DATA-BASED PID CONTROLLER DESIGNS FOR NONLINEAR SYSTEMS

IMMA NUELLA

NATIONAL UNIVERSITY OF SINGAPORE

2008

DATA-BASED PID CONTROLLER DESIGNS FOR NONLINEAR SYSTEMS

IMMA NUELLA
(S. T., ITB, INDONESIA)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF CHEMICAL AND BIOMOLECULAR ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2008

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my research supervisor, Dr. Min-Sen, Chiu, for his constant support, invaluable guidance and suggestions throughout my research work at National University of Singapore. He showed me different ways to approach a research problem and the need to be persistent to accomplish any goal. My special thanks to Dr. Chiu for his invaluable time to read this manuscript.

I greatly appreciate the valuable advices and concerns I received from Dr. Li Jia, Dr. Cheng Cheng, and Ankush Ganeshreddy Kalmukale to my research work. Special thanks and appreciation to my lab mates, Yasuki Kansha, Martin Wijaya Hermanto, and Xin Yang for actively participating discussion related to my research work and the help that they have rendered to me. I would also wish to thank technical and administrative staffs in the Chemical and Biomolecular Engineering Department for the efficient and prompt help. I am indebted to the National University of Singapore for providing me the excellent research facilities. I am also greatly indebted to the AUN-SEED-Net JICA for providing me research scholarship in National University of Singapore.

I cannot find any words to thank my parents, sisters, and all of my friends for their unconditional support, affection and encouragement, without which this research work would not have been possible. I also wish to thank my best partner, Husein, for his understanding, continuous support and encouragement during my research work.

TABLE OF CONTENTS

SUMMARY

In process industries, large numbers of process variables are regularly measured and automatically recorded in historical database. Therefore, how to extract useful information from data for controller design is one of the challenges in chemical industries. In this regard, data-based methods arise as an attractive alternative for nonlinear system modeling. In this thesis, the data-based controller designs for nonlinear process are developed. The main contributions of this thesis are as follows.

In the fuzzy neural network modeling framework, an adaptive PID control scheme is proposed. A fuzzy neural network model is employed to approximate the controlled nonlinear process. By utilizing Lyapunov method, an updating algorithm is derived to adjust the PID parameters to guarantee the convergence of the predicted tracking error. Next, a self-tuning PID controller design is designed based on the JITL modeling technique. This proposed design method exploit the current process information from controller database and modeling database to realize on-line tuning of PID parameters. The controller database is constructed to store the PID parameters together with their corresponding information vector, and the modeling database is employed for the standard use by JITL for the modeling purpose. The PID parameters are obtained from controller database according to the current process dynamics characterized by the information vector at every sampling instant. Furthermore, the PID parameters can be updated during on-line implementation and the resulting updated PID parameters together with their corresponding information vector are then stored into the controller database.

Simulation results are presented to demonstrate that the proposed control strategies give better performances than their conventional counterpart.

NOMENCLATURE

C_I , $C_{I_{in}}$	Initiator concentration
C_m , $C_{m_{in}}$	Monomer concentration
c_L	Corresponding cluster in FNN modeling when S_L is obtained
c_{li}	Center of cluster
D	Number of process database for JITL modeling
d_i	Similarity measure of controller database
e	Error between process output and set-point
e_r	Error between set-point and predictive output
e_x	Vector of errors
F , F_i	Flow rate
F_i^l	Fuzzy sets
$f*$	Parameter of polymerization reaction
f_i	Crisp function of fuzzy input vector
h , h_c	Number of nearest neighbors
I	Input
O	Output
J	Objective function
k_{I} , k_{f_m} , k_p , k_{T_c} , k_{T_d}	Kinetic parameters of polymerization reaction
$k_{\min}$, $k_{\max}$	Number of minimum and maximum relevant data set
M	Number of rule antecedent
M_m	Molecular weight of monomer
N	Number of fuzzy rules in FNN
N_0	Number of initial controller database
N_k	Number of information vectors stored in the current controller database
N_t	Number of database in FNN modeling

n_y , n_u , n_d	Integers related to the system's order and time delay
R^l	Fuzzy rules
r	Set-point
S_L , s_k	Similarity measure
u , $\hat{u}$	Process input
$u_{\min}$, $u_{\max}$	Minimum and maximum values of process input in the database
V	Reactor volume
v	Lyapunov function
$\mathbf{W}_{h^*}$	Weight matrix of JITL
w^0 , w_j , w_1 , w_2 , w_3	Parameters of PID controller
$\mathbf{x}(k)$	Regression vector in FNN and JITL modeling
$\mathbf{x}_q$	Query data
$\mathbf{x}_{\text{cl}}$	Information and query vector of controller database
x_i, x_M	Inputs of fuzzy system
y	Process output
$\hat{y}$	Model prediction
$\hat{y}_l$	Model prediction of the l-th fuzzy rule
y_h, y_{h^*}	Relevant process output of JITL
$y_{\min}$, $y_{\max}$	Minimum and maximum values of process output in the database
$\tilde{y}$, $\tilde{u}$	Scaled process output and input

Greek Symbols

α_l , β_l, α_1^k , α_2^k , β_1^k	First-order model parameters
ψ	Pre-specified threshold in FNN modeling
ε	Pre-specified threshold
ϕ	JITL algorithm parameter
λ_0	Parameter for updating center of cluster

μ_l	Membership of the *i*-th rule antecedent
η_j , η_1 , η_2 , η_3	Learning rates
ς_j	Mapping variable of PID parameters
θ_k	Angle between $\Delta\mathbf{x}(k)$ and $\Delta\mathbf{x}_q$
γ_i	Weighting factor of PID parameters
χ_l	Fuzzy width
ξ	Positive constant of Lyapunov function
$\sigma(x)$	Function of neural network
ρ	Overlap parameter
Ω	Relevant data set with largest similarity number in JITL
κ	Process input weighting factor
$\Phi(i)$	Controller database

Abbreviations

AIBN	Azo-bis-isobutyronitrile
ARX	Autoregressive exogenous
CSTR	Continuous stirred tank reactor
FNN	Fuzzy neural network
FNNM	Fuzzy neural network model
IMC	Internal model control
JITL	Just-in-time learning
MAE	Mean absolute error
MMA	Methyl methacrylate
NAMW	Number average molecular weight
NN	Neural network
PI	Proportional-integral
PID	Proportional-integral-derivative
RBF	Radial basis function
RLS	Recursive least square
STPI	Self-tuning proportional-integral

STPID	Self-tuning proportional-integral-derivative
T-S	Takagi-Sugeno
TSK	Takagi-Sugeno-Kang

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Motivation

Process control research has been an area of growing importance over the past several decades. The performance requirement in tightening product quality specifications have become increasingly difficult to satisfy due to stronger global competition, promptly changing economic conditions, tougher environmental and safety regulations, higher energy and material costs, and higher demand for robust and fault-tolerant systems. Furthermore, the rapid advances in computer technologies have enabled high-performance measurement and control systems to become an essential part of industrial plants. Hence, engineers and researchers are still motivated to develop more efficient and reliable techniques for process modeling, control, and monitoring for more flexible and complex processes.

In process industries, large numbers of process variables are regularly measured and automatically recorded in historical database. However, how to extract

valuable information and knowledge from database for process control, optimization and monitoring is still one of the challenges in the process industries. Although an accurate process model is required for many advanced control design method, the construction of first-principle models is usually time-consuming and costly. Furthermore, model-based controller design by incorporating these models would lead to complex controller structure, not to mention that many chemical processes are not amenable to this modeling approach due to the lack of precise knowledge about the processes (Babuška and Verbruggen, 2003). To this end, data-based methods arise as an attractive alternative for nonlinear system modeling in the last two decades (Pearson, 1999; Nelles, 2001).

In the literature, many data-based modeling methods have been proposed. They can be roughly classified into two modeling approaches: global modeling and local or memory-based modeling approach (Bontempi et al., 2001). The most well-known example for global modeling approach is neuro-fuzzy or fuzzy neural-network (FNN) which can facilitate the effective development of models by combining information from different sources, such as empirical models, heuristics, and data. Moreover, FNN has been proven to have ability to approximate any continuous function to a desired degree of accuracy through learning (Horikawa et al., 1992; Chen and Teng, 1995; Zhang and Morris, 1995, 1999; Cao et al., 1997; Wai and Lin, 1998; Gao et al., 2000; Zhang, 2001; Babuška and Verbruggen, 2003; Andrášik et al., 2004; Hsu et al., 2007). FNN describes systems by means of fuzzy *if – then* rules represented in a network structure, to which learning algorithms known from the area of artificial neural networks can be applied.

In comparison, the local modeling approach can be represented by instance-based learning algorithm which has attracted much research attention under various

notions, for example locally weighted learning (Atkeson et al., 1997a, 1997b), lazy learning and just-in-time learning (JITL, Cybenko, 1996; Stenman, 1996; Bontempi et al., 1999, 2001; Cheng and Chiu, 2004). The JITL technique uses the concept of memory-based modeling which focuses on approximating the function only in the neighborhood of the point to be predicted and select the best local model by assessing and comparing different alternatives in cross-validation. JITL has no standard learning phase because it merely stores the data in the database and the models are built dynamically upon query. Moreover, JITL has inherent adaptive nature which is achieved by storing the on-line measured data into the database.

PID controllers have been widely used in the process industries due to simple control structure, ease of implementation, and robustness in operation. However, the conventional PID controller is not adequate to deal with highly nonlinear and time varying chemical processes. To improve the control performance, various adaptive PID controller designs have been developed in the literature. In the context of neural network and FNN frameworks, Lu et al. (2001) constructed a predictive fuzzy PID controller by combining a fuzzy PID controller with model predictive controller. Chen and Huang (2004) designed adaptive PID controller based on the instantaneous linearization of a neural network model. Sun et al. (2006) developed a self-tuning PID controller based on adaptive genetic algorithm and neural networks. Most of the previous works update the parameters of the process model with respect to the current process condition and then PID parameters are computed by the corresponding adaptation algorithm and implemented. However, these adaptation algorithms are inadequate to address the convergence of the predicted tracking error. Recently, Chang et al. (2002) derived a stable adaptation mechanism in the continuous time domain by the Lyapunov approach such that the PID controller tracks a pre-specified

feedback linearization control asymptotically. Motivated by this work, a self-tuning algorithm derived from Lyapunov method in the discrete time for adaptive PID design based on FNN modeling technique will be developed in this thesis.

In the JITL modeling framework, an adaptive PID controller has been developed by Cheng (2006). In this work, the JITL technique served as the process model to provide information for controller design. However, the initialization of PID parameters required trial and error effort which made its application in control practice less attractive. To alleviate this shortcoming, Takao et al. (2006) proposed a memory-based IMC-PID controller design. However, the PID controller considered in Takao et al. (2006) was formulated by assuming a first-order plus time delay model, which is too restrictive to be applied in practical applications. Inspired by these previous results, a self-tuning PID controller based on the memory-based method and JITL modeling technique will be developed in this thesis as well.

1.2 Contribution

Motivated by the various modeling frameworks developed for nonlinear process modeling, two distinct modeling frameworks are explored and investigated in the proposed controller designs. One controller design uses FNN approach while another controller design is based on memory-based and JITL techniques. The main contributions of this thesis are as follows.

Firstly, an adaptive PID control scheme is developed. A fuzzy neural network-based model is employed to approximate the controlled nonlinear process. By utilizing Lyapunov method, an updating algorithm is derived to adjust the PID parameters to guarantee the convergence of the predicted tracking error.

Secondly, a self-tuning PID controller design is proposed by exploiting the current process information from controller database and modeling database to realize on-line tuning of PID parameters. The controller database contains the PID parameters and the corresponding information vectors, while the modeling database is employed by the JITL technique for modeling purpose. The PID parameters are obtained from controller database according to the current process dynamics characterized by the information vector at every sampling instant. Whenever these PID parameters need to be updated during on-line implementation, the resulting updated PID parameters together with their corresponding information vector are stored into the controller database to enhance the database for the operating conditions where the information is not available in the construction of the initial controller database.

1.3 Thesis Organization

The thesis is organized as follows. Chapter 2 comprises the literature review of nonlinear process modeling and control. By incorporating FNN technique into controller design, an adaptive PID controller design based on Lyapunov approach is proposed in Chapter 3. A self-tuning PID controller design using JITL modeling approach is developed in Chapter 4. Lastly, the general conclusions from the present work along with some suggestions for future work are given in Chapter 5.

Chapter 2

Literature Review

2.1 Nonlinear Process Modeling

To overcome the difficulty of obtaining accurate first-principle models due to the lack of complete physicochemical knowledge of chemical processes, empirical models are attractive alternatives. This modeling approach or so called data-based method extracts models from process data measured in industrial processes even when very little a priori knowledge is available. Recently, various data-based methods for nonlinear process modeling have been proposed (Pearson, 1999; Nelles, 2001). They can be broadly classified into two main opposing paradigms, the global versus the local models (Bontempi et al., 2001). Global models have two main properties. First, they cover the entire operating conditions of the system underlying the available data. Second, global models solve the problem of learning an input-output mapping as a problem of function estimation. Fuzzy neural network (FNN) is one of well-known examples of this modeling approach.

On the other hand, the local paradigm originates from the idea of relaxing one or both of the global modeling features. Given that the problem of function estimation is hard to solve in a general setting, this method focuses on approximating the function only in the neighborhood of the point to be predicted. Memory-based learning turns out to be a single-step approach where the learning problem is seen as value estimation rather than a function estimation problem. Furthermore, memory-based method requires the storage of database in opposition to functional methods which discard the data after training. One representative modeling technique of this class of method is just-in-time learning (JITL) technique.

FNN and JITL share the *divide-and-conquer* approach (Bontempi et al., 2001) to enhance the modeling accuracy by decomposing complex global problems into simpler local sub-problems. The main difference of these two modeling approaches lays in the model identification procedure. FNN aims at estimating a global description which covers the whole system operating domain, whereas JITL technique focuses simply on the current operating point. FNN is more time-consuming in the identification phase but it is faster in prediction. However, when a new piece of data is observed, it may need to update the model from scratch. On this matter, JITL is more advantageous because it is enough to update its database when a new input-output data is observed. Therefore, JITL is intrinsically adaptive while FNN requires proper on-line procedures to deal with the model updating. In the next section, these two different modeling approaches will be briefly reviewed.

2.1.1 Neural network modeling technique

Neural network (NN) that makes use of the organizational principles of human brains can provide an excellent framework for modeling the nonlinear systems

because of its capability of approximating any smooth function to an arbitrary degree of accuracy with a certain number of hidden layer neuron (Hornik et al., 1989). According to Hunt and Sbarbaro (1991), features of NN in the control context are:

(i) the ability to represent arbitrary non-linear relations

(ii) the adaptation and learning in uncertain systems, provided through both off-line and on-line weight adaptation

(iii) the information transformed to internal representation allowing data fusion, with both quantitative and qualitative signals

(iv) the parallel distributed processing architecture allowing fast processing for large-scale dynamical systems

(v) the architecture providing a degree of robustness through fault tolerance and graceful degradation.

Two classes of NN which have received considerable attention in the past two decades (Narendra and Parthasarathy, 1990) are: (1) multilayer feedforward neural network and (2) recurrent neural network. From systems theoretic point of view, multilayer neural network represents static nonlinear maps while recurrent neural network is represented by nonlinear dynamic feedback systems.

The NN as shown in Figure 2.1 is a feedforward neural network that consists of neurons arranged in layers, which are connected via weight parameters such that the signals at the input are propagated through the network to the output. Through the weight parameters, the input of each neuron is computed as the weighted sum of the outputs from the neurons in the preceding layer. The output of each neuron is computed by a transfer function to yield the nonlinear behavior of the network. The

most popular functions are the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and the radial basis function (RBF) $\sigma(x) = e^{-x^2}$, where x is the input of each neuron.

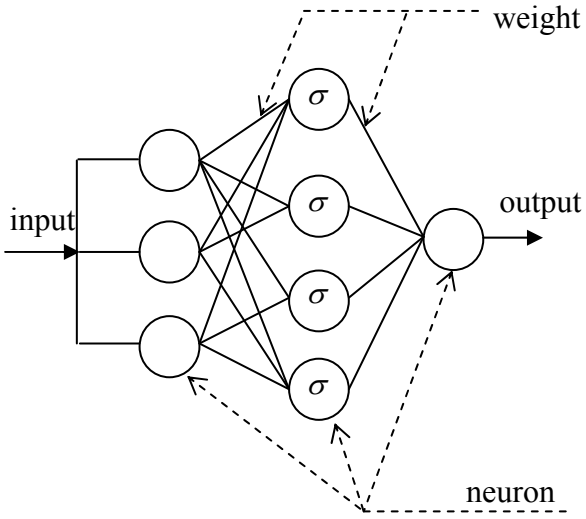

Figure 2.1 Structure of a multilayer feedforward neural network

During the training of NN, the weights are adjusted and learned from a given set of data aiming to achieve the 'best' approximation of the dynamics of the system. For modeling the dynamic systems, the output of the NN can be represented by:

$$\hat{y}(k) = f(y(k-1), \cdots, y(k-n_y), u(k-1-n_d), \cdots, u(k-n_u-n_d)) \qquad (2.1)$$

where $\hat{y}(k)$ is the predicted output of NN at the k-th sampling instant, y is the system output, u is the system input, n_y, n_u, and n_d are integers related to the system's order and time delay, and f is the unknown nonlinear function to be approximated by the NN, respectively.

Another class of NN is a recurrent neural network. The advantage of the recurrent neural network, as depicted in Figure 2.2, over the feedforward network is its better capability in long term prediction for chemical processes and thus it is more

suitable for predictive control application (Su et al., 1992; Su and McAvoy, 1997). Mathematically, the output of recurrent network is described by

$$\hat{y}(k) = f(\hat{y}(k-1), \cdots, \hat{y}(k-n_y), u(k-1-n_d), \cdots, u(k-n_u-n_d)) \tag{2.2}$$

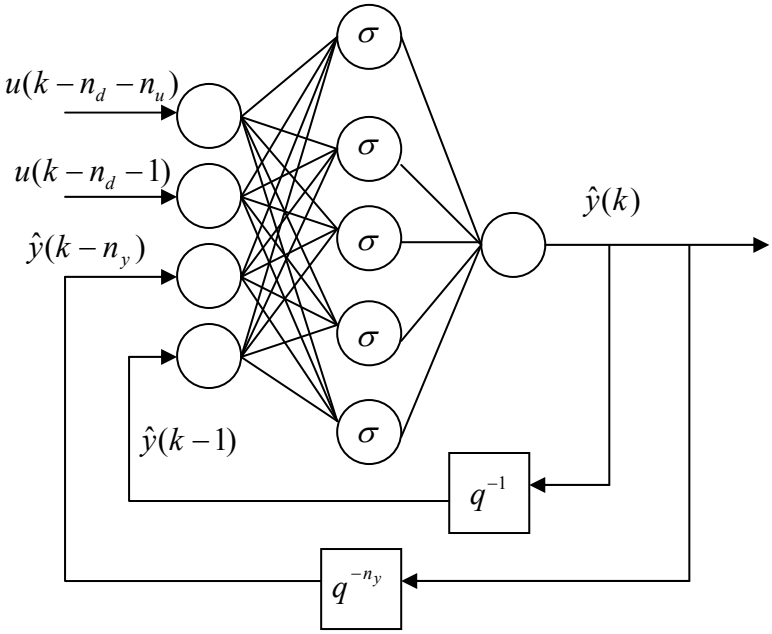

Figure 2.2 Structure of a recurrent neural network

2.1.2 Fuzzy neural network modeling technique

Both NN and fuzzy systems are motivated by imitating human reasoning processes. Fuzzy reasoning is already proven in handling imprecise and uncertain information. However, there are several difficulties associated with fuzzy logic methods. In a conventional fuzzy approach, the membership functions and the consequent models are chosen by the designer according to his/her priori knowledge. However, this fuzzy approach is often time-consuming and not straightforward because it relies on process experts who may not be able to transcribe their knowledge into requisite fuzzy rule form. Moreover, there are no formal frameworks to choose

the parameters of fuzzy models. To overcome those drawbacks, fuzzy logic methods are integrated together with NN to construct the fuzzy neural network (FNN). By using the learning capability of the NN, FNN can identify fuzzy rules and optimize membership function of fuzzy model (Lin and Lee, 1991; Jang, 1993; Jang and Sun, 1995).

In the context of FNN, the fuzzy model commonly used is the Takagi-Sugeno (T-S) fuzzy model (Takagi and Sugeno, 1985). Applying T-S model to describe dynamic system is equivalent to dividing the operating space of a dynamic system into several local operating regions. Within each local region, one fuzzy rule R^l is used to represent the process behavior. Specifically, in T-S model, the rule antecedents describe fuzzy region in the input space and the rule consequents are crisp function of the model inputs:

$$\begin{aligned} R^l &: IF\ x_1\ is\ F_1^l\ AND\ x_2\ is\ F_2^l \ldots AND\ x_M\ is\ F_M^l, \\ &THEN\ \hat{y}_l = f(\mathbf{x});\ \ l = 1,2,\cdots,N \end{aligned} \tag{2.3}$$

where R^l denotes the l-th fuzzy rule, $\mathbf{x} = (x_1, x_2, ..., x_M)$ is the input variable of the FNN system, $\hat{y}_l$ is the model prediction of the l-th fuzzy rule, F_i^l denotes the fuzzy sets defined on the corresponding universe [0, 1], and N is the total number of fuzzy rules. Normally, the consequents employ a liner model, i.e. $\hat{y}_l = \sum_{i=1}^{M} w_{li} x_i + b_l$, where w_{li} and b_l are the model parameters.

The output of the model is calculated by the center of gravity defuzzification as follows:

$$\hat{y} = \frac{\sum_{l=1}^{N} \mu_l \hat{y}_l}{\mu_l} \tag{2.4}$$

where μ_l is the membership of the l-th rule antecedent.

2.1.3 Just-in-time learning modeling technique

Aha et al. (1991) developed instance-based learning algorithms for modeling nonlinear systems. This approach is inspired by ideas from local modeling and machine learning techniques. Subsequent to Aha's work, different variants of instance-based learning are developed, such as locally weighted learning (Atkeson et al., 1997a, 1997b) and just-in-time learning (JITL, Cybenko, 1996; Stenman, 1996; Bontempi et al., 1999, 2001). The JITL was recently developed as an attractive alternative for modeling the nonlinear systems because of its prediction capability for nonlinear processes and its inherently adaptive nature. JITL uses a query-based approach to select the best local model by assessing and comparing different alternatives in cross-validation.

JITL assumes that all available observations are stored in a database, and the models are built dynamically upon query. Compared with other learning algorithms, JITL exhibits three main characteristics. First, the model-building phase is postponed until an output for a given query data is requested. Next, the predicted output for the query data is computed by exploiting the stored data in the database. Finally, the constructed answer and any intermediate results are discarded after the answer is obtained (Atkeson et al., 1997a, 1997b; Bontempi et al., 2001; Nelles, 2001).

There are many benefits offered by the JITL technique. JITL has no standard learning phase because it merely stores the data in the database and the computation is not performed until a query data arrives. Moreover, JITL constructs local approximation of the dynamic systems characterized by the current query data. Therefore, a simple model structure can be chosen, e.g. a first-order or second-order ARX model. In addition, JITL inherent adaptive nature is achieved by storing the current measured data into the database. It is important to point out that the selection

of relevant data is carried out individually for each incoming query data. This allows one to change the model architecture, model complexity, and the criteria for relevant data selection on-line according to the current situation (Nelles, 2001).

To achieve better predictive performance of JITL algorithm, Cheng and Chiu (2004) recently proposed an enhanced JITL algorithm by using a new similarity measure by combining the conventional distance measure with the angular relationship. In the following, the JITL algorithm developed in Cheng and Chiu (2004), which is used in this thesis, is described.

JITL consists of three main steps in order to calculate the model output corresponding to the query data: (i) finding the relevant data samples in the database corresponding to the query data by the nearest-neighborhood criterion; (ii) constructing a low-order local model based on the relevant data; and (iii) obtaining the model output based on the local model and the current query data. When the next query data is available, a new local model will be built based on the aforementioned procedure.

To proceed with the JITL technique, a required initial database is constructed by using process input and output data obtained around nominal operating condition. This database can be updated subsequently during its on-line implementation when modeling error between process output and predicted output by JITL is greater than the pre-specified threshold. In those cases, the current process data is considered as new data that is not adequately represented by the present database and is thus added to the database to improve its prediction accuracy for new operating region where the process data may not be available to construct the initial database for JITL.

The JITL technique is mainly used to identify the current process dynamics at each sampling instant by focusing on the relevant region around the current operating

condition. Therefore, a simple first-order or second-order ARX model is usually used as a local model.

$$\hat{y}(k) = \alpha_1^k y(k-1) + \alpha_2^k y(k-2) + \beta^k u(k-1) \tag{2.5}$$

where $\hat{y}(k)$ is the predicted output by JITL model, $y(k-1)$ and $u(k-1)$ denote the process output and input at the (k – 1)-th sampling instant, and the model parameters α_1^k, α_2^k, and β^k are calculated by JITL at the k-th sampling instant.

Based on Eq. (2.5), the regression vector for the ARX model is defined as

$$\mathbf{x}(k) = \left[y(k-1) \;\; y(k-2) \;\; u(k-1) \right] \tag{2.6}$$

Suppose the present JITL's database consists of D data $(y(k), \mathbf{x}(k))_{k=1\text{-}D}$. The following similarity measure, s_k, is used to select the relevant regression vectors from the database that resembles the query data $\mathbf{x}_q$:

$$s_k = \phi \sqrt{e^{-\left\| \mathbf{x}_q - \mathbf{x}(k) \right\|^2}} + (1-\phi)\cos(\theta_k), \quad \text{if } \cos(\theta_k) \geq 0 \tag{2.7}$$

where ϕ is a weight parameter constrained between 0 and 1, $\|\cdot\|$ is an Euclidean norm, and θ_k is the angle between $\Delta\mathbf{x}_q$ and $\Delta\mathbf{x}(k)$, where $\Delta\mathbf{x}_q = \mathbf{x}_q - \mathbf{x}_{q-1}$ and $\Delta\mathbf{x}(k) = \mathbf{x}(k) - \mathbf{x}(k-1)$. The value of s_k is bounded between 0 and 1. When s_k approaches to 1, it indicates that $\mathbf{x}(k)$ resembles closely to $\mathbf{x}_q$.

After all s_k are computed by Eq. (2.7), for each $h \in [k_{\min}\ k_{\max}]$, where $k_{\min}$ and $k_{\max}$ are the pre-specified minimum and maximum numbers of relevant data, the relevant data set $(\mathbf{y}_h, \Phi_h)$ is constructed by selecting the h most relevant data $(y(k), \mathbf{x}(k))$ corresponding to the largest s_k to the h-th largest s_k. Next, the leave-one-out cross validation test is conducted and the validation error is calculated. Upon the completion of the above procedure, the optimal h, h^*, is determined by that giving

the smallest validation error. Subsequently, the predicted output for query data is calculated as $\mathbf{x}_q^{\mathrm{T}} \left(\mathbf{P}_{h^*}^{\mathrm{T}} \mathbf{P}_{h^*} \right)^{-1} \mathbf{P}_{h^*}^{\mathrm{T}} \mathbf{W}_{h^*} \mathbf{y}_{h^*}$, where $\mathbf{P}_{h^*}^{\mathrm{T}} = \mathbf{W}_{h^*} \Phi_{h^*}$ and $\mathbf{W}_{h^*}$ is a diagonal matrix with entries being the first h^* largest s_k.

2.2 Adaptive Controller Design for Nonlinear Processes

Even though most processes in the chemical process industry are nonlinear in nature, most controller designs have used linear control techniques to control such systems. The prevalence of linear control strategies is partly due to the fact that, over the normal operating region, many of the processes can be approximated by linear models, which can be obtained by the well-established identification methods. In addition, the theories for the stability analysis of linear control systems are quite well developed so that linear control techniques are widely accepted. In contrast, controller design for nonlinear models is considerably more difficult than that for linear models. However, linear control design methodologies may not be adequate to achieve satisfactory control performance for nonlinear chemical processes. This has led to an increasing interest in the nonlinear controller design for the nonlinear dynamic processes.

Process control systems inevitably require adjustable controller settings to facilitate process operation over a wide range of conditions. Typically, controller settings are designed after the implementation of control system. If the process operating condition or the environment changes significantly, the controller may then have to be retuned. If these changes occur frequently, adaptive control techniques should be considered. Most adaptive control techniques integrate a set of techniques for automatic adjustment of controller parameters in real time in order to achieve or

maintain desired control performance when the process dynamics are unknown or vary in time. Adaptive control schemes provide systematic and flexible approaches for dealing with uncertainties, nonlinearities, and time-varying process parameters. The diagram of adaptive control concept is depicted in Figure 2.3.

In recent years, there has been extensive interest in adaptive control systems. With the progressing of control theories and computer technology, various adaptive control methodologies were proposed for process control in the last three decades. There are two distinct adaptive control categories (Narendra and Parthasarathy, 1990; Chen and Teng, 1995): (1) direct adaptive control and (2) indirect adaptive control. In direct adaptive control, the parameters of the controller are directly adjusted to reduce the error between the plant and the reference model. On the other hand, in indirect adaptive control, the parameters of the plant are estimated and the controller is chosen assuming that the estimated parameters represent the true values of the plant parameters.

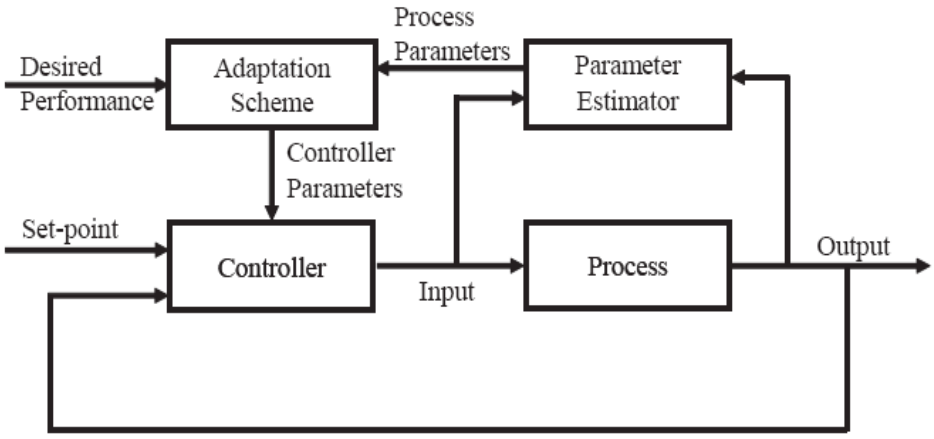

Figure 2.3 Block diagram of adaptive control scheme

There are three main technologies for adaptive control: gain scheduling, model reference control, and self-tuning regulators. The purpose of these methods is to find a

convenient way of changing the controller parameters in response to changes in the process and environment dynamics.

Gain scheduling is one of the earliest and most intuitive approaches for adaptive control. The idea is to find process variables that correlate well with the changes in process dynamics and then possible to compensate for process parameter variations by changing the parameters of the controller as function of the process variables. The advantage of gain scheduling is that the parameters can be changed quickly in response to changes in the process dynamics. It is convenient especially if the process dynamics are in a well-known fashion on a relatively few easily measurable variables. Despite of the benefits, gain scheduling concept also suffers some drawbacks, such as open-loop compensation without feedback and no straightforward approach to select the appropriate scheduling variables for most chemical processes.

Model reference control is a class of direct self-tuners since no explicit estimate or identification of the process is made. The specifications are given in terms of "reference model" which tells how the process output ideally should respond to the command signal. The desired performance of the closed-loop system is specified through a reference model, and the adaptive system attempts to make the plant output match the reference model output asymptotically. The third class of adaptive control is self-tuning controller. The general strategy of this controller is to estimate model parameters on-line and then adjust the controller settings based on the current parameter estimate (Åström, 1983). In the self-tuning controller, the parameters in the process model are updated using on-line estimation methods from input-output data, and then the control calculations are based on the updated model. The self-tuning control strategy generally consists of three steps: (i) information gathering of the

present process behavior; (ii) control performance criterion optimization; and (iii) adjustment of the controller parameters. The first step implies the continuous determination of the actual condition of the process to be controlled based on measurable process input and output data and appropriate modeling approaches selected to identify the model parameters. Various types of model identification can be distinguished depending on the information gathered and the method of estimation. The last two steps calculate the control loop performance and the decision as to how the controller will be adjusted or adapted. These characteristics make self-tuning controller very flexible with respect to its choice of controller design methodology and to the choice of process model identification (Seborg et al., 1986; Seborg et al., 2004).

Chapter 3

Fuzzy Neural Network-Based Adaptive PID Controller Design

3.1 Introduction

The design of control systems is currently driven by a large number of requirements posed by increasing competition, environmental requirements, energy and material costs and the demand for robust, fault-tolerant systems. These considerations introduce extra needs for effective process modeling techniques. However, the construction of first-principle models is usually time-consuming and costly. Furthermore, model-based controller design by incorporating these models would lead to complex controller structure, not to mention that many chemical processes are not amenable to this modeling approach due to the lack of precise knowledge about the process (Babuška and Verbruggen, 2003). To this end, data-based methods arise as an attractive alternative for nonlinear system modeling in the

last two decades (Pearson, 1999; Nelles, 2001). One of the most well-known examples for data-based methods is neuro-fuzzy or fuzzy neural-network (FNN).

FNN has been recognized as a powerful approach which can facilitate the effective development of models by combining information from different sources, such as empirical models, heuristics and data, to solve many engineering problems. Chen and Teng (1995) proposed a model reference control structure using a FNN controller which is trained on-line using a FNN identifier with adaptive learning rates. Jang and Sun (1995) reviewed the fundamental and advanced developments in neuro-fuzzy models for modeling and control based on an adaptive network. Zhang and Morris (1995) described a technique for modeling of nonlinear systems using two different FNN topologies. Jang and Sun (1995) reviewed the fundamental and advanced developments in neuro-fuzzy synergisms for modeling and control based on an adaptive network. Wai and Lin (1998) applied a FNN controller with adaptive learning rates to control a nonlinear slider-crank mechanism system. Zhang and Morris (1999) designed a recurrent neuro-fuzzy network to build long-term prediction models for nonlinear processes. Lin and Wai (2001) developed a hybrid control system using recurrent fuzzy neural network to control linear induction motor servo drive. Juang (2002) proposed a Takagi-Sugeno-Kang (TSK) recurrent fuzzy neural network for dynamic system identification and controller design.

Fink et al. (2003) described three commonly used nonlinear model-based approaches for process model architectures originating from the fields of neural networks and fuzzy systems. Similar work is given by Babuška and Verbruggen (2003) which reviewed the neuro-fuzzy modeling methods for nonlinear systems identification with an emphasis on the tradeoff between accuracy and interpretability. Lee and Lin (2005) developed an adaptive filter which uses periodic fuzzy neural

network to treat the equalization of nonlinear time-varying systems. Lin and Chen (2006) proposed a compensation-based recurrent fuzzy neural network which employed adaptive fuzzy operations.

As the most widespread used controller in the process industries, PID controllers have the advantage of simple control structure, ease of implementation, and robustness in operation. Nevertheless, the conventional PID controller might be difficult to deal with highly nonlinear and time varying chemical processes. To improve the control performance, various adaptive PID controller designs have been developed in the literature. Riverol and Napolitano (2000) proposed the use of neural network to update the PID controller parameters on-line. Lu et al. (2001) constructed a predictive fuzzy PID controller by combining a fuzzy PID controller with model predictive controller. Andrasik et al. (2004) made use of two neural networks for on-line tuning of PID controller. Chen and Huang (2004) designed adaptive PID controller based on the instantaneous linearization of a neural network model. Sun et al. (2006) developed a self-tuning PID controller based on adaptive genetic algorithm and neural networks.

In the abovementioned works, the parameters of the process model are updated with respect to the current process condition and the PID parameters are then computed by the corresponding adaptation algorithm and implemented. However, these adaptation algorithms employed in the previous results are inadequate to address the convergence of the predicted tracking error. To this end, Chang et al. (2002) derived a stable adaptation mechanism in the continuous time domain by the Lyapunov approach such that the PID controller tracks a pre-specified feedback linearization control asymptotically. Motivated by this work, a self-tuning algorithm

derived from Lyapunov method in the discrete time for adaptive PID design based on FNN modeling technique will be developed in this thesis.

In the following sections, FNN modeling strategy is presented and the detail of the proposed PID controller design is discussed. Literature examples are then presented to illustrate the proposed control strategy and a comparison with its conventional counterpart is made.

3.2 Fuzzy Neural Network-Based Modeling

FNN is recently developed neural network-based fuzzy logic control and decision system which is suitable for on-line nonlinear systems identification and control. The FNN is a multilayer feedforward network which integrates the TSK-type fuzzy logic system and radial basis function neural network into a connection structure. Without loss of generality, the following first-order TSK-type fuzzy rule is considered:

$$\begin{aligned} &R^l : IF\ x_1\ is\ F_1^l\ AND\ x_2\ is\ F_2^l \ldots AND\ x_M\ is\ F_M^l, \\ &\quad THEN\ \hat{y}_l(k) = \alpha_l y(k-1) + \beta_l u(k-1); \quad l = 1, 2, \cdots, N \end{aligned} \tag{3.1}$$

where R^l denotes the l-th fuzzy rule, $\mathbf{x}(k) = \left(x_1(k)\ x_2(k) \ldots x_M(k)\right)$ is the input variable of the FNN system, $\hat{y}_l(k)$ is the model prediction of the l-th fuzzy rule, $y(k-1)$ and $u(k-1)$ denote the output and input of the system at the $(k-1)$-th sampling instant, F_i^l denotes the fuzzy sets defined on the corresponding universe [0, 1], and N is the total number of fuzzy rules.

The FNN consists of five layers as depicted in Figure 3.1. The first layer is called input layer. The nodes in this layer just transmit the input variables to the next layer, expressed as:

$$\begin{aligned} &Input \quad : I_i^{(1)} = x_i, \ \ i = 1, 2, \cdots, M \\ &Output : O_i^{(1)} = I_i^{(1)}, \ \ i = 1, 2, \cdots, M \end{aligned} \tag{3.2}$$

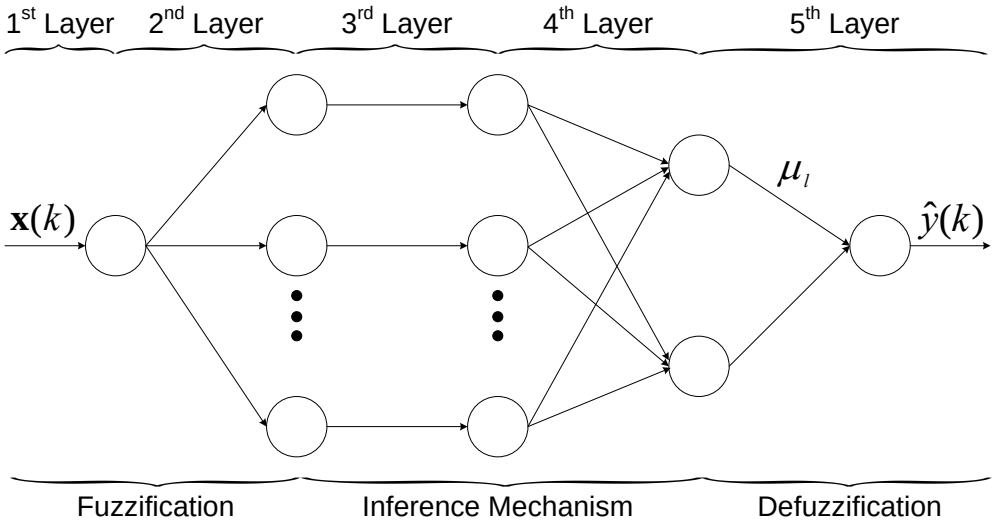

Figure 3.1 The structure of FNN system

The second layer is composed of N fuzzy *if – then* rules. Each rule has M neurons to receive inputs from every neurons of the first layer, by which the membership function of each fuzzy rule is calculated. In this thesis, Gaussian membership function is chosen, and thus the membership function of l-th rule in this layer can be expressed as:

$$\begin{aligned} &Input \quad : I_{li}^{(2)} = x_{li} \\ &Output : O_{li}^{(2)} = \exp\left(-\frac{\left(I_{li}^{(2)} - c_{li} \right)^2}{\chi_l^2} \right); \ \ i = 1, 2, \cdots, M; \quad l = 1, 2, \cdots, N \end{aligned} \tag{3.3}$$

The third layer consists of N neurons, which compute the fired strength of a rule. The l-th neuron receives only inputs from the corresponding neurons of the second layer. The input and output of every neuron is represented as follows:

$$\begin{aligned} &Input \quad : I_l^{(3)} = O_l^{(2)} \\ &Output : O_l^{(3)} = \prod_{i=1}^{M} O_{li}^{(2)}; \quad l = 1, 2, \cdots, N \end{aligned} \tag{3.4}$$

There are two neurons in fourth layer. One neuron connects with all neurons of the third layer through unity weight and another one connects with all neurons of the third layer through the weights $\hat{y}_l$, as described below:

$$\begin{aligned} Input \quad &: I_1^{(4)} = \left[O_1^{(3)}, O_2^{(3)}, \cdots, O_N^{(3)} \right] \\ &\quad I_2^{(4)} = \left[O_1^{(3)}, O_2^{(3)}, \cdots, O_N^{(3)} \right] \\ Output &: O_1^{(4)} = \sum_{l=1}^{N} O_l^{(3)} \\ &\quad O_2^{(4)} = \sum_{l=1}^{N} \hat{y}_l O_l^{(3)} \end{aligned} \tag{3.5}$$

The last layer has a single neuron to compute the predicted output $\hat{y}$. It is connected with two neurons of the fourth layer through unity weights in which defuzzification is performed. The integral function and activation function of the node can be expressed as:

$$\begin{aligned} Input \quad &: I^{(5)} = \left[O_1^{(4)}, O_2^{(4)} \right] \\ Output &: O^{(5)} = \frac{O_2^{(4)}}{O_1^{(4)}} \end{aligned} \tag{3.6}$$

The output of the whole FNN is then obtained as:

$$\hat{y} = O^{(5)} = \sum_{l=1}^{N} \mu_l \hat{y}_l \tag{3.7}$$

where

$$\mu_l = \frac{\exp\left(-\sum_{i=1}^{M} \frac{\left(x_i - c_{li} \right)^2}{\chi_l^2} \right)}{\sum_{l=1}^{N} \exp\left(-\sum_{i=1}^{M} \frac{\left(x_i - c_{li} \right)^2}{\chi_l^2} \right)} \tag{3.8}$$

3.3 Adaptive FNN-PID Control Scheme

In this section, the proposed adaptive PID control scheme as shown in Figure 3.2 will be described in details. The nonlinear processes under PID control are approximated by a fuzzy neural network model (FNNM), which provides information

to adjust the PID parameters by an updating algorithm derived from Lyapunov method.

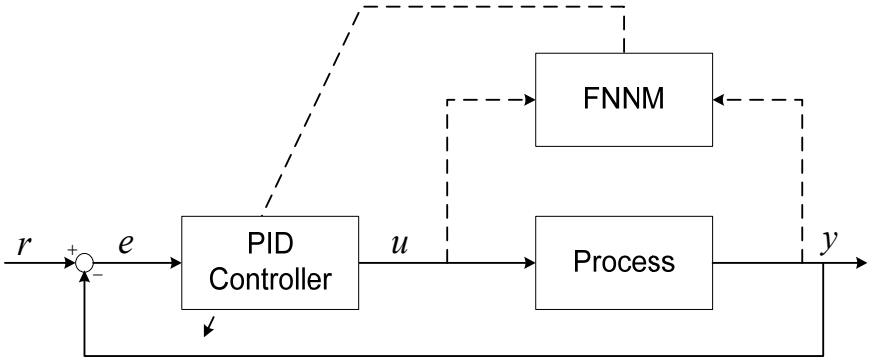

Figure 3.2 The structure of FNN-PID controller system

The nonlinear process can be represented by the following discrete nonlinear function

$$y(k+1) = f(\mathbf{z}(k)) \tag{3.9}$$

where

$$\mathbf{z}(k) = \left(y(k), y(k-1), \cdots, y(k-n_y), u(k-n_d), u(k-n_d-1), \cdots, u(k-n_d-n_u)\right)^{\mathrm{T}} \tag{3.10}$$

where n_y, n_u, and n_d are integers related to the system's order and time delay, respectively.

The FNNM is employed for nonlinear process modeling due to its capability of uniformly approximating any nonlinear function to any degree of accuracy, namely,

$$\hat{y}(k+1) = FNNM(\mathbf{x}(k)) \tag{3.11}$$

The input $\mathbf{x}(k)$ used in this thesis is defined by first-order as follows

$$\mathbf{x}(k) = \left(y(k-1), u(k-1)\right)^{\mathrm{T}} \tag{3.12}$$

The method employed for the identification of FNNM can be summarized as follows:

1. The first input data point, $\mathbf{x}(1)$, is chosen as the first cluster (fuzzy rule) and its cluster center is set as $c_1 = \mathbf{x}(1)$. The number of input data point belonging to the first cluster, N_1, and the number of fuzzy clusters, N, at this time are respectively $N_1 = 1$ and $N = 1$;
2. For the k-th training data point, $\mathbf{x}(k)$, determine the largest similarity measure, S_L, between $\mathbf{x}(k)$ to every cluster centers, $c_l\,(l = 1, 2, \cdots, N)$, according to Eq. (3.13), and the corresponding cluster is denoted by c_L.

$$S_L = \max_{1 \le l \le N} \left(\sqrt{e^{-\|\mathbf{x}(k) - c_l\|_2^2}} \right) \tag{3.13}$$

3. Next, decide whether a new cluster (fuzzy rule) should be added or not, according to the following criteria:
   - If $S_L < \psi$ where ψ is a pre-specified threshold, the k-th training data point does not belong to all the existing cluster and a new cluster will be established with its center located at $c_{N+1} = \mathbf{x}(k)$, and set $N = N + 1$ and $N_{N+1} = 1$, while other clusters remain unchanged;
   - If $S_L \ge \psi$, the k-th training data point belong to the L-th cluster and its corresponding center is adjusted as follows

$$c_L = c_L + \frac{\lambda_0}{N_L + 1}\left(\mathbf{x}(k) - c_L\right); \quad \lambda_0 \in [0,1] \tag{3.14}$$

   and set $N_L = N_L + 1$.
4. Set $k = k + 1$ and go to step 2 until all training data points are clustered to the corresponding cluster. After finishing the first three steps, the width of each fuzzy rule can be calculated as:

$$\chi_l = \min_{j = 1, 2, \cdots, N,\ j \ne l} \frac{|c_l - c_j|}{\rho} \tag{3.15}$$

where ρ is overlap parameter, usually $1 \le \rho \le 2$.

5. The consequent parameters, α_l and β_l $(l = 1,2,\cdots,N)$, are obtained by using least square method as given by:

$$\begin{bmatrix} \alpha_1 \\ \beta_1 \\ \alpha_2 \\ \beta_2 \\ \vdots \\ \alpha_N \\ \beta_N \end{bmatrix} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}} \begin{bmatrix} y(2) \\ y(3) \\ y(4) \\ \vdots \\ y(N_t) \end{bmatrix} \tag{3.16}$$

where N_t is the total number of training data and

$$\mathbf{A} = \begin{bmatrix} \mu_{11}y(1) & \mu_{11}u(1) & \cdots & \mu_{N1}y(1) & \mu_{N1}u(1) \\ \mu_{12}y(2) & \mu_{12}u(2) & \cdots & \mu_{N2}y(2) & \mu_{N2}u(2) \\ \mu_{13}y(3) & \mu_{13}u(3) & \cdots & \mu_{N3}y(3) & \mu_{N3}u(3) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \mu_{1N_t}y(N_t-1) & \mu_{1N_t}u(N_t-1) & \cdots & \mu_{NN_t}y(N_t-1) & \mu_{NN_t}u(N_t-1) \end{bmatrix} \tag{3.17}$$

where μ_{lj} is the membership function of the l-th fuzzy rule corresponding to the input $x_l(j)$ $(j = 1,2,\cdots,N_t)$.

With the FNN model obtained off-line according to the abovementioned procedure, it will then be incorporated into the proposed adaptive PID controller design to be detailed in the sequel. The PID control law of the proposed design is expressed as follows:

$$u(k) = u(k-1) + \Delta u(k) \tag{3.18}$$

$$\Delta u(k) = w_1(k)e(k) + w_2(k)\Delta e(k) + w_3(k)\delta e(k) \tag{3.19}$$

where $w_1(k)$, $w_2(k)$ and $w_3(k)$ are the PID controller parameters obtained at the k-th sampling instant, $e(k)$ is the error between process output, y, and its set-point, r, at the k-th sampling instant, $\Delta e(k) = e(k) - e(k-1)$, and $\delta e(k) = \Delta e(k) - \Delta e(k-1)$.

Since the controller parameters, w_j, are constrained to be positive or negative, the following function is introduced to map the set of positive (or negative) number to the set of real number:

$$w_j(k)=\begin{cases} e^{\varsigma_j(k)}, & \text{if} \quad w_j(k)\geq 0 \\ -e^{\varsigma_j(k)}, & \text{if} \quad w_j(k)<0 \end{cases}, \quad j=1\sim 3 \tag{3.20}$$

where $\varsigma_j(k)$ is a real number. In the sequel, an updating algorithm will be developed to adjust $\varsigma_j(k)$ on-line, and subsequently the FNN-PID parameters $w_j(k)$ can be easily calculated by Eq. (3.20).

To facilitate the subsequent development, the following notations are introduced:

$$e_x(k)=[e(k)\ \ \Delta e(k)\ \ \delta e(k)] \tag{3.21}$$

$$w(k)=[w_1(k)\ \ w_2(k)\ \ w_3(k)]^{\mathrm{T}} \tag{3.22}$$

$$\varsigma(k)=[\varsigma_1(k)\ \ \varsigma_2(k)\ \ \varsigma_3(k)]^{\mathrm{T}} \tag{3.23}$$

In order to update the parameter $\varsigma_j(k)$ at each sampling time so that the FNNM's predicted output converges to the desired set-point trajectory, the following theorem gives the theoretical basis for the convergence property of the proposed updating algorithm for $\varsigma(k)$.

Theorem 1. Considering nonlinear processes of Eq. (3.9) controlled by the FNN-PID controller of Eq. (3.18) with the following updating law and the learning rates η_1, η_2, and $\eta_3<2$,

$$\begin{aligned}
\varsigma(k+1) &= \varsigma(k) + \Delta\varsigma(k) \\
\Delta\varsigma(k) &= \frac{1}{\dfrac{\partial\hat{y}(k+1)}{\partial u(k)}} \cdot \begin{bmatrix} \eta_1 & 0 & 0 \\ 0 & \eta_2 & 0 \\ 0 & 0 & \eta_3 \end{bmatrix} \cdot \left[\frac{\partial w(k)}{\partial\varsigma(k)}\right]^{-1} \cdot \frac{e_x(k)^{\mathrm{T}} e_r(k)}{e_x(k) e_x(k)^{\mathrm{T}}} \\
\frac{\partial w(k)}{\partial\varsigma(k)} &= \begin{bmatrix} w_1(k) & 0 & 0 \\ 0 & w_2(k) & 0 \\ 0 & 0 & w_3(k) \end{bmatrix}
\end{aligned} \tag{3.24}$$

If the Lyapunov function candidate is chosen as

$$v(k) = \xi e_r^2(k) \tag{3.25}$$

where $e_r(k) = r(k) - \hat{y}(k)$ and ξ is a positive constant, then $\Delta v(k) < 0$ always holds. Thus, the predicted tracking error is guaranteed to converge to zero asymptotically.

Proof. Define

$$e_r(k+1) = e_r(k) + \Delta e_r(k+1) \tag{3.26}$$

By considering Eqs. (3.25) and (3.26), the following relationship can be obtained:

$$\begin{aligned}
\Delta v(k) &= v(k+1) - v(k) = \xi e_r^2(k+1) - \xi e_r^2(k) \\
&= 2\xi e_r(k)\Delta e_r(k+1) + \xi \Delta e_r^2(k+1)
\end{aligned} \tag{3.27}$$

In Eq. (3.27), $\Delta e_r(k+1)$ can be further expressed as

$$\begin{aligned}
\Delta e_r(k+1) &= \frac{\partial e_r(k+1)}{\partial k}\Delta k \\
&= \frac{\partial\left[r(k+1) - \hat{y}(k+1)\right]}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial w(k)} \cdot \frac{\partial w(k)}{\partial\varsigma(k)} \cdot \frac{\partial\varsigma(k)}{\partial k}\Delta k \\
&= -\frac{\partial\hat{y}(k+1)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial w(k)} \cdot \frac{\partial w(k)}{\partial\varsigma(k)} \cdot \Delta\varsigma(k)
\end{aligned} \tag{3.28}$$

where the partial derivative $\dfrac{\partial\hat{y}(k+1)}{\partial u(k)}$ can be derived from the FNNM as follows:

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{l=1}^{N} 2u(k)\beta_l g_l \cdot \frac{\sum_{l=1}^{N} \frac{(u(k)-c_{lm})g_l}{\chi_l^2} - \frac{(u(k)-c_{lm})}{\chi_l^2}\sum_{l=1}^{N} g_l}{\left(\sum_{l=1}^{N} g_l\right)^2} + \sum_{l=1}^{N} \beta_l \frac{g_l}{\sum_{l=1}^{N} g_l}$$
$$+ \sum_{l=1}^{N} 2y(k)\alpha_l g_l \cdot \frac{\sum_{l=1}^{N} \frac{(u(k)-c_{lm})g_l}{\chi_l^2} - \frac{(u(k)-c_{lm})}{\chi_l^2}\sum_{l=1}^{N} g_l}{\left(\sum_{l=1}^{N} g_l\right)^2} \tag{3.29}$$

and $g_l = \exp\left(-\sum_{i=1}^{M} \frac{(x_i - c_{li})^2}{\chi_l^2}\right)$.

According to Eqs. (3.24) and (3.29), Eq. (3.27) is then expressed as

$$\begin{aligned}
&\Delta v(k) \\
&= -2\xi e_r(k) \cdot \frac{\partial \hat{y}(k+1)}{\partial u(k)} \cdot e_x(k) \cdot \frac{\partial w(k)}{\partial \varsigma(k)} \cdot \frac{1}{\frac{\partial \hat{y}(k+1)}{\partial u(k)}} \cdot \begin{bmatrix} \eta_1 & 0 & 0 \\ 0 & \eta_2 & 0 \\ 0 & 0 & \eta_3 \end{bmatrix} \cdot \left[\frac{\partial w(k)}{\partial \varsigma(k)}\right]^{-1} \frac{e_x(k)^{\mathrm{T}} e_r(k)}{e_x(k) e_x(k)^{\mathrm{T}}} \\
&\quad + \xi \left(-\frac{\partial \hat{y}(k+1)}{\partial u(k)} \cdot e_x(k) \cdot \frac{\partial w(k)}{\partial \varsigma(k)} \cdot \frac{1}{\frac{\partial \hat{y}(k+1)}{\partial u(k)}} \cdot \begin{bmatrix} \eta_1 & 0 & 0 \\ 0 & \eta_2 & 0 \\ 0 & 0 & \eta_3 \end{bmatrix} \cdot \left[\frac{\partial w(k)}{\partial \varsigma(k)}\right]^{-1} \frac{e_x(k)^{\mathrm{T}} e_r(k)}{e_x(k) e_x(k)^{\mathrm{T}}} \right)^2 \\
&= -2\xi e_r(k) \cdot e_x(k) \cdot \begin{bmatrix} \eta_1 & 0 & 0 \\ 0 & \eta_2 & 0 \\ 0 & 0 & \eta_3 \end{bmatrix} \cdot \frac{e_x(k)^{\mathrm{T}} e_r(k)}{e_x(k) e_x(k)^{\mathrm{T}}} + \xi \left(-e_x(k) \cdot \begin{bmatrix} \eta_1 & 0 & 0 \\ 0 & \eta_2 & 0 \\ 0 & 0 & \eta_3 \end{bmatrix} \cdot \frac{e_x(k)^{\mathrm{T}} e_r(k)}{e_x(k) e_x(k)^{\mathrm{T}}} \right)^2 \\
&= -2\xi e_r^2(k) \cdot e_x(k) \cdot \begin{bmatrix} \eta_1 & 0 & 0 \\ 0 & \eta_2 & 0 \\ 0 & 0 & \eta_3 \end{bmatrix} \cdot \frac{e_x(k)^{\mathrm{T}}}{e_x(k) e_x(k)^{\mathrm{T}}} + \xi e_r^2(k) \left(e_x(k) \cdot \begin{bmatrix} \eta_1 & 0 & 0 \\ 0 & \eta_2 & 0 \\ 0 & 0 & \eta_3 \end{bmatrix} \cdot \frac{e_x(k)^{\mathrm{T}}}{e_x(k) e_x(k)^{\mathrm{T}}} \right)^2 \\
&= -2\xi e_r^2(k) \cdot \frac{\eta_1 e^2(k) + \eta_2 \Delta e^2(k) + \eta_3 \delta e^2(k)}{e^2(k) + \Delta e^2(k) + \delta e^2(k)} + \xi e_r^2(k) \left(\frac{\eta_1 e^2(k) + \eta_2 \Delta e^2(k) + \eta_3 \delta e^2(k)}{e^2(k) + \Delta e^2(k) + \delta e^2(k)} \right)^2 \\
&= \xi e_r^2(k) \cdot \frac{\eta_1 e^2(k) + \eta_2 \Delta e^2(k) + \eta_3 \delta e^2(k)}{e^2(k) + \Delta e^2(k) + \delta e^2(k)} \cdot \left(\frac{(\eta_1 - 2) e^2(k) + (\eta_2 - 2) \Delta e^2(k) + (\eta_3 - 2) \delta e^2(k)}{e^2(k) + \Delta e^2(k) + \delta e^2(k)} \right)
\end{aligned} \tag{3.30}$$

It is evident from Eq. (3.30) that $\Delta v(k)$ is always negative if $0 < \eta_j < 2$ holds, meaning that tracking error $e_r(k)$ is guaranteed to converge to zero by using the updating algorithm, Eq. (3.24), to design $\varsigma(k)$. This completes the proof.

The implementation of the proposed FNN-PID control algorithm is summarized as follows:

1. Given the learning rates η_j and initial FNN-PID controller parameters w_j;
2. Given the measured process output $y(k)$, compute the manipulated variable $u(k)$ from Eq. (3.18);
3. Update $\varsigma_j(k)$ by using Eq. (3.24) and consequently, FNN-PID parameters at the next sampling instant, $w_j(k+1)$, are calculated by using Eq. (3.20).
4. Set $k = k+1$ and go to step 2.

3.4 Examples

Example 1 The first example considered is a continuous polymerization reaction that takes place in a jacketed CSTR as depicted in Figure 3.3, where an isothermal free-radical polymerization of methyl methacrylate (MMA) is carried out using azo-bis-isobutyronitrile (AIBN) as initiator and toluene as solvent. Under the following assumptions (Doyle et al., 1995): (i) isothermal operation; (ii) perfect mixing; (iii) constant heat capacity; (iv) no polymer in the inlet stream; (v) no gel effect; (vi) constant reactor volume; (vii) negligible initiator flow rate (in comparison with monomer flow rate); and (viii) quasi-steady state and long-chain hypothesis. The dynamics of this reactor can be described by the following equations:

$$\frac{dC_m}{dt} = -(k_p + k_{fm})C_m P_0 + \frac{F(C_{m_{in}} - C_m)}{V} \tag{3.31}$$

$$\frac{dC_I}{dt} = -k_I C_I + \frac{F_i C_{I_{in}} - FC_I}{V} \tag{3.32}$$

$$\frac{dD_0}{dt} = (0.5k_{T_c} + k_{T_d})P_0^2 + k_{f_m} C_m P_0 - \frac{FD_0}{V} \tag{3.33}$$

$$\frac{dD_I}{dt} = M_m(k_p + k_{f_m})C_m P_0 - \frac{FD_I}{V} \tag{3.34}$$

$$y = \frac{D_I}{D_0} \tag{3.35}$$

$$P_0 = \left[\frac{2f * k_1 C_I}{k_{T_d} + k_{T_c}} \right]^{0.5} \tag{3.36}$$

The control objective is to regulate the product number average molecular weight (y = NAMW) by manipulating the flow rate of the initiator ($u = F_i$). The operating space considered is NAMW $\in$ [12500 25000]. The model parameters and steady-state operation condition are given in Tables 3.1 and 3.2.

To apply FNNM for process modeling, input and output data are generated by introducing uniformly random steps with distribution of [0.01 0.08] in process input. The process input and output (depicted in Figure 3.4) are then scaled by $\tilde{u} = \frac{u - 0.016783}{0.016783}$ and $\tilde{y} = \frac{y - 25000.5}{25000.5}$, respectively. Both process input and output are corrupted by 5% Gaussian white noise. With sampling time of 0.03h, input and output data thus obtained are used to build the database.

Validation tests (see Figure 3.5 for an illustration) are carried out to determine the optimal parameters for FNNM algorithm as follows: ψ = 0.9984, λ_0 = 0.4, and ρ = 1.28. To design FNN-PID controller, initial PID parameters $w_1 = -1.39$, $w_2 = -7.81$, and $w_3 = -2.3$ are designed and their corresponding learning rates are specified as $\eta_1 = 1.35 \times 10^{-4}$, $\eta_2 = 1.16 \times 10^{-3}$, and $\eta_3 = 7.17 \times 10^{-4}$.

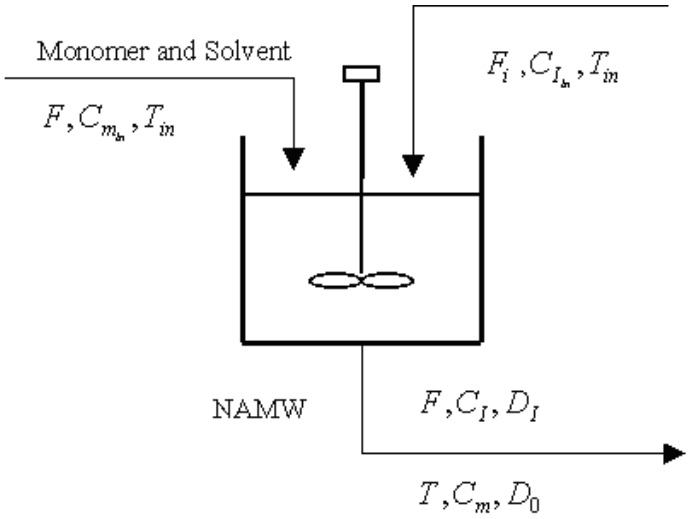

Figure 3.3 Polymerization reactor

Table 3.1 Model parameters for polymerization reactor

k_{T_c}	$= 1.3281\times10^{10}$ m^3/(kmol h)	F	$= 1.00$ m^3/h
k_{T_d}	$= 1.0930\times10^{11}$ m^3/(kmol h)	V	$= 0.1$ m^3
k_I	$= 1.0225\times10^{-1}$ L/h	$C_{I_{in}}$	$= 8.0$ kmol/m^3
k_p	$= 2.4952\times10^{6}$ m^3/(kmol h)	M_m	$= 100.12$ kg/kmol
k_{f_m}	$= 2.4522\times10^{3}$ m^3/(kmol h)	$C_{m_{in}}$	$= 6.0$ kmol/m^3
f^*	$= 0.58$		

Table 3.2 Steady-state operating condition of polymerization reactor

C_m	$= 5.506774$ kmol/m^3	D_I	$= 49.38182$ kmol/m^3
C_I	$= 0.132906$ kmol/m^3	u	$= 0.016783$ m^3/h
D_0	$= 0.0019752$ kmol/m^3	y	$= 25000.5$ kg/kmol

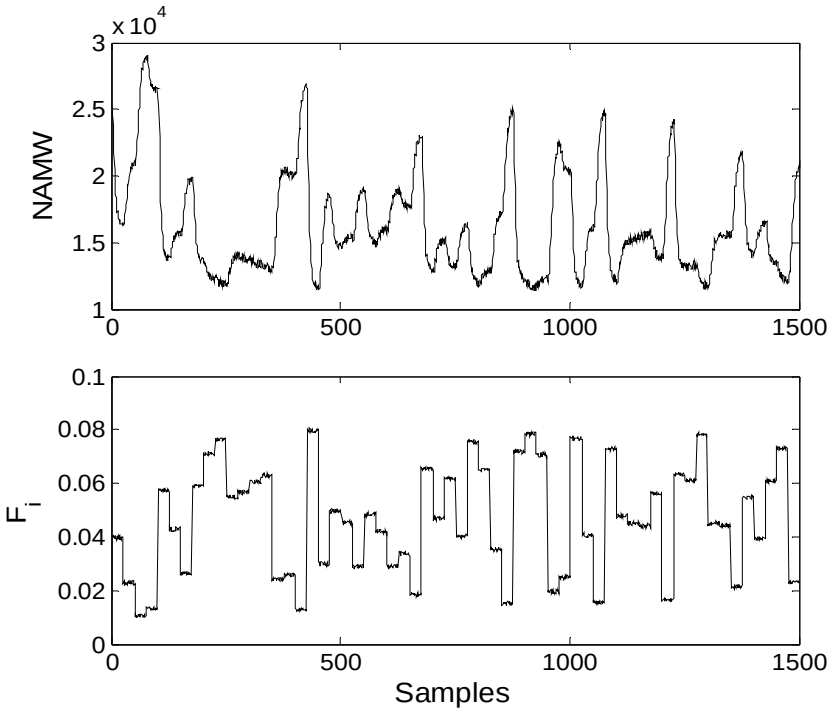

Figure 3.4 Input and output data used to construct the FNN model in polymerization reactor example

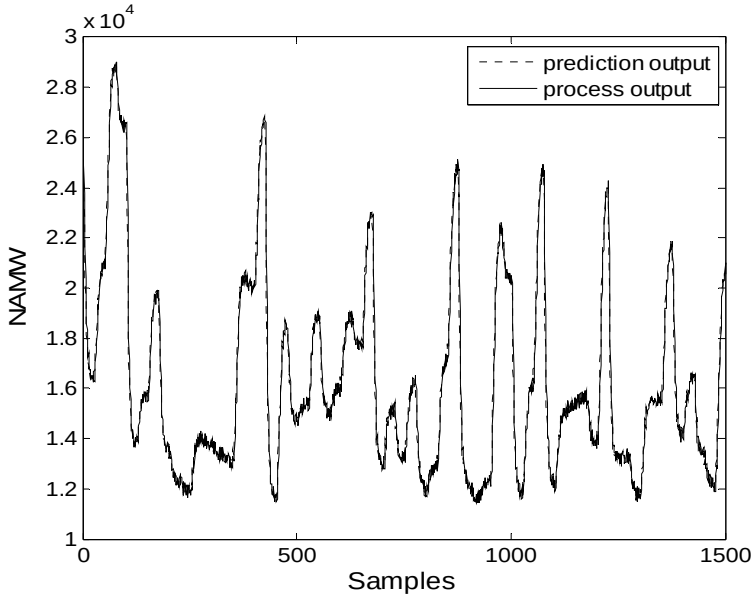

Figure 3.5 Validation of FNN model

For the comparison purpose, an adaptive PID controller is designed based on a second-order ARX model with parameter adaptation by the recursive least-square (RLS) identification procedure (Shahrokhi and Baghmisheh, 2005). To compare the performances of two PID designs, successive set-point changes between 25000.5 and 12500 kg/kmol are conducted. As can be seen from Figure 3.6, it is obvious that the proposed FNN-PID controller has better performance than that achieved by the RLS-based PID controller, resulting in the reduction of Mean Absolute Error (MAE) by 23.4%. Figure 3.7 shows the updating of controller parameters in the FNN-PID design.

By assuming ±25% step disturbances in the monomer initiator concentration, the resulting performances of two controllers at different operating conditions are compared in Figures 3.8 and 3.9. The FNN-PID controller achieves better control performance by giving shorter settling time compared to RLS-based PID controller, as evidenced by the reduction of MAE ranging from 14% to 49%. To evaluate the robustness of the proposed controller, it is assumed that there exist 10% modeling error in the kinetic parameter k_{I} and 20% error in the gain coefficients of the D_I and M_m. It is clear from Figure 3.10 that the proposed controller still maintains better control performance by achieving 23.3% reduction of MAE relative to RLS-based PID controller.

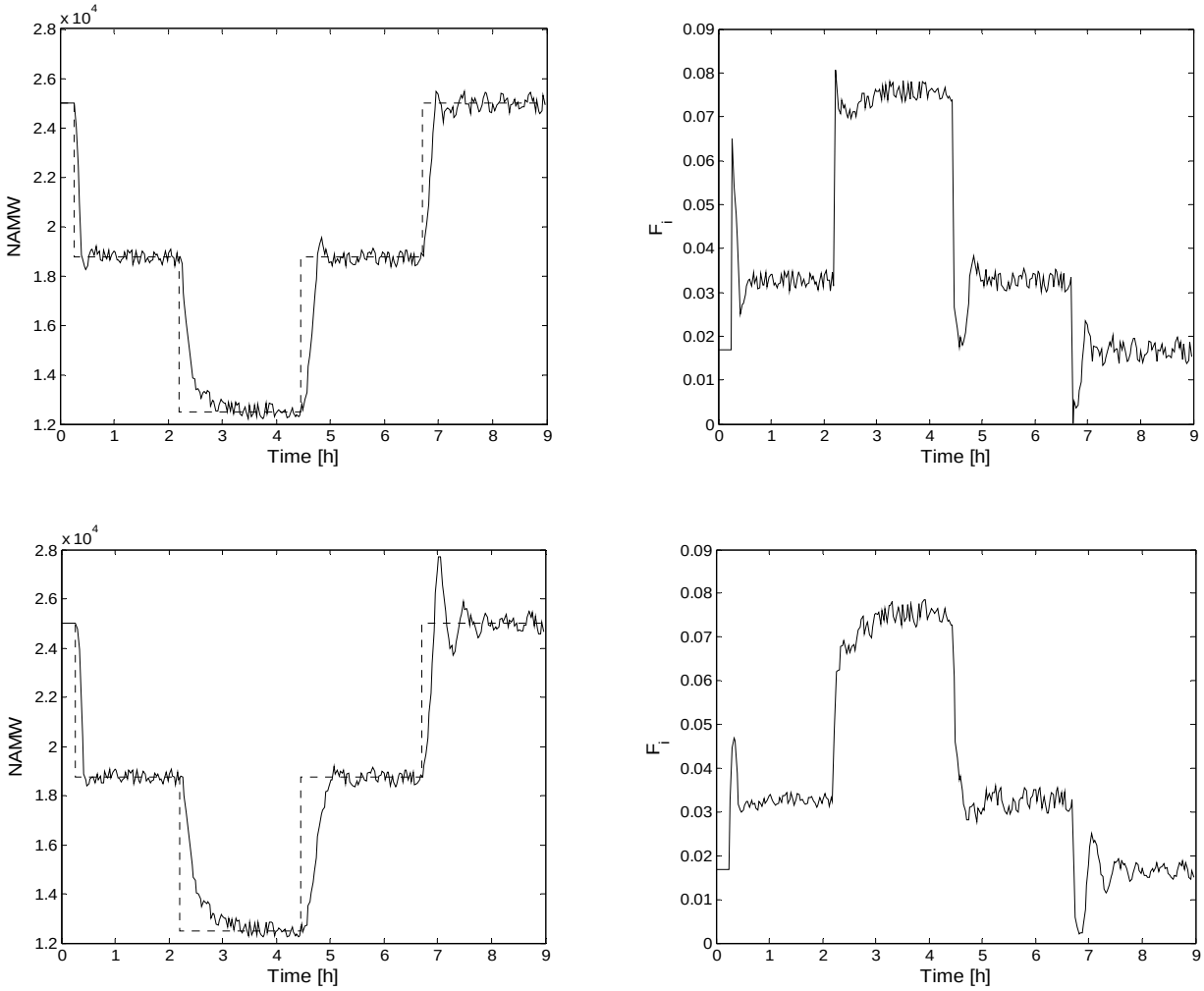

Figure 3.6 Servo responses of FNN-PID (top) and RLS-based PID (bottom)

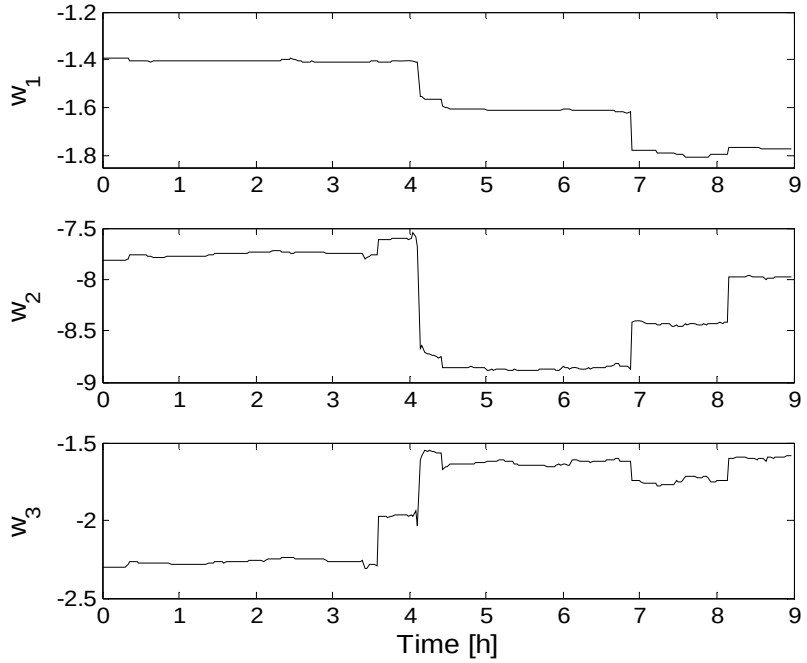

Figure 3.7 Updating of the FNN-PID parameters

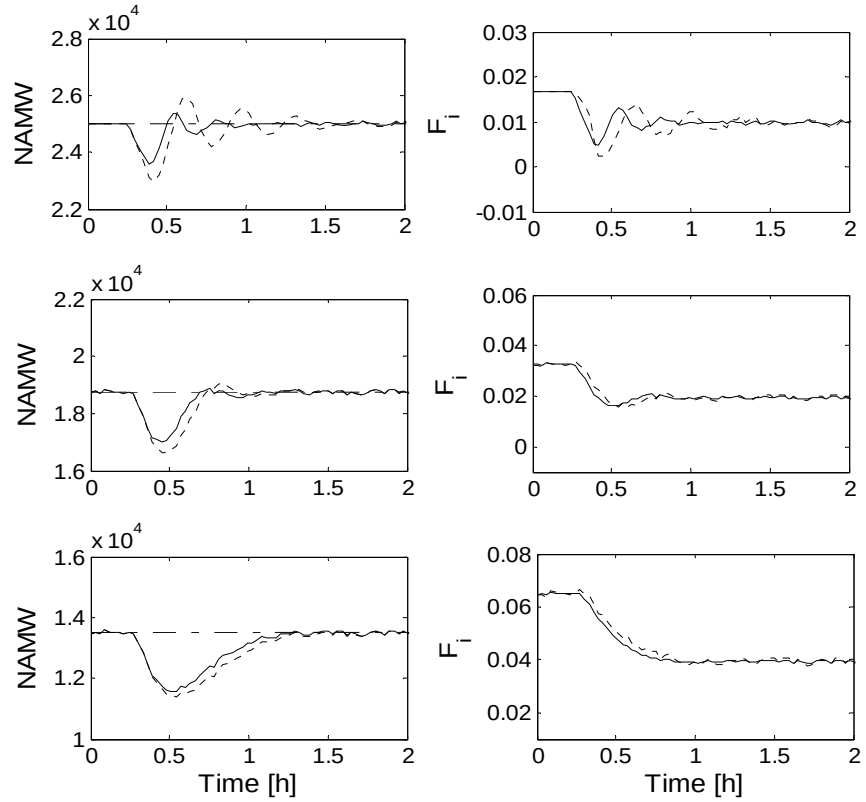

Figure 3.8 Closed-loop responses of two PID designs for -25% step change in $C_{m_{in}}$

Dashed: set-point; solid: FNN-PID; dotted: RLS-based PID

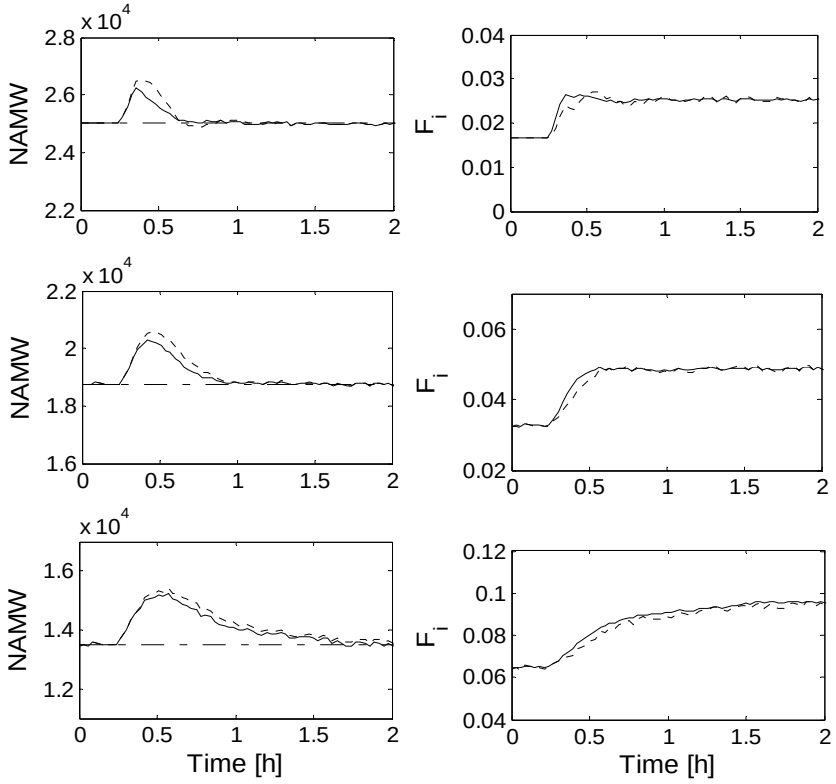

Figure 3.9 Closed-loop responses of two PID designs for +25% step change in $C_{m_{in}}$

Dashed: set-point; solid: FNN-PID; dotted: RLS-based PID

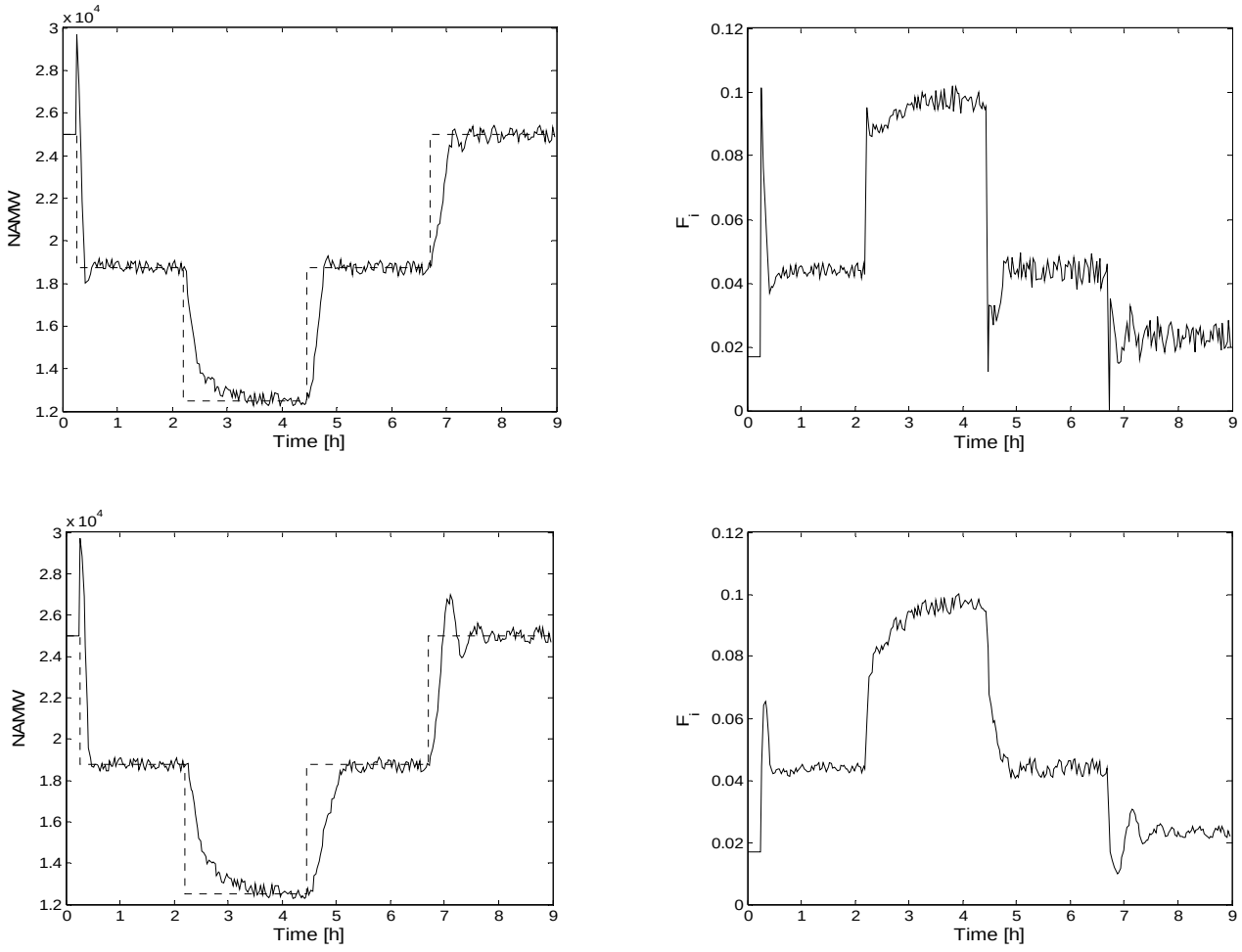

Figure 3.10 Servo responses of FNN-PID (top) and RLS-based PID (bottom)

in the presence of modeling error

Example 2 Consider a distillation process where the output variable is the top column composition, *y*, and the input variable is the reflux flow rate, *u*. This process can be defined by the following equations (Eskinat et al., 1991):

$$y(k) = 0.757y(k-1) + 0.243g(u(k-1)) \tag{3.37}$$

$$g(k) = 1.04x - 14.11x^2 - 16.72x^3 + 562.7x^4 \tag{3.38}$$

where the input and output variables are both defined as derivations from their respective nominal values.

To apply FNNM for process modeling, input and output data are generated by introducing uniformly random steps with distribution of $[-0.052\ 0.052]$ in process input. The process input and output are scaled by $\tilde{u} = \frac{2(u-u_{\min})}{u_{\max}-u_{\min}} - 1$ and $\tilde{y} = \frac{2(y-y_{\min})}{y_{\max}-y_{\min}} - 1$, respectively, where $u_{\min}$ and $u_{\max}$ are the minimum and maximum values of process input in the database, while $y_{\min}$ and $y_{\max}$ are the minimum and maximum values of process output in the database. Both process input and output are corrupted by 5% Gaussian white noise as depicted in Figure 3.11.

Again, validation tests are carried out to determine the optimal parameters for FNNM algorithm as follows: ψ = 0.997, λ_0 = 0.7, and ρ = 2. The validation result using these optimal parameters is shown in Figure 3.12. To design FNN-PI controller, initial PI parameters $w_1 = 0.64$ and $w_2 = 1.43$ is designed and their corresponding learning rates are specified as $\eta_1 = 1\times10^{-5}$ and $\eta_2 = 8\times10^{-5}$.

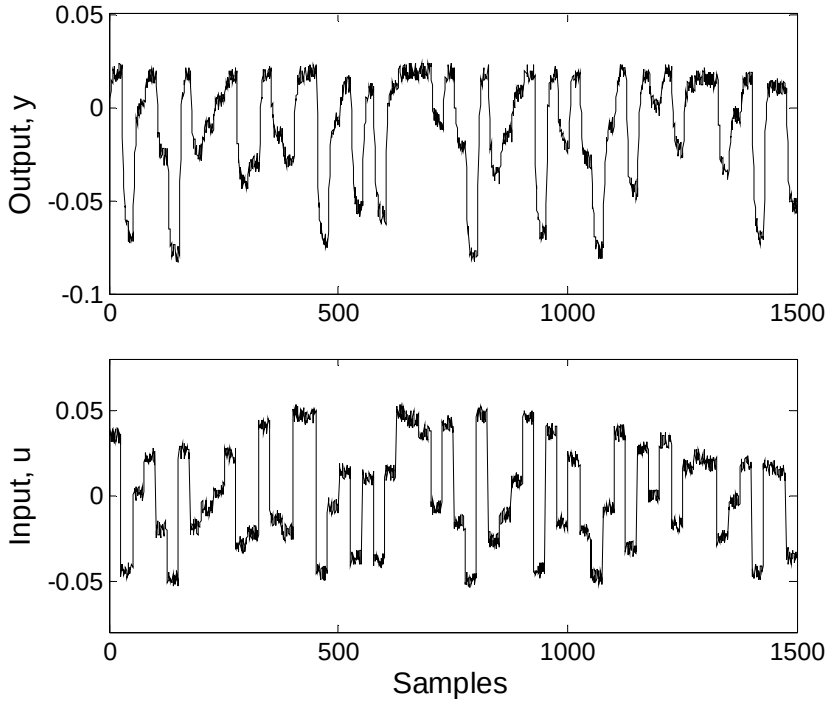

Figure 3.11 Input and output data used to construct the FNN model in distillation column example

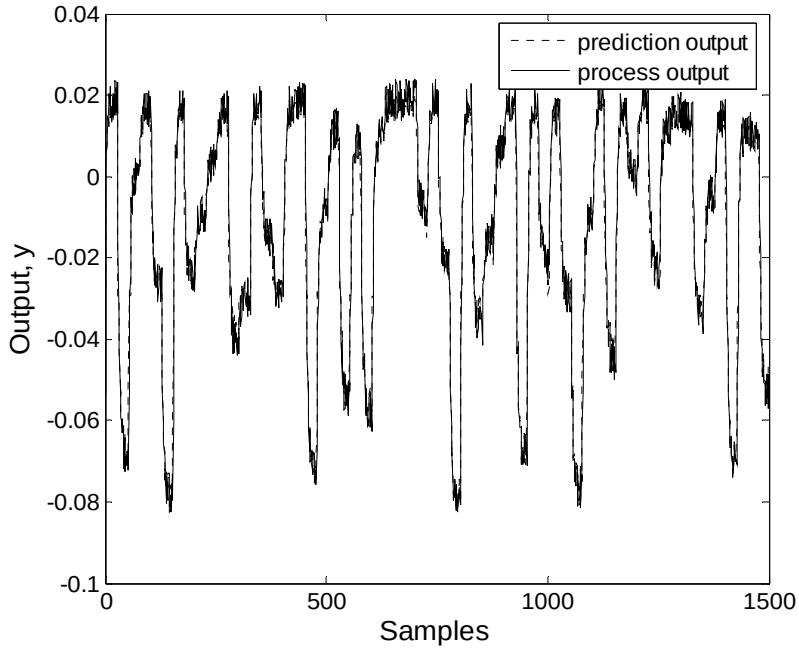

Figure 3.12 Validation of FNN model

To evaluate the servo performance of the proposed FNN-PI controller, successive set-point changes between 0.018 and -0.01 are conducted. For the purpose of comparison, an adaptive PI controller based on a first-order ARX model with parameter adaptation by the RLS identification procedure (Shahrokhi and Baghmisheh, 2005) is designed. As can be seen from Figure 3.13, the proposed FNN-PI has better control performance than that achieved by RLS-based PI controller, resulting in the reduction of MAE by 14.3%. Figure 3.14 shows the updating of the FNN-PI controller parameters in the abovementioned servo response.

To compare the disturbance rejection capability of these two controllers, unmeasured +30% step disturbances in the top column composition, y, are considered. The resulting closed-loop responses at three different operating points are compared in Figure 3.15. Again, the FNN-PI controller gives smaller deviation from the respective set-point compared to RLS-based PI controller, as evidenced by the reduction of MAE summarized in Table 3.3.

Table 3.3 Control performance comparison of two PI designs

	Tracking Error (MAE)		% Decrease in MAE
	FNN-PI	RLS-based PI	
Servo Response	5.080×10^{-4}	5.930×10^{-4}	14.35
Load Response			
at y = -0.01	1.965×10^{-4}	2.673×10^{-4}	26.48
at y = 0	2.144×10^{-4}	3.001×10^{-4}	28.57
at y = 0.01	2.544×10^{-4}	3.680×10^{-4}	30.88

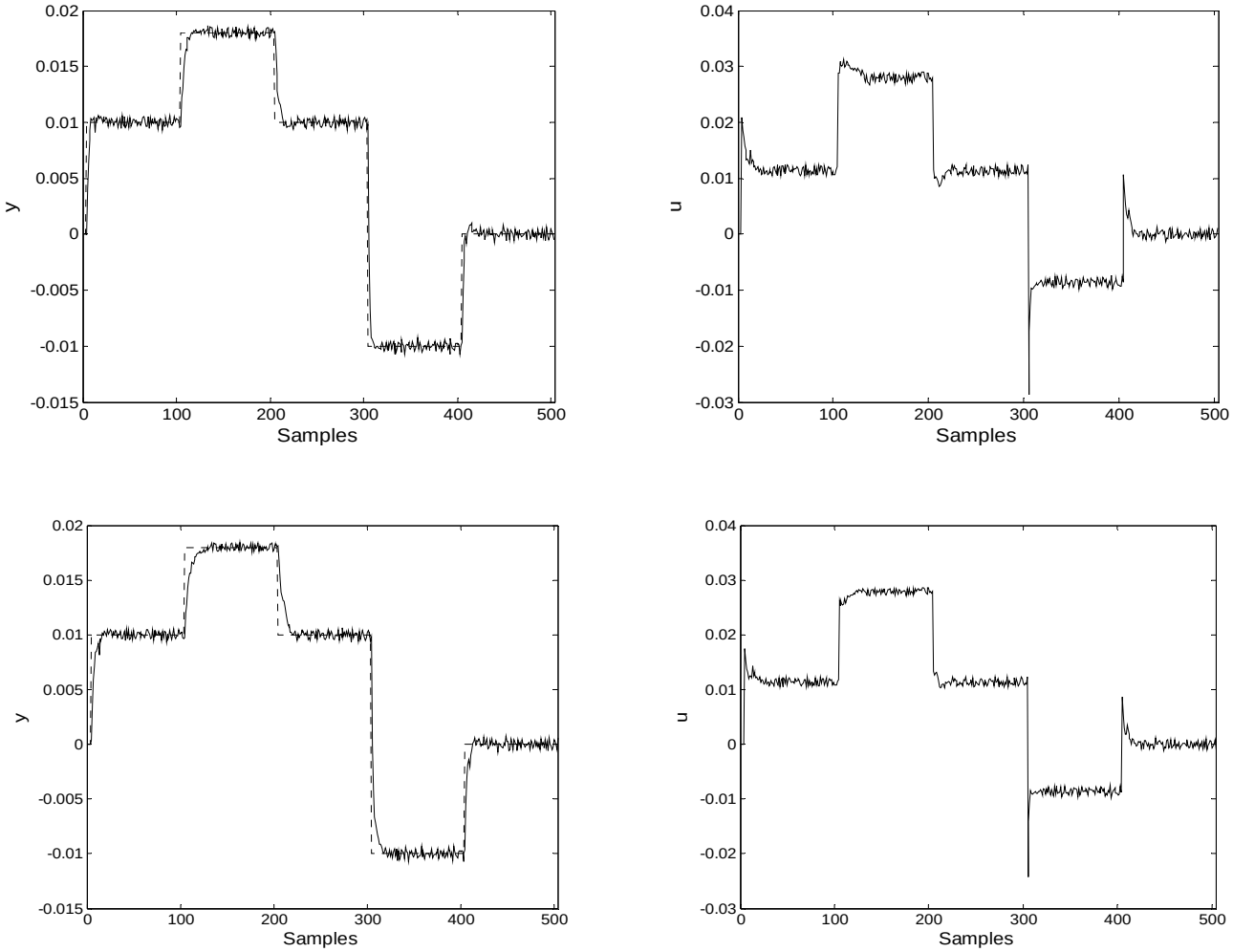

Figure 3.13 Servo responses of FNN-PI (top) and RLS-based PI (bottom)

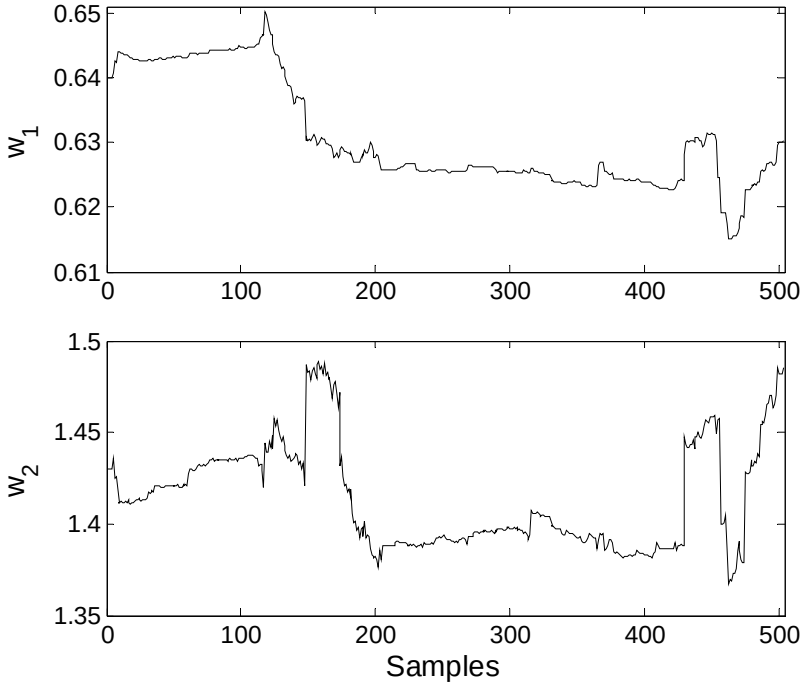

Figure 3.14 Updating of the FNN-PI parameters

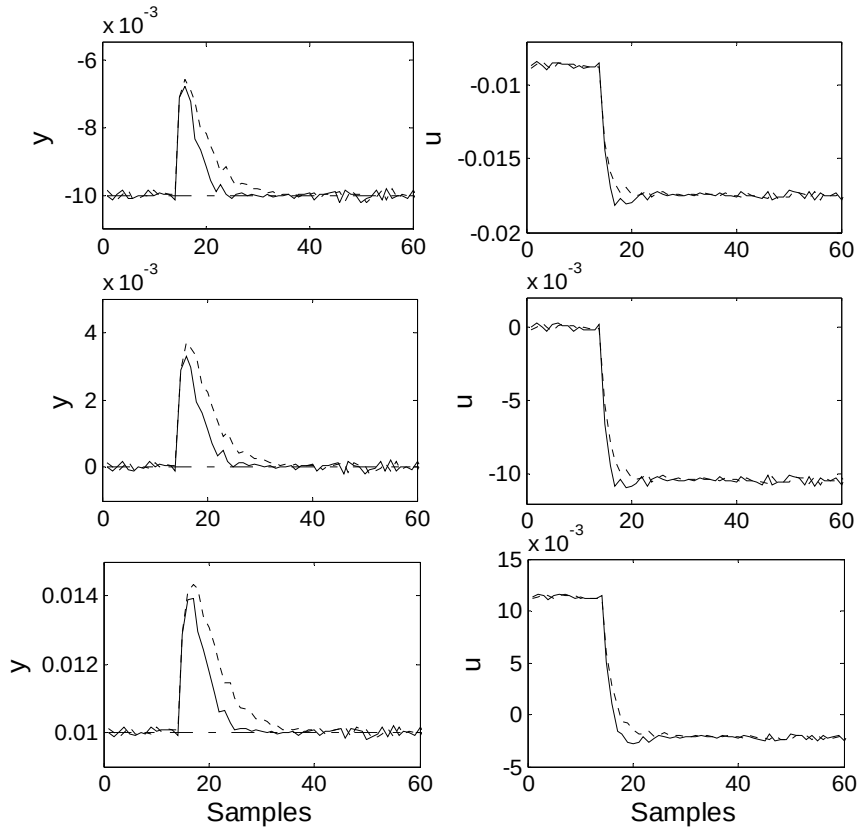

Figure 3.15 Closed-loop responses of two PI designs under +30% step disturbance

Dashed: set-point; solid: FNN-PI; dotted: RLS-based PI

3.5 Conclusion

An adaptive FNN-PID controller is developed for nonlinear process control in this chapter. A fuzzy neural network-based model is employed to approximate the controlled nonlinear process. By utilizing Lyapunov method, an updating algorithm is derived to adjust the PID parameters to guarantee the convergence of the predicted tracking error. Simulation results illustrate the performance and applicability of the proposed adaptive PID design.

Chapter 4

Self-Tuning PID Controller Design for Nonlinear Systems

4.1 Introduction

The proportional-integral-derivative (PID) controller has gained widespread use in many process control applications due to its simplicity in structure, robustness in operation, and easy comprehension in its principle (Åström and Hägglund, 1995). Numerous tuning methods have already been proposed to design PID controller, like Cohen-Coon, Zieglar-Nichols, model-based and relay feedback test (Tan et al., 2002; Huang et al., 2005), and dominant pole design (Åström and Hägglund, 1995). However, most of the tuning rules for PID controllers are based on a linear process model obtained experimentally around the nominal operating condition. Therefore, the performance of the conventional PID controller might degrade or even become unstable for nonlinear processes with a range of operating conditions.

To improve the control performance, several schemes of incorporating nonlinear control techniques in the design of PID controller have been developed in the literature. For example, Krishnapura and Jutan (2000) utilized neural network framework to mimic nonlinear PID design. Riverol and Napolitano (2000) proposed the use of neural network to update the PID controller parameters on-line. Chang et al. (2002) developed a stable adaptation mechanism such that the PID parameters are adjusted to track certain feedback linearization control previously designed. Andrasik et al. (2004) made use of two neural networks for on-line tuning of PID controller. In their method, a hybrid model consisting of a neural network and a simplified first-principle model is constructed as an estimator, while the second neural network is a neural PID-like controller, which is pre-trained off-line as a black-box model inverse of the controlled process. Bisowarno et al. (2004) developed a nonlinear PI controller to accommodate the directionality of the process gain for a reactive distillation column. Hirata et al. (2004) designed a nonlinear PID controller whose parameters are calculated based on the local models identified based on least squares method. Likewise, the recursive least-square method was employed to develop local models for an adaptive IMC-PID design to control a fixed-bed reactor (Shahrokhi and Baghmisheh, 2005). Using the genetic algorithm, the PID controller is optimized for nonlinear processes, such as activated sludge aeration process (Zhang et al., 2006) and jacketed batch polystyrene reactor (Altinten et al., 2007). Wang et al. (2007) proposed an adaptive PID controller based on reinforcement learning for complex and time-varying systems. The tuning of PID parameters is conducted using Actor-Critic learning based on RBF network. Pan et al. (2007) developed a two-layer supervised control method for tuning PID controller parameters. A conventional PID controller is adopted in the lower layer while the upper layer is composed of a tuning and an

identification module. System parameters are estimated based on the lazy learning algorithm to obtain better accuracy of system identification for nonlinear systems. However, the aforementioned previous results require trial and error procedure for initialization of PID parameters, which is computationally intensive and thus hampers the use of these results in practical applications.

To alleviate the aforementioned drawbacks, a memory-based IMC-PID controller design was proposed by Takao et al. (2006). In this design method, initial PID parameters are designed based on the local model obtained around the nominal operating condition, which can be carried out straightforwardly. In on-line application, PID parameters are initially calculated using both modeling and controller databases, where the latter consists of controller parameter previously implemented and the relevant information vector. Whenever required, an updating algorithm is used to tune the controller parameter in proportional to control errors. However, the PID controller considered in Takao et al. (2006) was formulated by assuming a first-order plus time delay model, which is too restrictive to be applied in practical applications.

To overcome the aforementioned limitation, a self-tuning PID design utilizing just-in-time learning (JITL) is proposed in this thesis. There are two databases employed in the proposed method. The first database is a controller database which contains the PID parameters and the corresponding information vectors. The initial controller database can be constructed from closed-loop data collected from successive set-point changes around nominal operating condition. Alternatively, the available historical closed-loop data can be used for the same purpose. Because the initial controller database can be easily obtained, the proposed method requires less trial and error effort compared to the previous methods. The second database is modeling database which is employed by the JITL technique for modeling purpose.

During the on-line implementation, the controller database is used to extract the relevant information based on the current process dynamics characterized by the information vector and the nearest-neighborhood criterion. Such information is subsequently utilized to calculate the PID parameters. Moreover, the PID parameters thus obtained can be further updated on-line when the predicted control error is greater than a pre-specified threshold and the resulting updated PID parameters together with their corresponding information vector are stored into the controller database. Literature examples are presented to illustrate the proposed control strategy and a comparison with its counterpart is made.

4.2 Self-Tuning PID Design for Nonlinear Systems

As discussed above, the proposed self-tuning PID (STPID) design as depicted in Figure 4.1 requires not only the database used by the JITL for modeling purpose but also the controller database to be exploited by the on-line tuning algorithm to extract the relevant information in order to compute PID parameters at every sampling instant.

The PID algorithm under consideration are given by:

$$u(k) = u(k-1) + \Delta u(k) \tag{4.1}$$

$$\Delta u(k) = w_1(k)e(k) + w_2(k)\Delta e(k) + w_3(k)\delta e(k) \tag{4.2}$$

where the notations used were previously defined in Eqs. (3.18) and (3.19).

The algorithm of the STPID control scheme based on JITL technique is discussed in the following subsections.

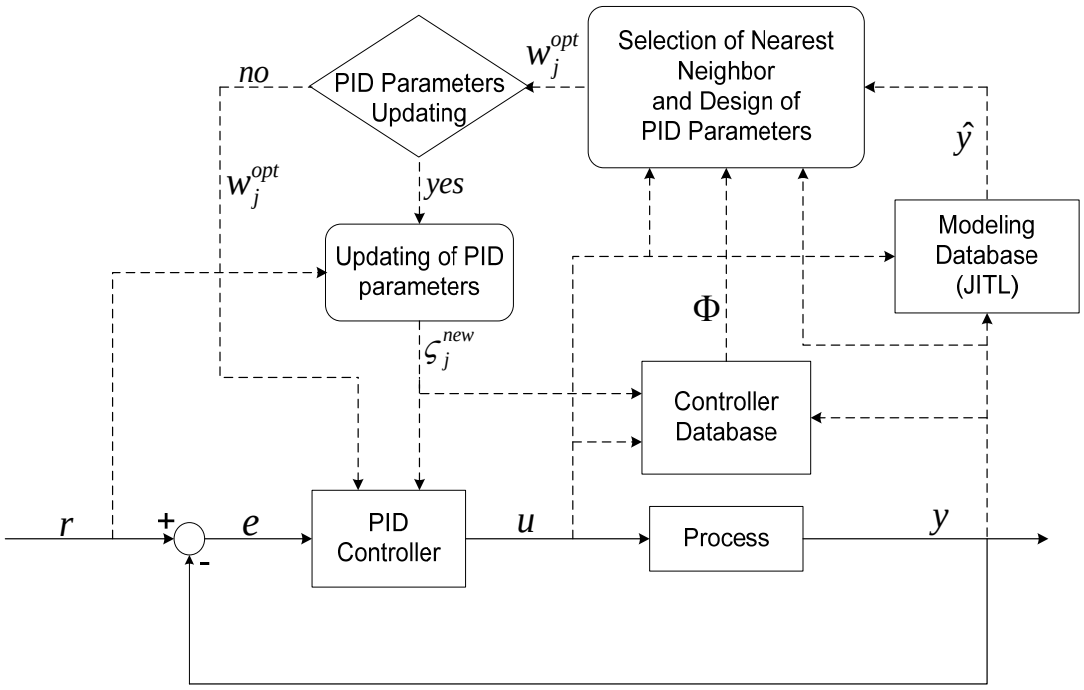

Figure 4.1 Self-tuning PID control scheme

4.2.1 Generation of initial controller database

The initial controller database can be easily constructed from the closed-loop data around the nominal operating condition, for example closed-loop data resulting from successive set-point changes around the nominal operating condition. Alternatively, the available historical closed-loop data can be used for the same purpose. It is assumed that PID parameters (w^0) chosen achieve satisfactory control performance. With the availability of measured process input and output data, the initial controller database is then generated as follows

$$\Phi(i) = (w(i), \mathbf{x}_{cl}(i)), \qquad i = 1, 2, \ldots, N_0 \tag{4.3}$$

where $\mathbf{x}_{cl}(i) = [y(i-1), u(i-1)]$ is information vector obtained from the available closed-loop data, $w(i) = [w_1(i)\ w_2(i)\ w_3(i)]$, and N_0 denotes the number of information vectors stored in the initial controller database. Because a fixed-

parameter PID controller is employed in the abovementioned closed-loop test, $w(1) = w(2) = \cdots = w(N_0) = w^0$ is specified in the initial database.

4.2.2 Calculation of initial PID parameters

At the *k*-th sampling instant during on-line application, the following measure is calculated between the query data $\mathbf{x}_{\mathrm{cl}}(k)$ and information vector $\mathbf{x}_{\mathrm{cl}}(i)$ in the controller database:

$$d_i = \sqrt{e^{-\left\|\mathbf{x}_{\mathrm{cl}}(k)-\mathbf{x}_{\mathrm{cl}}(i)\right\|^2}}, \qquad i = 1, 2, \ldots, N_k \tag{4.4}$$

where N_k denotes the number of information vectors stored in the current controller database. To extract PID parameters from controller database, h_c relevant information vectors or nearest-neighbors in the controller database that resemble $\mathbf{x}_{\mathrm{cl}}(k)$ are selected to be those corresponding to the largest d_i to the h_c-th largest d_i. As the number of nearest-neighbors may vary with respect to the operating condition, in the proposed STPID design, a selection procedure is developed to determine the optimal h_c in a pre-specified range as discussed in what follows.

With h_c nearest-neighbors chosen, a weight is assigned for each neighbor by using the following equation:

$$\gamma_i = \frac{d_i}{\sum_{i=1}^{h_c} d_i}, \qquad \sum_{i=1}^{h_c} \gamma_i = 1 \tag{4.5}$$

Next, the corresponding PID parameters are obtained from the controller database by using the following formula:

$$w^0(k) = \sum_{i=1}^{h_c} \gamma_i w(i) \tag{4.6}$$

and the resulting controller output is calculated by Eq. (4.1) as:

$$\hat{u}(k) = u(k-1) + w_1^{\,0}(k)e(k) + w_2^{\,0}(k)\Delta e(k) + w_3^{\,0}(k)\delta e(k) \tag{4.7}$$

Then, the process output at the $(k+1)$-th sampling instant can be predicted by employing the JITL technique as follows:

$$\hat{y}(k+1) = \alpha_1^{k+1} y(k) + \alpha_2^{k+1} y(k-1) + \beta^{k+1}\hat{u}(k) \tag{4.8}$$

The optimal nearest-neighbors at the *k*-th sampling instant is then determined by that giving the smallest deviation from the set-point at the $(k+1)$-th sampling instant. After the optimal h_c is determined, the fitness of its corresponding PID parameters, $w^{opt}(k)$, is then further evaluated in the next step.

4.2.3 Refinement of controller database and PID parameters

Because the initial controller database is constructed by using the process data around the nominal operating condition, it may not provide adequate information to adjust PID parameters effectively when the operating condition is away from the nominal one. In this situation, the PID parameters $w^{opt}(k)$ need further refinement and this resulting PID parameters together with the current information vector are added into the controller database to improve the controller database for the operating conditions where the information is not available in the construction of the initial controller database. To determine whether $w^{opt}(k)$ is satisfactory or not, the following criterion is introduced:

$$\left|\frac{r(k+1)-\hat{y}^{opt}(k+1)}{r(k+1)}\right| < \varepsilon \tag{4.9}$$

where $\hat{y}^{opt}(k+1)$ is the JITL's predicted output by using $w^{opt}(k)$, and ε is the threshold. The PID controller designed with $w^{opt}(k)$ is considered to give good control performance if the above inequality is satisfied and thus there is no need for further refinement. On the other hand, when the above inequality does not hold, $w^{opt}(k)$ can be improved further by the steepest descent method discussed in what follows.

The following quadratic function is used as the objective function for the updating law of PID parameters:

$$Min\ J = (r(k+1)-\hat{y}^{opt}(k+1))^2 + \kappa(\hat{u}^{opt}(k)-u(k-1))^2 \tag{4.10}$$

where κ is a weight parameter and $\hat{u}^{opt}(k)$ is the control action calculated by Eq. (4.7) using $w^{opt}(k)$.

As mentioned in Chapter 3, the PID parameters are constrained to be positive or negative. Therefore, to consider this constraint in the updating of PID parameters, the mapping function in Eq. (3.20) is applied.

$$w_j(k) = \begin{cases} e^{\varsigma_j(k)}, & \text{if} \quad w_j(k) \geq 0 \\ -e^{\varsigma_j(k)}, & \text{if} \quad w_j(k) < 0 \end{cases}, \quad j = 1 \sim 3 \tag{3.18}$$

where $\varsigma_j(k)$ is a real number. Likewise, $\varsigma_j^{opt}(k)$ denotes the mapping variable corresponding to $w^{opt}(k)$. The following updating algorithm can now be derived to improve the design of $\varsigma_j(k)$ for $j = 1 \sim 3$:

$$\begin{aligned}\varsigma_j^{new}(k) &= \varsigma_j^{opt}(k) - \eta_j \frac{\partial J}{\partial \varsigma_j^{opt}(k)} \\ &= \varsigma_j^{opt}(k) - \eta_i \frac{\partial J}{\partial w_j^{opt}(k)} w_j^{opt}(k)\end{aligned} \tag{4.11}$$

where η_j are the respective learning rates for ς_j^{opt} and

$$\frac{\partial J}{\partial w_j^{opt}(k)} = \frac{\partial J}{\partial \hat{u}^{opt}(k)} \frac{\partial \hat{u}^{opt}(k)}{\partial w_j^{opt}(k)} \tag{4.12}$$

$$\frac{\partial J}{\partial \hat{u}^{opt}(k)} = -2\beta^{k+1}(r(k+1) - \hat{y}^{opt}(k+1)) + 2\kappa(\hat{u}^{opt}(k) - u(k-1)) \tag{4.13}$$

$$\frac{\partial \hat{u}^{opt}(k)}{\partial w_1^{opt}(k)} = e(k) \tag{4.14}$$

$$\frac{\partial \hat{u}^{opt}(k)}{\partial w_2^{opt}(k)} = e(k) - e(k-1) \tag{4.15}$$

$$\frac{\partial \hat{u}^{opt}(k)}{\partial w_3^{opt}(k)} = e(k) - 2e(k-1) + e(k-2) \tag{4.16}$$

After $\varsigma_j^{new}(k)$ is calculated by Eq. (4.11), the corresponding new PID parameters are obtained from Eq. (3.20). Furthermore, these new PID parameters and their corresponding information vector are stored into controller database. The implementation of the proposed STPID control algorithm is summarized as follows:

1. Given the weight parameter κ, learning rate η_j, and range of nearest-neighbors for h_c;
2. At each sampling instant, determine the optimal nearest-neighbors and calculate PID parameters $w^{opt}(k)$ according to Eq. (4.6) and the corresponding controller output $\hat{u}^{opt}(k)$ by Eq. (4.7), by which $\hat{y}(k+1)$ is obtained from the JITL;

3. Evaluate the fitness of $w^{opt}(k)$ by the criterion given in Eq. (4.9). If this criterion is satisfied, $\hat{u}^{opt}(k)$ is implemented to the process and set $k = k+1$ and go to step 2. Otherwise, go to step 4;
4. Update $\varsigma_j^{opt}(k)$ by Eq. (4.11) and calculate the resulting new PID parameters using Eq. (3.20), by which controller output is obtained and implemented to the process. In addition, the controller database is updated by adding new PID parameters thus obtained and their corresponding information vector. Set $k = k+1$ and go to step 2.

4.3 Examples

Example 1 The proposed self-tuning PID design is applied to the polymerization reactor discussed in Chapter 3. The model parameters and steady-state operating condition can be found in Tables 3.1 and 3.2. To apply the JITL method for process modeling, input and output data are generated by introducing uniformly random steps with distribution of $[0.016\ \ 0.02]$ to the process input F_i. The process input and output are scaled by $\tilde{u} = \dfrac{u - 0.016783}{0.016783}$ and $\tilde{y} = \dfrac{y - 25000.5}{25000.5}$, respectively. Both process input and output are corrupted by 5% Gaussian white noise. With sampling time of 0.03h, input and output data thus obtained (see Figure 4.2) are used to build the database. A second-order ARX model is used as the local model and the parameters chosen for JITL algorithm are $\phi = 0.95$, $k_{\min} = 7$, and $k_{\max} = 50$.

To construct the initial controller database, the PID controller with $w_1 = -1.86$, $w_2 = -8.35$, and $w_3 = -3.40$ is designed to give good control performance around the nominal operating condition. This PID controller is then used in a closed-loop

experiment consisting of multiple set-point changes as illustrated in Figure 4.3, from which the initial controller database is constructed with $N_0 = 150$. In addition, the following parameters are chosen in the proposed design: the threshold $\varepsilon = 0.05$, the weight parameter $\kappa = 0.003$, nearest-neighbors $h_c \in [3\ 20]$, and the learning rates $\eta_1 = 0.96$, $\eta_2 = 0.1$, and $\eta_3 = 0.3$.

To evaluate the servo performance of the proposed STPID controller, successive set-point changes between 25000.5 and 12500 kg/kmol are conducted. For comparison purpose, the RLS-based adaptive PID controller provided in Chapter 3 serves as the benchmark design. As can be seen from Figure 4.4, the proposed STPID has consistent better control performance than that achieved by RLS-based PID controller, resulting in the reduction of MAE by 26.4%. Figures 4.5 and 4.6 show the updating of controller parameters and the profile of corresponding optimal nearest-neighbors in the STPID design respectively.

To compare the disturbance rejection capability of these two controllers, unmeasured ±25% step disturbances in the monomer initiator concentration are considered. The resulting closed-loop responses at three different operating points are shown in Figures 4.7 and 4.8. STPID controller gives smaller deviation from the respective set-point and shorter settling time compared to RLS-based PID controller, as evidenced by the reduction of MAE ranging from 29% to 51.9%. To evaluate the robustness of the proposed controller, it is assumed that there exist 10% modeling error in the kinetic parameter k_{I} and 20% error in the gain coefficients of the D_I and M_m. As can be seen in Figure 4.9, the proposed controller still maintains better control performance by achieving 21.5% reduction of MAE relative to RLS-based PID controller.

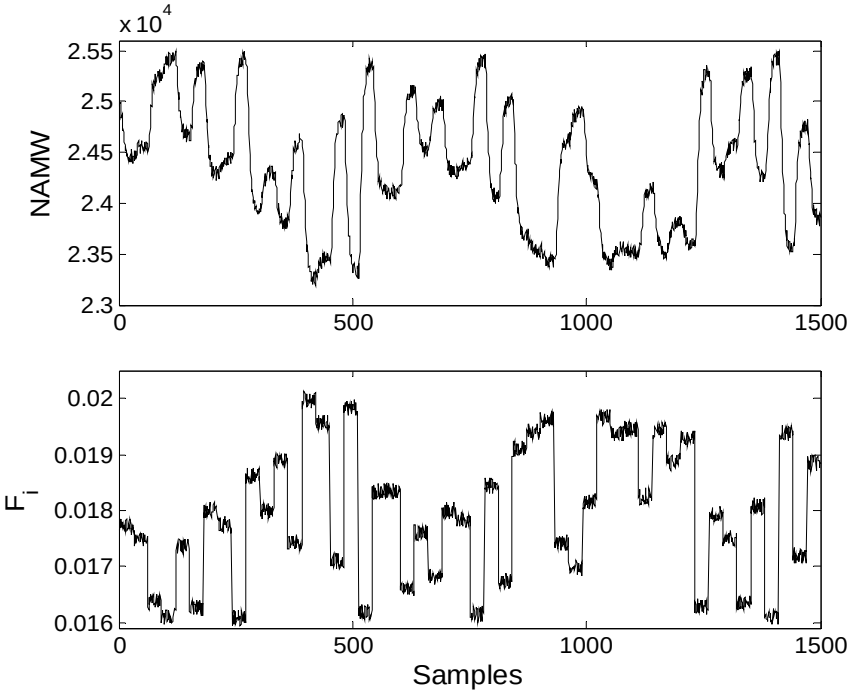

Figure 4.2 Input and output data used to construct the modeling database for JITL in polymerization reactor example

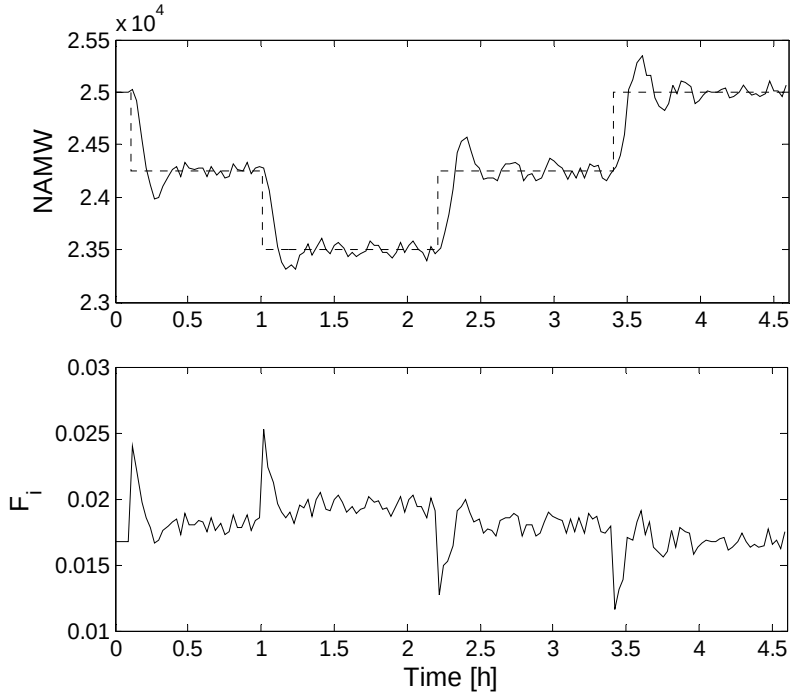

Figure 4.3 Input and output data used to construct the initial controller database in polymerization reactor example

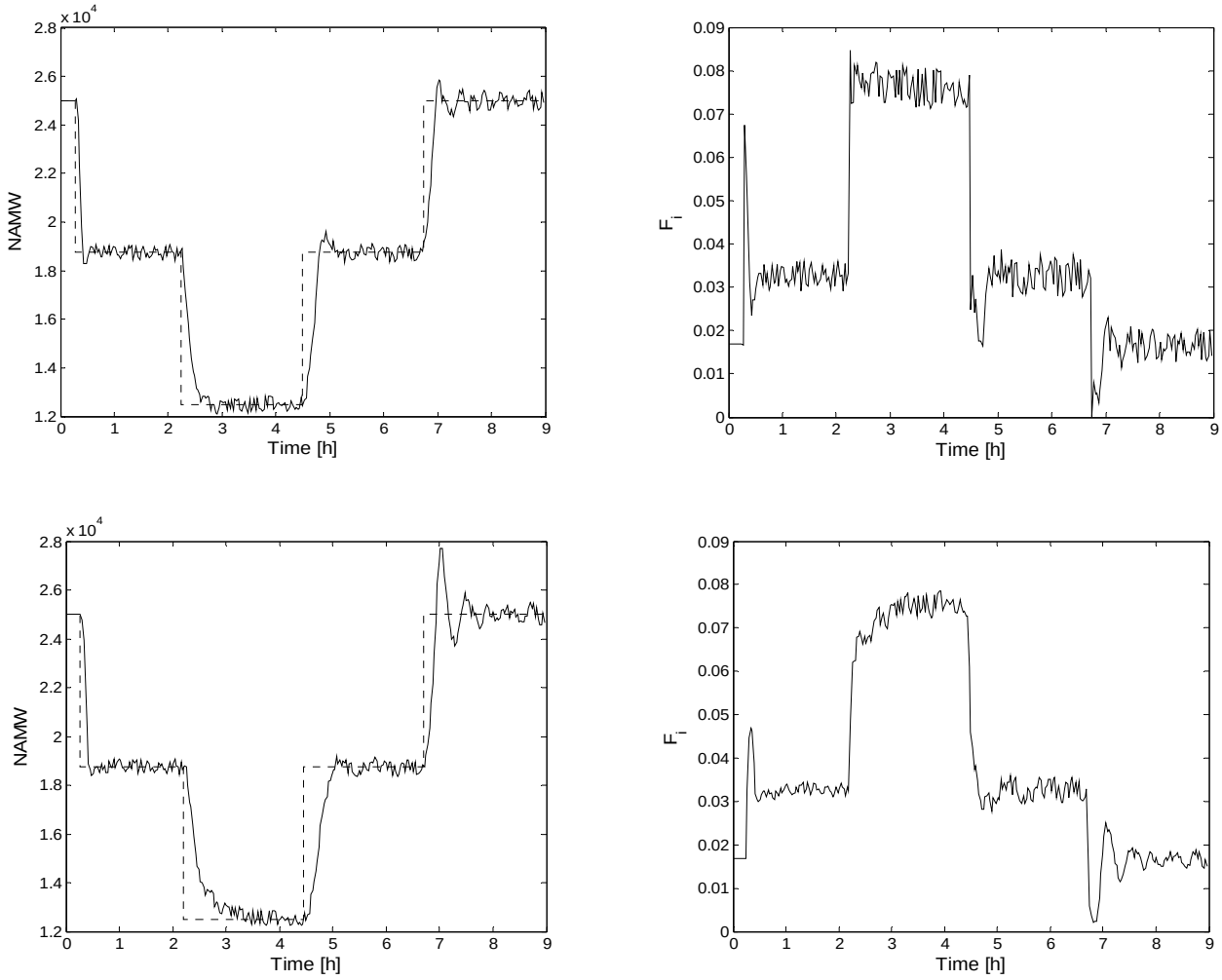

Figure 4.4 Servo responses of STPID (top) and RLS-based PID (bottom)

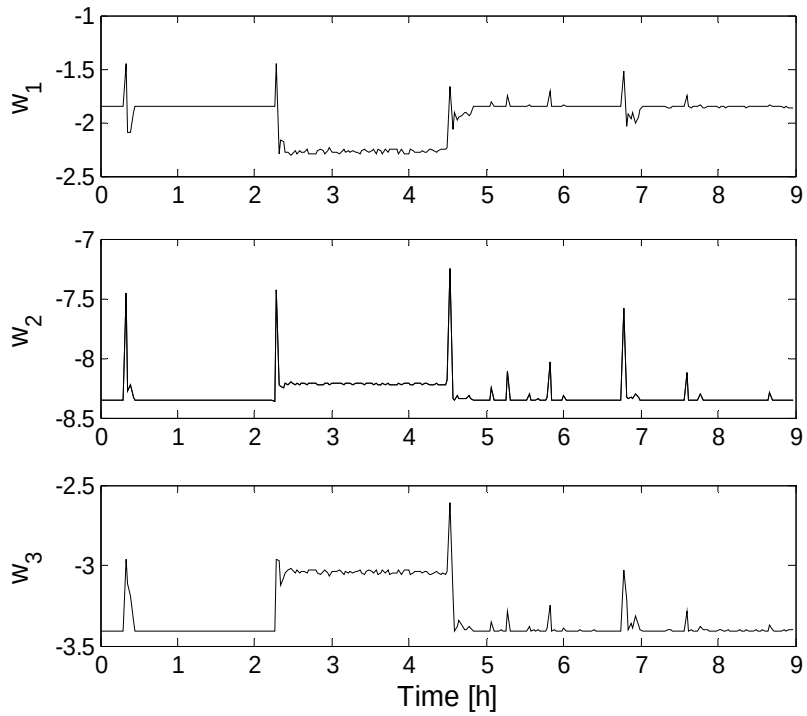

Figure 4.5 Updating of the STPID parameters

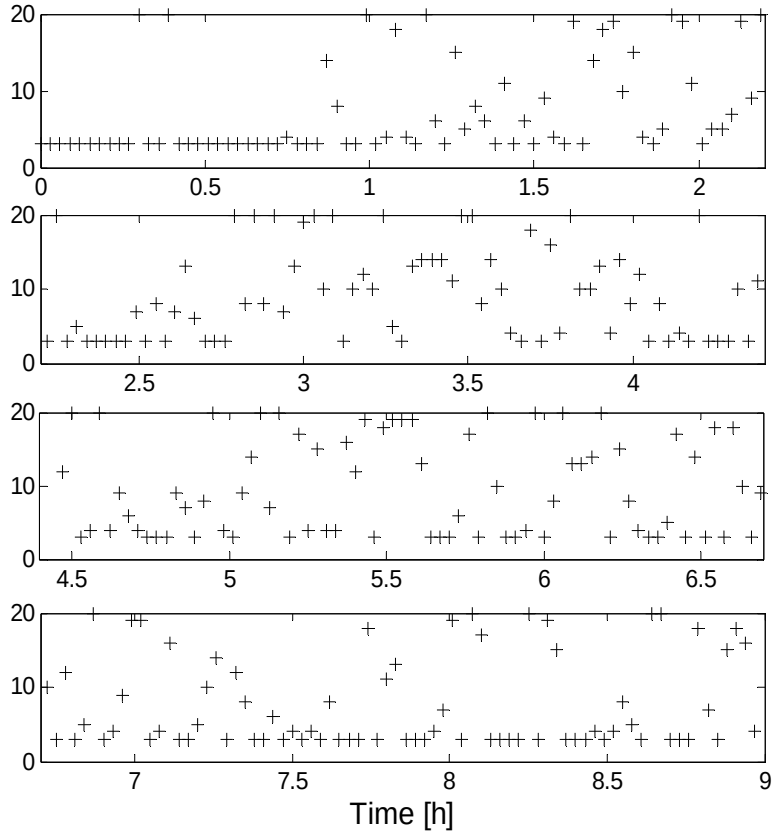

Figure 4.6 The profile of optimal nearest-neighbors in STPID design

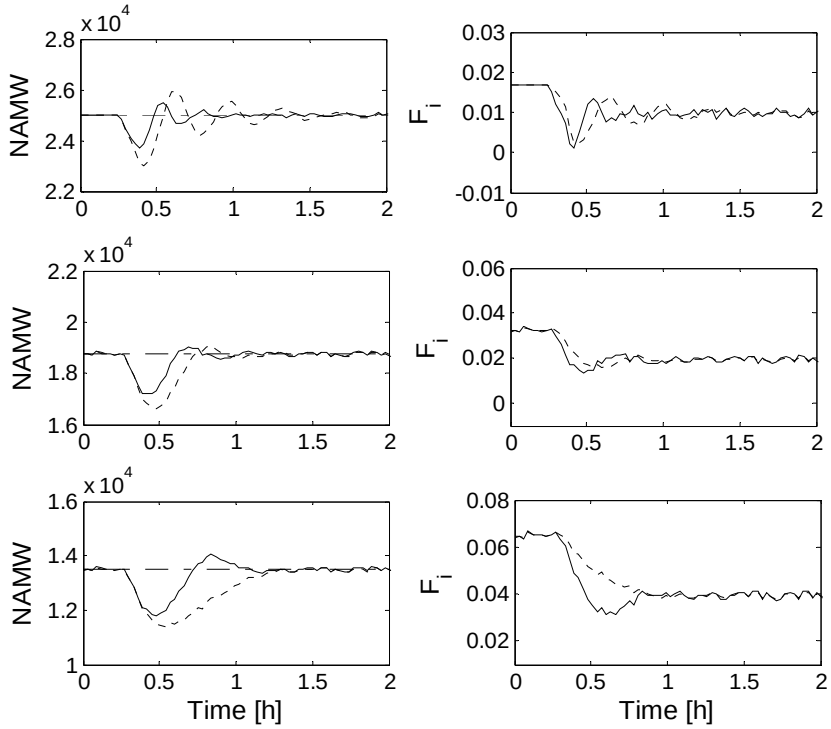

Figure 4.7 Closed-loop responses of two PID designs for -25% step change in $C_{m_{in}}$

Dashed: set-point; solid: STPID; dotted: RLS-based PID

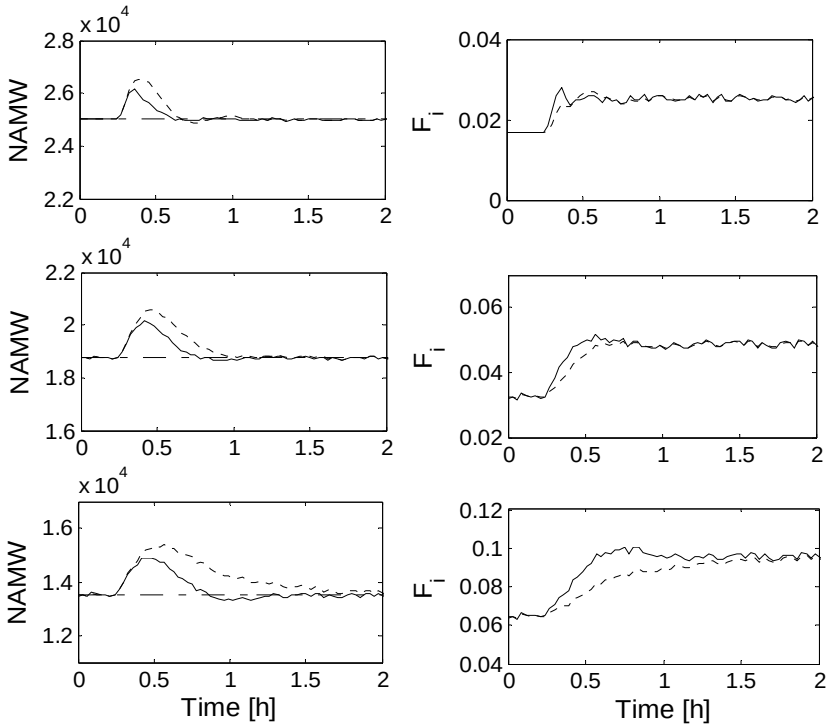

Figure 4.8 Closed-loop responses of two PID designs for +25% step change in $C_{m_{in}}$

Dashed: set-point; solid: STPID; dotted: RLS-based PID

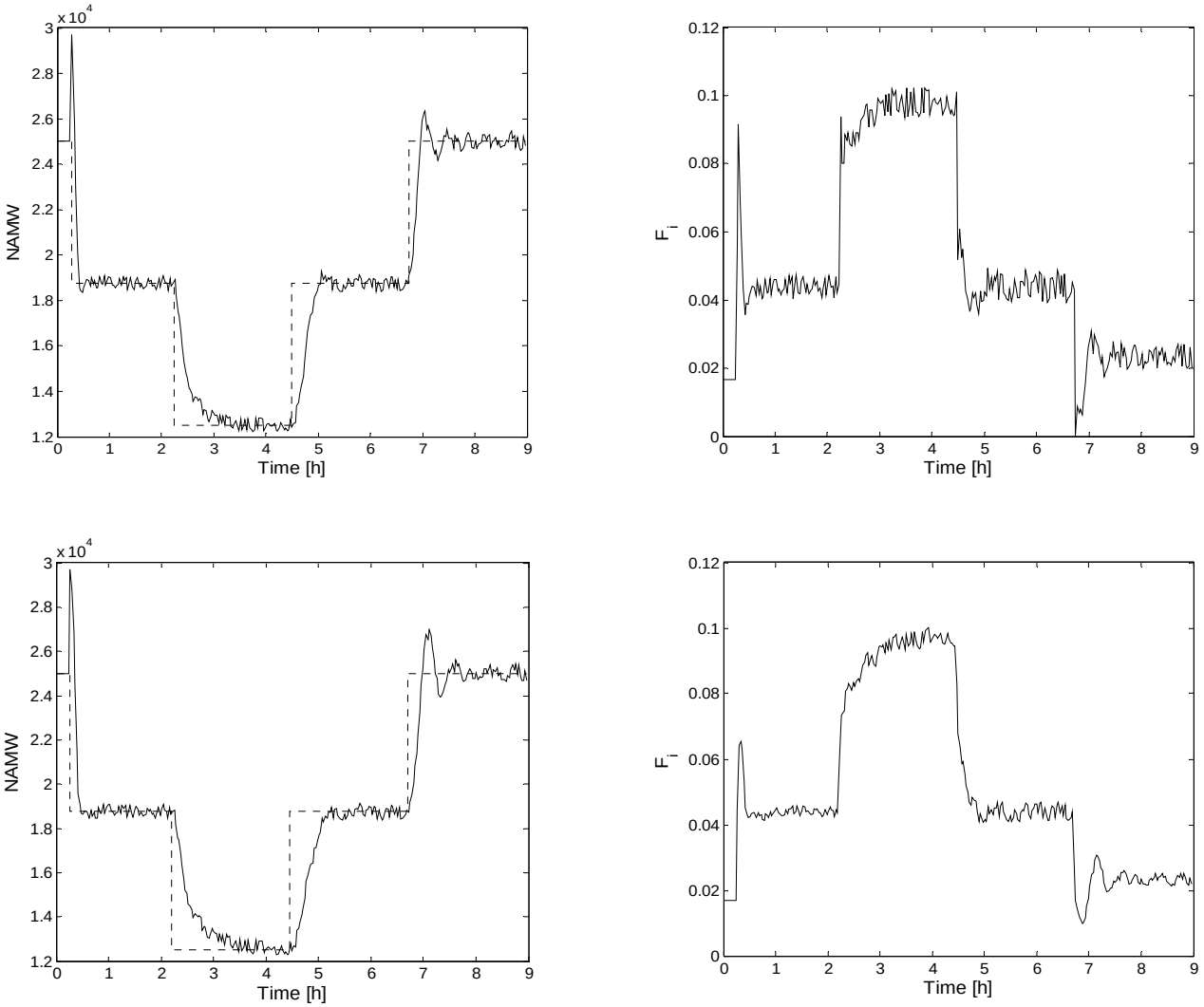

Figure 4.9 Servo responses of STPID (top) and RLS-based PID (bottom)

in the presence of modeling error

Example 2 The next example considered is the control of a distillation process discussed in Chapter 3. To apply the JITL method for process modeling, input and output data are generated by introducing uniformly random steps with distribution of $[-0.00145\ 0.00163]$ in process input. The process input and output are scaled by $\tilde{u} = \frac{2(u-u_{\min})}{u_{\max}-u_{\min}} - 1$ and $\tilde{y} = \frac{2(y-y_{\min})}{y_{\max}-y_{\min}} - 1$, respectively, where $u_{\min}$ and $u_{\max}$ are the minimum and maximum values of process input in the database, while $y_{\min}$ and $y_{\max}$ are the minimum and maximum values of process output in the database. Both process input and output are corrupted by 5% Gaussian white noise as depicted in Figure 4.10. A first-order ARX model is used as the local model and the parameters chosen for JITL algorithm are as follows: $\phi = 0.95$, $k_{\min} = 5$, and $k_{\max} = 50$.

A PI controller with $w_1 = 0.510$ and $w_2 = 1.853$ is designed to give good control performance around the nominal operating condition. This PI controller is then used in a closed-loop experiment consisting of multiple set-point changes as illustrated in Figure 4.11, from which the initial controller database is constructed with $N_0 = 150$. In addition, the following parameters are chosen in the proposed design: the threshold $\varepsilon = 0.05$, the weight parameter $\kappa = 0.1$, range of nearest-neighbors $h_c \in [3\ 20]$, and the learning rates $\eta_1 = 5\times10^{-5}$ and $\eta_2 = 9\times10^{-5}$. Again, the RLS-based adaptive PI controllers provided in Chapter 3 serve as the benchmark for the purpose of comparison.

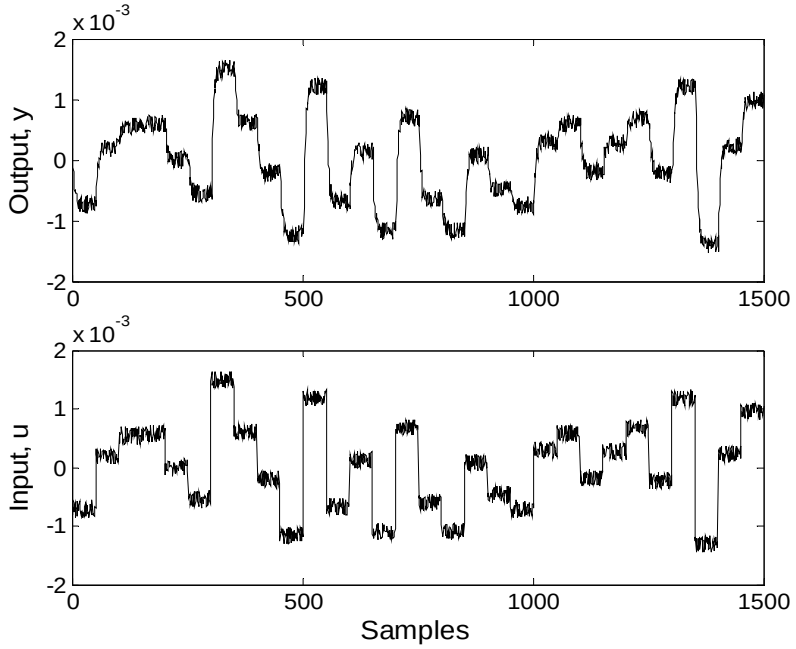

Figure 4.10 Input and output data used to construct the modeling database for JITL in distillation column example

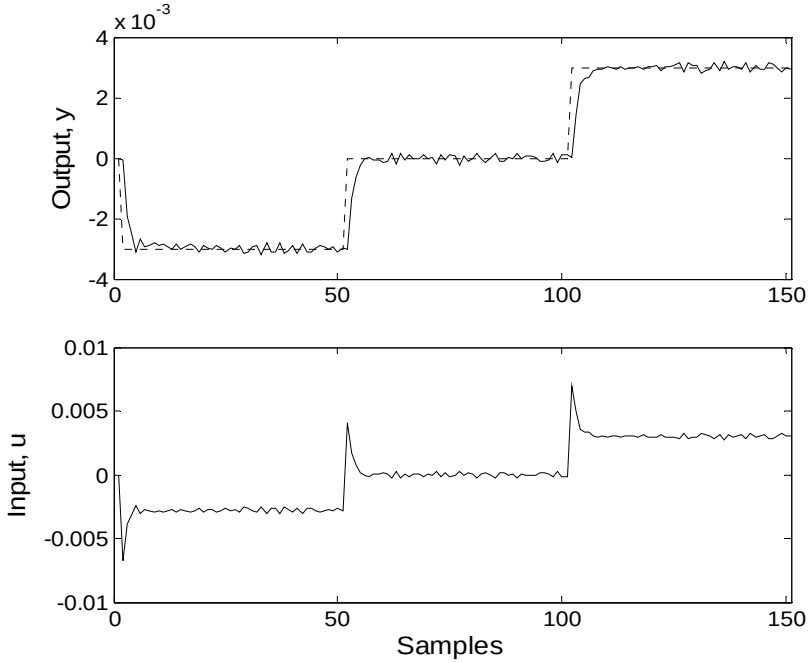

Figure 4.11 Input and output data used to construct the initial controller database in distillation column example

To compare the performances of two PI designs, successive set-point changes between 0.018 and -0.01 are conducted. As can be seen from Figure 4.12, it is obvious that the proposed STPI controller has better performance than those achieved by the RLS-based PI controller, resulting in the reduction of MAE by 14.2%. Figures 4.13 and 4.14 show the updating of the STPI controller parameters and the corresponding optimal nearest-neighbors in the aforementioned closed-loop responses. By assuming unmeasured +30% step disturbance in the top column composition *y*, the resulting performances of two controllers at different operating conditions are compared in Figure 4.15. Again, the STPI controller achieves better control performance, as evidenced by the reduction of MAE summarized in Table 4.1.

Table 4.1 Control performance comparison of two PI designs

	Tracking Error (MAE)		% Decrease in MAE
	STPI	RLS-based PI	
Servo Response	5.088×10^{-4}	5.930×10^{-4}	14.20
Load Response			
at y = -0.01	1.723×10^{-4}	2.673×10^{-4}	35.56
at y = 0	2.550×10^{-4}	3.001×10^{-4}	15.04
at y = 0.01	2.182×10^{-4}	3.680×10^{-4}	40.70

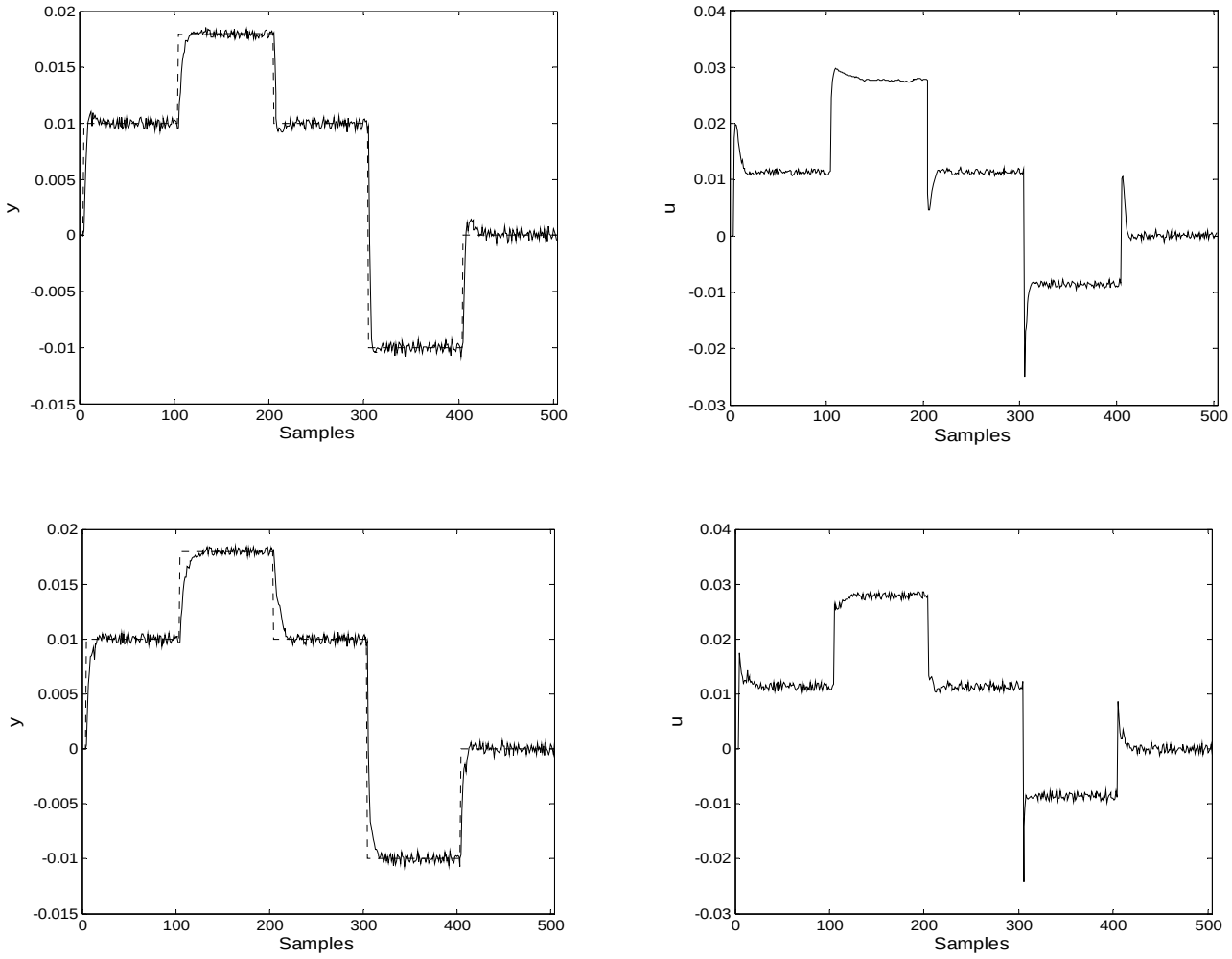

Figure 4.12 Servo responses of STPI (top) and RLS-based PI (bottom)

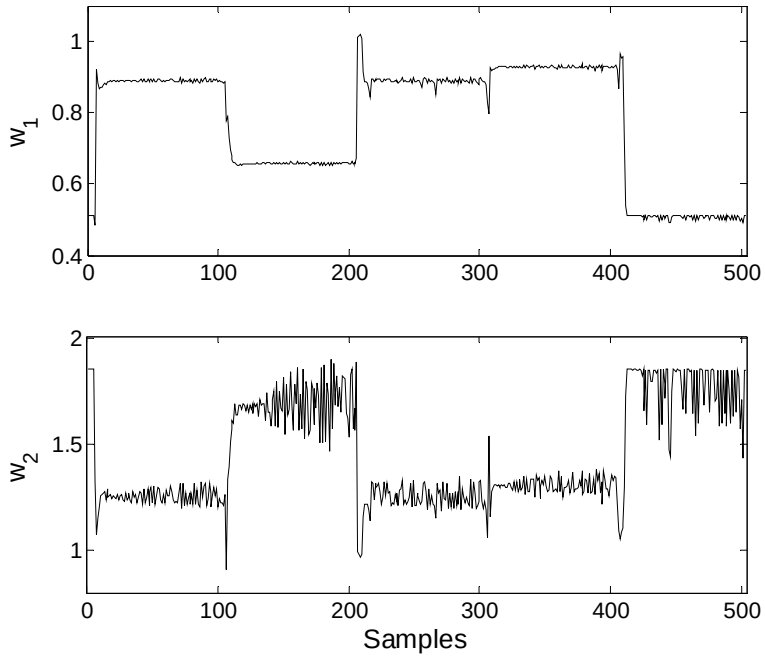

Figure 4.13 Updating of the STPI parameters

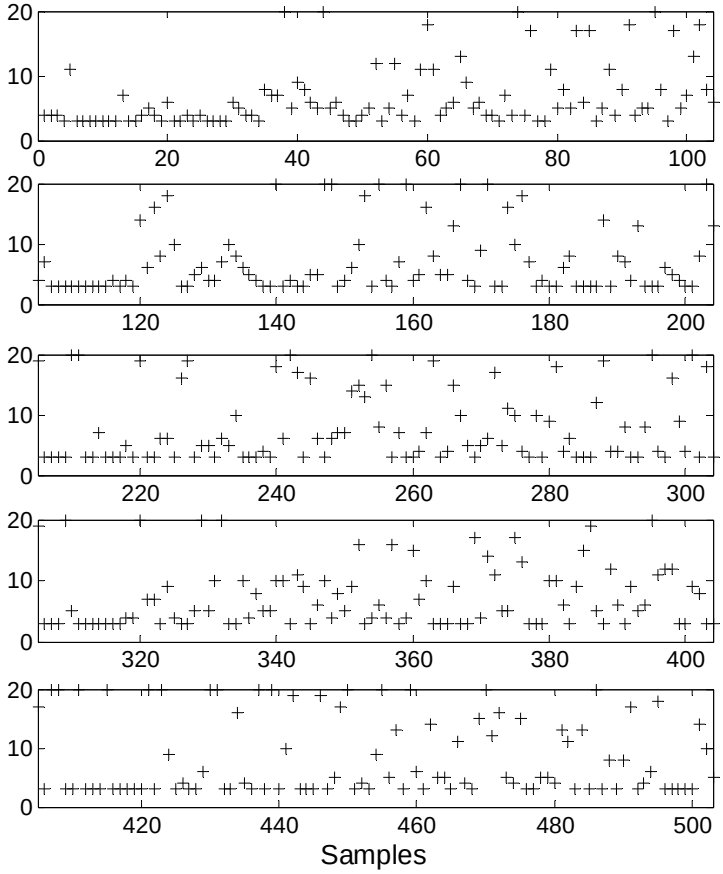

Figure 4.14 The profile of optimal nearest-neighbors in STPI design

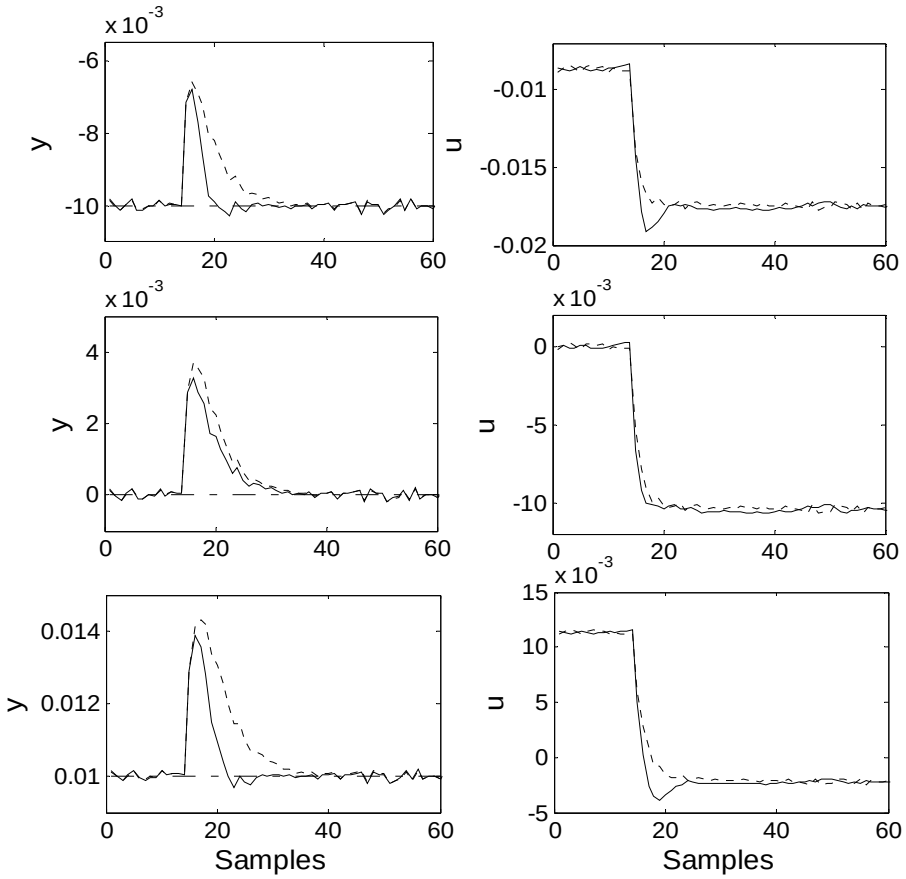

Figure 4.15 Closed-loop responses of two PI designs under +30% step disturbance

Dashed: set-point; solid: STPI; dotted: RLS-based PID

4.4 Conclusion

A new self-tuning PID controller is proposed for nonlinear process control in this chapter. This proposed design method exploit the current process information from controller database and modeling database to realize on-line tuning of PID parameters. The controller database contains the PID parameters and the corresponding information vectors, while the modeling database is employed by the JITL technique for modeling purpose. The PID parameters are obtained from controller database according to the current process dynamics characterized by the information vector at every sampling instant. Then, these PID parameters are updated

during on-line implementation based on certain criterion and the resulting updated PID parameters with their corresponding information vector are then stored into the database. The simulation results reveal that the proposed STPID controller gives better control performance than its counterpart.

Chapter 5

Conclusions and Further Work

5.1 Conclusions

In this thesis, two modeling frameworks are investigated in nonlinear process controller design. Adaptive FNN-PID controller design uses fuzzy neural network (FNN) approach while self-tuning PID controller design is based on the just-in-time learning (JITL) technique. These controllers make use of the information provided by their respective modeling techniques to realize on-line tuning of control parameters and calculation of the manipulated variable for that matter.

In FNN-PID controller design, a FNN model is employed to approximate the controlled nonlinear process. By utilizing Lyapunov method, an updating algorithm is derived to adjust the PID parameters to guarantee the convergence of the tracking error. Next, a new self-tuning PID controller is proposed for nonlinear process control. This proposed method consists of two databases, (i) controller database which is constructed to store the PID parameters together with their corresponding information

vector, and (ii) modeling database which is employed for the standard use by JITL for the modeling purpose. The PID parameters are then obtained on-line based on the nearest-neighborhood criterion and controller database. Compared with the previous adaptive PID controller design methods, the initialization of the proposed design requires less trial and error effort because the initial controller database can be easily constructed from the closed-loop data available in the historical operating data.

Simulation results are presented to illustrate the improved control performance obtained by the proposed controller designs over their conventional counterpart. Using the polymerization reactor and distillation column as examples, the control performances of two proposed controller designs are compared in Tables 5.1 and 5.2. In terms of MAEs, the self-tuning PID controller design is comparable with the FNN-PID controller design in servo performance. However, the self-tuning PID controller gives better load performance than the FNN-PID controller.

Moreover, it can be seen from simulation results that FNN aims at obtaining a global model to describe the process dynamics in the entire operating space, whereas JITL technique focuses simply on the current operating point. FNN is more time-consuming in the identification phase but it is faster in prediction. However, when new data is observed, model update may need to start from scratch. In comparison, JITL is more advantageous because it is enough to update its database when a new input-output data is observed.

Table 5.1 Comparison of two proposed PID designs for polymerization reactor example

	Tracking Error (MAE)	
	FNN-PID	STPID
Servo Response	596.230	572.814
Servo Response*	694.530	711.351
-25% in $C_{m_{in}}$ at y = 25000.5	89.929	86.155
-25% in $C_{m_{in}}$ at y = 18750	148.508	135.982
-25% in $C_{m_{in}}$ at y = 13500	240.324	172.096
+25% in $C_{m_{in}}$ at y = 25000.5	82.184	70.116
+25% in $C_{m_{in}}$ at y = 18750	146.057	119.704
+25% in $C_{m_{in}}$ at y = 13500	266.206	159.746

* In the presence of modeling error

Table 5.2 Comparison of two proposed PI designs for distillation column example

	Tracking Error (MAE)	
	FNN-PI	STPI
Servo Response	5.080×10^{-4}	5.088×10^{-4}
Load Response		
at y = -0.01	1.965×10^{-4}	1.723×10^{-4}
at y = 0	2.144×10^{-4}	2.550×10^{-4}
at y = 0.01	2.544×10^{-4}	2.182×10^{-4}

5.2 Suggestions for Further Work

The data-based control strategies developed in this thesis are restricted to the single-input single-output systems. Therefore, it is of practical importance to generalize the proposed design methods to the multivariable systems, which are often encountered in industrial control practices.

References

Aha, D.W., Kibler, D., and Albert, M.K. (1991). Instance-based learning algorithms. *Machine Learning*, **6**, 37-66.

Altinten, A., Ketevanlioğlu, F., Erdoğan, S., Hapoğlu, H., and Alpbaz, M. (2007). Self-tuning PID control of jacketed batch polystyrene reactor using genetic algorithm. *Chemical Engineering Journal*. *In press*.

Andrášik, A., Mészáros, A., and de Azevedo, S.F. (2004). On-line tuning of a neural PID controller based on plant hybrid modeling. *Computers and Chemical Engineering*, **28**, 1499-1509.

Åström, K.J. (1983). Theory and applications of adaptive control – A survey. *Automatica*, **19**, 471-486.

Åström, K. J. and Hägglund, T. (1995). PID controllers: theory, design and tuning, 2nd edition. International Society for Measurement and Control, North Carolina.

Atkeson, C.G., Moore, A.W., and Schaal, S. (1997a). Locally weighted learning. *Artificial Intelligence Review*, **11**, 11-73.

Atkeson, C.G., Moore, A.W., and Schaal, S. (1997b). Locally weighted learning for control. *Artificial Intelligence Review*, **11**, 75-113.

Babuška, R. and Verbruggen, H. (2003). Neuro-fuzzy methods for nonlinear system identification. *Annual Reviews in Control*, **27**, 73-85.

Bisowarno, B.H., Tian, Y.C., and Tadé, M.O. (2004). Adaptive control of an ETBE reactive distillation column. *Journal of Chemical Engineering of Japan*, **37**, 210-216.

Bontempi, G., Birattari, M., and Bersini, H. (1999). Lazy learning for local modeling and control design. *International Journal of Control*, **72**, 643-658.

Bontempi, G., Bersini, H., and Birattari, M. (2001). The local paradigm for modeling and control: from neuro-fuzzy to lazy learning. *Fuzzy Sets and Systems*, **121**, 59-72.

Cao, S.G., Rees, N.W., and Feng, G. (1997). Analysis and design for a class of complex control systems. Part I: Fuzzy modeling and identification. *Automatica*, **33**, 1017-1028.

Chang, W.D., Hwang, R.C., and Hsieh, J.G. (2002). A self-tuning PID control for a class of nonlinear systems based on the Lyapunov approach. *Journal of Process Control*, **12**, 233-242.

Chen, J.H. and Huang, T.C. (2004). Applying neural networks to on-line update PID controllers for nonlinear process control. *Journal of Process Control*, **14**, 211-230.

Chen, Y.C. and Teng, C.C. (1995). A model reference control structure using a fuzzy neural network. *Fuzzy Sets and Systems*, **73**, 291-312.

Cheng, C. (2006). Data-based methods for modeling, control and monitoring of chemical processes. Ph.D Thesis, National University of Singapore.

Cheng, C. and Chiu, M.S. (2004). A new data-based methodology for nonlinear process modeling. *Chemical Engineering Science*, **59**, 2801-2810.

Cybenko, G. (1996). Just-in-time learning and estimation. In S. Bittanti and G. Picci, Identification, adaptation, learning: the science of learning models from data, Springer, New York, 423-434.

Doyle, F.J.III, Ogunnaike, B.A., and Pearson, R.K. (1995). Nonlinear model-based control using second-order Volterra models. *Automatica*, **31**, 697-714.

Eskinat, E., Johnson, S.H., and Luyben, W.L. (1991). Use of Hammerstein models in identification of nonlinear systems. *AIChE Journal*, **37**, 255-268.

Fink, A., Töpfer, S., and Isermann, R. (2003). Nonlinear model-based control with local linear neuro-fuzzy models. *Archive of Applied Mechanics*, **72**, 911-922.

Gao, F., Wang, F., and Li, M. (2000). An analytical predictive control law for a class of nonlinear processes. *Industrial and Engineering Chemistry Research*, **39**, 2029-2034.

Hirata, M., Ohnishi, Y., and Yamamoto, T. (2004). A design of nonlinear PID control systems by using local model identification. *Proceedings of the 30th Annual Conference of the IEEE Industrial Electronics Society*, Busan, Korea, 2464-2469.

Horikawa, S., Furuhashi, T., and Uchikawa, Y. (1992). On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Transactions on Neural Networks*, **3**, 801-806.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**, 359-366.

Hsu, C.F., Chen, G.M., and Lee, T.T. (2007). Robust intelligent tracking control with PID-type learning algorithm. *Neurocomputing*, *In press*.

Huang, H.P., Jeng, J.C., and Luo, K.Y. (2005). Auto-tune system using single-run relay feedback test and model-based controller design. *Journal of Process Control*, **15**, 713-727.

Hunt, K.J. and Sbarbaro, D. (1991). Neural networks for nonlinear internal model control. *IEE Proceedings-D*, **138**, 431-438.

Jang, J.S.R. (1993). ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, **23**, 665-685.

Jang, J.S.R. and Sun, C.T. (1995). Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, **83**, 378-406.

Juang, C.F. (2002). A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. *IEEE Transactions on Fuzzy Systems*, **10**, 155-170.

Krishnapura, V.G. and Jutan, A. (2000). A neural adaptive controller. *Chemical Engineering Science*, **55**, 3803-3812.

Lee, C.H. and Lin, Y.C. (2005). An adaptive neuro-fuzzy filter design via periodic fuzzy neural network. *Signal Processing*, **85**, 401-411.

Lin, C.J. and Chen, C.H. (2006). A compensation-based recurrent fuzzy neural network for dynamic system identification. *European Journal of Operational Research*, **172**, 696-715.

Lin, C.T. and Lee, C.S.G. (1991). Neural-network-based fuzzy logic control and decision system. *IEEE Transactions on Computers*, **40**, 1320-1336.

Lin, F.J. and Wai, R.J. (2001). Hybrid control using recurrent fuzzy neural network for linear induction motor servo drive. *IEEE Transactions on Fuzzy Systems*, **9**, 102-115.

Lu, J., Chen, G., and Ying, H. (2001). Predictive fuzzy PID control: Theory, design, and simulation. *Information Sciences*, **137**, 157-187.

Narendra, K.S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, **1**, 4-27.

Nelles, O. (2001). Nonlinear system identification: from classical approaches to neural networks and fuzzy models. Springer, Berlin.

Pan, T., Li, S., and Cai, W.J. (2007). Lazy learning-based online identification and adaptive PID control: A case study for CSTR process. *Industrial and Engineering Chemistry Research*, **46**, 472-480.

Pearson, R.K. (1999). Discrete-time dynamic models. Oxford University Press, New York.

Riverol, C. and Napolitano, V. (2000). Use of neural networks as a tuning method for an adaptive PID application in a heat exchanger. *Chemical Engineering Research and Design*, **78**, 1115-1119.

Seborg, D.E., Edgar, T.F., and Shah, S.L. (1986). Adaptive control strategies for process control: a survey. *AIChE Journal*, **32**, 881-913.

Seborg, D.E., Edgar, T.F., and Mellichamp, D.A. (2004). Process dynamics and control. 2nd edition. John Wiley & Sons, New Jersey.

Shahrokhi, M. and Baghmisheh, G.R. (2005). Modeling, simulation and control of a methanol synthesis fixed-bed reactor. *Chemical Engineering Science*, **60**, 4275-4286.

Stenman, A., Gustafsson, F., and Ljung, L. (1996). Just in time models for dynamical systems. *Proceedings of the 35th Conference on Decision and Control*, Kobe, Japan, 1115-1120.

Su, H.T. and McAvoy, T.J. (1997). Artificial neural networks for nonlinear process identification and control. In M.A. Henson and D.E. Seborg, Nonlinear Process Control, Prentice Hall, New Jersey, 371-428.

Su, H.T., McAvoy, T.J., and Werbos, P. (1992). Long-term predictions of chemical processes using recurrent neural networks: a parallel training approach. *Industrial and Engineering Chemistry Research*, **31**, 1338-1352.

Sun, L., Mei, T., yao, Y., Cai, L., and Meng, M.Q.H. (2006). PID controller based adaptive GA and neural networks. *Proceedings of the 6th World Congress on Intelligent Control and Automation*, Dalian, China, 6564-6568.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, **15**, 116-132.

Takao, K., Yamamoto, T., and Hinamoto T. (2006). Memory-based IMC tuning of PID controllers for nonlinear systems. *IEEJ Transactions on Electrical and Electronic Engineering*, **1**, 364-373.

Tan, K.K., Huang, S., and Ferdous, R. (2002). Robust self-tuning PID controller for nonlinear systems. *Journal of Process Control*, **12**, 753-761.

Wai, R.J. and Lin, F.J. (1998). A fuzzy neural network controller with adaptive learning rates for nonlinear slider-crank mechanism. *Neurocomputing*, **20**, 295-320.

Wang, X.S., Cheng, Y.H., and Sun, W. (2007). A proposal of adaptive PID controller based on reinforcement learning. *Journal of China University of Mining and Technology*, **17**, 40-44.

Zhang, J. (2001). A nonlinear gain scheduling control strategy based on neuro-fuzzy networks. *Industrial and Engineering Chemistry Research*, **40**, 3164-3170.

Zhang, J. and Morris, A. J. (1995). Fuzzy neural networks for nonlinear systems modeling. *IEE Proceeding – Control Theory and Applications*, **142**, 551-561.

Zhang, J. and Morris, A. J. (1999). Recurrent neuro-fuzzy networks for nonlinear process modeling. *IEEE Transactions on Neural Networks*, **10**, 313-326.

Zhang, P., Yuan, M., and Wang, H. (2006). Self-tuning PID based on adaptive genetic algorithms with the application of activated sludge aeration process. *Proceedings of the 6th World Congress on Intelligent Control and Automation*, Dalian, China, 9327-9330.