detect departures from steady-state. The technique examines successive time periods and consists of two tests: the first one establishes whether the unknown covariance matrices were equal, and the second test establishes whether the means of the two periods were equals (using the Hotteling's T^2 test). This method presents a similar

drawback to that presented above, which is that it requires extensive computational effort.

In another work, Narasimhan et al. (1987) applied the mathematical theory of evidence to the detection of changes in steady-states which is an alternative to their earlier method, but it cannot be applied if the variables to be tested are not independent.

An alternative method (Betha and Rhinehart, 1991) is to perform linear regression over a data window, and use a t-test on the regression slope. The system is said to be at steady-state if the slope is equal to zero. This method also requires considerable data storage and computational effort as well as user-required choice of the data window length.

More recently, Loar (1994) presented a method based on a Statistical Process Control (SPC) moving average chart. In the same year, Alekman (1994) proposed a technique which compares the average calculated from a recent history to a standard based on an earlier history, then applies the t-statistic test to analyse whether the average is unchanged: the steady-state hypothesis. Again, storage and data processing is a computational burden.

Perhaps the most practical of the methods reviewed by authors in literature, is the Cao and Rhinehart (1995) method, which is a modification of the primitive F-test type of statistics of, crow et al. (1955). The ratio of two variances as measured on the same set of data by two different methods is calculated. However, in order to reduce computational effort, exponentially weighted moving average and variances are used instead of the conventional average or variance. These values are calculated from exponentially weighted moving average filters. In this case, data can be treated sequentially for steady-state identification without the need to select a time window required in most of the earlier methods which is the main drawback of these methods. For this technique to be effective on-line, the filter constants must be chosen judiciously and optimally. Critical values for R (ratio of variances), based on the process being at steady-state with independent and identically distributed variation, were also developed by Cao and Rhinehart (1997). An extension to the multivariable case was presented and experimentally demonstrated on a distillation column by Brown and Rhinehart (2000).

The method of Cao and Rhinehart (1995) is given below:

We consider the discrete filtered value of the measurement value X given by:

$$X_{f,i} = L_1 X_i + (1 - L_1) X_{f,i-1} \tag{8.1}$$

where:

X_i : is the process measured variable at time i

$X_{f,i}$: is the filtered value of X at time i

$X_{f,i-1}$: is the filtered value of X at time $i-1$

L_1 : is a filter factor

In this method, one needs to calculate two variances in order to obtain the R-statistic value. The first variance uses a filtered mean square deviation from the previous filtered values $v_{f,i}^2$ and the other one uses a filtered mean square difference of successive data $d_{f,i}^2$. The ratio of the two gives the value of the R-statistic.

The filtered mean square deviation from the previous filtered values $v_{f,i}^2$ is computed as follows:

$$v_{f,i}^2 = L_2 (X_i - X_{f,i-1})^2 + (1 - L_2) v_{f,i-1}^2 \tag{8.2}$$

where:

$v_{f,i}^2$: is the curent filtered mean square deviation

$v_{f,i-1}^2$: is the previous filtered mean square deviation

L_2 : is a filter factor

While filtered mean square difference of successive data $d_{f,i}^2$ is given by:

$$d_{f,i}^2 = L_3 (X_i - X_{i-1})^2 + (1 - L_3) d_{f,i-1}^2 \tag{8.3}$$

where:

$d^2_{f,i}$: is the curent filtered square difference of succesive data

$d^2_{f,i-1}$: is the previous square difference of succesive data

L_3 : is a filter factor

From the above equations, we can compute the R-statistic which can be used to ascertain the existence of the steady-state as follows (Appendix B):

$$R = \frac{(2 - L_1)v^2_{f,i}}{d^2_{f,i}} \tag{8.4}$$

The four equations given above represent the only requirements needed in order to check for steady-state condition. These requirements are direct, need no-logic, have low storage and low computational operation calculations. In total, there are three variables to be stored, ten multiplications, eight additions, and one comparison per observed variable.

The R-value found in equation (8.4) is compared with some critical value of R-statistic (R_{crit}). The system is said to be at steady-state if the R-value is found to have a distribution of values close to R_{crit}.

An R_{crit} value is selected and determined by the level of significance, α, alternately the confidence level, [100(1- α)], that we want to achieve. The null hypothesis is that the process is at steady-state. If the computed R-statistic from equation (8.4) is greater than R_{crit}, then we are 100(1- α) percent confident that the process is not at steady-state. Consequently, a value of R-statistic less than or equal to R_{crit} means the process may be at steady-state. We assign values of either "0" or "1" to a variable, say SS, which represents the state of the process. If R-calculated > R_{crit} (α) "reject" steady-state with 100(1- α) confidence, assign SS=0. Alternately, if R-calculated< R_{crit}(α) "accept" that the process may be at steady-state, and assign SS=1.

Cao and Rhinehart (1997) suggested some critical values for R, together with the filter factors L_1, L_2 and L_3. They came to a conclusion that filter values of

$L_1 = 0.2$ and $L_2 = L_3 = 0.1$ produce the best balance of Type I and Type II errors. Type I error is the error associated with wrongly rejecting the null hypothesis (process at steady-state) when it is true. While Type II error is the error associated with wrongly accepting the null hypothesis when it is false.

An alternative procedure to find the optimal values of L_1, L_2 and L_3 is given here (Bhat et al., 2003):

1. Select a value of L_3 (say 0.01)
2. Select a value of L_2 (say 0.05)
3. For these value of L_2 and L_3, start with a low value of L_1 (say 0.02), and calculate R_{crit}, Type II errors as well as steady-state detection lag.
4. Increase L_1 and repeat calculation of step 3 until allowable Type II error limit is crossed.
5. Increment L_2 by 0.05 and return to step 3. Keep on incrementing L_2 until the local minimum in terms of earlier detection of steady-state is obtained corresponding to given value of L_3.
6. Increment L_3 and return to step 2 until a global minimum is obtained.

A small drawback to the steady-state detection method outlined above is illustrated in the fact that data points cannot be auto-correlated at steady-state. Commonly, we get around this disadvantage by adjusting the sampling interval to eliminate auto-correlation when at steady-state.

The extension of the Cao and Rhinehart method for steady-state detection presented in this chapter to multivariable analysis was performed by Brown and Rhinehart (2000), and is given as follows:

It is assumed that a system is not at steady-state if at least one process variable is not at steady-state, and might be at steady-state if all variables might be at steady-state.

This can be easily tested with a single statistic:

$$SS_{process} = \prod_{i=1}^{N} SS_i \tag{8.5}$$

where N is the total number of process variables.

A steady-state condition is identified when:

$$P(SS_{process} = 1) = \prod_{i=1}^{N} P(SS_i = 1) = \prod_{i=1}^{N} (1 - \alpha_i) \tag{8.6}$$

$$(1 - \alpha_{process}) = \prod_{i=1}^{N} (1 - \alpha_i) \tag{8.7}$$

$\alpha_{process}$ is the global level of significance, while α_i are individual level of significance for each variable. From equation (8.7) it can be deducted that each individual level of significance α_i is equal to: $\alpha_i = 1 - \sqrt[N]{(1 - \alpha_{process})}$.

Two requirements are necessary for this method to be used. The first requirement is the traditional non auto-correlation of the data in every single variable. The second one is that there should not be cross-correlation between the variables (at steady-state). This means that in steady-state condition, the noise on one variable should not be correlated to the noise on another (Brown and Rhinehart, 2000).

8.2.1. Application

The above steady-state identification algorithm is applied and tested on the two CSTR system presented in detail in chapter 3.

The system has four outputs which are the concentrations of the two components A and B in the two tanks, i.e.: $y = (C_{a1}, C_{b1}, C_{a2}, C_{b2})^T$. In our example, these four variables are to be monitored for steady-state identification. Temperatures in the two tanks, T_1 and T_2, are the set points.

Equation (8.4), used for calculating the R-statistic was rearranged in order to avoid dividing by zero. The rearrangement was: If $(2-L_1)*v^2 > R_{crit}*d^2$ then SS = 0, else SS=1.

Choosing a global level of significance $\alpha_{process} = 0.05$, and four variables, then the individual level of significance for each variable is given by: $\alpha_i = 1-\sqrt[N]{(1-\alpha_{process})}$, where $i = 1,2,..,4$ and $N = 4$. Which results in $\alpha_i = 0.012$ for each variable. This means that, we were [100(1- α_i)] confident that variable i might be at steady-state when its corresponding SS_i value was high, and we rejected steady-state with [100(1-α_i)] confidence when its corresponding SS_i was low. This confidence level is equal to 98.8.

The three filter parameters L_1, L_2 and L_3 were chosen, following some trial and error procedure, to be: $L_1 = 0.06$, $L_2 = 0.01$ and $L_3 = 0.01$. These values proved to be the best values when applied amongst those tested. The values of R_{crit} were chosen to be as follows: R_{crit} = [1.4 ; 1.35 ; 1.4 ; 1.35]. R_{crit} is a vector of four values, as each element corresponds to one output variable of the process.

The simulations were carried out using MATLAB® and were started from the same initial operating point given by $T_1 = 307K$ and $T_2 = 302K$, yielding the following steady-state output values of the concentration of products A and B in the two tanks 1 and 2, $C_{a1}(0) = 0.04835$ [kmol/m^3], $C_{b1}(0) = 0.05165$ [kmol/m^3]. $C_{a2}(0)$=0.04137 [$kmol/m^3$] and $C_{b2}(0)$=0.058638 [$kmol/m^3$]. In order to perform the steady-state detection scheme, the set-points were changed four times during simulations to enable us to test the steady-state detection scheme in multiple cases. These changes were random.

Simulation results are presented in figures (8-3) to (8-13).

Figure (8-3) shows how sampled data taken from variable C_{a1} are organised. It is clear that there is no auto-correlation between the samples (which is requirement for this method to work). This is mainly due to the well choice of sampling time.

Cross-correlation between variables is illustrated in figure (8-4). It is shown that there is no cross-correlation (another requirement for the method to be applied and used) for example during steady-state between C_{b1} and C_{b2}, and between rest of variables in general.

Figures (8-4), (8-5), (8-6) and (8-7) show trajectories of the different values of R for different outputs, together with their corresponding SS variable values. It has to be mentioned that an SS_i variable is at a high level "1" if the corresponding output i is probably at steady-state, and low "0" otherwise. It is clear from the graphs that when the R-value for each output is below a selected point, steady-state is detected and the corresponding output might be at steady-state. In this case the corresponding SS variable is put to a high level. However, if the R-value is above the selected point given by R_{crit} the corresponding output is probably not at steady-state and its SS variable value is put to a low level.

Figures (8-8), (8-9), (8-10) and (8-11) show the trajectories of the different outputs of the system and their corresponding SS values. In these figures, it is demonstrated that when an output might be at steady-state, the corresponding SS variable value is set to a high level, indicating the variable might have reached steady-state with a confidence level of $[100(1-\alpha_i)] = 98.8$.

Figure (8-12), is a summary of the previous figures, in which all four output trajectories are shown, together with the overall $SS_{process}$ variable, which indicates that the system might be at steady-state when $SS_{process}$ is at a high level (which value is divided by a given factor in order to have all variables plotted on the same graph). By visual inspection, the method is shown to be working well as during the transient, the value of the $SS_{process}$ variable was always null, which indicates system not at steady-state with a confidence level of $[100(1-\alpha)] = 95$. While it goes up to a high level when the system might be at steady-state with the same level of confidence.

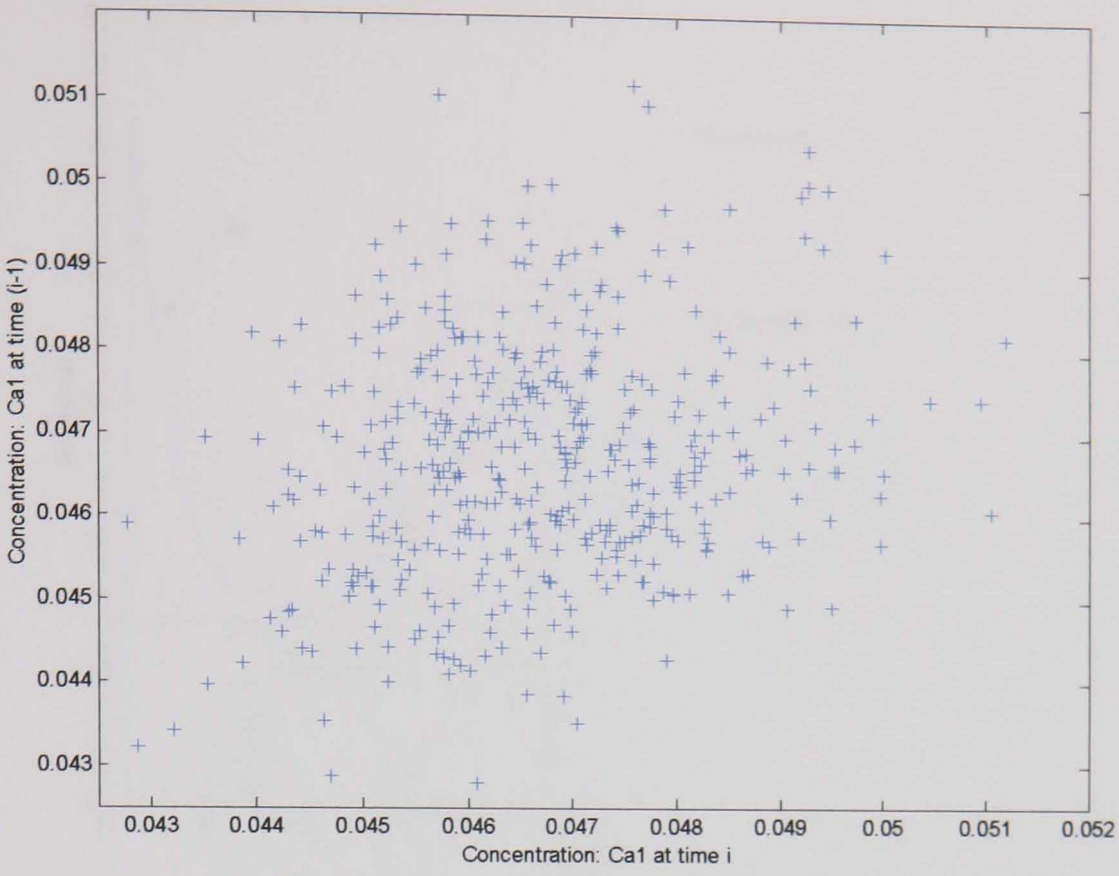

Figure (8-2): No auto-correlation in variable C_{a1}

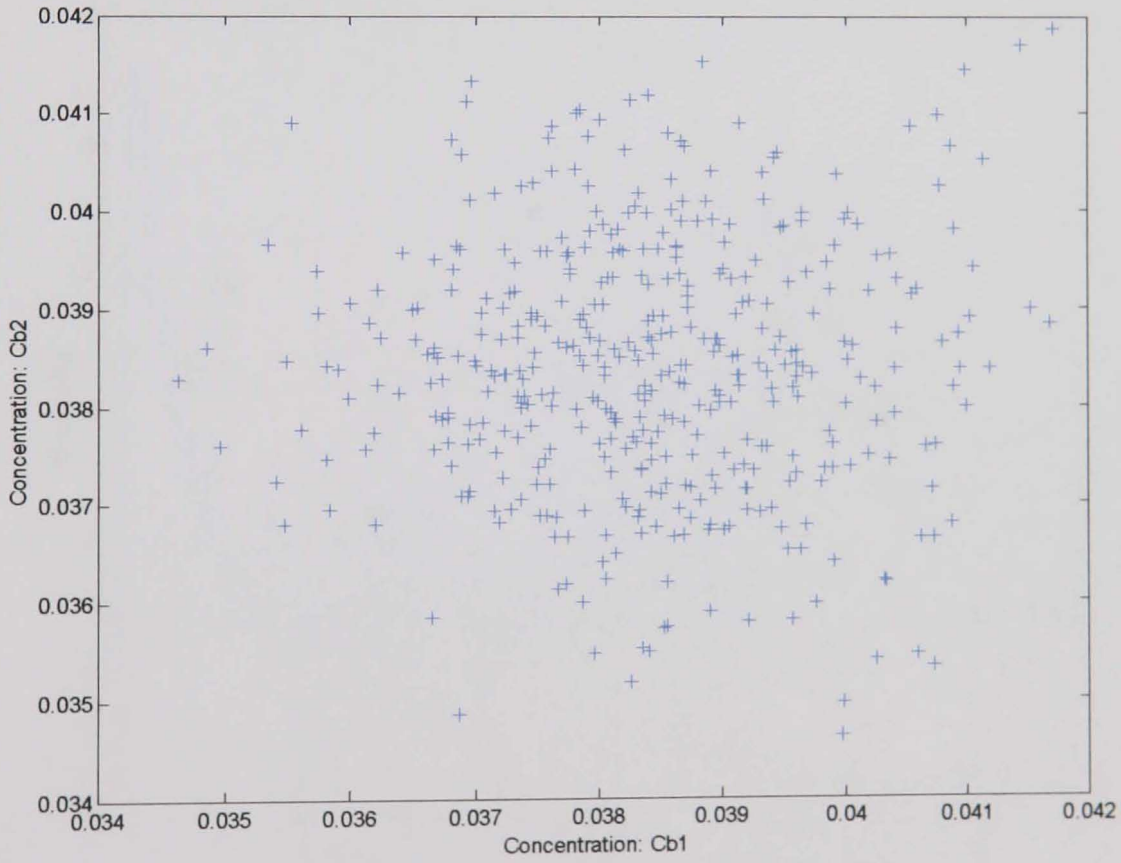

Figure (8-3): No cross-correlation between C_{b1} and C_{b2}

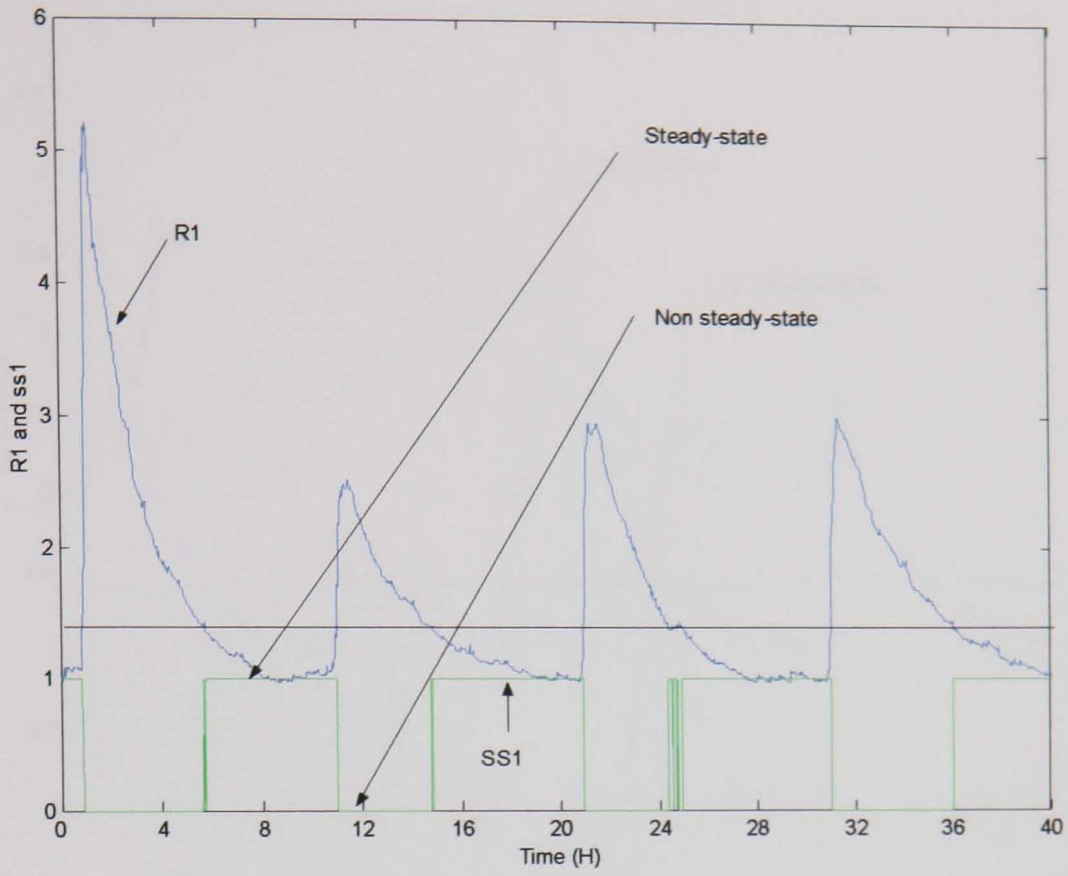

Figure (8-4): R-value and SS trajectories for C_{a1}

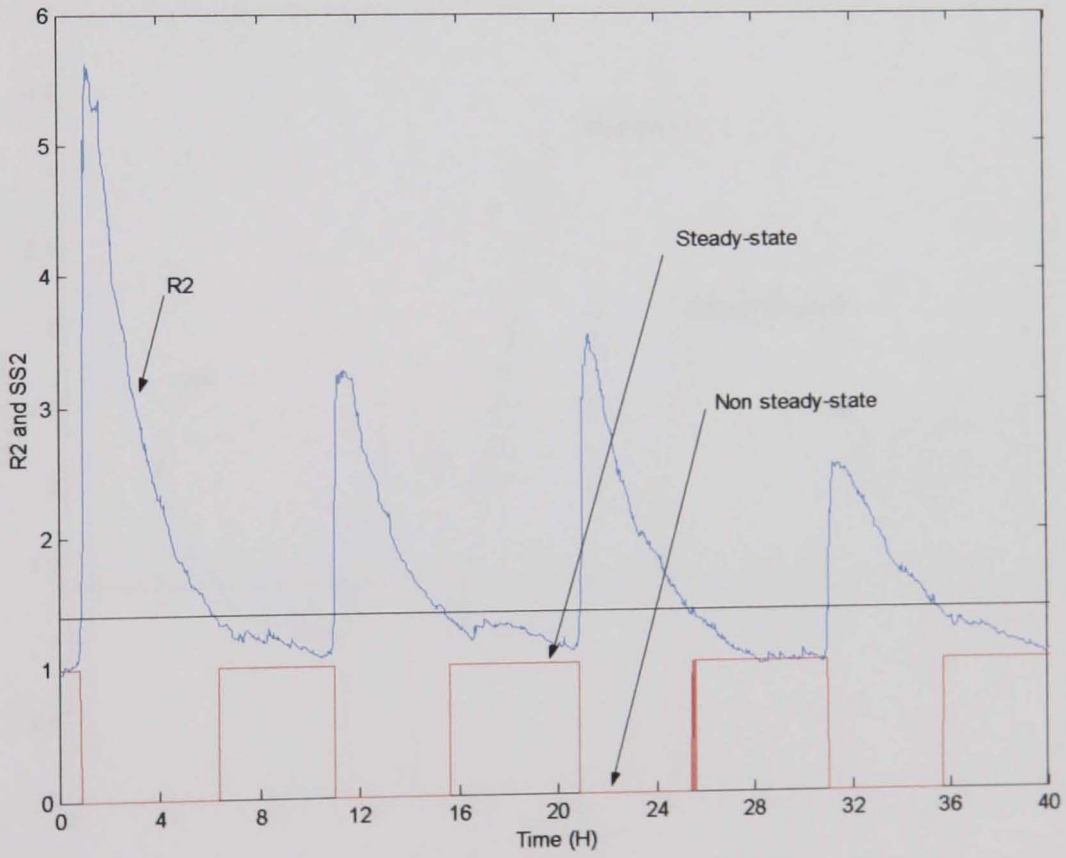

Figure (8-5): R-value and SS trajectories for C_{b1}

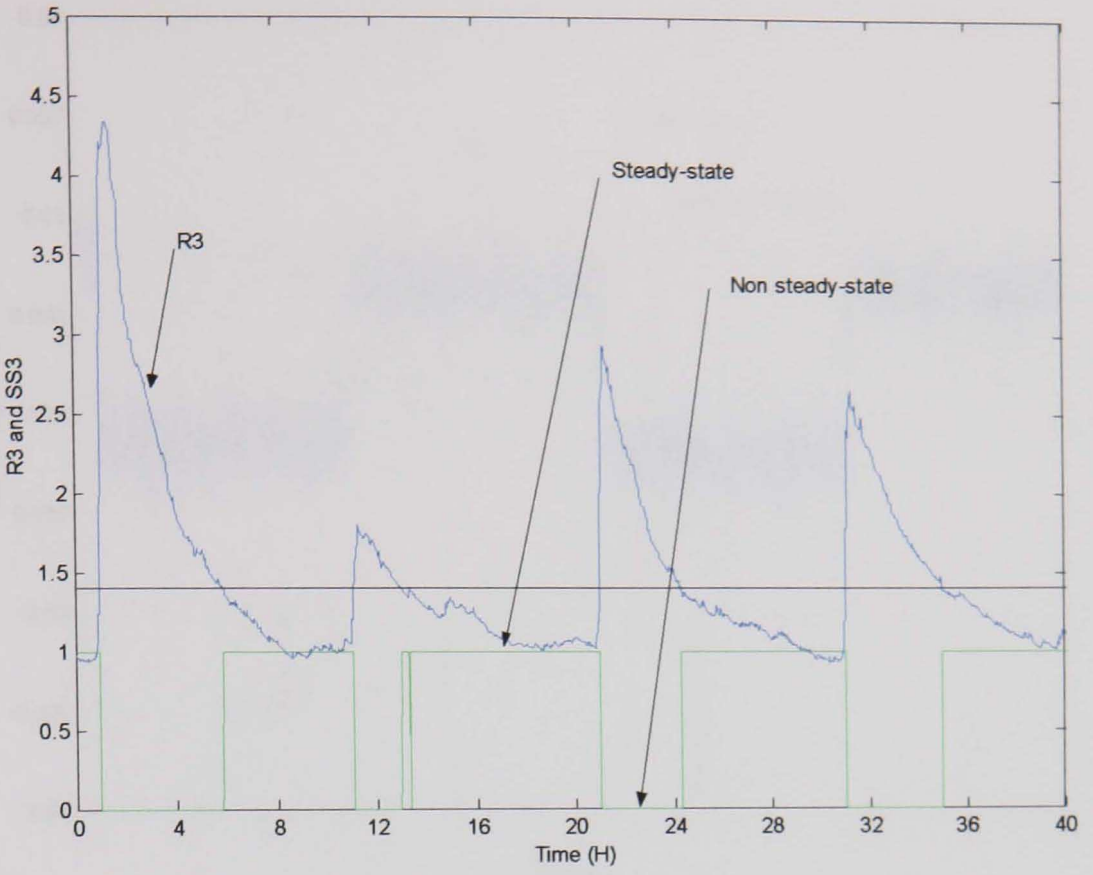

Figure (8-6): R-value and SS trajectories for C_{a2}

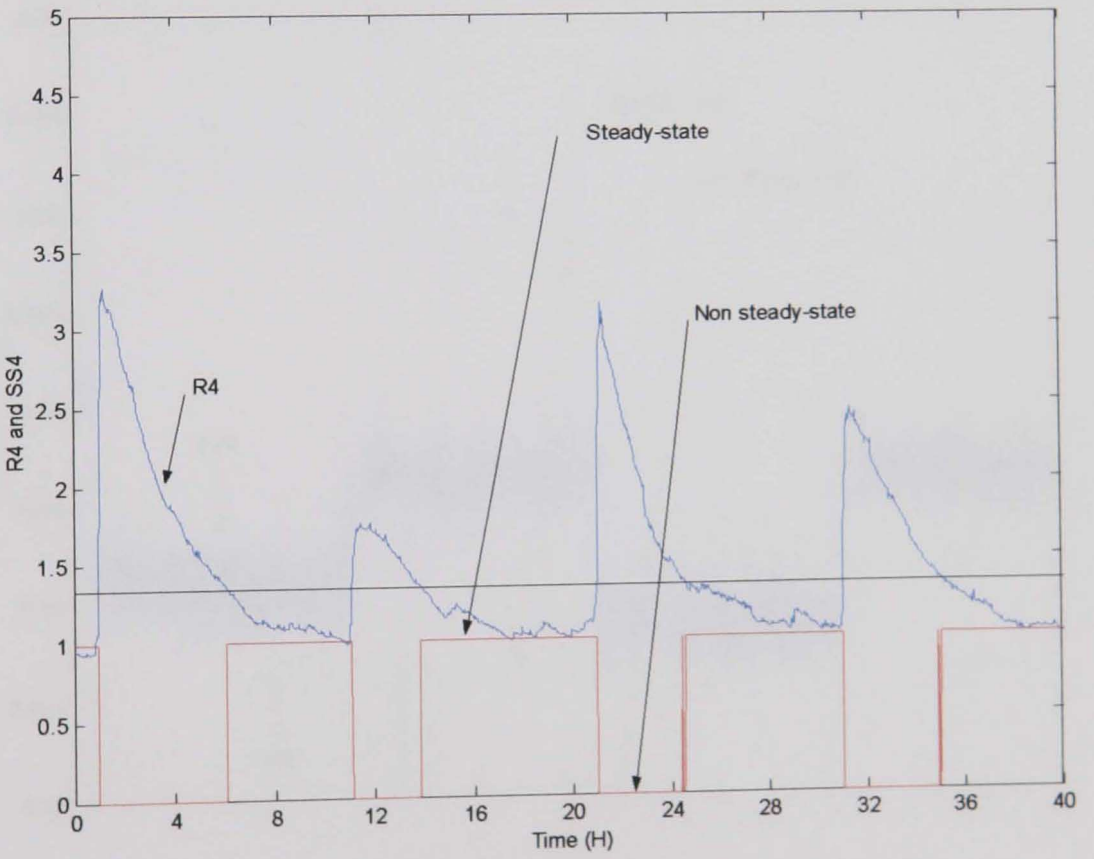

Figure (8-7): R-value and SS trajectories for C_{b2}

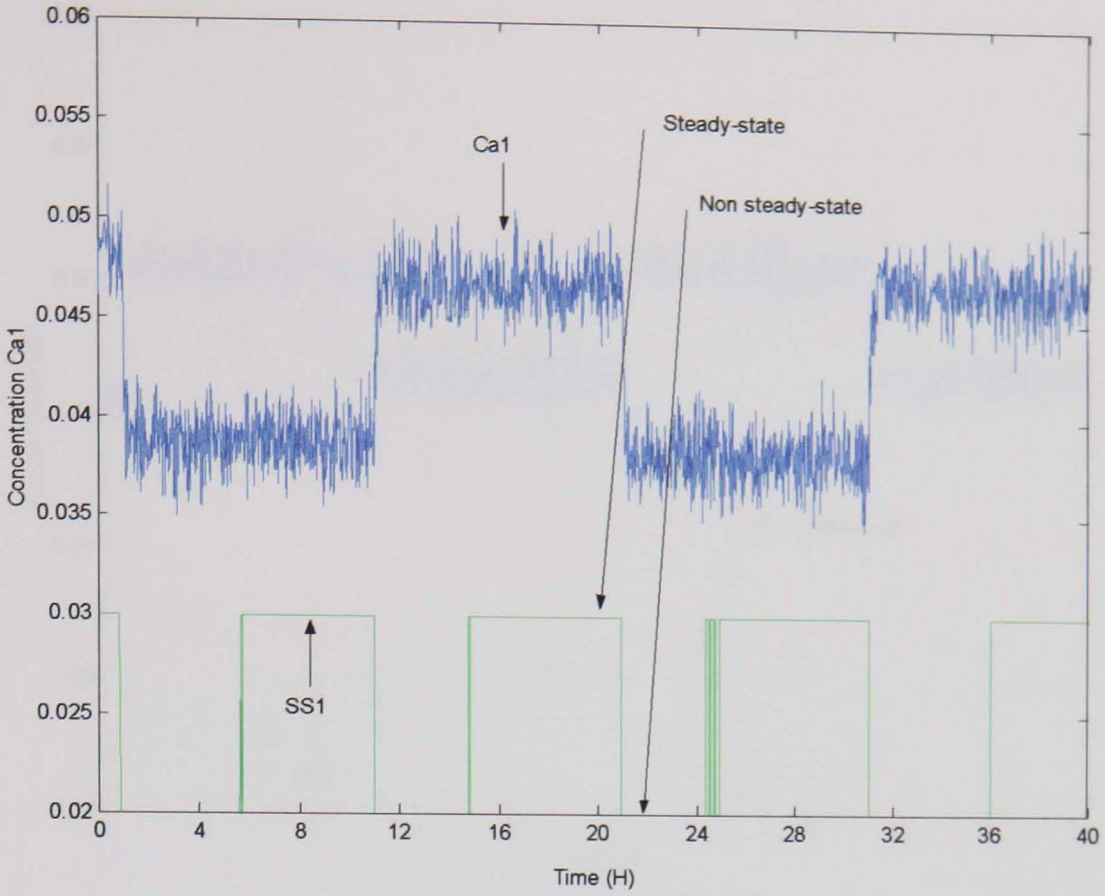

Figure (8-8): C_{a1} and SS trajectories

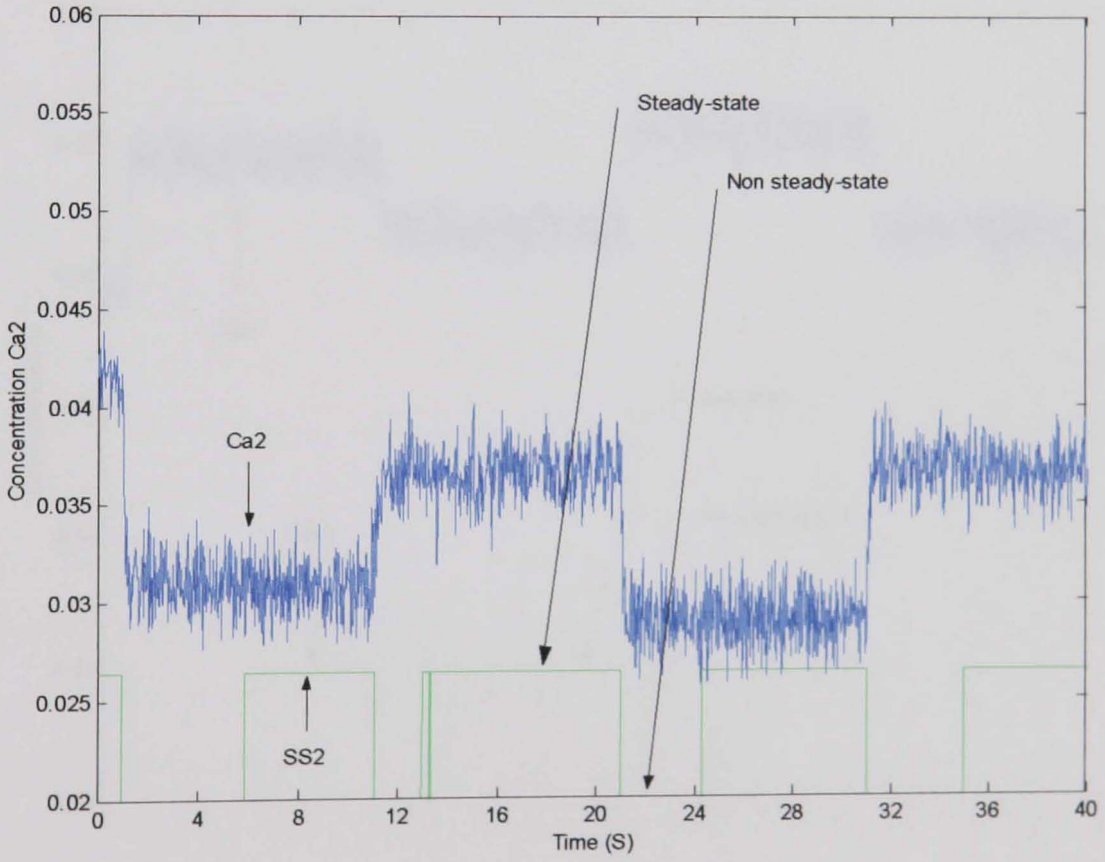

Figure (8-9): C_{a2} and SS trajectories

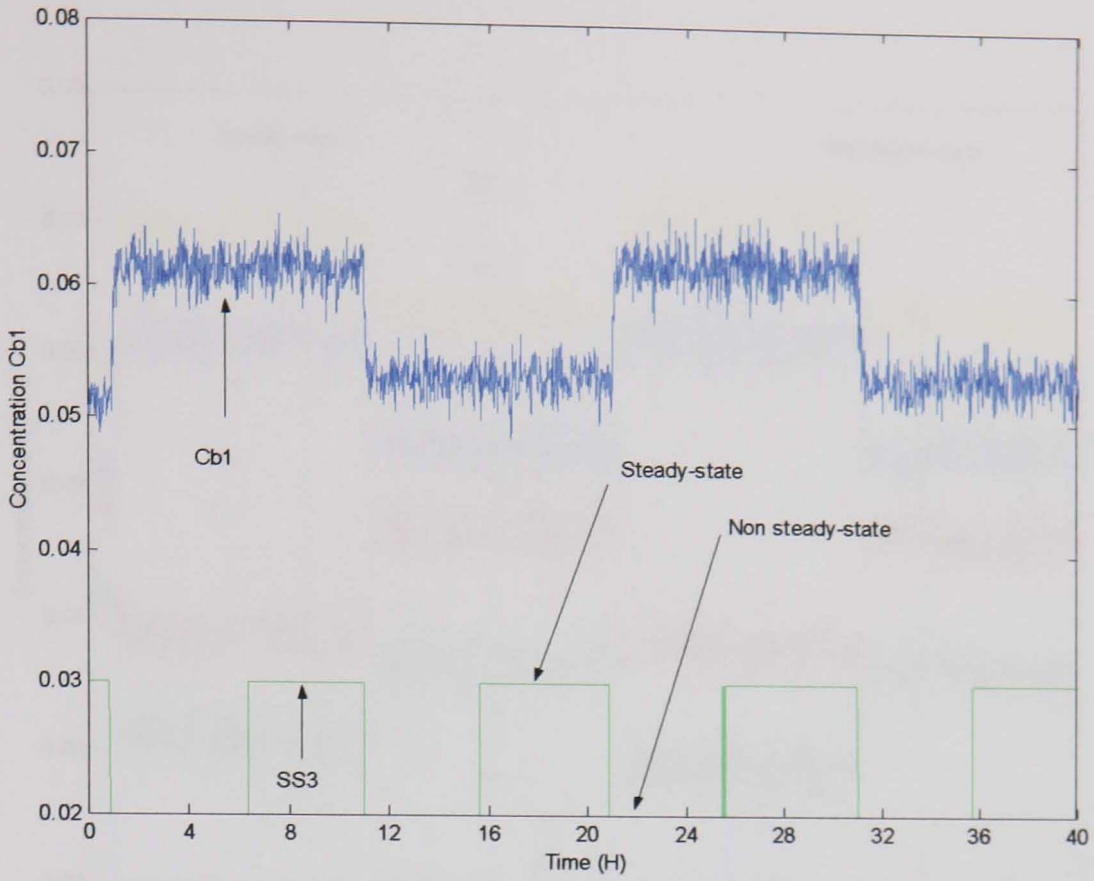

Figure (8-10): C_{b1} and SS trajectories

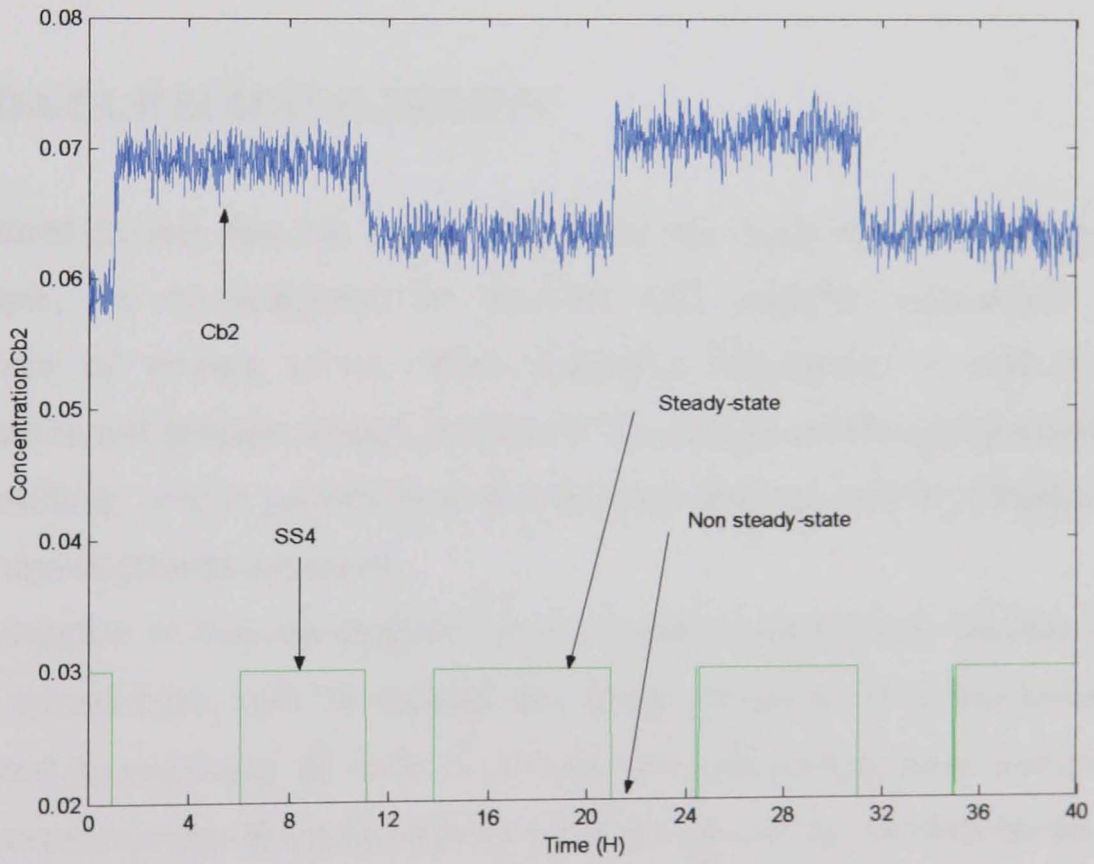

Figure (8-11): C_{b2} and SS trajectories

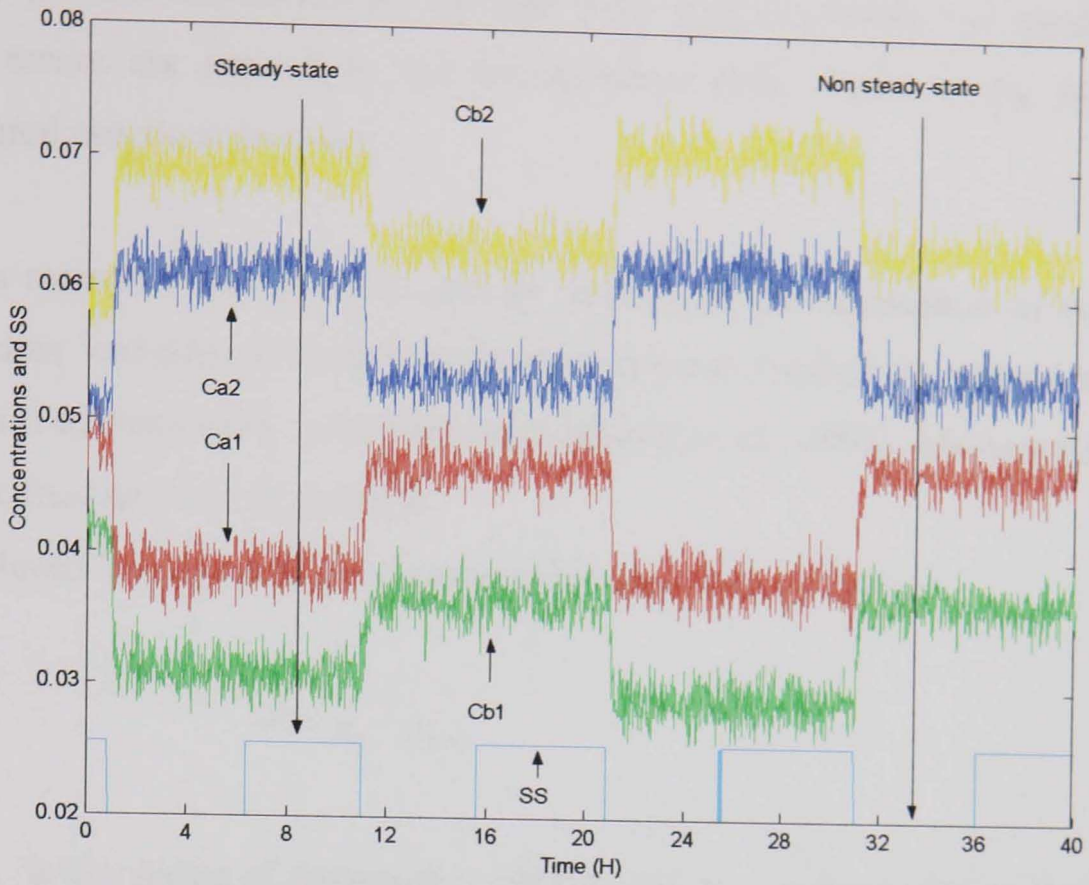

Figure (8-12): Output vector and overall SS trajectories

8.3 DATA RECONCILIATION

Measured process data may contain inherently inaccurate information because, for example, the measurements are obtained with imperfect instruments or the presence of random errors. When imperfect information is used for state estimation and process control, the state of the system is thus misrepresented and the resulting control performance may be poor and can lead to suboptimal and even unsafe process operation.

The objective of data reconciliation is to correct measured data variables so they obey natural laws, such as material and energy balances. Data reconciliation is achieved by removing all sorts of random errors that might have corrupted the data measurements. Because random errors are caused by the randomness of the measurements, they possess a zero-mean and are assumed to be normally distributed. This contrasts with gross errors, which are usually caused by non-random events such as process leaks and biases. Data reconciliation techniques

cannot remove such gross errors, and therefore gross error detection techniques are used in such circumstances (section 8.4). In this section, we assume only random errors are present in the measurement data, which is the frequently encountered practical situation.

The data reconciliation problem can be defined as the estimation of measured process data variables to reduce measurement error through the use of temporal and special or functional, redundancies (Liebman et al., 1992). Mathematically, it can be defined as an NLP problem.

As introduced in chapter 6, the measurement error ε is given by:

$$\varepsilon = y_m - y_{true} \tag{8.8}$$

where y_m is the vector of measured variables, and y_{true} is the vector of true values of variables.

The measurement errors are estimated by minimising sum-squares of standardised measurement errors, $\varepsilon^T V^{-1} \varepsilon$, subject to a set of constraints that describe the relationship among the variables, i.e., the process model. In other words:

$$\begin{aligned} \underset{y_{true}}{\text{Min}} \quad & \frac{1}{2} (y_m - y_{true})^T V^{-1} (y_m - y_{true}) \\ \text{Subject to: } & h(y_{true}) = 0 \end{aligned} \tag{8.9}$$

where V is the variance-covariance matrix where each element V_{ii} is σ_i^2, and is assumed to be the same for all data sets, and h is a set of algebraic equality constraint equations. h can be linear as well as nonlinear. The above equation is a general formulation of the data reconciliation problem. It is a NLP problem as stated earlier. Solving equation (8.9) gives the reconciled values of the process variables and the estimated measurement errors.

In the case of linear constraint equations, where material balances are considered only, so,

$$h(y_{true}) = Ay_{true} = 0 \tag{8.10}$$

where A is the Jacobian of the constraint equations. Then the optimisation problem of equation (8-9) has an analytical solution which can be written as:

$$y_{true} = y_m - VA^T(AVA^T)^{-1}Ay_m \tag{8.11}$$

and the vector of measurement adjustments is:

$$a = y_{true} - y_m = -VA^T(AVA^T)^{-1}Ay_m \tag{8.12}$$

However, if the constraint equations are nonlinear, that is h includes material and energy balances, chemical reaction rate equations, thermodynamic relations ...etc; the solution found in equation (8.11) is no longer applicable. In this case, the problem of equation (8.9) is solved by nonlinear programming techniques.

8.4 GROSS ERROR DETECTION

Gross error detection techniques are used to detect errors which are of a non-random nature. As mentioned previously, raw process data are subject to two types of errors: random errors and gross errors. Gross errors are of different types. Figure (8-13) shows some examples of these types (Narasimhan and Jordache, 2000). As data reconciliation techniques only remove random errors with the condition of non-existence of gross errors, a gross error detection phase is needed in order to deal with such non random errors.

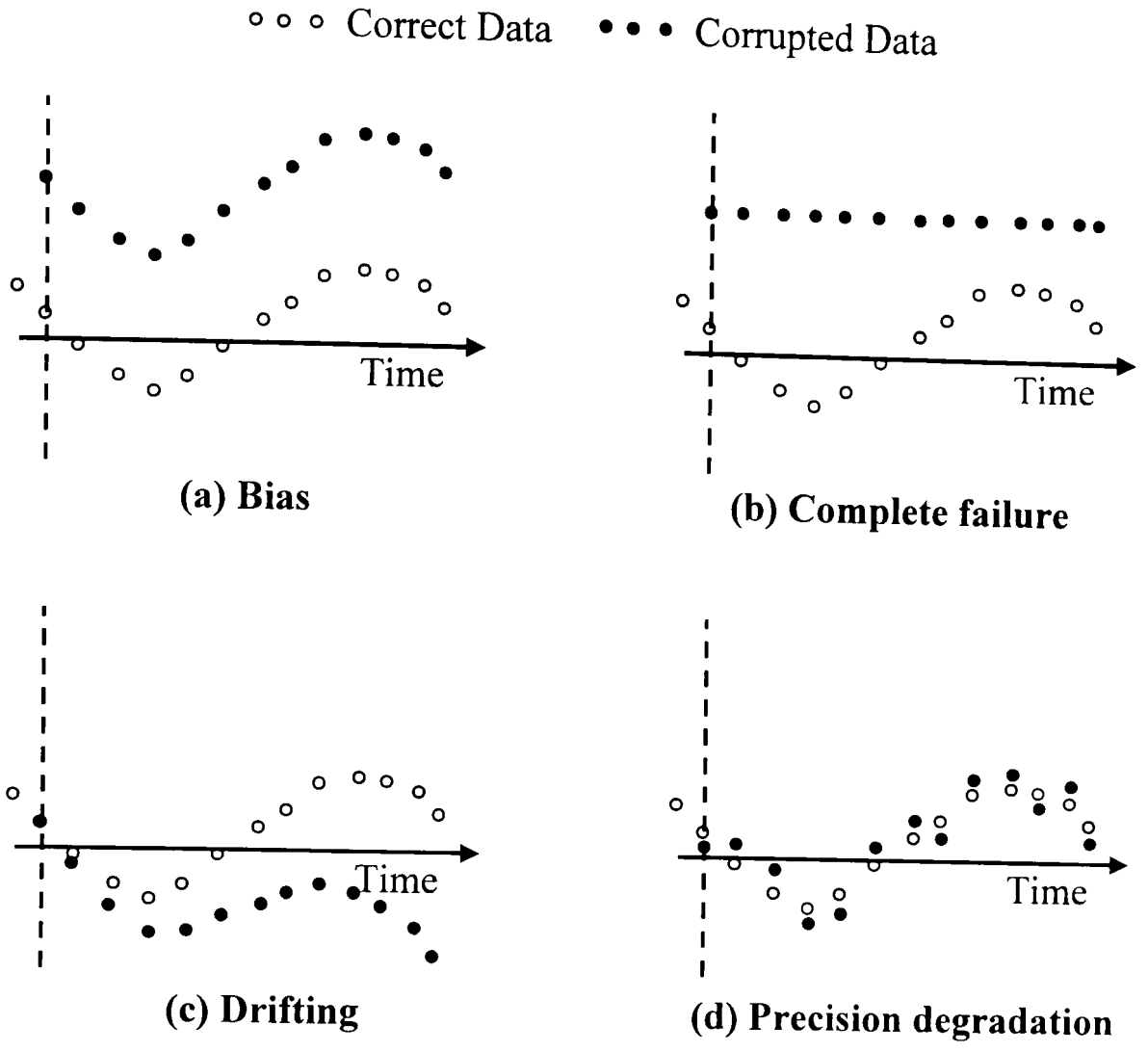

Figure (8-13): Types of gross errors (Narasimhan and Jordache, 2000).

Ideally, the main tasks of a gross error detection technique are to:
Detect the existence of gross errors, identify their locations, identify their types and to determine their sizes. After that, the gross errors are either corrected or eliminated.

Several approaches, such as time series screening, statistical, or neural network methods have been proposed and developed for gross error detection.

In the time screening approach, horizontal time screening is used to check for steady-state data, while vertical screening is used to filter out the gross errors in sampled data. This approach has practised in industrial applications. But because instrument errors and process leaks usually result in persistent gross errors, they cannot be detected or eliminated by time screening methods which are insensitive to such types of errors (Chen, 1998).

In the neural network approach, trained ANN's can be used for effective fault detection and diagnosis of chemical plants. Artificial Neural networks (ANN's) can be considered as collections of simple computational units which can take a numerical input and transform it into an output. Because they possess a considerable ability to learn from and adapt to their environment, they can be trained to learn associations between faults in systems and the vector of sensor measurement. Noise in process measurement is accommodated and therefore, effectively detected and identified. However, this approach presents one major drawback: it is computationally expensive. In other words, the complexity of computations increases with the number of sensors. For example, if the number of sensors is: 1000 sensor, training and its consequent computations become prohibitive.

The statistical approach however, requires a detailed model of the process. Knowledge of the measurement error structure is also essential. Generally speaking, a statistical method involves solving a NLP problem in order to estimate the errors. This kind of approach has been reported to be the most effective method for gross error detection.

We consider a linear (or linearised) measurement model represented by:

$$y_m = y_{true} + \varepsilon \tag{8.13}$$

$$\varepsilon = \varepsilon_r + \varepsilon_g e_i \tag{8.14}$$

where y_m, y_{true}, and ε are as above, the measurement error ε contains random errors (ε_r) and gross errors. ε_g is a vector of gross errors values, while e_i is a unit vector which all elements are zero except the ith element is 1.

The constraint residuals vector r is given by (Mah, 1990):

$$r = Ay_m - c \tag{8.15}$$

where A is as above, and c is a constant vector in the constraint.
The vector of measurement adjustments a can be given by:

$$a = y_{true} - y_m \tag{8.16}$$

It is typically assumed that there are no unmeasured variables (Sequiera, 2003).
Statistical methods for gross error detection can be divided into two categories:
The first category uses the distribution of constraint residuals r, while the second one uses the distribution of measurement adjustments a.
Methods based on constraint residuals include: the Global test, Nodal test and GLR (see chapter 6). In these methods, linearity of the constraints is assumed, and all variables must be measured.
Methods based on measurement adjustments include: the measurement test, Tjoa and Biegler's contaminated Gaussian distribution method, and the robust function method. Unlike the methods based on constraint residuals, the methods based on measurement adjustments need to reconcile process data first, then the reconciled data is tested or examined for errors (the reconciled data should follow a normal distribution (Sequiera, 2003)). Also, these methods allow unmeasured variables to be included in the plant model and can handle nonlinear constraints as well as linear. This kind of methods is classified as gross error detection and data reconciliation methods and will be covered in detail in the next section.
Below, is a list of the main contributions in the area of gross error detection:

- The Global Test (Ripps, 1965): This method uses hypothesis testing to test for gross errors presence. The null hypothesis test H_0, that there is no gross error, is used. It is based on the fact that the objective function of the data reconciliation problem has a Chi-square distribution at the minimum if the sampled data measurements are independent and normally distributed around their true values. This method was later modified by Almasy and Sztano (1975) to include cases where the variances of the measurements are not known.

- The Nodal Test (Reilly and Carpani, 1963; Mah et al.. 1976): Based on the constraint residuals r. In the absence of gross errors, r follows a m-variate normal distribution. Therefore

$$\phi_i = N_M^{1/2} \frac{r_i}{\sqrt{(AVA^T)_{ii}}} \tag{8.17}$$

follows a standard normal distribution, $N(0,1)$, under H_0. If ϕ_i is larger than the critical value based on a confidence level α, then it is concluded that there is at least one gross error present in the set of measurement that participates in the corresponding node balance. Rollins et al. (1996) proposed a strategy using this test on linear combination of nodes.

- The Measurement Test (Mah and Tamhane, 1982): this method is for combined gross error detection and data reconciliation and will be treated in section 8.5.

- The Generalised Likelihood Ratio Test (GLR): Originally by Wilsky and Jones (1974) and then Narasimhan and Mah (1987). This method was developed to identify different types of gross errors caused by either measurement biases and/ or process leaks with the GLR test. This method assumes linearity (or linearised) of the constraint equations, and requires a model that describes the effect of each type of gross error. For instance, the model of the system is given by: $y_m = y_{true} + \varepsilon + be_i$ for a measurement bias of magnitude b, and $h(y_{true}) = Ay_{true} - bm_j = 0$ for a process leak. Although suited for single gross error identification, a serial compensation strategy is often adopted in combination with the GLR method when dealing with multiple gross errors (Narasimhan and Mah, 1987).

- The Unbiased Estimation Technique (UBET, Rollins and Davis (1992, 1993)): This method considers both biased measurements and process leaks. It is only applicable to normally distributed errors, steady-state and linear constraints. First,

the global test is applied to detect any gross errors, then UBET is used to detect the number and location of gross errors by trial and error. using two test statistics (F and *Bonferoni* tests). Rollins and Roelfs (1992) extended this approach to the case where the constraints are bilinear.

- The Principle Component Analysis Test (PCA, Tong and Crowe (1996)): In this technique, a set of correlated variables is transformed into a new set of uncorrelated variables, known as principal component (PC), through an orthonormal matrix constructed by the eigenvectors of the covariance matrix H for the projected constraint residuals, i.e.,

$$d = W^T r \tag{8.18}$$

where W is constructed from the eigenvector of covariance matrix H of constraint residuals and satisfies

$$W = U\Lambda^{-1/2} \tag{8.19}$$

where matrix Λ is diagonal, consisting of the eigenvalues of H on its diagonal and satisfies

$$\Lambda = U^T HU \tag{8.20}$$

The matrix U consists of the orthonormalised eigenvectors of H so that

$$UU^T = I \tag{8.21}$$

Through this transformation, the new vector d becomes a new set of uncorrelated variables and is normally distributed, i.e., d - $N(0, 1)$. Then the gross errors are detected by the nodal test method as discussed previously.

8.5 COMBINED GROSS ERROR DETECTION AND DATA RECONCILIATION

In combined gross error detection and data reconciliation, data reconciliation is required to reconcile process data and to estimate the measurement errors for gross error identification. These methods, can be applied to models that are highly nonlinear and accept process variables that are unmeasured or unmeasurable.

There are several efficient methods for combined gross error detection and data reconciliation available. As mentioned previously, they are all based on the distribution function of the measurement adjustments. The process to proceed for gross error detection and data reconciliation is as described bellow:
First data reconciliation is conducted to reconcile all process data by maximising the joint distribution function subject to the process constraints. When all process data have random errors removed, a test statistic is applied to identify the gross errors.
Below are some of the most recognised methods in combined gross errors detection and data reconciliation:

8.5.1. Measurement Test

Developed by Mah and Tamhane (1982), this method is based on the distribution function of measurement errors. These errors are estimated by minimising the sum squares of the standardised measurement errors subject to the process model constraints. In other words:

$$\begin{aligned} \underset{y_m}{Minimise}: \quad & (y_m - y_{true})^T V^{-1} (y_m - y_{true}) \\ Subject\ to: \quad & h(y_m) = 0 \end{aligned} \tag{8.22}$$

This formulation is exactly the same as the one in equation (8.9) for solving the data reconciliation problem. This NLP problem can be solved by nonlinear

programming techniques, or can have an analytical solution if the constraints are linear as shown in equation (8.11). The measurement errors (equation 8.12) can then be used for a test to determine whether gross errors exist or not. This test is given as follows:

If $|a_i|/\sigma_i \geq C$ then measurement *i* contains a gross error.

where $a_i = y_{mi} - y_{truei}$ is the measurement error *i*, and *C* is a certain critical value. *C* is chosen from a table of a standard normal distribution function based on the selected significant level β for individual measurement, which is given by:

$$\beta = 1-(1-\alpha)^{1/m} \tag{8.23}$$

where α is the overall significance level, and m is the number of distinct values of $|a_i|/\sigma_i$ for all measurement errors.

The main advantages of using the measurement test is that it can detect sources of gross errors, can be applied to nonlinear constraint cases, and allows unmeasured variables in the model. A crucial disadvantage is that, as in most traditional gross error detection methods, it assumes a normal distribution of the measurement errors which is not always true.

The implementation of the measurement test algorithm as described in Serth and Heenan (1986) is given bellow:

Step 1: Compute reconciled values y_{true} and measurement adjustments $\boldsymbol{a}$ for the full system using equations (8.11) and (8.12).

Step 2: Compute standardised measurement adjustments: $\varepsilon_i = a_i/\sigma_i$ for each measurement.

Step 3: Compare each ε_i with the critical value of test statistic, C, selected from the table of standard normal distribution at the selected significant level β. If $|\varepsilon_i| > C$, then denote measurement i as a suspected measurement containing systematic errors and add the suspected measurements to set S. If $|\varepsilon_i| < C$ for all measurements, then go to Step 7.

Step 4: If the set S is empty, proceed to step 7. Otherwise, remove measurements contained in S from the system by nodal aggregation. This process eliminates some of the constraints and variables and yields a new system with reduced number of constraints and variables, and the original constraints ($A\, y_{true} = 0$) are reduced as $Bd = 0$. In the reduced constraints, d represents the variable vector as y_{true} excluding the variables that are eliminated by the nodal aggregation, and B represents the constraint coefficient matrix as A excluding the rows and columns that are corresponding to the eliminated constraints and variables from the nodal aggregation. Also, the measurement vector y_m is reduced to vector w that excludes the eliminated measurements from nodal aggregation, and let T denote the set of measurements contained in w. In addition, the variance and covariance matrix of measurement errors V is reduced to matrix P that excludes the variances and covariances of the eliminated measurements.

Step 5: Repeat Step 1 to compute the estimated values of process variables and measurement adjustments by equations (8.11) and (8.12) with A, y_m, and V replaced by B, w, and P, respectively.

Step 6: Compute corrected values of variables in S by solving $A\, y_{true} = 0$ with the variables in set T specified with the estimated values from step 5 and the variables in set R specified with the original measured values. R is a set of variables that were eliminated during the nodal aggregation and whose measured data does not contain gross error, i.e., R = U - (S$\cup$T), where U is the set of all variables in the system. Then go back to Step 2.

Step 7: If the set S is empty, then all measurements do not contain gross error, and the estimated values of process variables in step 1 by equation (8.11) are the reconciled values of all process variables. Otherwise, the set of reconciled values is obtained from the values computed in step 6 for the variables containing gross errors in set S, the reconciled values computed in step 5 for the variables in set T, and the original measured values for the variables in set R.

The above procedure ensures the detection of gross errors in systems with nonlinear model constraints by minimising the sum squares of the standardised measurement errors. However, the measurement test as it was originally proposed, helps spread the gross error to all the measurements, leading to large residuals corresponding to good measurements. This results in the measurements being erroneously identified as containing gross errors (Type I errors). An iterative elimination method was developed (Ripps, 1965; Serth and Heenan, 1986), to overcome this problem, and was incorporated in the measurement test method to form the Iterative Measurement Test (IMT). In this method, only the measurement corresponding to the largest standardised error is automatically identified as containing a gross error and is deleted each time the measurement test is applied. The IMT reduces type I errors (a gross error is identified, while there is none) significantly, but again, this method encounters one major problem: the set of reconciled flow rates may contain negative values or absurdly large values remains. Therefore, the Modified Iterative Measurement Test (MIMT) was proposed by Kim et al. (1997) to avoid this problem. This technique was implemented on a simple CSTR system and results were compared when using nonlinear programming techniques with those using successive linearisation. The results showed that the MIMT with nonlinear programming technique performs better and provides more accurate results (Kim et al., 1997). The IMT and MIMT algorithm procedures can be found in Appendix C.

To summarise, the measurement test is based on measurement errors. It necessitates the elimination of random errors first by applying data reconciliation.

The method then uses the error estimates to directly detect and locate gross errors. It can be applied to nonlinear constraint cases and allows unmeasured variables in the model. However, the major drawback of the method is that it assumes a normal distribution of the errors, which cannot describe the distribution behaviour of gross errors. The IMT and MIMT versions of the measurement test overcome this kind of problems.

8.5.2. Tjoa and Biegler's Contaminated Gaussian distribution

This method uses a two mode (random and gross errors) Gaussian distribution of the function of measurement error. It was proposed by Tjoa and Biegler (1991) for gross error detection and data reconciliation. The proposed distribution function of the measurement errors was given as:

$$P(y_{mi} \mid y_{truei}) = (1-\eta)P(y_{mi} \mid y_{truei}, R) + \eta P(y_{mi} \mid y_{truei}, G) \tag{8.24}$$

where $P(y_{mi} \mid y_{truei}, R)$ is the probability distribution function for the random error, $P(y_{mi} \mid y_{truei}, G)$ is the probability distribution function for the gross error, with a gross error occurring with a prior probability η, and $1-\eta$ for a random error. Because the distribution function of random errors is normal, with zero mean and known variance σ^2, it can be written as follows:

$$P(y_m \mid y_{true}, R) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(y_m - y_{true})^2}{2\sigma^2}} \tag{8.25}$$

So much so, the distribution function for a gross error which is assumed to be normally distributed with zero mean and a larger variance $(b\sigma)^2$ (with b>>1), can be expressed as:

$$P(y_m \mid y_{true}, G) = \frac{1}{\sqrt{2\pi}b\sigma} e^{\frac{-(y_m - y_{true})^2}{2(b\sigma)^2}} \tag{8.26}$$

Tjoa and Biegler used the global likelihood function for all measurements, which is the product of the individual distribution functions for each measurement and called it the Contaminated Gaussian Distribution function. It is given by:

$$P(y_m \mid y_{true}) = \prod_{i=1}^{N} P(y_{mi} \mid y_{truei}) = \prod_{i=1}^{N} \{(1-\eta)P(y_{mi} \mid y_{truei}, R) + \eta P(y_{mi} \mid y_{truei}, G)\} \tag{8.27}$$

This function is maximised or its negative logarithm minimised subject to the constraints in plant model in order to reconcile process measurements as follows:

$$\underset{y_{true}}{Minimise}: \quad -\sum_i \left\{ \ln\left[(1-\eta) e^{\frac{-(y_{mi}-y_{truei})^2}{2\sigma_i^2}} + \frac{\eta}{b} e^{\frac{-(y_{mi}-y_{truei})^2}{2(b\sigma_i)^2}} \right] - \ln\left[\sqrt{2\pi}\sigma_i\right] \right\} \tag{8.28}$$

$$Subject \quad to: \quad \begin{matrix} h(y_{true}) = 0 \\ y_{true}^L < y_{true} < y_{true}^U \end{matrix}$$

where all the variables are as previously defined, and y_{true}^L and y_{true}^U are lower and upper bounds on the process variables.

The above nonlinear data reconciliation problem (equation 8.28) is solved using nonlinear programming techniques, where values for η and b are needed.

After data reconciliation is accomplished, the measurements are analysed for gross error presence by applying the following test statistic:

If: $\eta P(y_{mi} \mid y_{truei}, G) \geq (1-\eta) P(y_{mi} \mid y_{truei}, R)$, then a gross error exists on measurement *i*. Alternatively, use

$$|\varepsilon_i| = \left| \frac{y_{mi} - y_{truei}}{\sigma_i} \right| > \sqrt{\frac{2b^2}{b^2-1} \ln\left[\frac{b(1-\eta)}{\eta} \right]} \tag{8.29}$$

If equation (8.29) is satisfied, then measurement *i* contains gross error. Otherwise, no gross error is present in this measurement.

The following procedure describes how the contaminated Gaussian distribution method is implemented:

1. Solve the data reconciliation problem given by equation (8.28) to give the reconciled values for measured and unmeasured variables, and then compute the vector of measurement adjustments $a = y_{true} - y_m$.

2. Apply the test statistic (equation 8.29) to determine if the measurement is contaminated by a gross error. If the test is positive, i.e.: the measurement is affected by a gross error, replace it with its appropriate reconciled value. This test is applied on all measurements. When finished, construct a new set of measurement using the reconciled data which replace the measurements affected by gross errors, and those original measurements which contain random errors only. This new set of measurements contains random errors only, and it is used in simultaneous data reconciliation and parameter estimation to update plant parameters for on-line optimisation.

As mentioned earlier, the values of η and b are two parameters needed by the contaminated Gaussian distribution algorithm. b is a tuning parameter to shape the distribution. Increasing b will reduce the effect of a gross error on the estimation and increases the robustness of this approach. However, it will decrease the asymptotic efficiency to the normality. In practice, the value of b is usually chosen between 10-20, and the weight coefficient for a measurement with a gross error is 100-400 times smaller than one with a random error. As for the second parameter which is the prior probability of a gross error, η, if no prior information about the errors is available, then the value of $\eta = 0.5$ is recommended (Chen, 1998).

It has to be said that the contaminated Gaussian distribution method is more effective than the measurement test method (Chen, 1998). This is illustrated in the fact that the contaminated Gaussian distribution method incorporates the distribution pattern for both random and gross errors, and it is able to rectify both random and gross errors in measurements. Another feature that the contaminated

Gaussian distribution method posses, is that it can directly locate the gross error and gives unbiased estimation for all reconciled data.

8.5.3. Robust Function Methods

In robust statistics, instead of assuming an ideal distribution of the measurement errors as in classical approaches, an estimator which will give unbiased results in presence of this ideal distribution is constructed. At the same time, this estimator is insensitive to deviations from ideality. The robust function is to be insensitive to the presence of gross errors in sampled data when this function is used to conduct data reconciliation, and it still maintains a high efficiency (lower dispersion) that indicates the accuracy of estimation (Huber, 1972 and Seber, 1984).

Several different classes of estimators have been developed: the L-estimators, R-estimators and M-estimators. The most important ones are the M-estimators, which are generalisations of the maximum-likelihood estimator (Albuquerque and Biegler, 1996). It is described in the following:

$$\begin{aligned} &\min_{y_{true}} -\sum_{i=1}^{n} \rho(y_{mi}, y_{truei}) \\ &\text{Subject to: } h(y_{true}) = 0 \\ &\qquad y_{true}^{L} \le y_{true} \le y_{true}^{U} \end{aligned} \tag{8.30}$$

Several distribution functions have been proposed in the literature. To note: the Lorentzian and Fair function.

The Lorentzian distribution is given by (Jonston and Kramer, 1995):

$$\rho(\varepsilon_i) = \frac{1}{1 + \frac{1}{2}\varepsilon_i^2} \tag{8.31}$$

where ε_i is the standardised measurement error comprising both random and gross error, i.e.: $\varepsilon_i = (y_{mi} - y_{truei})/\sigma$.

The Fair function which is convex and has got continuous first and second order derivative. It is given by (Albuquerque and Biegler, 1996):

$$\rho(\varepsilon, c) = c^2 \left[\frac{|\varepsilon|}{c} - \log\left(1 + \frac{|\varepsilon|}{c}\right) \right] \tag{8.32}$$

where ε_i is the standardised measurement error and c is a tuning parameter.

A boxplot technique is used to test for gross error presence. The centre of the box represents the median, and the sides represent the quartiles. Outliers are detected by computing the order statistics (median and quartiles) and their distances from these. In this test, the interquartile-range is defined by:

$$d_F = F_u - F_l \tag{8.33}$$

where F_u and F_l are the third and first quartiles respectively. The outlier cutoffs are given by: $F_l - \alpha d_F$ and $F_u + \alpha d_F$, with α usually set to 1/3 (Albuquerque and Biegler, 1996). Measurements outside the cutoff are considered as outliers.

Robust statistical methods were developed to overcome difficulties with data that contain gross errors and that does not follow the ideal normal distribution. This kind of method uses an objective function that is insensitive to gross errors in sampled data and known to have the advantage of having a very simple mathematical form and also for having very convenient properties for optimisation. Moreover, robust methods do not need any prior knowledge of the error structure of the outliers and of the data. However, the accuracy of estimation from these methods will be slightly lost because robust functions have a flatter shape that gives larger variation in the estimation. Also, the test used to detect gross errors for robust methods is not as straight forward as the contaminated Gaussian distribution, although the boxplot and dotplot methods from exploratory statistics (Albuquerque and Biegler, 1996) may be used to identify the gross errors

of sampled data. Moreover, such robust methods tend to cause more type I errors that a gross error does not exist but was positively identified.

Chen (2003), reporting the results of theoretical and numerical evaluations of the three methods for combined data reconciliation and gross error detection presented above, showed that:

The Tjoa and Biegler's contaminated Gaussian distribution method has the best performance for measurements contaminated by random errors and moderate gross errors of the range (3σ -30σ).

The robust method using Lorentzian distribution function proved to be more effective for measurements with very large gross errors (larger than 30σ).

The measurement test provides a more accurate estimation for measurements containing random errors only. However it gives significantly biased estimation when gross errors larger than 10σ are present in the measurements

8.6 PARAMETER ESTIMATION

In on-line optimisation, some model parameters are regularly updated. In reality, there is generally a difference between the model and the real process it is representing, so the aim is to reduce model-reality (or plant-model) differences.

For an unconstrained problem, a model of a plant can be expressed function of process variables x, fixed parameters P_α, and variable or updated parameters P_β, as:

$$h(x, P_\alpha, P_\beta) = 0 \tag{8.34}$$

The vector of updated parameters P_β is estimated from process measurements y with the help of an appropriate parameter estimation problem formulation. In case the data measurements are affected by noise, data reconciliation can be used to reduce the effect of noise and data variability, and hence improve the estimation.

Deming (1943) originally formulated the general problem of parameter estimation by taking into account all the errors in measured variables. Britt and Luecke (1973) presented general methodology for the parameter estimation of error-in-variables model. According to them, there are two types of models for parameter estimation. The first type is the explicit model. In this model, measurements are divided into two sets: dependent and independent variables. The independent variables are measured with greater accuracy than the dependent ones. In fact, the dependent variables are expressed as an explicit function of measured variables. Parameters can be estimated by such procedures by minimising the sum of squared errors of dependent variables (least squares method) or maximizing the likelihood function, a probability distribution function of the measurement errors of dependent variables (maximum likelihood method). This is an unconstrained optimisation problem, and linear regression method is one of examples for this type of estimation.

The second type of model is the implicit or error-in-variables model. Where errors are present in all measurements and the variables cannot be divided into dependent and independent variables as in the explicit model. The constraints of process models are implicit. Therefore, the optimisation problem of parameter estimation must be formulated as a constrained optimisation problem. In error-in-variables models, the vector of measurements y is divided into measured and unmeasured variables y_m and y_u.

Often, data reconciliation and parameter estimation are joint and performed together in what's called Simultaneous data reconciliation and parameter estimation. They use error-in-variables models, where all the measured variables have errors in them, and are given by the following traditional relationship:

$$y_m = y_{true} + \varepsilon \tag{8.35}$$

where all the variables are as previously defined.

The vector of updated parameters and reconciled variables P_β and y_{true} are found by minimising the objective function subject to the equality constraints in equation (8.34).

Therefore, the general formulation of the Simultaneous data reconciliation and parameter estimation is a least squares constrained optimisation problem.

$$\begin{aligned} &\min_{P_\beta}: \ (y_m - y_{true})^T V^{-1} (y_m - y_{true}) \\ &\text{subject to: } h(y_{true}, P_\alpha, P_\beta) = 0 \end{aligned} \tag{8.36}$$

Britt and Luecke (1973) described the use of Lagrange multiplier method to solve the optimisation problem of equation (8.36). The constraints are implicit nonlinear, and there is no analytical solution for it. The authors developed an iterative linearisation technique to solve this nonlinear problem. They linearised the nonlinear constraints using Taylor expansion at the solution point of the last linearisation, and then iteratively searched for the optimal solution. They came to a conclusion that their algorithm provided a feasible approach to the general parameter estimation problems.

In order to eliminate plant-model mismatch, there have been several proposals to integrate the parameter estimation problem with the system optimisation problem in one whole formulation (Haimes and Wismer, 1972; Roberts, 1979; Cheng and Zafiriou, 2000). The main contribution to the subject and relevant to our study is of no doubt the Integrated System Optimisation and Parameter Estimation (ISOPE) algorithm developed by Roberts (1979). The general idea behind the ISOPE algorithm is to replace the model-based optimisation problem, by an equivalent problem which is ultimately decomposed into a parameter estimation and a modified model-based optimisation problems. This algorithm was introduced in chapter 2, and will be reviewed in the following subsection.

8.7 VARIANCE-COVARIANCE MATRIX ESTIMATION

One crucial point when solving data reconciliation, gross error detection and parameter estimation problems, is the choice of the variance-covariance matrix, V. Several methods to estimate this matrix have been reported in the literature. To note, the direct method based on the mean of the measurement samples, and the indirect method which uses the estimates of the constraint residuals calculated using the direct method.

In the direct method, the variance-covariance is given by:

$$\sigma_{ij}^2 = \mathrm{cov}(y_{mi}, y_{mj}) = \frac{1}{n-1}\sum_{k=1}^{n}(y_{mik} - \bar{y}_{mi})(y_{mjk} - \bar{y}_{mj}) \qquad (8.37)$$

where $\bar{y}_{mi}$ is the mean of the measured variable y_{mi} over a number of samples n, and is given by:

$$\bar{y}_{mi} = \frac{1}{n}\sum_{k=1}^{n} y_{mik} \qquad (8.38)$$

The variance-covariance matrix of the measurement errors V is formed from the above formulation, and is given by:

$$V = [\sigma_{ij}^2] \qquad (8.39)$$

The above formulation of the variance-covariance matrix is only valid for cases where there are no gross errors present on the measurements, and the sampled data are independent of each other. In other words, the n samples have to be taken from the same steady-state operating point.

From equation (8.39), one can deduct the covariance matrix of constraint residuals H (in the case of linear constraints) as follows:

$$H = AVA^T \tag{8.40}$$

This formulation was used by Keller et al. (1992), in his indirect method to avoid or eliminate the dependency between sampled data which occur in the direct method.

Equation (8.40), which represents the constraint residuals matrix H found by the direct method, is used to estimate the variance of the measurement errors using a simple optimisation procedure. This procedure is the minimisation of the squared differences between H and the estimated constraint residual variances AV^*A, with V^* in the unknown:

$$\min \ (H - AV^*A^T)^T (H - AV^*A^T) \tag{8.41}$$

Equation (8.41) is solved to determine V^* which is the variance-covariance matrix of the measurement errors.

8.8 OPTIMISATION

Once the data and models are reliable, i.e.: after the data measurements have been tested for steady-state detection, data reconciliation to remove random errors, gross error detection to remove gross errors and parameter estimation to update the model parameters, optimisation, which is the determination of those optimal set-points for which the system operates most efficiently, is required to be carried out.

Generally speaking, optimisation is concerned with the mathematical problem defined by minimising (or maximising) an objective function of n variables say f, subject to some m equality and p inequality constraints:

$$\underset{x_{i,(i=1,\ldots n)}}{Minimize}\ f(x_1, x_2, \ldots\ldots, x_n)$$

$$\text{Subject to: } h_j(x_1, x_2, \ldots\ldots, x_n) = 0, \text{ for } j = 1, 2, \ldots\ldots m \qquad (8.42)$$

$$g_k(x_1, x_2, \ldots\ldots, x_n) \leq 0, \text{for } k = 1, 2, \ldots\ldots p$$

Many methods and techniques have been developed and used to solve the mathematical problem (8.42). Depending on the nature of the objective function, equality and inequality constraints, it is possible to divide them into three categories (Ellis, 1994):

- *Category 1:* Linear programming problems, where the objective function $f(x)$, the equality and inequality constraints ($h(x)$ and $g(x)$) are all linear functions of the independent variable x. It is without doubt the most natural mechanism for formulating a vast array of problems with modest effort. Linear programming formulations are popular because it lends itself more readily to a mathematical formulation, the theory is richer, and the computation simpler for linear problems than for nonlinear ones. The Solutions of this problem always lay on constraint boundaries and algorithms exploiting this fact are well established.

- *Category 2:* Quadratic programming problems, where the objective $f(x)$ is a quadratic function of x, with perhaps, linear or quadratic constraints. Quadratic programming arises in many applications and it forms a basis of some specific algorithms and techniques. As it is usually solved using calculus, many problems (which are highly non-linear) are converted into quadratic formulations.

- *Category 3:* The general non-linear programming problem. This problem is usually too complex to solve using calculus because of the nonlinearity of the objective function and constraints. Usually numerical techniques are used to solve this kind of problems.

Optimisation techniques can be divided into two general categories: direct and indirect (model-based) optimisation methods (Garcia and Morari, 1981).

In the direct approach, measurements are taken directly from the real process as it is moving from one operating point to another, and a suitable optimisation technique is applied to find the optimum operating point. The way to proceed in choosing a suitable optimisation method for the direct approach depends upon several criterions, such as stability and convergence (Mansour and Ellis, 2003). Although these classical techniques have proved to deliver good performance in optimising some processes 'off-line', they still have two major problems when applied on-line (Ellis et al, 1988): the first one related to the process dynamics, while the second to noise presence.

As all real processes are dynamic in their nature, enough time (waiting period) must be given to the system to settle down, as the optimising control problem is essentially a steady-state problem, before taking any measurements or applying any inputs. This procedure can be time consuming especially in the case of slow processes where the waiting period may become prohibitive (Ellis et al, 1988). Also in practice, measurements might be affected by some noise, thus giving wrong values of the measured variables yielding to sub-optimality.

In the indirect or model-based approach, optimisation is performed on a model of the system instead of the physical system itself. When found the results are then applied to the real process. In this case, measurement noises are highly unlikely to occur, and the problem of the system dynamic is overcome via using a steady-state model.

In practice, such problems require a realistic representation of the physical system by means of a suitable mathematical model and the explicit or implicit formulation of an appropriate performance criterion. The mathematical model must describe correctly at least the qualitative features of the practical system in

the complete range of the probable operating conditions. and the optimality criterion must be a valid representation of the practical meaning of optimality. Although proved effective in some cases, this approach presents some major difficulties, as we could never provide the right model of the system, as there always exist differences or mismatches between the model and reality. This has been acknowledged for some time now, and it has been welcomed that the model must be adaptive so that some model parameters can be periodically updated (Lowe and Hidden, 1971).

8.8.1 The ISOPE algorithm

The key feature of the ISOPE algorithm is to replace the model-based optimisation problem, after an analysis of first-order optimality conditions (Appendix *A*), by an equivalent problem which is ultimately decomposed into a parameter estimation problem and a modified model-based optimisation problem (Roberts et al, 1988). As an on-line optimisation procedure, the ISOPE algorithm (developed by Roberts in 1979) has some features which can either be taken as direct or indirect. It is based on derivatives calculation provided by real process measurements (can be thought of as using elements of the direct technique) to update an unfaithful model used in the model-based optimisation, thus reaching the real optimum of the process in spite of model-reality differences (refer to chapter 2 for a detailed description of the ISOPE algorithm).

However, this method suffers from a major problem, which is that the derivatives have to be estimated by means of measurements which increases geometrically with problem dimensionality.

Methods and techniques have since been developed for the purpose of estimating these process derivatives, such as: Finite Difference Approximation Method (FDAM), Dual Control Optimisation, Broydon's method and Dynamic Model Identification method (DMI), with a linear model. These methods have been described and applied under simulation, to a cascade Continuous Stirred Tank Reactor (CSTR) system in chapter 4 of this thesis, together with a new version of

the DMI which uses a nonlinear model instead of a linear one. Results of the comparison simulations showed the superiority of the dynamic model identification method. In chapter 5, an Artificial Neural Network method was also introduced for the same purpose of estimating real process derivatives. Simulation results showed the method produces extremely accurate estimates of the derivatives in absence of noise, and results in a faster convergence of the ISOPE algorithm.

8.9 SIMULATION CASE STUDIES

In this final section of this chapter, two simulation case studies are conducted. The two studies are carried out using the two CSTR system of chapter 3 (Figure 3-2). In the first simulation case study, the system with its measurements contaminated by noise and bias is optimised without recourse to steady-state detection, gross error detection or data reconciliation. While in the second case study, on-line optimisation is applied as a package on the system with its measurements contaminated by noise and bias. The package includes: steady-state detection, static data reconciliation, gross error detection, and the actual optimisation given by the ISOPE algorithm.

The whole package was implemented under a MATLAB®/SIMULINK platform in groups of interconnected modules. The first module obtains the measurements from the system. It is directly connected to the steady-state detection module where the data is treated for automatic identification of steady-state using the Brown and Rhinehart, (2000) method presented earlier in this chapter. This module is connected to the gross error detection module. If steady-state is reached, the data measurements are passed to the gross error detection module to be cleared of any gross errors. After that, the data reconciliation module uses mass and energy balances equations to reconcile the data measurements resulting from the gross error detection module. After being cleared from gross and random errors, the data is passed to the optimisation module, where the ISOPE algorithm performs the integrated system optimisation and parameter estimation. The whole concept is illustrated in figure (8-14).

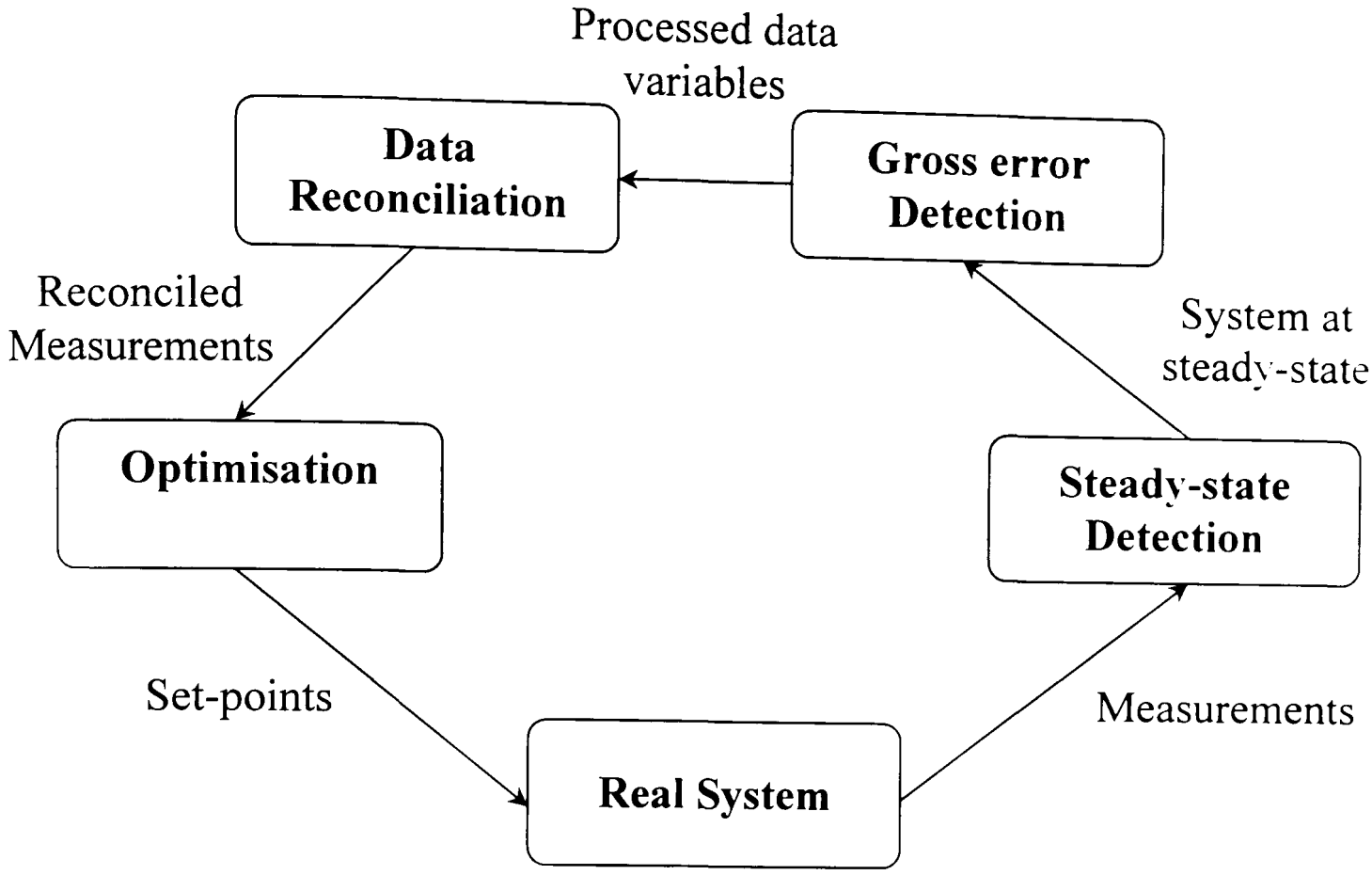

Figure (8-14): On-line Optimisation Procedure

In our case, the concentrations of species B in both tanks are considered as the measured variables, $y_m = [C_{b1}\ C_{b2}]^T$.

Both simulations were carried out using MATLAB® and were started from the same initial operating point given by $T_1 = 307K$ and $T_2 = 302K$, yielding the following steady-state output values of the concentration of products A and B in the two tanks 1 and 2, $C_{b1}(0) = 0.05165$ [kmol/m^3], and $C_{b2}(0)$=0.058638 [$kmol/m^3$].

For the steady-state detection module, a global level of significance was chosen to be $\alpha_{process} = 0.05$. Which means that the individual level of significance for each variable is given by: $\alpha_i = 1 - \sqrt[N]{(1-\alpha_{process})}$, where $i = 1,\ldots,N$. and $N = 2$. This results in $\alpha_i = 0.025$ for each variable. The three filter parameters L_1, L_2 and L_3 were chosen by trial error to be: $L_1 = 0.06$, $L_2 = 0.01$ and $L_3 = 0.01$. These values proved to be the best values when applied amongst the various tested. The value

of R_{crit} was chosen to be: R_{crit} = [1.35 ; 1.35]. R_{crit} is a vector of two elements: each element corresponds to one measured variable of the process.

For the gross error detection and static data reconciliation, the added noise was simulated as a normally distributed noise with zero mean, where the variance-covariance matrix V was obtained as shown in section 8.7. After a short training period of time, which included applying some given set-points, and measuring the corresponding outputs, the matrix V was built upon these values obtained by trial and error. The matrix V can be updated regularly using on-line measurements.

A bias was added (in simulation) to one of the two variables C_{b2}, and was of a value of 0.01172 [kmol/m^3] which corresponds to 20% of the initial value of C_{b2}. Finally, the optimisation was performed on a linear objective function of the measured variable C_{b2}. This choice of the objective function manifests a desire to maximise the amount of component B in tank 2. Therefore, the mathematical form of this function is given as:

$$L(y,v) = -C_{b2} \tag{8.43}$$

8.9.1. Results

Results of simulations are shown on figures (8-15) to (8-22).

For the first case study, figures (8-15) and (8-17) show the trajectories of the true and measured values of the output variables C_{b1} and C_{b2} when no steady-state detection, data reconciliation or gross error detection were applied, but only the ISOPE algorithm to optimise the objective function given by equation (8.43). The first figure when only noise, and no bias was present on the measurement, C_{b1}, while the second figure has both noise and bias. Figures (8-16) and (8-18) are the corresponding set-points trajectories for each case.

In the second case study, the full methodology of on-line optimisation was applied as a package on the two CSTR system. Gross error detection and elimination, data reconciliation and also the ISOPE algorithm were not applied until we were sure that the system was at steady-state with a certain confidence level. Figures (8-19) to (8-22) show the results obtained with this methodology.

8.9.2. Discussion of the Results

In the first case study, it is clear that the system does not converge to what we exactly want it to. Especially in the case when one of the measurements was biased. However, when no bias was present on either measurements the outputs tend to follow a certain pattern leading to a near-optimum point. But, at that stage, the output doesn't settle for a given value, but keeps fluctuating up and down with a certain error. In the second study however, we can see from the results obtained that the algorithm converges even in presence of a bias on one of the measurements. The slowness of the procedure shown in figures (8-19) and (8-20) is due to the fact we had to leave enough time to the steady-state detection algorithm to confidently detect steady-state before applying data reconciliation or the ISOPE algorithm. However, this huge waiting time is reduced in figures (8-21) and (8-22). Reducing the wasted time was achieved by activating the data reconciliation, and thus stop the steady-state detection procedure as soon as we are certain (of course with the given confidence level) that steady-state is reached. In fact, and as seen from both sets of figures this waiting time is reduced by half.

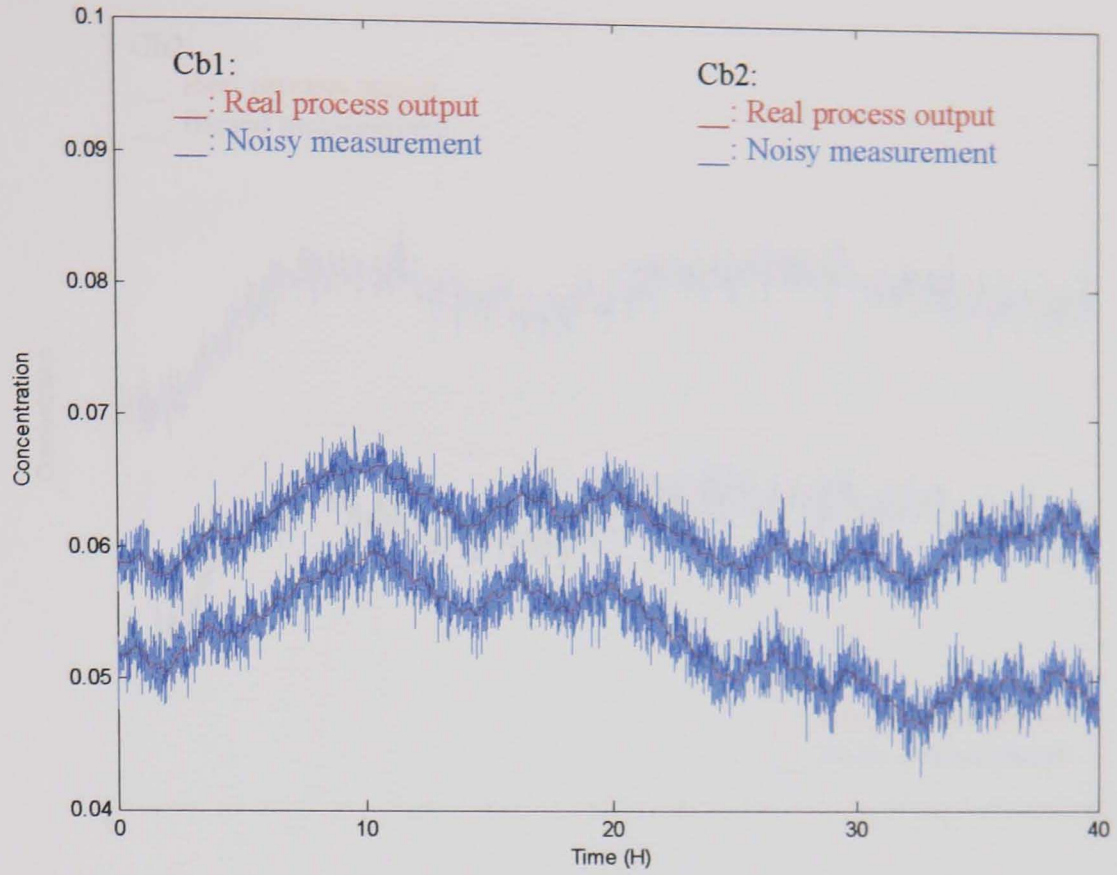

Figure (8-15): Real process output and noisy measurement trajectories, case when the methodology was not applied with both measurements affected by noise only.

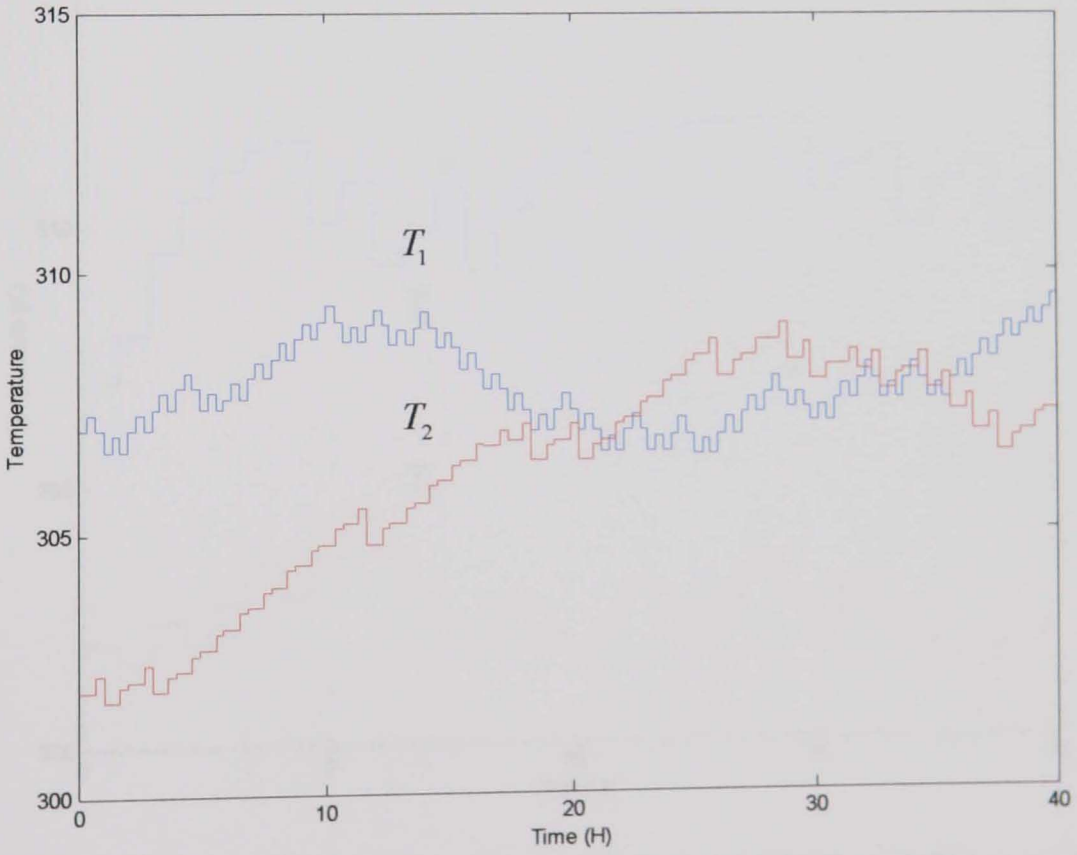

Figure (8-16): Set-point trajectories, case when the methodology was not applied with both measurements affected by noise only.

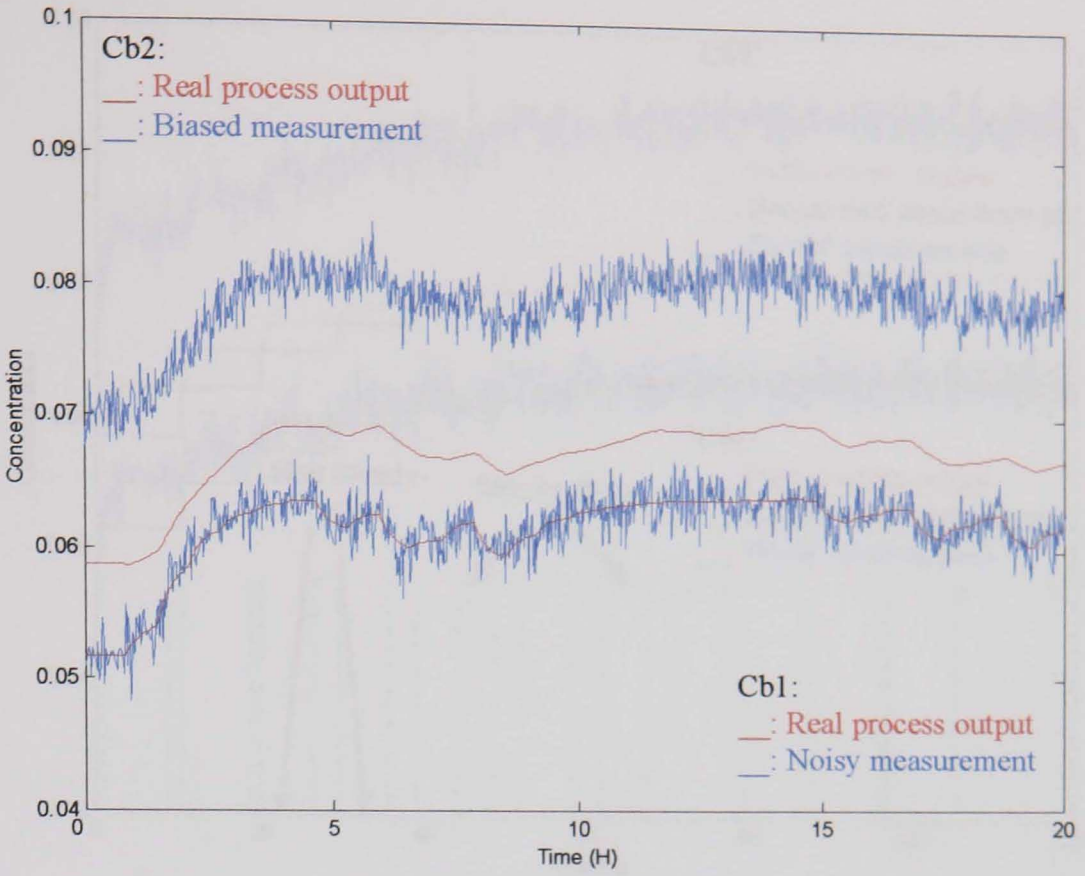

Figure (8-17): Real process output and noisy measurement trajectories, case when the methodology was not applied with both measurements affected by noise and C_{b2} is biased.

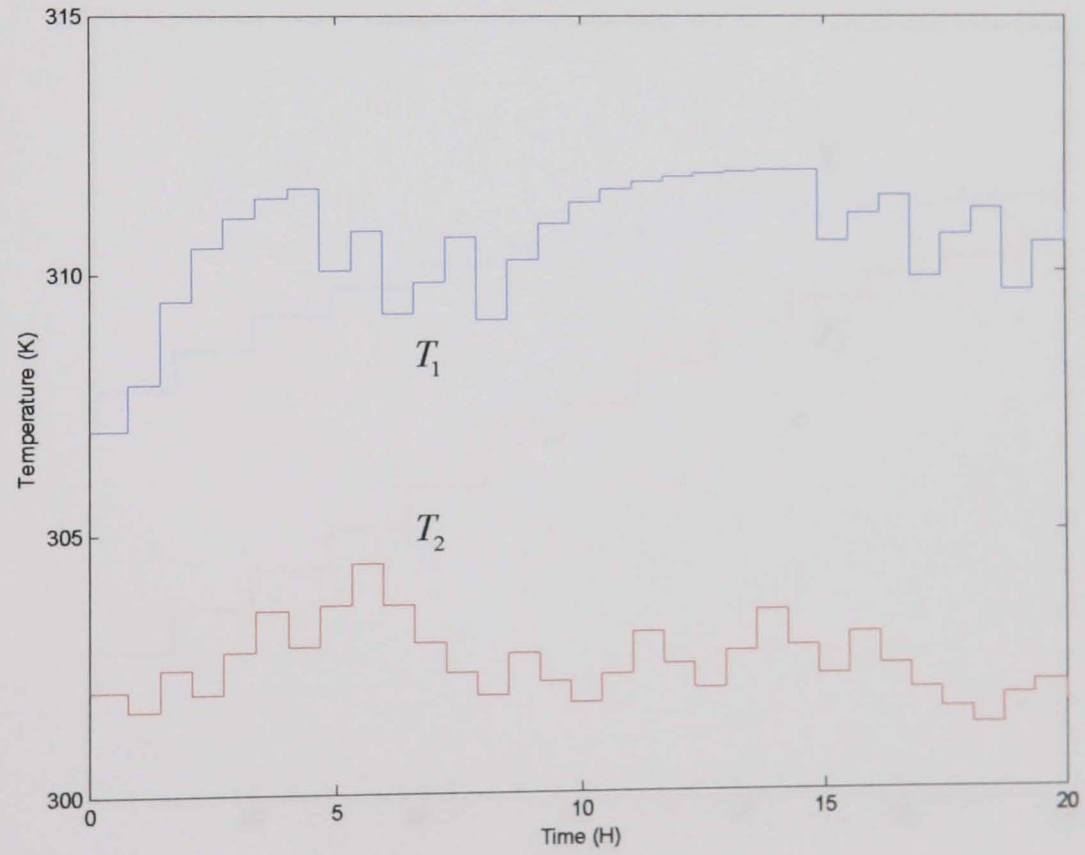

Figure (8-18): Set-point trajectories, case when the methodology was not applied with both measurements affected by noise and C_{b2} is biased.

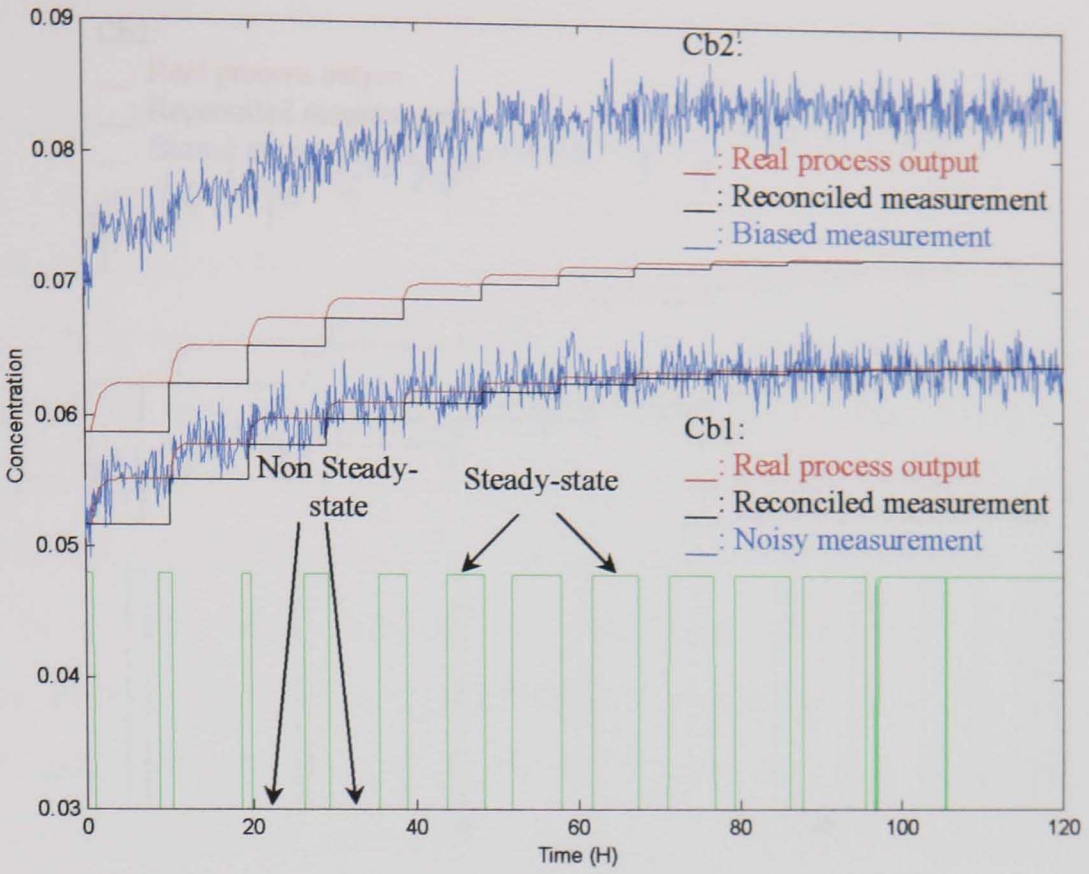

Figure (8-19): Real process output, noisy and reconciled measurement trajectories, case when the methodology was applied with C_{b2} biased.

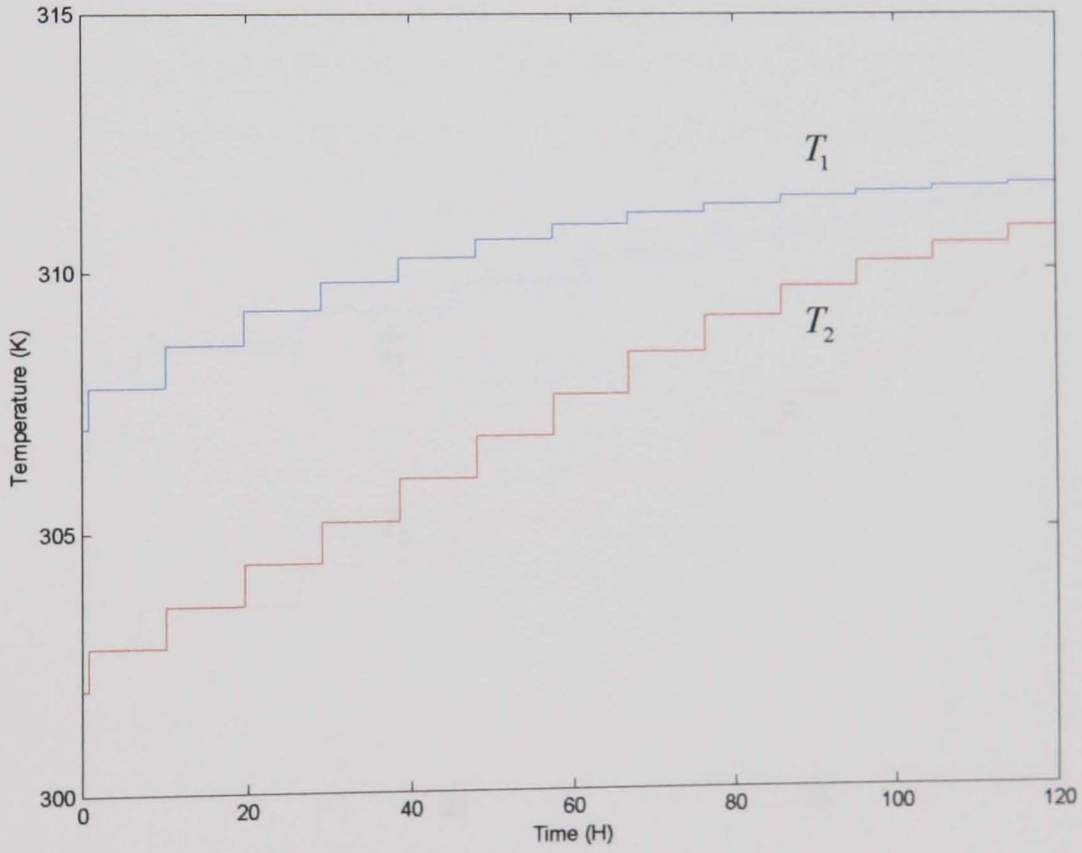

Figure (8-20): Set-point trajectories, case when the methodology was applied with C_{b2} biased.

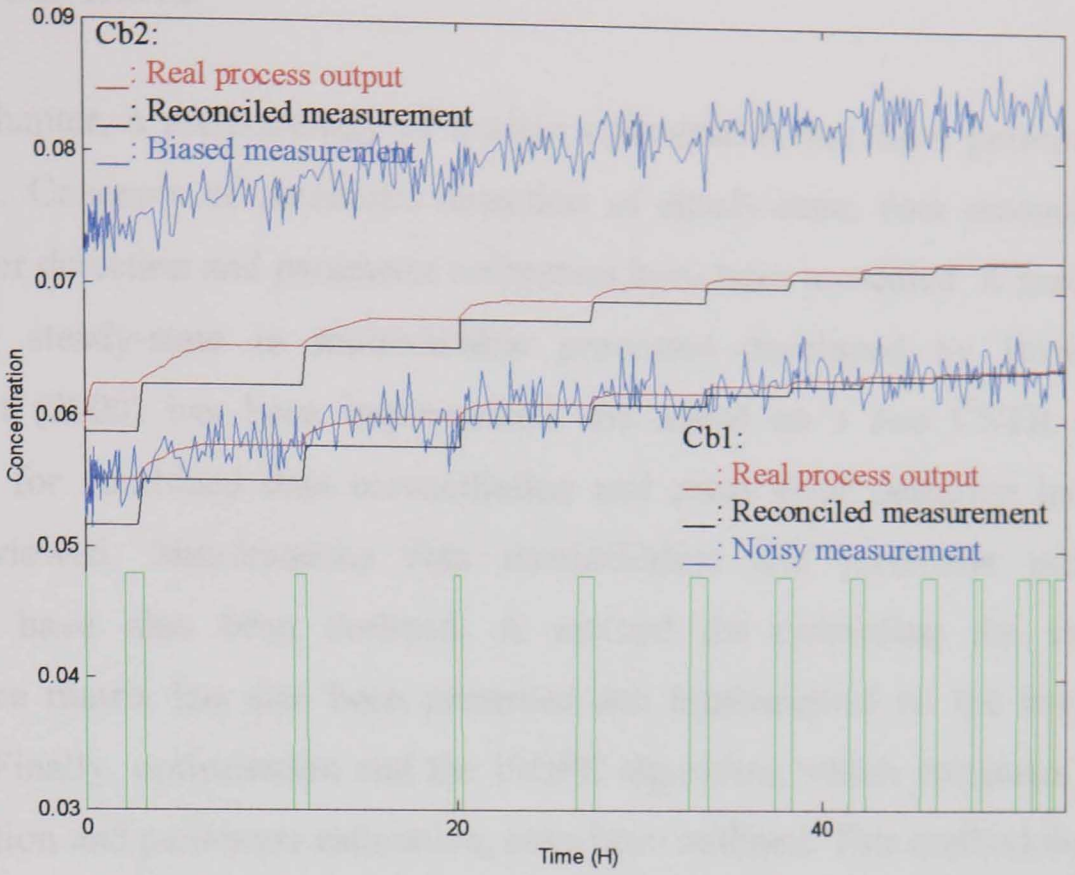

Figure (8-21): Real process output, noisy and reconciled measurement trajectories, case when the methodology was applied with C_{b2} biased after eliminating wasted time.

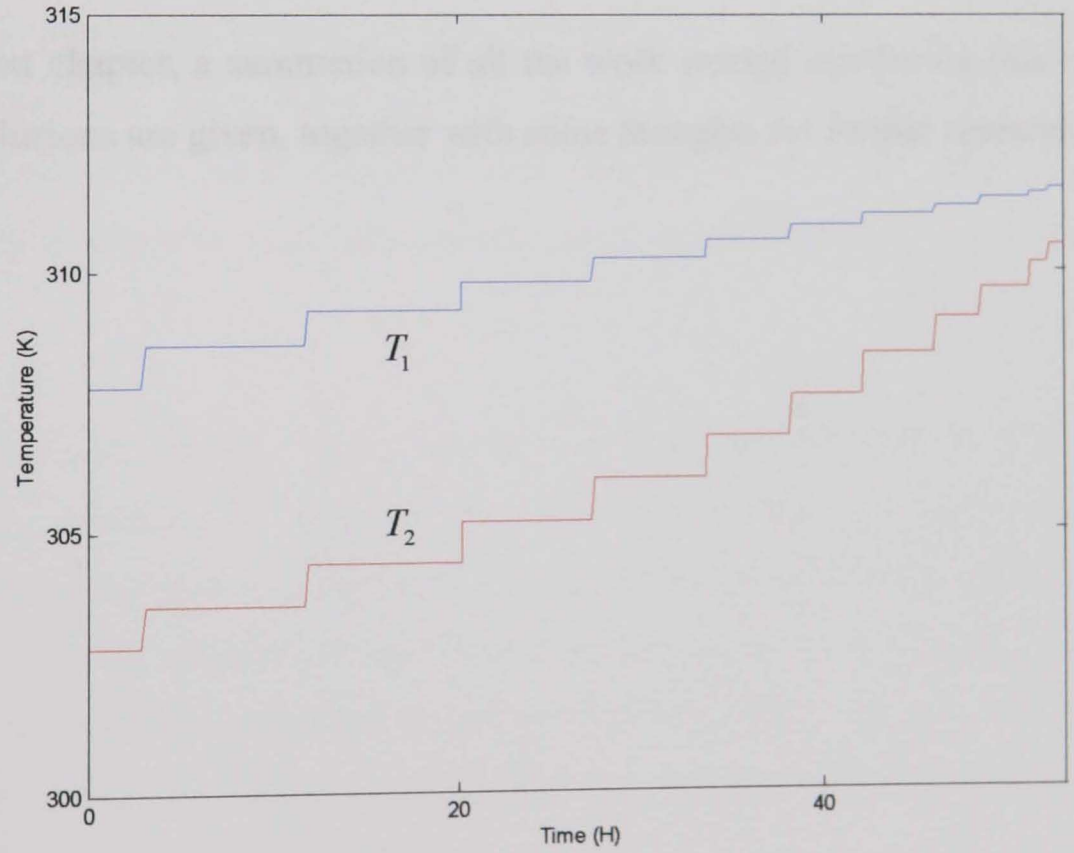

Figure (8-22): Set-point trajectories, case when the methodology was applied with C_{b2} biased after eliminating wasted time.

8.10 SUMMARY

In this chapter, a methodology of on-line optimisation has been presented and reviewed. Concepts of automatic detection of steady-state, data reconciliation, gross error detection and parameter estimation have been presented. A method for detecting steady-state in multivariable processes developed by Brown and Rhinehart (2000) has been implemented and tested on a two CSTR system. Methods for combined data reconciliation and gross error detection have also been reviewed. Simultaneous data reconciliation and parameter estimation methods have also been outlined. A method for estimating the variance-covariance matrix has also been presented and implemented on the two CSTR system. Finally, optimisation and the ISOPE algorithm, which integrates system optimisation and parameter estimation, have been outlined. This methodology was implemented successfully on a two Continuous Stirred Tank Reactors (CSTR) system.

In the next chapter, a summation of all the work carried out during this research and conclusions are given, together with some thoughts for further research.

CHAPTER 9

CONCLUSIONS AND RECOMMENDATIONS

9.1 CONCLUSIONS

This thesis is concerned with certain on-line optimisation structures and the general ISOPE algorithm which integrates an optimisation scheme together with parameter estimation. Algorithms to improve the performance of the algorithm were presented. These algorithms were for derivative estimation, steady-state detection, data reconciliation, gross error detection and optimisation. These topics were also reviewed and tested to asses their performance and effectiveness on a two CSTR system.

The concern here has been issues related to the development and use of algorithms for on-line optimisation and control. The methodology to apply an on-line optimisation procedure is complex, and may involve several steps and stages from different areas. A number of topics have been covered in this thesis. These are: Derivatives estimation, automatic detection of steady-state, static data reconciliation, gross error detection, parameter estimation and process optimisation. These topics have been extensively considered.

In order to assess and compare the performance and effectiveness of the techniques presented in this thesis, two examples of systems have been used. The first system is

a simple SISO nonlinear discrete time system, while the second one is a more complex system which consists of a two Continuous Stirred Tank Reactors (CSTR) connected in series. These are presented in Chapter 3. A practical version of the ISOPE algorithm developed by Becerra and Roberts (2000) has been implemented on these systems and used throughout each time the ISOPE algorithm was called. Several methods and techniques presented in this thesis have also been implemented and tested under simulation on models of these systems. The implementations of the models of these systems together with all the different techniques have been performed using a MATLAB®/SIMULINK software platform. During simulations, the SIMULINK model calls the subroutines containing the appropriate algorithms stored in M-files. These M-files acquire the information data under the form of measurements from the SIMULINK model, and process it as appropriate.

The solution of an on-line optimisation problem can be achieved in two ways:
The direct approach and the indirect or model-based approach. Both approaches possess advantages as well as disadvantages. For instance, the disadvantages associated with the direct approach are mainly due to the presence of noise and disturbances in the measurements on one hand, and the internal nature of the process in terms of its time response on the other hand. As in the case of a very slow process, the algorithm used to optimise the system might take a long time to converge. The disadvantages associated with the indirect approach however, are mainly caused by the mismatch that exist between the system and the model representing it, as it is very difficult, even unlikely to obtain a correct model of the system and its environment. One way to overcome the problems of measurements and noise in the direct approach, and model-reality differences in the indirect one, is to use model-adaptive technique in which some model parameters are regularly updated using real process measurements. The ISOPE algorithm is one of these techniques. It has got features from both the direct and indirect approaches, and do achieve the real optimum in spite of model-reality differences.

However, it has been acknowledged for some time now that probably the major drawback of the ISOPE algorithm is the requirement of real process derivatives to be computed at each iteration of the algorithm. These process derivatives are needed by the ISOPE algorithm in order to satisfy necessary optimality conditions. Attempts have been made to overcome this problem by either developing alternative techniques to successfully estimate these derivatives or totally eliminate this necessity. In this thesis, several algorithms and techniques for estimating process derivatives have been presented, implemented and tested under simulation on a two CSTR system. These are: Finite difference approximation, dual control optimisation, Broydon's method and dynamic model identification with linear model method. A comparison study has also been conducted using these techniques, and results have been presented. Also, a novel technique developed during the course of this research based on a nonlinear dynamic model identification has been presented and tested under simulation. The aim was to provide accurate estimates of the process derivatives, while avoiding the difficulties encountered in the previous techniques. These difficulties are mainly caused by excessive excitation of the set-points for some methods, and slowness and sensitivity to noise for others. The technique when implemented has shown to be successful and gave leading performance when compared to the other techniques. The only exception was the dynamic model identification based on a linear model. The results of the simulations show that this method is the most suitable method to be used for process derivative estimation amongst those tested in our example, because it makes the ISOPE algorithm converge faster, and its least square estimator plays a filter role against noise. Also, it was shown that all the techniques do achieve convergence to the optimum point, but with a small difference in the time taken to converge, with the exception of FDAM, which proved to converge slowly due to the need for extra set-point changes, which is problematic in the case of large-scale and slow processes.

In the same context but with a different approach, an Artificial Neural network (ANN) method has been implemented and applied on two different systems. The

method exploits the ability of ANN's to learn from the environment and produce accurate approximations of functions, and train an ANN model in order to imitate the behaviour of the real system. At the beginning of the procedure, and before the optimisation is applied, input/output candidates are collected from the real system by applying a set of inputs to the system. A waiting period is allowed to transpire so that a steady-state is reached and then the corresponding outputs are gathered. This data is then used to train the neural network in order to obtain the model that would represent the real system. Although the method makes the ISOPE algorithm converge faster compared to that using FDAM to estimate the derivatives, as it is based on a steady-state model, changes in process parameters can result in the whole scheme producing erroneous estimates of the process derivatives and hence lead to sub-optimality. In such cases, the algorithm needs some considerable time to retrain the neural network model to adapt to the new changes, which can be prohibitive in the case of slow processes. In practice, to cover against system parameter changes, retraining may be carried out at periodic intervals.

The reliability and accuracy of measured data is of a great importance in monitoring, evaluating process performance, and for process models that are used in optimisation and control. The objective of data reconciliation is to correct the measured variables by removing random errors from the data set, and to estimate the values of the unmeasured variables, so that we can obtain an estimate of the true state of the plant. The procedure is to reconcile process data by requiring it to be consistent with natural laws such as energy and mass balances. The data reconciliation problem can be solved using a number of efficient approaches. Nonlinear programming, quadratic programming and successive linearisation methods are the common methods used. However, in presence of gross errors, the least-squares objective function used in data reconciliation can be severely biased leading to incorrect reconciliation and estimation. Gross errors as opposed to random errors are considered to be caused by non-random events such as process leaks, biases in instruments, and so on.

Therefore, a gross error detection phase is usually needed before data reconciliation is applied. Ideally, the aim of a gross error detection technique is to:

1 Detect the existence of the gross error
2 Identify its location
3 Identify its type
4 Determine its size

After the gross errors are identified, two responses are possible and/or desired :

1 Eliminate the measurement with the bias, or
2 Correct the model such as the case of a leak and run the reconciliation again.

In this thesis, static data reconciliation and gross error detection have been implemented in a module, and tested under simulation on the two CSTR system. Errors and biases of different values have been added to the outputs of the real system to simulate erroneous measurements. The results of the simulations have shown that the scheme has been successfully implemented to detect and eliminate errors from flawed measurements. Given values of biases added to the measurements to simulate errors ranging from -40% to +40% of the nominal values, the whole data reconciliation and gross error detection procedure produce good results. However, this was only observed in this example using the CSTR system, in other situations, other factors may need to be considered.

The application of the above data reconciliation and gross error detection within the on-line optimisation procedure, the ISOPE algorithm has been implemented in software. The resulting scheme (data reconciliation and gross error detection + ISOPE) collects data measurements from the system, and applies the data reconciliation and gross error detection to remove both random and gross errors.

After that, the reconciled measurements (free of errors) are used in the optimisation. The performance of the scheme has been demonstrated under simulation on the two CSTR system, where the measurements were contaminated by random errors and biases. Optimisation was performed using the ISOPE algorithm on a linear objective function which reflects the desire of maximising the concentration of one component in the second tank of the CSTR system. The implementation of the whole procedure was achieved by allocating a separate module for each task. These tasks interact where and when necessary. The simulation results have shown that the application of data reconciliation and gross error detection on corrupted data measurements within the ISOPE algorithm proved to improve optimisation. This is mainly due to the improved parameter estimation, which improves the derivative estimation as well, resulting in a more efficient operation of the system.

In order for data reconciliation, gross error detection and steady-state optimisation to be applied, data measurements have to be collected from the system when at the steady-state. For this purpose, a method for automatic detection of steady-state that can be used in multivariable analysis has been presented, implemented and tested under simulation on the two CSTR system. This method is capable of detecting steady-state with a certain confidence level. The testing of the algorithm has proved to be successful given the right choice of values of the parameters used in the algorithm. These parameters are highly important, and a wrong tuning of these parameters could lead to an early, late, or a non detection of steady-state at all.

As most methods for data reconciliation and gross error detection are based on the knowledge of the Variance-covariance matrix V of measurements. The choice of a suitable V matrix is crucial and sometimes proves to be very difficult. Several methods to estimate this matrix have been reported in the literature. We mentioned the direct method based on the mean of the measurement samples, and the indirect method which uses the estimates of the constraint residuals calculated using the direct

method. The direct method has been presented and implemented under simulation on the two CSTR system, and the results have been presented.

In the final section of this thesis, the separate modules implemented for steady-state detection, data reconciliation and gross error detection, Variance-covariance matrix estimation, process derivative estimation, and the actual optimisation (ISOPE algorithm) have been assembled and implemented sequentially to form a methodology of on-line optimisation. This methodology has been implemented in software and tested on the two CSTR system. Each task in the methodology was carried out sequentially in a modular way starting from detecting steady-state and finishing by obtaining the optimum set-points that achieve the most efficient operation of the system. The modules are interconnected together where and when necessary in order to enable an easy and reliable interaction and transfer of the information. Simulation results have shown that this methodology can successfully be used to achieve the optimum operating point of the system if all parameters of each task are tuned appropriately. It also reduces the time wasted waiting for the system to settle down. This time can be either too short or wasted by a wrong choice of the waiting time. This can be avoided by activating the steady-state detection algorithm, which can tell us when the system settles down, within a certain confidence level.

9.2 RECOMMENDATIONS FOR FUTURE RESEARCH

As an extension to the work carried out in this thesis, the following items are most recommended for further research:

For the ISOPE algorithm, instead of estimating the process derivatives which some times proves to be difficult, it might be more advantageous to develop techniques which are indirect, but do not require derivative information to be obtained. Initial thoughts are to use some of the features of the well established Powell's conjugate direction method in conjunction with the ISOPE. This could be an area of possible further research.

In the area of data reconciliation and gross error detection, one of the challenges that should attract more attention and seek more consideration, is the elimination of the uncertainties on the location of the gross errors and uncertainties that are independent of the method of detection and compensation.

As for the simulations carried out in this work, only biases have been simulated as a type of gross error. Further testing with different types of gross errors might be beneficial to assess the data reconciliation and gross error detection methods implemented in this work.

All the techniques presented in this thesis have been implemented and tested on the two CSTR system. A similar task would be to test these techniques on different type of systems.

The development of an on-line optimisation package based on the methodology presented in this thesis might also be very useful. With the new capabilities that MATLAB®/ SIMULINK offers, it is possible to develop this sort of packages which can implement, apply and simulate the methodology on either linear or nonlinear systems.

REFERENCES

Abdullah N. (1988): "Augmented Integrated System Optimisation and Parameter Estimation Techniques for On-line hierarchical Control of Large Scale Industrial Processes", PhD. Thesis, City University, London, U.K.

Abu-el-zeet Z.H. (1998): "Developments in Predictive Optimisation using DISOPE", Technical Report no. 166, CERC, City University, London, UK, March 1998.

Abu-el-zeet Z.H., Becerra V.M. and Roberts P.D. (1999): "Steady State Data Reconciliation and Estimation of Systematic Biases", Technical Report no. 171, CERC, City University, London, UK, October 1999.

Abu-el-zeet Z.H. (2000): "Optimisation Techniques for Advanced Process Supervision and Control ", Ph.D. Thesis, City University, London, U.K.

Abu-el-zeet Z.H., Becerra V.M. and Roberts P.D. (2000): "Data reconciliation and steady-state detection applied to a chemical process", UKACC International Conference (Control 2000), University of Cambridge, Cambridge, U.K., 4-7th September 2000.

Albuquerque J.S. and Biegler L.T. (1996): "Data Reconciliation and Gross-Error Detection for Dynamic Systems", AIChE Journal, Vol. 42, No. 10, October 1996.

Almasy G.A. and Sztano T. (1975): "Checking and Correction of Measurements on the Basis of Linear System Model", Problems of Control and Information Theory, Vol. 4(1), pp. 57-69.

Alekman S.L. (1994): "How Can Steady-state Conditions Be Identified Automatically?", Control For The Process Industries, Vol. 8, pp. 62.

Amand T., Heyen G. and Kalitventzeff (2000): "Plant Monitoring and Fault Detection: Synergy between Data Reconciliation and |Principle Component Analysis", Computers and Chemical Engineering, vol. 24S, pp 819-822.

Arora N., Biegler L.T., and Heyen G. (2002): "Optimization Formulations for Data Reconciliation", Computer Aided Process Engineering.

Ayala, J. S., (1997): "State of the Art in CLRTO", AspenWorld 97 Conf. Proc., Aspen Technology, Inc. Boston, MA, Oct. 15, 1997.

Bagajewicz M. (1997): "Optimal Sensor Location in Process Plants", AIChE Journal, Vol. 43, No. 9, pp. 2300.

Bagajewicz M. and Sánchez M. (1999a): "Sensor Network Design and Upgrade for Plant Parameter Estimation", Computers and Chemical Engineering, Vol. 23, Supp., pp. S593-S596.

Bagajewicz M. and Sánchez M. (1999b): "Duality of Sensor Network Design Models for Parameter Estimation", AIChE Journal, Vol. 45, No. 3, pp. 661-664.

Bagajewicz M. and Sánchez M. (1999c): "Design and Upgrade of Nonredundant and Redundant Linear Sensor Networks" AIChE Journal, Vol. 45, No. 9, pp. 1927-1939.

Bagajewicz M. and Sánchez M. (2000a): "On the Impact of Corrective Maintenance in the Design of Sensor Networks", Industrial and Engineering Chemistry Research, Vol. 39, No. 4, pp. 977-981.

Bagajewicz M. and Sánchez M. (2000b): "Reallocation and Upgrade of Instrumentation in Process Plants", Computers and Chemical Engineering, Vol. 24, No. 8, pp. 1961-1980.

Bagajewicz M. and Sánchez M. (2000c): "Cost-Optimal Design of Reliable Sensor Networks" Computers and Chemical Engineering, Vol. 23, No. 11/12, pp. 1757-1762.

Bagajewicz M.J. (2003): "Data Reconciliation and Instrumentation Upgrade. Overview and Challenges", FOCAPO (Foundations of Computer Aided Process Operations), Coral Springs, FL, USA, January 2003.

Bagajewicz M.J. and Rollins D. (2003): "On the consistency of the measurement and generalized likelihood ratio tests for gross error detection", Computer & Chemical Engineering.

Bamberger W. and Isermann R. (1978): "Adaptive On-Line steady-State optimization of slow dynamic Processes", Automatica, Vol. 14, pp. 223-230.

Becerra V.M. and Roberts P.D. (2000): "Implementation of Nonlinear Optimal Process Control with Model Reality Differences", Technical Report no. 142, CERC, City University, London, UK, July 1995.

Becerra V.M. and Roberts P.D. (2000): "Formulation, implementation and application of a practical version of the Modified Two-Step method for optimising control", Technical Report no. 173, CERC, City University, London, UK, April 2000.

Becerra V.M., Roberts P.D. and Griffiths, V.W. (1998): "Novel developments in process optimisation using predictive control", Journal of Process Control, Vol. 8, No. 2, pp. 117-138.

Betha R.M. and Rhinehart R.R. (1991): "Applied Engineering Statistics", Marcel Dekker, New York, NY.

Binder T., Blank L., Dahmen W. and Marquardt W. (1998): "Towards Multiscale Dynamic Data Reconciliation", In: Nonlinear Model Based Process Control (R. Berber and C. Kravaris, Eds.). NATO--ASI Series.

Brdy's M., Roberts P.D., Badi M.M. and Kokkinos I.C. (1986): "Model based double iterative strategy for integrated system optimisation and parameter estimation of large scale industrial processes" Proc. 4th IFAC/IFORS Symp. On Large Scale Systems: Theory and Applications, Zurich, Switzerland, pp. 634-639.

Brdy's M. and Roberts P.D. (1987): "Convergence and optimality of modified two-step algorithm for integrated system optimization and parameter estimation", International Journal of Systems Science, Vol. 18, pp. 1305-1322.

Brdy's M. and Tatjewski P. (1994): "An algorithm for steady-state optimising dual control of uncertain plants", IFAC workshop on New trends in Design of Control Systems, Smolenice, Slovak Republic.

Brian D.B. (1984): "Basic optimisation Methods", London: Edward Arnold.

Brown P.R and Rhinehart R.R. (2000): "Demonstration of a method for Automated Steady-State Identification in Multivariable systems", Hydrocarbon processing, Vol. 79, No. 9, pp. 79-83.

Box M.J., Davies D. and Swann W.H. (1969): "non-linear Optimization techniques", Oliver and Boyd.

Britt H.I. and Luecke R.H. (1973): "The Estimation of Parameters in Nonlinear, Implicit Models", Technometrics, Vol. 15, No. 2, pp. 233-247.

Cao S. and Rhinehart R.R. (1995): "An Efficient Method for on-line identification of steady state", Journal of Process Control, Vol. 5, No. 6, pp. 363-374.

Cao S. and Rhinehart R.R. (1997): "Critical Values for a steady-state identifier", Journal of Process Control, Vol. 7, No.2, short note, pp. 149-152.

Chen X. (1998): "The optimal implementation of On-line Optimization for chemical and refinery processes", Ph.D. Thesis, Louisiana State University, Louisiana, U.S.A.

Chen X., Pike R.W., Hertwig T.A. and Hopper J.R. (1998): "Optimal implementation of On-Line Optimization", Computers and Chemical Engineering, Vol. 22, pp. 435-442.

Chen X., Derya B.O. and Pike R. W. (2003):"Gross Error Detection in Chemical Plants and Refineries for On-Line Optimization", Workshop on Systems Safety, LAWSS 2003.

Cheng J. and Zafiriou E. (2000): "Robust model-based iterative feedback optimization of steady-state plant operations", Ind. Eng. Chem. Res., Vol. 39, pp. 4215.

Crow E.L., Davis, F.A. and Maxfield, M.W. (1955): "Statistics Manual", Dover Publications, New York, N.Y, 63.

Crowe C.M. (1986): "Reconciliation of Process Flow Rates by Matrix Projection – Part II: The Nonlinear Case", AIChE Journal, Vol. 32, No. 4, pp. 616-623.

Crowe C.M. (1996): "Data reconciliation-progress and challenges", Journal of Process Control, Vol. 6, No. 2/3, pp. 89-98.

Crowe C.M, Garcia C.Y.A. and Hrymak A. (1983): "Reconciliation of Process Flow Rates by Matrix Projection – Part I: The Linear Case", AIChE Journal, Vol. 29, No.6, pp. 881-888.

Deming W.E. (1943): "The Statistical Adjustment of Data", New York: Wiley, 1964 New York: Dover.

Dennis J.E, Roberts B.S. (1983): "Numerical methods for unconstrained optimization and nonlinear equations", Prentice-Hall.

Duane H. and Bruce L. (1996): "Mastering MATLAB, A Comprehensive Tutorial and Reference", The MATLAB Curriculum Series, Prentice-Hall.

Edgar T.F. and Himmelblau D.M. (1988): "Optimization Of Chemical Processes", Mc Graw-hill.

Ellis J.E. and Roberts P.D. (1981): "Simple models for integrated optimization and parameter estimation", International Journal of Systems Science, Vol. 12, No. 4, pp. 455-472.

Ellis J.E., Kambhampati C., Sheng G. and Roberts, P.D. (1988): "Approaches to the Optimising Control Problem", International Journal of Systems Science, Vol. 19, No. 10, pp. 1969-1985.

Ellis J.E., Kandamby N.H. and Roberts P.D. (1993): "Rationalization and Application of Algorithms for Integrated System optimization and Parameter Estimation", International Journal of Systems Science, Vol. 24, No. 2, pp. 219-229.

Ellis J.E., Lopez S. and Roberts P.D. (1994): "Model Certainty in Optimising Control", IEEE Conference, Glasgow, Scotland.

Ellis J. E. (1994): "Principles of Optimisation", MSc Lecture Notes, City University, London, UK.

Farid M. (1999): "Optimisation and Optimal Control", MSc lecture notes, City University, London, UK

Fletcher R. (1980): "Practical Methods of Optimization", A Wiley-Interscience Publication, Vol. 1.

Fletcher R. (1980): "Practical Methods of Optimization", A Wiley-Interscience Publication, Vol. 2.

Forbes J.F. (1994): "Model structure and adjustable parameter selection for operations optimization", PhD. Thesis, McMaster University, Ontario, Canada.

Forbes J.F. and Marlin T.E. (1996): "Design Cost: A systematic approach to technology selection for model-based real-time optimization systems", Computer & Chemical Engineering, Vol. 20, No6/7, pp. 717-734.

Garcia C.E. and Morari M. (1981): "Optimal Operation of Integrated processing Systems", AICHE Journal, Vol. 27, No. 6, pp. 960-968.

Gelb A. (1974): "Applied Optimal Estimation", MIT Press, Cambridge, MA (1974).

Gene F.F. and David J.P. (1980): "Digital control of dynamic systems", Addison-Wesley Pub. Co.

Gill P.E., Murray W. and Wright M.H. (1981): "Practical Optimization", Academic Press.

Gumowski I., and Mira C. (1968): "Optimization In Control Theory And Practice", Cambridge University Press.

Haimes Y. and Wismer D. (1972): "A computational approach to the combined problem of optimization and parameter estimation", Automatica, Vol. 8, pp. 337.

Hardin A.J., Joshi A. and Jones J. (1995): "Rigorous crude unit optimization", In **NPRA** computer conference, 1995.

Haykin, S. (1994): "Neural networks: a comprehensive foundation", Macmillan London, 1995.

Hebb D.O. (1949): "The Organization of behavior: A Neuropsychological Theory", Wiley, New York.

Hornik, K., Stinchcombe, M., and White, H. (1990): "Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks", Neural Networks, Vol. 3, pp. 551-560.

Holly W., Cook R. and Crowe C.M. (1989): "Reconciliation of Mass Flow Rate Measurements in a Chemical Extraction Plant", The Canadian Journal of Chemical Engineering, Vol. 67, pp. 595-601.

Hopfield J.J. (1982): "Neural Networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences of the U.S.A, Vol. 79, pp. 2554-2558.

Huber, P.J. (1972): "Robust statistics: A review", Ann. Math. Statist., Vol. 43, pp.1041-1067.

Jang S.S., Joseph B. and Mukai H. (1986): "Comparison of two approaches to on-line parameter and state estimation of nonlinear systems", Ind. Engng Chem. Process. Des. Dev., Vol. 25, pp. 809-814.

Johnson L.P.M. and Kramer M. A. (1995): "Maximum Likelihood Data Rectification: Steady-State Systems", *AIChE* Journal, Vol. 41, No. 11, pp. 2415-2426.

Kim I.W., Kang M.S., Park S. and Edgar T.F. (1997): "Robust Data Reconciliation and Gross Error Detection: The Modified MIMT Using NLP", Computers and Chemical Engineering, Vol. 21, No.7, pp. 775-782.

Kambhampati C. (1988): "Algorithms for Optimising Control", PhD. Thesis, City University, London, U.K.

Kambhampati C., Ellis J.E. and Roberts, P.D. (1989): "generalization of Integrated System Optimization and Parameter Estimation Techniques", Automatica, Vol. 25, No. 2, pp. 307-310.

Kambhampati C., Mason J. D. and Warwick K. (2000): "A stable one-step-ahead predictive control of non-linear systems", Automatica, Vol. 36, pp. 485-495.

Kao C.S., Tamahane A.C. and Mah R.S.H. (1992): "Gross Error Detection in Serially Correlated Process Data", Ind. Eng. Chem. Res., Vol. 31, pp.254-262.

Kim, I-W., Liebman M. J. and Edgar T. F., (1990): "Robust Error-in-Variable Estimation Using Nonlinear Programming Techniques," AIChE Journal., Vol. 36, No. 7, pp. 985-93.

Kim I., Kang M.S., Park S. and Edgar T.F. (1997): "Robust data reconciliation and gross error detection: The modified MIMT using NLP", Comp. chem. Eng., Vol. 21, pp. 775-782.

Kuehn D.R. and Davidson H. (1961): "Computer Control, II Mathematics of Control", Chemical Engineering Progress, Vol. 57, No. 6, pp.44-47.

Lauks V., Vasbinder R., Vallenberg P. and Van Leuwen C. (1992): "On-line optimization of an ethylene plant", Comp. Chem. Eng., Vol. 16, pp. 213-219.

Lewis F.L. (1986): "Optimal Control", New York: Wiley.

Liebman M.J. and Edgar T.F. (1988): "Data reconciliation for nonlinear processes", Paper Presented at the AIChE Annual Meeting, Washington, DC.

Liebman M.J., Edgar T.F. and Lasdon L.S. (1992): "Efficient data reconciliation and estimation for dynamic process using nonlinear programming techniques", Computer & Chemical Engineering, Vol. 16, No. 10/11, pp. 963-986.

Lin J., Hendawy Z.M. and Roberts P.D. (1988): "A modified integrated system optimisation and parameter estimation technique for hierarchical control of steady-state systems", Adv. Model. Simul., Vol. 12, pp. 43-52.

Loar J. (1994): "How Can Steady-state Conditions Be Identified Automatically?", Control For The Process Industries, Vol. 8, pp. 62.

Loeblein C. and Perkins J.D. (1996): "Economic Analysis of Different Structures of On-Line Process Optimization Systems", Computer & Chemical Engineering, Vol. 20, Suppl., pp. S551-S556.

Lowe E.I and Hidden A., (1971): "Computer Control in Process Industries", Peter Peregrinus, London.

Luenberger D. G. (1984): "Linear and Nonlinear Programming", Addison-Wesley.

Madron F. (1985): "A New Approach to the Identification of Gross Errors in Chemical Engineering Measurements", Chemical Engineering Science , Vol. 40, No. 10, 1985.

Madron, F. (1992): "Process plant performance : Measurement and data processing for optimization and retrofits", Ellis Horwood, Chichester, London (1992).

Mah R.S.H. (1982): "Design and Analysis of Process Performance Monitoring Systems", In Chemical Process Control II, Engineering Foundation, New York, 1982, pp.525, Proceedings of the Engineering Foundation Conference, January 18-23, 1981, Sea Island, Georgia, Editors: T.F. Edgar and D.E. Seborg.

Mah R.S.H. (1990): "Chemical Process Structures and Information Flows", Butterworths, Stoneham, MA, Chapters 8 and 9.

Mah R.S.H., Stanley G.M. and Downing D.M. (1976): "Reconciliation and rectification of process flow and inventory data", Ind. Engng Chem. Process. Des. Dev., Vol. 15, pp. 175-183.

Mah R.S.H. and Tamhane A.C. (1982): "Detection of Gross Errors in Process Data", AIChE Journal, Vol. 28, No.5, pp. 828-830.

Mansour M. and Ellis J.E. (2000): "Issues in On-line Optimisation", Technical Report no. 174, CERC, City University, London, UK, May 2000.

Mansour M. and Ellis J.E. (2003): "Comparison of methods for estimating real process derivatives in on-line optimisation", Applied Mathematical Modelling, Vol. 27, pp. 275-291.

McBrayer K.F. and Edgar T.F. (1995): "Bias detection and estimation in dynamic data reconciliation", Journal of Process Control, Vol. 5, No. 4, pp. 285-289.

McBrayer K.F., Soderstrom T.A., Edgar T.F. and Young R., E. (1998): "The application of nonlinear dynamic data reconciliation to plant data", Computers & Chemical Engineering, Vol. 22, No. 12, pp. 1907-1911.

McCulloch W.S and Pitts W. (1943): "A logical calculus of the ideas immanent in nervous activity", Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-133.

Minsky M.L. (1961): "Steps Towards Artificial Intelligence", Proceedings of the Institute of Radio Engineers, Vol. 49, pp. 8-30.

Narasimhan S. (1984): "Detection of Steady States and its Application to Process Data Reconciliation", M.S. Thesis, Northwestern University.

Narasimhan S. and Mah R.S.H. (1987): "Generalized Likelihood Ratio Method for Gross Error Identification", AIChE Journal, Vol. 33, No. 9, pp. 1514-1521.

Narasimhan S. and Jordache C. (2000): "Data Reconciliation and Gross error detection: An Intelligent Use of Process Data", Houston: Gulf Publishing Company.

Narasimhan S., Mah R.S.H., Tamhane A.C., Woodward J.W. and Hale J.C. (1986): "A Composite Statistical Test for Detecting Changes of Steady States", AIChE Journal, September 1986, Vol. 32, No. 9, pp.1409 – 1418.

Narasimhan S., Chen Shan Kao and Mah R.S.H. (1987): "Detecting Changes of Steady States Using the Mathematical Theory of Evidence", AIChE Journal, November 1987, Vol. 33, No. 11, pp. 1930 – 1932.

Narendra K.S. and Parthasarathy K. (1990): "Identification and control of dynamical systems using neural networks", IEEE Trans. on Neural Networks, Vol. 1, pp. 4-27.

Polak E. (1971): "Computational Methods in Optimization A Unified Approach", Academic Press.

Reilly P. and Carpani R. (1963): "Application of statistical theory of adjustment to material balances", In 13th Can. Chem. Eng. Conf., Montreal, 1963.

Richard L.F. (1971): "Optimization Methods For Engineering Design", Addison-Wesley Pub. Co.

Ripps D.L. (1965): "Adjustment of experimental data", Chemical Engineering Prog. Symp. Ser., Vol. 61, pp. 8-13.

Robert H.B. (1993): "Modern Control Systems Analysis and Design Using MATLAB", Addison-Wesley Pub. Co.

Roberts P.D (1979): "An algorithm for steady-state system optimization and parameter estimation", International Journal of Systems Science, Vol. 10, pp. 719-734.

Roberts P.D. and Williams T.W.C. (1981): "On an Algorithm for Combined System Optimisation and Parameter Estimation", Automatica, Vol. 17, pp. 199-209.

Roberts P.D. (1992): "Optimal control of nonlinear systems with model-reality differences", Proceedings of the 31st Conference on Decision and Control, Tucson, Arizona, December 1992.

Roberts P.D. (1995): "Coping with model-reality differences in industrial process optimisation – A review of integrated system optimisation and parameter estimation (ISOPE) ", Computers in Industry, pp. 281-290.

Rollins D.K. and Roelfs S.D. (1992): "Gross Error Detection When Constraints are Bilinear", AIChE J., Vol. 38, 1295-1299.

Rollings D. and Davis J. (1992): "Unbiased estimation of gross errors in process measurements", AICHE Journal, Vol. 38, No. 4, pp. 563.

Rollings D. and Davis J. (1993): "Gross error detection when variance-covariance matrices are unknown", AICHE Journal, Vol. 39, No. 8, pp. 1335-1341.

Rollings D.K., Cheng Y. and Devanathan S. (1996): "Intelligent Selection Of Hypothesis tests to Enhance Gross Error Identification", Comp. And Chem. Eng., Vol. 20, No. 5, pp. 517-530.

Rumelhart D.E., Hinton G.E. and Williams R.J. (1986): "Learning representations by back-propagating errors", Nature (London), pp. 533-536.

Rumelhart D.E. and McClelland J.L. (1986): "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", Vol. 1, MA: MIT Press.

Seber G.A.F. (1984): "Multivariate Observations", John Wiley & Sons, New York.

Sequeira, S.E. (2003): "Real Time Evolution (RTE) for on-line optimisation of continuous and semi-continuous chemical processes", Ph.D. Thesis, Universitat Politècnica de Catalunya, Spain.

Serth R.W. and Heenan W.A. (1986): "Gross Error Detection and Data Reconciliation in Steam-Metering Systems", AIChE Journal, Vol. 32, No. 5, pp. 733-742.

Shearer J.L., Kulakowski B.T. and Gardner J.F. (1997): "Dynamic modelling and control of engineering systems", Prentice Hall, 2nd edition.

Singh and Titli, (1978): "Systems: Decomposition, Optimisation and Control", Oxford Pergamon Press.

Stanley G.M. and Mah R.S.H (1981): "Observability and redundancy in Process Data", Chemical Engineering Science, Vol. 36, pp. 259-272.

Tamhane A.C. and Mah R.S.H. (1985): "Data Reconciliation and Gross Error Detection in Chemical Process Networks", Technometrics, November 1985, Vol. 27, No. 4, pp. 409-422.

The MathWorks, (1996): "Using MATLAB", The MathWorks.

Tjoa I.B. and Biegler L.T. (1991): "Simultaneous Strategies for Data Reconciliation and Gross Error Detection of Nonlinear Systems". Computers and Chemical Engineering, Vol. 15, No. 10, pp. 679-690.

Tong H. and Crowe C.M. (1996): "Detecting Persistent Gross Errors by Sequential Analysis of Principal Components", Computers and Chemical Engineering, Vol. 20, Suppl., pp. S733-S738.

Uttley A.M. (1956): "A theory of the mechanism of learning based on the computation of conditional probabilities", Proceedings of the 1st International Conference on Cybernitics, Namur, Gauthier-Villars, Paris.

Van Wijk R. and Pope M. (1992): "advanced process control and on-line optimization in Shell refineries", Comp. Chem. Eng., Vol. 16, pp. 69-77.

Widrow B. and Hoff M.E. (1960): "Adaptive Switching Circuits", IRE WESCON Convention Record, pp. 96-104.

Widrow B. (1962): "Generalization and Information storage in networks of adaline 'neurons'", In Self-Organizing Systems (Yovitz M.C., Jacobi G.T. and Goldstein G.D. eds.), Washington, D.C.: Sparta.

Willsky A.S. and Jones H.L. (1974): "A Generalized Likelihood Ratio Approach to State Estimation in Linear Systems Subject to Abrupt Changes", Proc. IEEE Conf. Decision and Control.

Zhang H. and Roberts P.D. (1990): "On-line steady-state optimisation of nonlinear constrained processes with slow dynamics", Trans. Ins. MC, Vol. 12, No. 5, pp. 251-261.

Zhang J. and Morris A.J. (1999): "Recurrent Neuro-Fuzzy Networks for Nonlinear Process Modeling", IEEE transactions on Neural Networks, Vol. 10, No. 2, pp. 313-326.

Appendix A

Necessary Optimality Conditions

Given the following optimisation problem:

$$\begin{aligned} &\min f(x) \\ \textbf{subject to :}\ &h(x) = 0 \\ &g(x) \le 0 \end{aligned} \tag{A1}$$

where $f(x)$ is the Objective function to be minimised, and $h(x)$ and $g(x)$ are the equality and inequality constraint functions respectively.

First-Order Necessary Conditions

The First-order necessary conditions for optimality, also known as: The Kuhn-Tucker Conditions (Luenberger, 1983) are:

$$\begin{aligned} &\mu \ge 0, \\ &\nabla f(x^*) + \lambda^t \nabla h(x^*) + \mu^t \nabla g(x^*) = 0 \\ &\mu^t g(x^*) = 0 \end{aligned} \tag{A2}$$

where x^* is a relative minimum point for the above optimisation problem, and λ and μ are the Kuhn-Tucker multipliers.

Second-Order Conditions

In addition to the First-order Necessary conditions given above, the Second-order Sufficiency conditions necessitates that the Hessian matrix

$$L(x^*) = F(x^*) + \lambda' H(x^*) + \mu' G(x^*), \tag{A3}$$

is positive definite on the subspace M given by:

$$\begin{aligned} M &= \{y : \nabla h(x^*) y = 0, \nabla g_j(x^*) y = 0 \text{ for all } j \in J\}, \\ J &= \{j : g_j(x^*) = 0, \mu_j > 0\}. \end{aligned} \tag{A4}$$

where F, H and G are the 2nd order derivatives of the objective function, the equality constraint and inequality constraint respectively.

Appendix B

The R-statistic

Given the filtered value of the measurement value X:

$$X_{f,i} = L_1 X_i + (1 - L_1) X_{f,i-1} \tag{B.1}$$

One can compute the filtered mean square deviation from the previous filtered values $v_{f,i}^2$ by:

$$v_{f,i}^2 = L_2 (X_i - X_{f,i-1})^2 + (1 - L_2) v_{f,i-1}^2 \tag{B.2}$$

If the process is stationary:

$$E(v_{f,i}{}^2) = E((X_i - X_{f_{i-1}})^2) = v^2 \tag{B.3}$$

Equation (B.2) is an unbiased estimate of v^2, and the variance of $v_{f,i}{}^2$ is given by:

$$Var(v_{f,i}{}^2) = \frac{L_2}{2 - L_2} Var((X_i - X_{f_{i-1}})^2) \tag{B.4}$$

This means that equation (B.2) provides a computationally efficient, unbiased estimate of $(X_i - X_{f_{i-1}})^2$.

Then the estimate of the noise variance with the first approach will be:

$$S_{1,i}{}^2 = \frac{2 - \lambda_1}{2} v_{f,i}{}^2 \tag{B.5}$$

Also, given the filtered mean square difference of successive data $d_{f,i}^2$:

$$d_{f,i}^2 = L_3(X_i - X_{i-1})^2 + (1 - L_3)d_{f,i-1}^2 \tag{B.6}$$

It is easily shown that the second estimate of the noise variance would be:

$$S_{2,i}{}^2 = \frac{d_{f,i}{}^2}{2} \tag{B.7}$$

The R-statistic is computed by taking the ratio of the two estimates of variance (measured by the two methods) as determined by equation (B.5) and equation (B.7):

$$R_i = \frac{S_{1,i}{}^2}{S_{2,i}{}^2} = \frac{(2 - L_1)v_{f,i}{}^2}{d_{f,i}{}^2} \tag{B.8}$$

Appendix C

The IMT and MIMT algorithms

The procedure of Iterative Measurement Test (IMT) is:

Step 1: Compute reconciled vector y_{true} and measurement adjustments vector a as in the Measurement test (MT).

Step 2: Calculate the standardised measurement adjustments $\varepsilon_i = a_i / \sigma_i$, as in MT.

Step 3: Compare each ε_i with the critical value C of test statistic as in MT. If $|\varepsilon_i| < C$ for all measurements, go to step 6. Otherwise, select the measurement corresponding to the largest value of $|\varepsilon_i|$ and add it to set S as suspected measurement that contains a gross error. If two or more measurements have the same maximum values of $|\varepsilon_i|$, select the one with lower index.

Step 4: If set S is empty, proceed to Step 6. Otherwise, remove the measurements contained in S from system by nodal aggregation to obtain a lower dimension of system with constraint coefficient matrix **B**, measurement vector **w**, and covariance matrix **P** as MT (**B**, **w**, and **P** have the same meaning as given in MT). Let T denote the measurements contained in **w**. Repeat Step 1 to compute y_{true} and a with **A**, y_m, and V replaced by **B**, **w**, and **P**, respectively.

Step 5: Compute corrected values for measurements in set S by solving equations $A\, y_{true} = 0$ with the variables in set T specified with the reconciled values from step 4 and the variables in set R specified with the original measured values. R is a set of variables that were eliminated during the nodal aggregation and whose measured data does not contain gross error, i.e., R = U - (S$\cup$T), where U is the set of all variables in the system. Then, go back to Step 2.

Step 6: If the set S is empty, then all measurements are free of gross errors, and the estimated values of process variables in step 1 are the reconciled values of all process

variables. Otherwise, the set of reconciled values is obtained from the computed values in step 5 for the variables affected by gross errors in set S, the reconciled values computed in step 4 for the variables in set T, and the original measured values for the variables in set R.

The procedure of modified iterative measurement test (MIMT) is:

Step 1: Compute reconciled vector y_{true} and measurement adjustments vector a as in the Measurement Test (MT).

Step 2: Calculate the standardised measurement errors $\varepsilon_i = a_i / \sigma_i$, as in the MT.

Step 3: Compare each ε_i with the critical value C of test statistic as in MT. If $|\varepsilon_i| < C$ for all measurements, go to step 7. Otherwise, select the measurement corresponding to the largest value of $|\varepsilon_i|$ and add it to set S as suspected measurement that contains a gross error. If two or more measurements have the same maximum values of $|\varepsilon_i|$, select the one with lower index.

Step 4: If set S is empty, proceed to Step 7. Otherwise, remove the measurements contained in S from system by nodal aggregation to obtain a lower dimension of system with constraint coefficient matrix **B**, measurement vector **w**, and covariance matrix **P** as MT (**B**, **w**, and **P** have the same meaning as given in MT). Let T denote the measurements contained in **w**. Repeat Step 1 to compute y_{true} and a with A, y, and V replaced by **B**, **w**, and **P**, respectively.

Step 5: Compute corrected values for measurements in set S by solving equations $A\, y_{true} = 0$ with the variables in set T specified with the reconciled values from step 4 and the variables in set R specified with the original measured values. R is a set of variables that were eliminated during the nodal aggregation and whose measured data does not contain gross error, i.e., R = U - (S$\cup$T), where U is the set of all variables in the system.

Step 6: Check the reconciled values of process variables with the pre-specified bounds. If one or more of reconciled data does not satisfy the bounds, then discard

the reconciled data and return to step 3, delete the last entry in set S, and replace it with the measurement corresponding to next largest value of $|\varepsilon_i|$. If no bound violation is found, go back to Step 2.

Step 7: If the set S is empty, then all measurements do not contain gross error, and the estimated values of process variables in step 1 are the reconciled values of all process variables. Otherwise, the set of reconciled values is obtained from the computed values in step 5 for the variables containing gross errors in set S, the reconciled values computed in step 4 for the variables in set T, and the original measured values for the variables in set R.