ADAPTIVE NEURO FUZZY INFERENCE SYSTEM APPLICATIONS IN CHEMICAL PROCESSES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

EVREN GÜNER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF CHEMICAL ENGINEERING

NOVEMBER 2003

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Timur Doğu
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis and for the degree of Master of Science.

Prof. Dr. Kemal Leblebicioğlu Co-supervisor	Prof. Dr. Canan Özgen Supervisor

Examining Committee Members

Prof. Dr. Güniz GÜRÜZ

Prof. Dr. Canan ÖZGEN

Prof. Dr. Kemal LEBLEBİCİOĞLU

Prof. Dr. Hayrettin YÜCEL

Assoc. Prof. Naime SEZGİ

ABSTRACT

ADAPTIVE NEURO FUZZY INFERENCE SYSTEM (ANFIS) APPLICATIONS IN CHEMICAL PROCESSES

Güner, Evren
M.S., Department of Chemical Engineering
Supervisor: Prof. Dr. Canan ÖZGEN
Co-supervisor: Prof. Dr. Kemal LEBLEBİCİOĞLU

November 2003, 133 pages

Neuro-Fuzzy systems are the systems that neural networks (NN) are incorporated in fuzzy systems, which can use knowledge automatically by learning algorithms of NNs. They can be viewed as a mixture of local experts. Adaptive Neuro-Fuzzy inference system (ANFIS) is one of the examples of Neuro Fuzzy systems in which a fuzzy system is implemented in the framework of adaptive networks. ANFIS constructs an input-output mapping based both on human knowledge (in the form of fuzzy rules) and on generated input-output data pairs.

Effective control for distillation systems, which are one of the important unit operations for chemical industries, can be easily designed with the known composition values. Online measurements of the compositions can be done using

direct composition analyzers. However, online composition measurement is not feasible, since, these analyzers, like gas chromatographs, involve large measurement delays. As an alternative, compositions can be estimated from temperature measurements. Thus, an online estimator that utilizes temperature measurements can be used to infer the produced compositions. In this study, ANFIS estimators are designed to infer the top and bottom product compositions in a continuous distillation column and to infer the reflux drum compositions in a batch distillation column from the measurable tray temperatures. Designed estimator performances are further compared with the other types of estimators such as NN and Extended Kalman Filter (EKF).

In this study, ANFIS performance is also investigated in the adaptive Neuro-Fuzzy control of a pH system. ANFIS is used in specialized learning algorithm as a controller. Simple ANFIS structure is designed and implemented in adaptive closed loop control scheme. The performance of ANFIS controller is also compared with that of NN for the case under study.

Keywords: NN, Fuzzy Systems, Estimator, EKF, ANFIS, pH control

ÖZ

KİMYASAL PROSESLERDE ADAPTİF SİNİRSEL BULANIK TAHMİN YÖNTEMİNİN UYGULAMALARI

Güner, Evren
Yüksek Lisans, Kimya Mühendisliği
Tez Yöneticisi: Prof. Dr. Canan ÖZGEN
Tez Yardımcısı: Prof. Dr. Kemal LEBLEBİCİOĞLU

Kasım 2003, 133 sayfa

Sinirsel bulanık sistemler bilgiyi otomatik olarak sinir ağları öğrenme algoritmalarıyla elde edebilen, bulanık sistemler ile sinir ağlarının birleştirildiği sistemlerdir. Bu sistemler yerel uzmanların karışımı olarak irdelenebilirler. Adaptif sinirsel bulanık tahmin yönetimi (ASBT) bulanık sistemin adaptif ağ sisteminin yapısında uygulandığı sinirsel bulanık sistemlerden biridir. ASBT uzman bilgisine (bulanık system kuralları şeklinde) ve elde edilen girdi-çıktı verilerine dayalı olarak sistemin girdi-çıktı yapısını oluşturur.

Kimya endüstrilerinde önemli temel işlemlerden biri olan Damıtma sistemlerinin etkili kontrolü ölçülebilen derişim değerleriyle kolayca tasarlanabilir. Gerçek zamanlı derişim ölçümleri doğrudan derişim analiz cihazları kullanarak

yapılabilir. Ancak, gerçek zamanlı derişim analizleri kolay değildir. Çünkü, bu analiz cihazları (gaz kromotografisi gibi) ölçümlerde zaman gecikmesi içerirler. Bu yüzden, ölçülebilen sıcaklık verilerini kullanan gerçek zamanlı tahmin ediciler derişimleri tahmin etmek için kullanılabilirler. Bu çalışmada, sürekli damıtma kolonunda alt ve tepe ürün, kesikli damıtma kolonunda geri döngü tankı derişimleri ölçülebilir tepsi sıcaklıklarını kullanarak tahmin edebilen ASBT tahmin ediciler tasarlanmıştır. Tasarlanan tahmin edicilerin performansları, önceden tasarlanan sinir ağı ve geliştirilmiş kalman filtre (GKF) sonuçları ile karşılaştırılmıştır.

Bu çalışmada, ASBT performansı ayrıca adaptif sinirsel bulanık pH sistem kontrolünde de araştırılmıştır. ASBT özel öğrenme algoritmasının içinde kontrol edici olarak kullanılmıştır. Basit ASBT yapısı dizayn edilmiş ve adaptif kapalı döngü kontrol planında uygulanmıştır. ASBT kontrol edici performansı ayrıca çalışılan sistem için tasarlanan sinir ağı kontrol ediciyle de karşılaştırılmıştır.

Anahtar Kelimeler: Sinir ağları, Bulanık sistemler, Tahmin edici, GKF, ASBT, pH kontrolü

Dedicated to My Family…

ACKNOWLEDGEMENTS

First, I would like to express my deepest appreciation to my supervisor Prof. Dr. Canan Özgen, without whom this thesis would not exist, for her valuable guidance and suggestions, for her kindly and lovely attitude and especially for believing in me during the course of this work.

I would like to thank to Prof. Dr. Kemal Leblebicioğlu, my co-supervisor, for his support and valuable comments and discussions on this study.

I am very grateful to my parents for their endless love, trust and sacrifices for me. I would like to thank to my colleagues; Almıla Bahar, who always motivates me to study harder with enthusiasm, for her assistance to this work, Uğur Yıldız for always listening my problems without grumbling and trying to suggest the best solutions. Thanks to other control team members; Necati Günaydın, and Işık Aşar for their help and friendship throughout this study. I would like to thank also to İbrahim Saygın, Hasan and Ebru Ali, Elif Dörtcan, Erkan Aslan, Önder Kemal Güler, Özgür Yazaydın, Nezih Yağşi, Alper Uzun, İsmail Doğan, Özkan Gül, Bahadır Türk, Serhan Çimen and Yalçın Yıldız who always motivate me and to the valuable person occupying the special part of my life. And, many thanks to my friends that I could not mention here, for listening my weeping and for being with me all the time.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

NOMENCLATURE

c_1	Concentration of acetic acid, mol/l
c_2	Concentration of sodium hydroxide, mol/l
e	Error between the plant output and set point
E	Error Measure
F	Feed flowrate (kmol/hr)
F_1	Acetic acid stream flow rate, l/min
F_2	Sodium hydroxide flow rate, l/min
k	Step Size
L	Reflux flowrate (kmol/hr)
Q	Boilup rate
R	Reflux
S	Small linguistic variable
M	Medium linguistic variable
B	Big linguistic variable
t	Time (hr)
p	Training data pair
T	Temperature (K)
w	Connection weights in NN
x	Mole fraction
z	Feed mole fraction

Greek Letters

η	Learning rate
μ	Membership grade
α	Structure parameter

Subscripts

b	Bottom
d	Top

F	Feed
k	sampling time
i	Number of neuron
j	Number of layer

Abbreviations:

ANFIS	Adaptive Neuro Fuzzy Inference System
ANN	Artificial Neural Network
ANW	Action Network
ARMA	Auto Regressive Maximum Average
CNW	Critic Network
CMPC	Constraint Model Predictive Controller
CSTR	Continuous Stirred Tank Reactor
DNN	Dynamic Neural Network
DMC	Dynamic Matrix Control
EKF	Extended Kalman Filter
ERN	Externally Recurrent Networks
FAM	Fuzzy Associative Memory
FLC	Fuzzy Logic Controller
FNN	Fuzzy Neural Network
FRLRPC	Fuzzy Relational Predictive Controller
GPC	Generalized Predictive Control
IAE	Integral of the Absolute Error
IMC	Internal Model Control
IRN	Internally Recurrent Networks
LVQ	Learning Vector Quantization
MLP	Multi Layer Perceptron
MPC	Model Predictive Control
NC	Number of Components
NF	Neuro-Fuzzy
NFC	Neuro Fuzzy Control
NN	Neural Network
RNN	Reccurent Neural Networks
RBF	Radial Basis Function
SLPFC	Self-Learning Predictive Fuzzy Controller
SVD	Singular Value Decomposition

CHAPTER 1

INTRODUCTION

An intelligence system is a system that is able to make decisions that would be regarded as intelligent if were done by humans. Intelligence systems adapt themselves using some example situations (inputs of a system) and they correct decisions automatically for future situations (Czogala and Leiski 2003). Neural networks (NNs), Fuzzy systems, and Neuro-Fuzzy systems are the examples of the artificial intelligence systems.

Fuzzy systems provide a unified framework for taking into account the gradual or flexible nature of variables, and representation of incomplete information (Prode and Dubio 1992). This is an alternative to classical approach and is based on the observations that, humans think using linguistic terms such as “small” or “large” and others rather than numbers. The concept is described in a natural language, by Zadeh using fuzzy sets introduced by himself in 1965. The essence of fuzzy systems is conditional if-then rules, which use fuzzy sets as linguistic terms in antecedent and conclusion parts. A collection of these fuzzy if-then rules can be determined from human experts or alternatively can be generated from observed data (examples). The main advantage of such fuzzy systems is the easiness to interpret knowledge in the rule base (Leiviska 2001).

NNs are the systems that get inspiration from biological neuron systems and mathematical theories for learning. They are characterized by their learning ability with a parallel-distributed structure and also can be considered as black box modeling. They are useful empirical modeling tools that have been used for process estimation and control since 1950's.

In most fuzzy systems, fuzzy if-then rules were obtained from a human expert. However, this method of knowledge acquisition has great disadvantages; not every expert can and/or wants to share his/her knowledge. For this reason, NNs were incorporated into fuzzy systems, which can acquire knowledge automatically by learning algorithms of NNs. These systems are called Neuro-Fuzzy systems and have advantages over fuzzy systems, i.e., acquired knowledge is easy to understand. Like in NNs, knowledge is saved in connection weights, but can also be easily interpreted as fuzzy if then rules. The Neuro-Fuzzy systems can be viewed as a mixture of local experts (rules operate dominantly in each fuzzy region) and their parameters are updated using gradient and least squares optimization methods. The most frequently used NNs in Neuro-Fuzzy systems are radial basis function networks (RBFN). Their popularity is due to the simplicity of structure, well-established theoretical basis and faster learning than in other types of NNs. Adaptive network based fuzzy inference system or adaptive Neuro-Fuzzy inference system (ANFIS), first proposed by Jang (1993), is one of the examples of RBFN Neuro Fuzzy systems in which a fuzzy inference system is implemented in the framework of adaptive networks. ANFIS constructs an input output mapping based both on human knowledge (in the form of fuzzy if then rules) and on generated input output data pairs by using a hybrid algorithm that is the combination of the gradient descent and least square estimates. ANFIS is not a black box model and works well with optimization techniques, which is computationally efficient and is also

well suited to mathematical analysis. Therefore, it can be used in modeling and controlling studies, and also for estimation purposes.

In batch and continuous distillation processes, composition control is very important. Especially, in order to meet the purity specifications, a batch column has to be operated as precisely as possible. If the current compositions are known, they can form a basis for improving the process performance through an operator decision making for the development of a closed loop control scheme (Venkateswarlu and Avantika 2001). An effective feedback control systems for continuous distillation columns can also be easily designed with the known composition values. Online measurements of the compositions can be done using direct composition analyzers. However, online composition measurement is not feasible, since, these analyzers, like gas chromatographs, involve large measurement delays as well as high investment and maintenance costs. As an alternative, compositions can be estimated from temperature measurements. Thus, an online estimator that utilizes temperature measurements can be used to infer the produced compositions. In this study, ANFIS estimators are designed to infer the top and bottom product compositions in a continuous distillation column and to infer the reflux drum compositions in a batch distillation column. Designed estimator performances are further compared with the other types of estimators such as NN and Extended Kalman Filter (EKF).

Since the control of pH is recognized as a difficult problem in literature due to its highly nonlinear nature. In this study, an ANFIS performance is also investigated in the adaptive Neuro-Fuzzy control of a pH system. Specialized learning algorithm is used for pH system to see the performance of the ANFIS as a controller. Therefore, simple ANFIS controller is designed and implemented in adaptive closed loop control scheme. The performance of the ANFIS controller is also compared with that of NN for the case under study.

CHAPTER 2

LITERATURE SURVEY

In this chapter, the literature survey on Neural Network (NN), Fuzzy Logic (FL) and Neuro-Fuzzy (NF) systems are given. In the first section, studies on NNs, in the second section cases in which fuzzy systems were applied are presented. In the last section, studies on NF systems development and implementation are given. At the end of each section, summary of the studies are also presented in tables.

2.1 Neural Networks

A neural network is a data processing system consisting of a large number of simple, highly interconnected processing elements (artificial neurons) in an architecture inspired by the structure of the cerebral cortex of the brain (Tsoukalas and Uhrig 1997). The formal realization that the brain in some way performs information processing tasks was first spelt out by McCulloch and Pitts (1943). They represented the activity of individual neurons using simple threshold logic elements, and showed how networks made out of many of these units interconnected could perform the logical operations. Rosenblat (1959) developed the concept of perceptron, a generalization of the McCulloch and Pitts concept of the functioning of the brain, by adding learning (Lisboa 1992). These studies were the initiations of NNs.

NNs are applied in many areas of science and engineering (Svrcek 2001). The use of NN is especially increased in the last two decades due to the following reasons:

- Recent advances in computer technology and parallel processing have made the use of NNs more economically feasible than past.

- Since NNs are composed of nets of nonlinear functions, they have the ability to evolve good process model from example data and require little or no priori knowledge.

- They have the potential to solve complex problems that have not been satisfactorily handled by traditional methods (Himmelblau and MacMurray 1995).

In chemical engineering, since NNs provide good empirical models of complex nonlinear processes, they have been frequently used in complex modeling problems. They have also used in control schemes as aid or controller. Fault detection, prediction of polymer quality and data rectification are some other examples of NN applications in chemical engineering (Himmelblau 2000).

Yamamura et. al. (1988) proposed three different methods; general learning, specialized learning and combination of them to train the NN controller to act as the inverse of the plant. Since using general learning can be very difficult to provide adequate performance in practical control applications, method of error propagation backwards through the plant is introduced to train the network exactly on the operational range of the plant. Also, combination of generalized and specialized training was proposed to use their advantages and to avoid their potential disadvantages. A plant that converts polar coordinates (r,θ) to Cartesian coordinates (x,y) was considered as a simple plant example.

Proposed learning methods were applied to train the NN controller. Hybrid learning algorithm showed better performance than general and specialized learning.

Bhat and McAvoy (1990) discussed the use of backpropagation, the most widely used NN, for non-linear dynamic modeling and model-based control of chemical process systems. They applied this algorithm to a CSTR to model the non-linear dynamic response of pH in the reactor and used this backpropagation model for control, including learning process inverses. In their paper, after a review of the backpropagation algorithm, a comparison between NN dynamic modeling and traditional modeling are presented. The backpropagation technique was shown to be able to pick up more of the non-linear characteristics of the CSTR than the traditional ARMA modeling. Then two approaches that use NNs for model-based control were discussed. One of them was the same as that used in DMC (Cutler and Ramaker, 1980), except that the nonlinear NN model was used in place of the linear convolution model. The other was the backpropagation dynamic modeling net that can be trained to learn the inverse of the process model and then this inverse can be employed in an Internal Model Control, IMC, structure.

Also in 1990, Mc Avoy et. al. used NNs for modeling nonlinear chemical systems such as; a steady state reactor and a dynamic pH CSTR. In steady state example, an isothermal CSTR reaction sequence was considered. Inputs to the net were the scaled feed composition and reactor space time. The outputs were the dimensionless product concentrations. In dynamic example, to model the pH response, past and present values of pH and flow rate of base and future values of manipulated variables were fed to the net. The output from the net was the pH in the future. It was shown that, in all cases, NNs were able to learn the underlying governing relationships.

Himmelblau and MacMurray (1995) implemented a NN for MPC in a packed distillation column. Their column had the interesting feature of multiple changes in the sign of the process gain as the operating conditions change. They used an External Reccurent Network (ERN) to model the process over a significant part of the state space and compared the model performance with a simplified principles model for MPC. Although both models gave similar performance, in some instances the NN model was better. They concluded that, when the process is too complex to be modeled by a first principles model, or takes too long to model, an ERN model might be a very effective choice for a model.

Fong et. al. (1995) presented a NN approach to model and control of a pH process in a CSTR. In the modeling task, their network consists of two single hidden layer nets connected in cascade. The first one was a recurrent net to reflect the dynamic nature of the CSTR, while the second one was a static net configured to reflect the static nature of the titration characteristics. For the control task, they simulated two control schemes. In the first scheme, a NN was trained to provide an inverse to the static titration curve. The training data were obtained from a model for the CSTR. In the second control scheme, a model reference adaptive NN control was studied. Thus, a direct method of adaptation was used.

Himmelblau et. al. (1996) discussed the feasibility of using Internal Recurrent Network (IRN) to identify a nonlinear dynamic process under closed loop control conditions. They used a direct closed loop identification method, identifying an open loop process model using process input and output data collected under closed loop control. A packed distillation column and CSTR were used to demonstrate the feasibility of using IRN for closed loop identification. They concluded that IRN is a promising nonlinear process identification using data taken under closed loop control.

Wang et. al. (1998) proposed a nonlinear predictive control framework , in which nonlinear processes are modeled using NNs. In this framework, a predictive controller, based on the NN model, calculates the optimal manipulated variable; while at the same time the available measured output (controlled variable) is used to modify the output predicted by the NN model. The proposed framework was implemented to the pH control problem in a CSTR.

Hussain (1999) provided an extensive review of the various applications utilizing NNs for chemical process control, both in simulation and also in online implementation. He categorized the review under three major control schemes; inverse model based control, predictive control and adaptive control methods. The review shows that using NNs in chemical process control is widespread and multilayered feedforward NN is the most popular network for such process control applications. However, it is emphasized that there is lack of actual successful online applications.

Budman and Kavchak (1999) worked on estimation and control of nonlinear processes using adaptive radial basis function NNs. They used NNs that are adapted online to model the process dynamics. The Radial Basis Functions (RBF) used were wavelets, which provide advantages because of their localization in space. The network used the overall prediction error and were incorporated into an adaptive controller and tested on an exothermic CSTR.

Jutan and Krishnapura (2000) proposed an adaptive NN controller for the control of nonlinear dynamical systems. The NN controller was adaptive in structure and used no explicit model of the process in the design. Contrasting traditional NNs, which have many connection weights, the proposed controller network had a few weights. This NN controller was applied to a nonlinear CSTR and a pH neutralization process and exhibited very good performance.

The summary of the literature on NNs is given in Table 2.1.

Table 2.1 Summary of the NN studies

Year	Authors	System	Structure	Specific
1988	Yamamura et. al.	No specific system	Backpropagation NN	Learning methods; general, specialized and conjunction of these two methods
1990	Bhat & McAvoy	CSTR	Backpropagation NN	Non-linear dynamic modeling and model-based control of pH in CSTR with NN
1990	McAvoy et. al.	Steady state reactor and a dynamic pH CSTR	Backpropagation NN	Modeling product concentrations and pH in reactors with NNs
1995	Himmelblau & MacMurray	A packed distillation column	ERN	Modeling the process with ERN and comparison performance with first principle models
1995	Fong et. al.	pH process in CSTR	NN consists of two single hidden layer nets connected in cascade	Layers; recurrent and static nets to reflect the dynamic nature of CSTR
1996	Himmelblau et. al.	A packed distillation column	IRN	A direct closed loop identification with input and output data collected under closed loop control
1998	Wang et. al.	pH process in CSTR	Backpropagation NN	NN predictive control of CSTR, which is modeled using NN
1999	M. A. Hussain	No specific system	NNs	A review of the various applications utilizing NNs for chemical process control
1999	Budman & Kavchak	CSTR	RBFNN	Modeling process dynamics with radial basis functions
2000	Jutan & Krishnapura	CSTR and pH neutralization process	Backpropagation NN	NN controller that uses no explicit model of the process

2.2 Fuzzy Logic Systems

"As the complexity of a system increases, our ability to make precision and yet significant statements about the behavior diminishes until a threshold is reached beyond which precise and significance become almost mutually exclusive characteristics." (Zadeh 1971)

The concept of fuzziness was first proposed by Zadeh (1965). He aimed to describe complex and complicated systems using fuzzy approximation and introduced fuzzy sets. "Generally, fuzzy logic can be considered as a logical system that provide a model for modes of human reasoning that are approximations rather than exact" (Rutkowska 2002).

Fuzzy logic systems had found successful applications in wide variety of fields such as: automatic control, pattern recognition, signal processing, expert systems, communication, system identification and time series prediction (Czogala and Leski 2000). In chemical engineering systems, they have been generally used in control studies. Since Fuzzy Logic Control (FLC) does not require a model and the control is based on expertise human reasoning, they have been applied in many control schemes. Also, they are used extensively in modeling.

Kim and Kim (1995) combined the fuzzy control and predictive control and applied it to the binary distillation column. They compared its performance with MPC by simulation and experimentation. They used a combined technique in which a switching scheme was implemented. When error or change of error between the controlled variable and the set point of the variable was larger than a certain value, the fuzzy control was used in control computation. Otherwise MPC was employed. It was concluded that combined method improved the control performance for set point tracking and disturbance rejection. Considering

the IAE scores it was found that combined control technique improved the control performance by 9% over the MPC.

Benz et. al. (1996) developed and tested a self adaptive computer based pH measurement and control system for the control of fermentation process in a laboratory reactor and for the neutralization of wastewater streams. The controller was based on the fuzzy logic, which permitted the inclusion of subjective knowledge, often based on experience and not on a theoretical model. It also employed data and used primary knowledge (experimental results) directly. At the end of the study, the experiments showed that the fuzzy controller was able to adjust the pH value faster than the common PID controller. It was also stated that, by using self adaptive fuzzy controller it was possible to control the pH value of systems with an extremely small buffer capacity, even using strong acids and bases.

In another research, Matko et. al. (1997) presented a new method of fuzzy model based predictive control for highly nonlinear pH process. Since a standard MPC technique uses linear process model and thus unable to deal with strong process nonlinearities, this method was based on a fuzzy model of the Takagi-Sugeno form. Different step responses for current operating points were extracted from the fuzzy model of the process online, and this concept was integrated into standard linear DMC predictive control. It was observed that despite its simplicity (fuzzy rules of zeroth order), the presented fuzzy model had good accuracy in steady state mapping, as well as in prediction of dynamic behavior.

Fisher et. al. (2000) investigated the performance of the two fuzzy relational controllers; the self-learning predictive fuzzy controller (SLPFC) and the fuzzy relational long range predictive controller (FRLRPC). They were

evaluated experimentally on a laboratory scale, nonlinear, interacting tank process. It was shown that, although both controllers gave good overall performance, the SLPFC was slightly better than the FRLRPC.

A fuzzy classifier that can be used as an adequate and reliable expert system to perform quality qualifications in chemical engineering system was proposed by Bafas et. al. (2002). The method builds a fuzzy logic model, which infers the quality variables from other accurately measured system parameters. It was applied to two chemical engineering problems; the wine distillate maturation and the tissue making process and compared with a feedforward NN methodology and a fuzzy identification method. It was confirmed that classifications of proposed fuzzy logic model were more accurate.

Burden et. al. (2003) constructed a linear constrained model predictive controller (CMPC) in order to control nonlinear systems. For this purpose, a fuzzy logic concept was used. They combined fuzziness with CMPC to extend the applicability of CMPC to nonlinear systems. 2-D and 3-D type fuzzy PI controllers were also presented. Some examples were given to show how to apply these structures to chemical engineering systems.

Roffel et. al. (2003) investigated the development of a simple model that describes the product of an experimental batch distillation column. Their objective was to develop a hybrid model that can simulate a batch run, including start-up. In hybrid modeling, Takagi-Sugeno type fuzzy models were used. For identification of fuzzy models, G-K (Gustafson and Kessel, 1978) clustering algorithm with structure optimization was used. Proposed framework included the behavior for bottom exhaustion, column warming-up that applies for a column under constant quality or constant temperature operation. The production as a function of a batch time was described with the derived model

without the need to describe internal column dynamics. The summary of the literature survey on FL is given in Table 2.2.

Table 2.2 Summary of the FL studies

Year	Authors	System	Structure	Specific
1995	Kim & Kim	A binary distillation column	FLC	Combining the fuzzy control and model predictive control
1996	Benz et. al.	pH systems ; fermentation reactor and neutralization wastewater	Adaptive FLC	Controlling pH value systems with an extremely small buffer capacity
1997	Matko et. al.	pH process	Fuzzy models	MPC with Takagi-Sugeno type fuzzy model
2000	Fisher et. al.	A laboratory scale, nonlinear, interacting tank process	FLC	Fuzzy relational controllers; SLPFC and FRLRPC
2002	Bafas et. al.	A wine distillation and tissue making processes	Fuzzy classifier	A fuzzy logic model, which infers the quality variables from other accurately measured system parameters
2003	Burden et. al.	No specific system	Fuzzy models	Combining fuzziness with CMPC to extend the applicability of CMPC to nonlinear systems
2003	Roffel et. al.	A batch distillation column	Fuzzy models	A hybrid modeling of the column with Takagi-Sugeno type fuzzy models

2.3 Neuro Fuzzy Systems

In most fuzzy systems, fuzzy rules were obtained from the human expert. However, every expert does not want to share his knowledge and there is no standard method that exists to utilize expert knowledge. As a result, ANNs were incorporated into fuzzy systems to be able to acquire knowledge automatically by learning algorithms. The learning capability of the NNs was used for automatic fuzzy if then rules generation (Czogala and Leski 2000).

The connection of fuzzy systems with an ANN is called neuro-fuzzy, NF, systems. Like in NNs where knowledge is saved in connection weights, it is interpreted as fuzzy if then rules in NF systems. The most frequently used NN in NF systems is radial basis function neural network, RBFNN in which each node has radial basis function such as Gaussian and Ellipsoidal. Their popularity is due to the simplicity of structure, well-established theoretical basis and faster learning than in other types of NNs. Also, there are many developed fuzzy neural networks (FNN) as NF algorithms in literature. Adaptive network based fuzzy inference system, ANFIS, is one of them. It is type of RBFNN.

Jang (1992) proposed to use the ANFIS architecture to improve the performance of the fuzzy controllers. The performance of the fuzzy controller relies on two important factors: knowledge acquisition and the availability of human experts. For the first problem, Jang proposed the ANFIS to solve the automatic elicitation of the knowledge in the form of fuzzy if then rules. For the second problem, that is how the fuzzy controller is constructed without using human experts; a learning method based on a special form of gradient descent (backpropagation) was used. The proposed architecture identified the near optimal membership functions and the other parameters of a controller rule base for achieving a desired input-output mapping. The backpropagation type gradient descent method was applied to propagate the error signals through

different time stages to control the plant trajectory. The inverted pendulum system was employed to show the effectiveness and robustness of the proposed controller.

In 1992, Uchikawa et. al. presented a fuzzy modeling method using fuzzy neural networks, FNNs, with the backpropagation algorithms. They proposed three types of NN structures of which the connections weights have particular meanings for getting fuzzy inference rules for tuning membership functions. These structures are categorized into FNNs and these different types FNNs realize three different types of reasoning.

Rao and Gupta (1994) described the basic notions of biological and computational neuronal morphologies and the principles and architectures of FNNs. Two possible models of FNN were given. In first one, the fuzzy interface provides an input vector to a multilayered network in response to linguistic statements. Then the NN can be trained to yield desired output. In the second scheme, a multilayered NN drives the fuzzy inference mechanism. It was pointed out that using FNN approaches having the potential for parallel computation could eliminate the amount of computation required.

In another paper, Uchikawa et. al. (1995) presented a new design method of adaptive fuzzy controller using linguistic rules of fuzzy models of the controlled objects. FNNs identify fuzzy models of nonlinear systems automatically with the backpropagation algorithm in this method. Authors also presented a rule-to-rule mapping method for describing the behavior of fuzzy dynamical systems. Using this methodology, first, the control rules are modified by considering rule-to-rule transitions. After that, designed controller was implemented with another FNN. The adaptive tuning of the control rules was done using the fuzzy model of the controlled object by utilizing the derivative

value from the fuzzy model. A second order system was simulated to show the feasibility of the proposed design method.

Aoyama et. al. (1995) proposed a FNN employed in IMC scheme for SISO nonlinear process. The control-affine model was described identified from transient and steady state data using backpropagation in this scheme. Inverse of the process is obtained through algebraic inversion of the process model to use as a controller. Two highly nonlinear process; CSTR and pH neutralization processes were studied. In the CSTR, effluent concentration was controlled using the coolant flow rate. In pH neutralization, manipulating the base flow rate controlled the effluent pH. The proposed strategy was compared to a conventional PID controller for set point and disturbance changes. The results showed that controller performance was significantly better than PID controller.

A methodology for batch process automation using reinforcement learning was presented Martinez and Wilson (1997). In this study, an autonomous controller continuously learned to implement control actions that can drive the process state very close to desired one with near optimal performance. Fuzzy Q-Learning algorithm was proposed to build the controller. This methodology was exemplified using a batch process involving simultaneous reaction and distillation.

Dagli et. al. (1997) combined the Dynamic Neural Networks, DNN, with the Fuzzy Associative Memory, FAM, that determines the performance of the controller by evaluating the error (e) and derivative of the error (Δe) of the system to find a better model for nonlinear control problems. The proposed model consisted of three major parts: action network, ANW, critic network, CNW and fuzzy membership adjustment procedure. ANW was the main controller of the model generates the control signal of the system. The CNW was used as the

FAM. The output of the CNW was sent to ANW to adjust the weight matrices of the DNN. CNW was composed of the fuzzifier, the rule base and deffuzifier. Fuzzy membership adjustment procedure was used to improve the quality of the output network. The proposed model was tested in chemical and real processes.

Peng and Chen (1999) developed an intelligent control system for the direct adaptive control of chemical processes in the presence of unknown dynamics, nonlinearities and uncertainties. They constructed a Neuro-Fuzzy Controller (NFC) with an equivalent four-layer connectionist network. With a derived learning algorithm, fuzzy rules and membership functions were updated adaptively by observing the process output error. A shape tunable NN with backpropagation algorithm was also suggested as the estimator in order to provide a reference signal to the controller. The proposed algorithm was implemented to direct adaptive control of an open loop unstable nonlinear CSTR. Comparisons were performed with a static fuzzy controller.

Belarbi et. al. (2000) proposed a FNN that learns rules of inference for a fuzzy system through classical backpropagation. The network was trained off-line in a closed loop simulation to design Fuzzy Logic Controller (FLC). Another network was used as a design model in order to backpropagate the error signal. Controller rules were extracted from the trained network to build the rule base of the FLC. The framework was applied to the estimation and control of a batch pulp digester. The Kappa number, the controlled variable, which cannot be measured online was estimated with same type of FNN through the measurements of the batch temperature and concentration of the alkali. Although the FLC was quite simple with nine rules, simulation results showed good degree of robustness in the face of parameter variations and changes in operating conditions.

Leiviska et. al. (2001) used linguistic equations (fuzzy models) and NN models in prediction of Kappa number in the continuous digester. Actual prediction data was collected from a continuous digester house. It included the extraction flow measurements and reactive index, temperature in the extraction flow, and the measurement of Kappa number from an online device after digester. Then the data was divided into training and testing data. ANFIS was used as one of the fuzzy model and gave the best performance in other fuzzy models.

Castillo and Melin (2001) used an ANFIS methodology in electrochemical process. The problem in battery manufacturing was to find how much the current could be increased without causing battery to explode due to the increase in temperature and at the same time minimizing the time of loading. Since ANFIS can be used to adapt the membership functions and consequents of the rule base according to the historical data of the problem, ANFIS was used as fuzzy controller in this research. Fuzzy logic toolbox of MATLAB was used with 5 membership functions and first order Sugeno function in the consequents. ANFIS controller input and output were temperature and electrical current, respectively. They found that, the ANFIS methodology gave better results than manual, conventional and fuzzy control methods.

In another study of adaptive FNN, Hancheng et. al. (2002) used the ANFIS to extract fuzzy rules from experimental data for material property modeling. Prediction of tensile strength based on compositions and microstructure was aimed. Hence, backpropagation NNs used in literature needed large amount of training data in order to acquire high learning precision, and had a poor generalization capability and obtaining experimental data was also expensive, authors tried to use ANFIS. To verify the generation of the model, 38 available patterns were divided into two categories: a training set of 29 cases and a test

of 9 cases. All the membership functions of the input variables were of the gaussian type, and parameters sub-spaces were determined by using *K*-means clustering of the training data set, 20 rules being obtained. Inputs to the ANFIS were the carbon equivalent, the graphite flake size, and the microhardness of the matrix, the amount of austenite dentrite and the eutectic cell. Output was the tensile strength. The results were compared with multiple statistical analyses, fuzzy regression and the generalized regression network and ANFIS showed good learning precision and generalization.

Sarimveis et. al. (2002) presented a new fast and efficient method for training RBFNN to model nonlinear dynamical MIMO discrete-time systems. The proposed training methodology was based on a fuzzy partition of the input space and combines self-organized and supervised learning. According to the algorithm, first, the centers of the nonlinear units were determined. Then, the widths of Gaussian functions were calculated. Finally, the connection weights between the hidden and the output layers were computed, by solving a simple quadratic optimization problem, which minimized the errors between the desired and predicted outputs. The developed RBF network models were used to predict the concentration and temperature in CSTR and Kappa number in a continuous pulp digester. The most important advantage of proposed algorithm was the ability to determine the network structures and parameters using a very limited computational time.

The summary of the literature survey on NF is given in Table 2.3.

Table 2.3 Summary of the NF studies

Year	Authors	System	Structure	Specific
1992	R. J. Jang	Inverted pendulum system	ANFIS	ANFIS, used as fuzzy controller to improve the performance of the controller
1992	Uchikawa et. al.	No specific system	FNN	Proposed three different type FNNs, used for getting FIS for tuning MFs
1994	Gupta & Rao	A second order system	FNN	Error-based and output-based learning algorithms
1995	Uchikawa et. al.	A second order system	FNN	Description of dynamical behavior of control system and adaptive tuning of a controller
1995	Aoyama et. al.	CSTR and pH neutralization	Modified FNN	Control- affine process model
1997	Dagli et. al.	Different kind of nonlinear real life processes	Dynamic neural networks with fuzzy associative memory	Fuzzy MFs adjustment procedure
1997	Martinez & Wilson	Batch process involving reaction and distillation	Fuzzy Logic System	Fuzzy controller constructed by Fuzzy Q learning algorithm
1998	Castillo & Melin	Electrochemical process	ANFIS	ANFIS, used as controller in battery manufacturing
1999	Cheng & Peng	CSTR	FNN and a shape tunable NN	NFC with a shape tunable NN that is estimator to provide a reference signal
2001	Leiviska et. al.	A continuous pulp digester	ANFIS	ANFIS estimator for prediction of Kappa number

Table 2.3 Continued….

2002	Hancheng et. al.	Material property modeling	ANFIS	ANFIS estimator for prediction of material tensile strength based on compositions and microstructure
2002	Sarimveis et. al.	CSTR and continuous pulp digester	RBFNN	RBFNNs, used as to predict the concentrations and temperature in CSTR and Kappa number in digester

CHAPTER 3

STRUCTURES OF ARTIFICIAL INTELLIGENCE SYSTEMS

In this chapter, the theoretical background and structures for artificial intelligence systems will be given. Characteristics of NNs and the theory of fuzzy logic will be presented. Also, the structure of ANFIS will be explained in detail.

3.1 Neural Networks (NN)

Basic characteristics of NNs will be summarized in this section. First, a neuron model and architecture of NN will be described. After that, learning in NNs will be explained. Also, adaptive network will be given as an example of NN with backpropagation and hybrid learning algorithms.

3.1.1 Models of Neuron

A neuron is a special nervous cell in organisms, which have electric activity. These cells are mainly intended for the operation of the organism. The biological neuron is shown schematically in Figure 3.1 . A neuron consists of a cell body, which is surrounded by a membrane. The neuron has dendrites and axons, which are its inputs and outputs of neuron. Axons of neurons join to dendrites of other neurons by forming synaptic contacts (synapses).

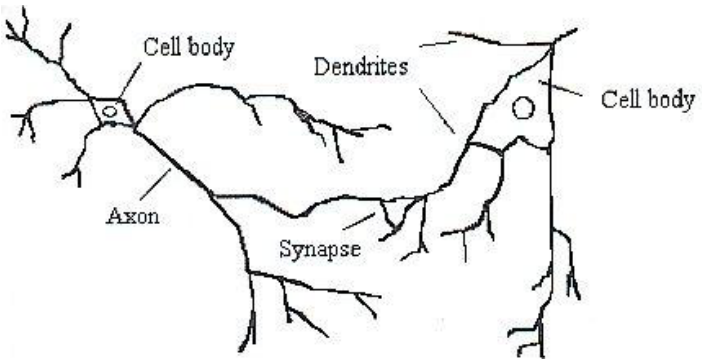

Figure 3.1 Biological neuron

Input signals of the dendrite tree are weighted and added in the cell body and formed in the axon, where the output signal is generated. The signal's intensity, consequently, is a function of a weighted sum of the input signal. The output is passed through the branches of the axon and reaches the synapses. Through the synapses the signal is transformed into a new input signal for neighbor neurons. The input signal can be either positive or negative (exciting or inhibiting), depending on the synapses (Aliev 2001).

In accordance with the biological model, different mathematical models were suggested. The mathematical model of the neuron, which is usually utilized under the simulation of NN, can be shown in Figure 3.2. The neuron receives a set of input signals x_1, x_2,…, x_n (vector X) which are usually the output signals of other neurons. Each input signal is multiplied to a corresponding connection weight, w, analogue of the synapse's efficiency. Weighted input signals come to the summation module corresponding to cell body, where their algebraic summation is executed and the excitement level of neuron is determined:

$$I = \sum_{i=1}^{n} x_i w_i \qquad (3.1)$$

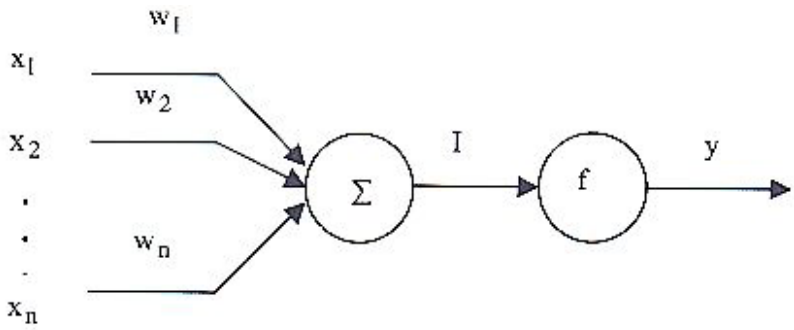

Figure 3.2 Mathematical neuron

The output signal of a neuron is determined by conducting the excitement level through the function f, called activation function as in Equation 3.2.

$$y = f(I) \tag{3.2}$$

The following activation functions can be utilized as function f:

Linear function:

$$y = k.I \text{ , } k\text{= constant} \tag{3.3}$$

Binary (threshold) function:

$$y = \begin{cases} 1, \; if \;\; I \geq \theta \\ 0, \; if \;\; I < \theta \end{cases} \tag{3.4}$$

Sigmoid function:

$$y = \frac{1}{1 + e^{-(I)}} \tag{3.5}$$

3.1.2 Architectures of Neural Networks

The totality of the neurons, connected with each other and with the environment, forms the NN. Figure 3.3 shows the basic structure of the neural

network. The input vector comes to the network by activating the input neurons. A set of input signals of a network's neurons is called the vector of input activeness. Connection weights of neurons are represented in form of matrix W, element w_{ij} of which is the connection weight between i-th and j-th neurons. During the network functioning process, the input vector is transformed into output one, i.e. some information processing is performed. The computational power of the network, thus, solves problems with its connections. Connections link inputs of one neuron with output of others. The connection strengths are given by weight coefficients. NN can also consist a bias term, which acts on a neuron like an offset. The function of the bias is to provide a threshold for the activation of neurons. The bias can be connected all neurons in network.

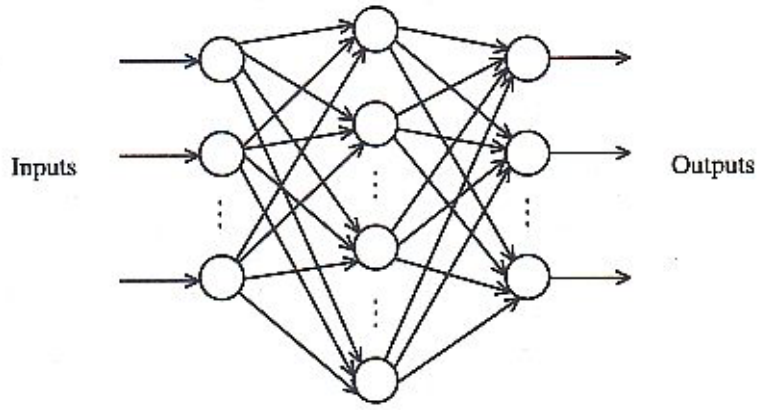

Figure3.3 Basic structure of neural network

NNs can be divided into two types of architectures: feedforward networks and recurrent NNs.

Feedforward networks have no feedback connections. In this type of network, neurons of the j-th layer receive signals from environment (when j=1) or the neurons of previous the (j-1)-th layer when (j>1) and pass their outputs to neurons of the next (j+1)-th layer to the environment (when j is the last layer). Feedforward networks can be single-layer or multi-layer. Multilayer NNs

consist of input, output and hidden layer. The use of hidden layers allows an increase in the computational power of the network. Choosing the optimal structure of a network provides an increase in reliability and computational power, and a decreased processing. The multiplayer perceptron (MLP), adaptive network, radial basis function neural network (RBFNN), the learning vector quantization (LVQ) network, and the group-method of data handling network (GMDH) can be given the examples of feedforward networks (Bulsari A.B. 1995).

Recurrent neural networks (RNN) have structures similar to standard feedforward NN with layers of nodes connected via weighted feed-forward connections, but also include time delayed feedback or recurrent connections in the architecture (Himmelblau 2000). The important advantage of the RNN is the ability to approximate a continuous or discrete nonlinear dynamic system by neural dynamics defined by a system of nonlinear differential equations. This offers the opportunities for applications to adaptive control problems. Examples of the RNN include the Hopfield network, the Elman network, and the Jordan network.

3.1.3 Learning in Neural Networks

Generally, learning is the process by which the NN adapts itself to a stimulus, and eventually it produces a desired response. It is also a continuous classification process of input stimuli: when a stimulus appears at the network, the network either recognizes it or it develops a new classification. Actually, during the process of learning, the network adjusts its parameters, the synaptic weights, in response to an input stimulus so that its actual output response converges to the desired output response. When the actual output response is the same as the desired one, the network has completed the learning phase. Learning rules for networks are described by mathematical expressions called learning equations. The neurons in NNs may be interconnected in different ways;

however, the learning process is not same for the all. It is known that, different learning methodologies suit different people. Like this, different learning techniques suit different NNs (Kartalopoulos 1996). There are two general categories of learning in NNs, supervised and unsupervised learning.

In supervised learning, both the input and the actual response and the desired response are available and are used to formulate a cost (error) measure. If the actual response differs from the target response, the NN generates an error signal, which is then used to calculate the adjustment that should be made to the network's weights so that actual output matches the target output (Jain 1997).

Unlike supervised learning, there is no target output in unsupervised learning. During the training period, the network receives at its input many different input patterns and it arbitrarily organizes the pattern into categories. When a stimulus is later applied, the network provides an output response indicating the class to which the stimulus belongs. If a class cannot be found for the stimulus, a new class is generated. This type of learning sometimes referred to as self-organizing learning.

3.1.4 Learning algorithms

A learning algorithm is a mathematical tool that outlines the methodology and the speed for NN to reach the steady state of its parameters, weights and thresholds successfully. It starts with an error function (energy function), which is expressed in terms of weights. The objective is to minimize the error in the set of weights. When the error function is zero, or small enough, the steady state of the network and of the weights is reached. During learning, the error function decreases and the weights are updated. The decrease may be accomplished with different optimization techniques such as the Delta rule, Boltzman's algorithm,

the backpropagation learning algorithm and simulation annealing. The selection of the error function and the optimization method is important, because it may increase stability, instability or a solution trapped in a local minimum.

Backpropagation learning algorithm (Rumelhart et al 1986) is the basic learning mechanism and it is very popular in the literature. In this algorithm, the network output, on presentation of input data, is compared with the desired output and a measure of the error is obtained. This error measure is then used to incrementally modify appropriate weights in the connection matrices in order to reduce the error. Following numerous presentations of training data, the overall error of the network is expected to be reduced to an acceptable level and the network has then learned how to solve the problem posed by training data. Example of the backpropagation learning algorithm will be given in the following section.

3.1.5 Adaptive networks

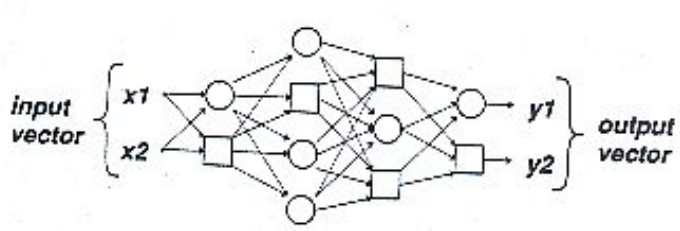

Figure 3.4 Adaptive network

An adaptive network (Figure 3.4) is an example of multilayer feedforward NN in which each node performs a particular function (node function) on incoming signals as well as a set of parameters pertaining to this node. The formulas for the node functions may vary from node to node, and the choice of each node function depends on the overall input-output function, which the adaptive network is required to carry out. The links in an adaptive network only

show the flow direction signals between nodes. No weights are associated with the links (Jang 1993).

The basic learning rule of adaptive networks is the backpropagation learning rule. However, since it is slow and tends to become trapped in local minima, a hybrid learning rule algorithm was proposed to speed up the learning algorithm by Jang in 1993.

3.1.5.1 Backpropagation of Adaptive Networks

Suppose that adaptive network in Figure 3.4 has *N* layers. (*j,i*) is the node in the *i*th position of the *j*th layer and $O_{j,i}$ is the node output. Since a node output depends on its incoming signal and its parameter set, it can be defined as follows:

$$O_{j,i} = f_{j,i}\left(O_i^{j-1},....O_{\#(j-1)}^{j-1}, a,b,c,.....\right) \tag{3.6}$$

a,b,c are the parameters pertaining to this node and f is the node function. Assuming the training data has P entries, an error measure can be defined for the *p*th ($1 \le p \le P$) entry of training data as the sum of the square errors and it is equal to:

$$E_p = \sum_{k=1}^{\#N}\left(d_k - O_{N,k}\right)^2 \tag{3.7}$$

where d_k is the *k*th component of the *p*th target vector, and $O_{N,k}$ is the *k*th component of the actual output vector produced by presentation of *p*th input vector. Therefore, the overall error measure can be expressed as:

$$E = \sum_{p=1}^{P} E_p \tag{3.8}$$

When E_p is equal to zero, the network is able to reproduce exactly the desired output vector in the *p*th training data pair. Therefore, aim is to minimize an overall measure. For this, first error rate $\frac{\partial E_p}{\partial O}$ for *p*th training data and for each node output should be calculated. The error rate for output node at layer N can be calculated from Equation 3.7 as:

$$\frac{\partial E_p}{\partial O_{i,p}^N} = -2\left(d_{i,p} - O_{i,p}^N\right) \tag{3.9}$$

For internal node at the *i*th position of layer j, the error rate can be derived by the chain rule of differential calculus as follows:

$$\frac{\partial E_p}{\partial O_{i,p}^j} = \sum_{k=1}^{\#j+1} \frac{\partial E_p}{\partial O_{k,p}^{j+1}} \frac{\partial O_{k,p}^{j+1}}{\partial O_{i,p}^j} \tag{3.10}$$

where ($1 \le j \le N-1$). That is, the error rate of an internal node at layer j can be expressed as a linear combination of the error rates of the layer j+1. Therefore, for all $1 \le j \le N$ and $1 \le i \le \#(j)$ error rates are found by applying Equation 3.9 and 3.10. The underlying procedure is called backpropagation since the error rates are obtained sequentially from the output layer back to the input layer. The gradient vector is defined as the derivative of the error measure with respect to α and equals to:

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha} \tag{3.11}$$

where S is the set of nodes whose outputs depend on α. Then, the derivative of the overall measure E with respect to α is written as:

$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha} \tag{3.12}$$

According to the simple steepest descent, the update formula for the generic parameter α is as follows:

$$\Delta\alpha = -\eta \frac{\partial E}{\partial \alpha} \tag{3.13}$$

in which η is a learning rate which can be further expressed as:

$$\eta = \frac{k}{\sqrt{\sum_{\alpha} \left(\frac{\partial E}{\partial \alpha} \right)^2}} \tag{3.14}$$

where k is the "step size", the length of each transition in the parameter space. It is observed that if k is small, the gradient method will closely approximate the gradient path, but convergence will be slow since gradient must be calculated many times. On the other hand, if k is large, convergence will initially be vary fast, but the algorithm will oscillate about the optimum. For this reason, it is proposed that if the error measure undergoes four consecutive reductions, k should be increased by 10%, if the error measure undergoes two consecutive combinations one increase and one reduction, it should be decreased by 10% (Jang 1993).

There are two learning paradigms for adaptive networks: offline learning (batch learning) and online learning (pattern learning). In offline learning, the update formula parameter α is based on Equation 3.12 and the update action takes place only after the whole training data set has been presented, i.e., only after each epoch. On the other hand, the parameters are updated immediately after each input-output pair has been presented in online learning, and the update formula is based on Equation 3.11 (Castillo and Melin 2001).

3.1.5.2 Hybrid Learning Rule: Offline Learning

Hybrid learning rule combines the gradient method and the least square estimate (LSE) to identify the parameters.

Assuming that the adaptive network has only one output, the output can be expressed as:

$$output = F(I, S) \tag{3.15}$$

where I is the set of input variables and S is the set of parameters. "If there exists a function H such that the composite function $H \circ F$ is linear some of the elements of S, these elements can be identified by the least square estimates." (Jang 1993). Hence parameter set S can be decomposed into two sets: S_1 and S_2 such that $H \circ F$ is linear in the elements of S_2. Then applying H to Equation 3.15, it becomes as follows:

$$output = H \circ F(I, S) \tag{3.16}$$

which is linear in the elements of S_2. Now given values of elements of S_1, P training data is plugged into Equation 3.15 and a matrix equation obtained:

$$AX = B \tag{3.17}$$

where X is an unknown vector whose elements are parameters in S_2. Since number of training data pairs is usually greater than number of linear parameters, this is an over-determined problem and there is no exact solution for Equation 3.17. To deal with this problem, a sequential method of LSE was proposed by Jang. According to this method X is calculated iteratively using the sequential formulas adopted in literature as:

$$\begin{aligned} X_{i+1} &= X_i + S_{i+1} a_{i+1} \left(b_{i+1}^T - a_{i+1}^T X_i \right) \\ S_{i+1} &= S_i - \frac{S_i a_i a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}}, \quad i = 0,1,....,P-1 \end{aligned} \tag{3.18}$$

where a_i^T is the *i*th row vector of matrix A defined in Equation 3.17 and b_i^T is the *i*th element of B. S_i is called the covariance matrix.

Each epoch of hybrid learning procedure consists of a forward pass and backward pass. In forward pass, input data and functional signals go forward are supplied to calculate each node output until the matrices A and B in Equation 3.17 are obtained. Then, the parameters in S_2 are identified by the sequential least squares formulas. After identifying parameters in S_2, the functional signals keep going forward till the error measure is calculated. In the backward pass, the error rates propagate from the output end toward the input end, and the parameters in S_1 are updated by the gradient method (Equation 3.13).

3.1.5.3 Hybrid Learning Rule: Online Learning

Online learning method is important for online parameter identification for systems with changing characteristics. The gradient descent is based on E_p instead of E in this method. A forgetting factor is added to the original sequential formula to decay the effects of the old data pairs as new data pairs become available. This approach gives higher factors to more recent data pairs and hence the time varying characteristics of the incoming data are accounted for. Original sequential formula for online learning is written as follows:

$$\begin{aligned} X_{i+1} &= X_i + S_{i+1} a_{i+1} \left(b_{i+1}^T - a_{i+1}^T X_i \right) \\ S_{i+1} &= \frac{1}{\lambda} \left[S_i - \frac{S_i a_i a_{i+1}^T S_i}{\lambda + a_{i+1}^T S_i a_{i+1}} \right] \end{aligned} \tag{3.19}$$

where λ is between 0 and 1. The smaller λ is, the faster effects of old data decay.

3.2 Fuzzy Logic Systems

This section presents general information about the theory of fuzzy logic. Definition of a fuzzy set and linguistic variable conception are presented. The

meaning of a fuzzy rule is explained and some rule examples are given. Fuzzy reasoning mechanism and fuzzy inference systems are also presented.

3.2 .1 Fuzzy set

A "fuzzy set" is a simple extension of the definition of a classical set in which the characteristic function is permitted to have any values between 0 and 1 (Castillo and Melin 2000). A "fuzzy set" A in X can be defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \tag{3.20}$$

where $\mu_A(x)$ is called membership function for the fuzzy set A. It maps each x to a membership grade between 0 and 1. Examples of membership functions (Triangular, Trapezoidal and Gaussian) can be seen in Figure 3.5 and described with the following formulas:

Triangular MFs:

$$triangle(x;a,b,c) = \begin{cases} 0, & x \le a \\ (x-a)/(b-a), & a \le x \le b \\ (c-x)/(c-b), & b \le x \le c \\ 0, & c \le x \end{cases} \tag{3.21}$$

Trapezoidal MFs:

$$trapezoidal(x;a,b,c,d) = \begin{cases} 0, & x \le a \\ (x-a)/(b-a), & a \le x \le b \\ 1, & b \le x \le c \\ (d-x)/(d-c), & c \le x \le d \\ 0, & d \le x \end{cases} \tag{3.22}$$

Gaussian MFs:

$$gaussian(x;c,\sigma) = e^{\frac{-(x-c)^2}{2\sigma}} \tag{3.23}$$

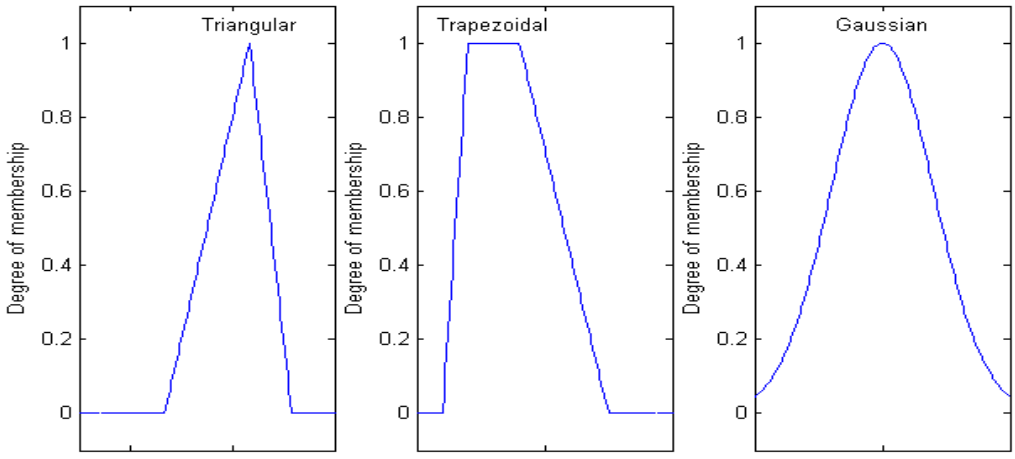

Figure3.5 Examples of membership functions

3.2 .2 Linguistic Variables

The concept of linguistic variables was introduced by Zadeh (1973) to provide a basis for approximate reasoning. A linguistic variable was defined as a variable whose values are words or sentences. For instance, Age can be linguistic variable if its values are linguistic rather than numerical, i.e., young, very young, old, very old, etc., rather than 20, 21, 23, 45…. Figure 3.6 illustrates the term set Age expressed by the Gaussian MFs.

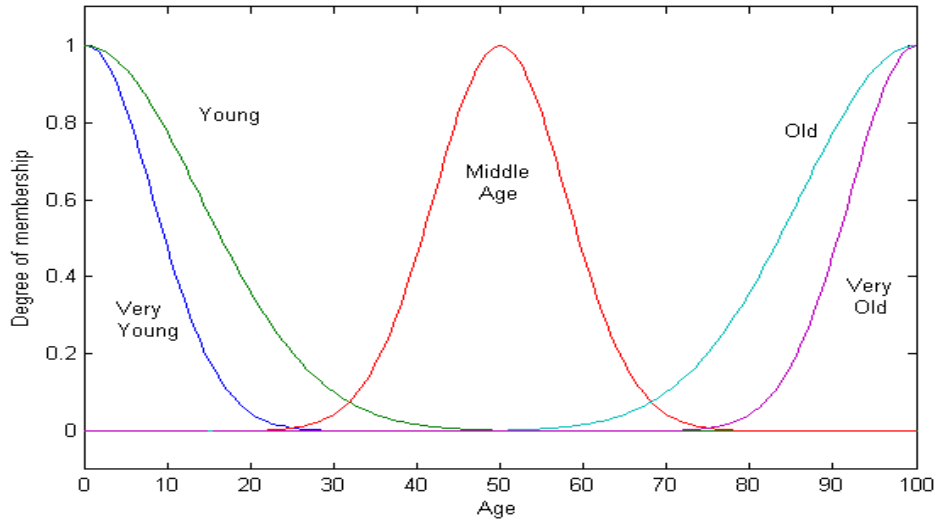

Figure3.6 Membership functions of the term set Age

3.2 .3 Fuzzy if then Rules

A fuzzy if-then rule (fuzzy rule, fuzzy implication, or fuzzy conditional statement) is expressed as follow:

If x is A **then** y is B

where A and B linguistic values defined by fuzzy sets. "x is A" is called "antecedent" or "premise", while "y is B" is called the "consequence" or "conclusion" (Castillo and Melin 2000). Some of the if-then rule examples can be given below:

- If pressure high, then volume is small.
- If the speed is low AND the distance is small, then the force on brake should be small.

3.2 .4 Fuzzy Reasoning

Fuzzy reasoning, approximate reasoning, is an inference procedure whose outcome is conclusion for a set of fuzzy if-then rules. The steps of fuzzy reasoning can be given as follows:

1. "Input variables are compared with the MFs on the premise part to obtain the membership values of each linguistic label (fuzzification).
2. The membership values on the premise part are combined through specific fuzzy set operations such as: min, max, or multiplication to get firing strength (weight) of each rule.
3. The qualified consequent (either fuzzy or crisp) is generated depends on the firing strength.

4. The qualified consequents are aggregated to produce crisp output according to the defined methods such as: centroid of area, bisector of area, mean of maximum, smallest of maximum and largest of maximum (deffuzification) (Jang 1993)".

3.2 .5 Fuzzy Systems

Fuzzy systems are made of a knowledge base and reasoning mechanism called fuzzy inference engine. The structure of fuzzy inference engine is shown in Figure 3.7. A fuzzy inference engine combines fuzzy if-then rules into a mapping from the inputs of the system into its outputs, using fuzzy reasoning methods (Czogala and Leski 2002). That is, fuzzy systems represents nonlinear mapping accompanied by fuzzy if-then rules from the rule base. Each of these rules describes the local mappings. The rule base can be constructed either from human expert or automatic generation that is extraction of rules using numerical input-output data.

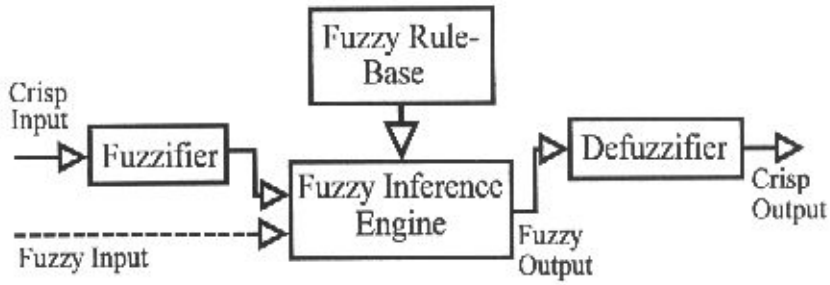

Figure3.7 Fuzzy Inference Engine

Mamdani and Takagi-Sugeno fuzzy systems are the examples of fuzzy inference systems. Mamdani fuzzy inference system was first used to control a steam engine and boiler combination by a set of linguistic rules obtained from human operators (Mamdani and Assilian 1975). Figure 3.8 illustrates how a two rule Mamdani fuzzy inference system derives the overall output z when

subjected to two numeric inputs x and y. Takagi-Sugeno fuzzy inference system was first introduced by Takagi and Sugeno (1985). The difference of Takagi-Sugeno model is that each rule has a crisp output, and the overall output is determined as weighted average of single rules output. This type of fuzzy inference system is shown in Figure 3.9

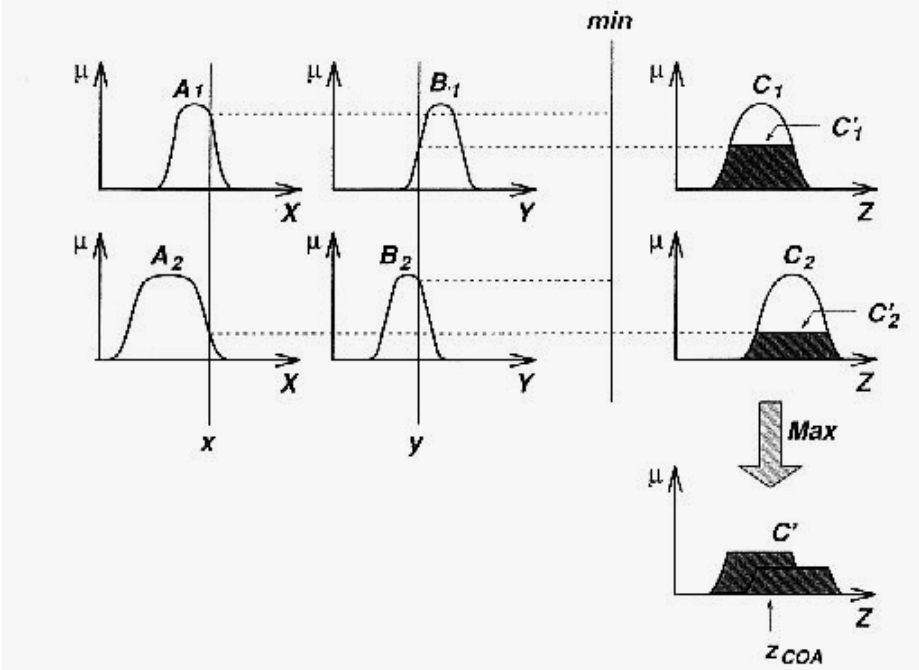

Figure3.8 Mamdani Fuzzy Inference System

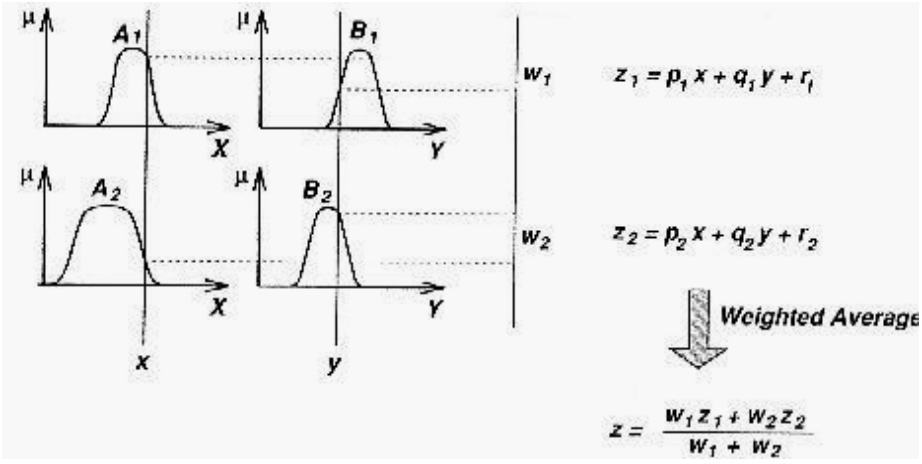

Figure3.9 Takagi-Sugeno Fuzzy Inference System

3.3 ANFIS

As previously stated, ANFIS is an adaptive network that is functionally equivalent to fuzzy inference system (Jang 1993), and referred in literature as "adaptive network based fuzzy inference system" or "adaptive neuro fuzzy inference sytem". In this section, architecture of ANFIS will be presented.

3.3.1 ANFIS architecture

In ANFIS, Takagi-Sugeno type fuzzy inference system is used. The output of each rule can be a linear combination of input variables plus a constant term or can be only a constant term. The final output is the weighted average of each rule's output. Basic ANFIS architecture that has two inputs x and y and one output z is shown in Figure 3.10. The rule base contains two Takagi-Sugeno if-then rules as follows:

Rule1: If x is A_1 and y is B_1, then $f_1 = p_1 x + q_1 y + r_1$

Rule2: If x is A_2 and y is B_2, then $f_2 = p_2 x + q_2 y + r_2$

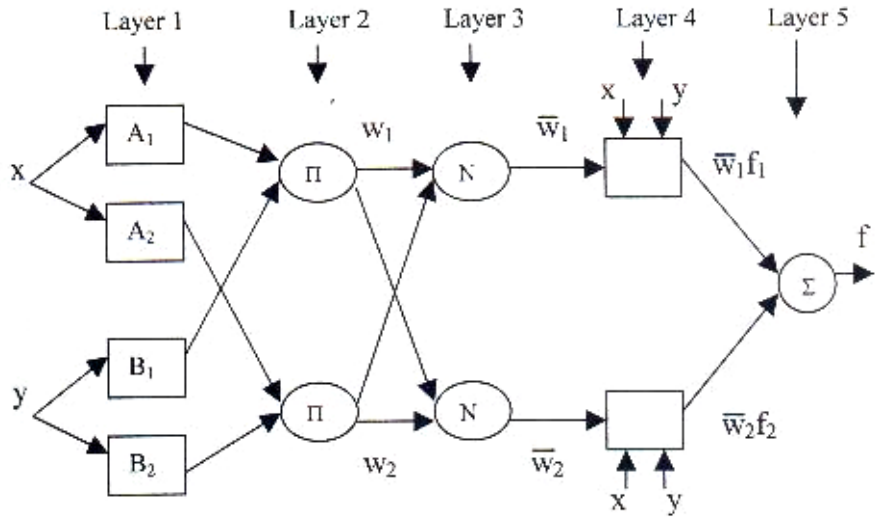

Figure3.10 Basic structure of ANFIS

The node functions in the same layer are the same as described below:

Layer 1: Every node i in this layer is a square node with a node function as:

$$\begin{aligned} 0_{1,i} &= \mu_{A_i}(x), \quad for\ i = 1,2 \\ 0_{1,i} &= \mu_{B_i - 2}(y), \quad for\ i = 1,2 \end{aligned} \tag{3.24}$$

where x is the input to node *i*, and A_i (or B_{i-2}) is a linguistic label (such as "small" or "large") associated with this node. In other words, $0_{l,i}$ is the membership grade of a fuzzy set A and it specifies the degree to which the given input x satisfies the quantifier A. The membership function for A can be any appropriate membership function, such as the Triangular or Gaussian. When the parameters of membership function changes, chosen membership function varies accordingly, thus exhibiting various forms of membership functions for a fuzzy set A. Parameters in this layer are referred to as "premise parameters".

Layer 2: Every node in this layer is a fixed node labeled as Π, whose output is the product of all incoming signals:

$$0_{2,i} = w_i = \mu_{A_i}(x)\mu_{B_i}(y), \quad i = 1,2 \tag{3.25}$$

Each node output represents the firing strength of a fuzzy rule.

Layer 3: Every node in this layer is a fixed node labeled N. The *i*th node calculates the ratio of the rule's firing strength to the sum of all rules' firing strengths:

$$0_{3,i} = w_i = w_i / (w_1 + w_2), \quad i = 1,2 \tag{3.26}$$

Outputs of this layer are called "normalized firing strengths".

Layer 4: Every node i in this layer is an adaptive node with a node function as:

$$0_{4,i} = \overline{w_i} f_i = \overline{w_i}\left(p_i x + q_i y + r_i\right) \tag{3.27}$$

where $\overline{w_i}$ is a normalized firing strength from layer 3 and $\{p_i, q_i, r_i\}$ is the parameter set of this node. Parameters in this layer are referred to as "consequent parameters".

Layer 5: The single node in this layer is a fixed node labeled Σ that computes the overall output as the summation of all incoming signals:

$$overall\ output = 0_{5,i} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f}{\sum_i w_i} \tag{3.28}$$

Thus an adaptive network, which is functionally equivalent to the Takagi-Sugeno type fuzzy inference system, has been constructed. Other example of ANFIS with nine rules can be shown in Figure 3.11. Three membership functions are associated with each input, so the input space partitioned into nine fuzzy subspaces. The premise part of a rule describes a fuzzy subspace, while the consequent part specifies the output within this fuzzy subspace.

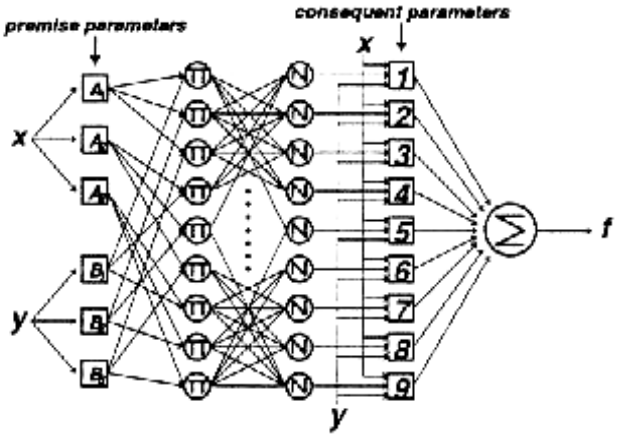

Figure3.11 ANFIS Architecture with nine rules

3.3.2 ANFIS Learning algorithm

From the proposed ANFIS architecture above (Figure 3.11), the output f can be defined as:

$$
\begin{aligned}
f &= \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \\
f &= \overline{w_1}\left(p_1 x + q_1 y + r_1\right) + \overline{w_2}\left(p_2 x + q_2 y + r_2\right) \\
f &= \left(\overline{w_1} x\right) p_1 + \left(\overline{w_1} y\right) q_1 + \left(\overline{w_1}\right) r_1 + \left(\overline{w_2} x\right) p_2 + \left(\overline{w_2} y\right) q_2 + \left(\overline{w_2}\right) r_2
\end{aligned}
\tag{3.29}
$$

where p_1, q_1, r_1, p_2, q_2 and r_2 are the linear consequent parameters. The methods for updating the parameters are listed as below:

1. *Gradient decent only*: All parameters are updated by gradient decent backpropagation.

2. *Gradient decent and One pass of Least Square Estimates (LSE)*: The LSE is applied only once at the very beginning to get the initial values of the consequent parameters and then the gradient descent takes over to update all parameters.

3. *Gradient and LSE*: This is the hybrid learning rule.

Since the hybrid learning approach converges much faster by reducing search space dimensions than the original backpropagation method, it is more desirable. In the forward pass of the hybrid learning, node outputs go forward until layer 4 and the consequent parameters are identified with the least square method. In the backward pass, the error rates propagate backward and the premise parameters are updated by gradient descent.

CHAPTER 4

ANFIS DESIGN AND CASE STUDIES

In this study, as a continuation of previous studies done by Bahar (2003), and Yıldız (2003), the aim is to use the ANFIS methodology, a hybrid structure, in the estimation of compositions using tray temperatures in continuous and batch distillation columns. The performance of the ANFIS estimator is compared with the performance of the NN and Extended Kalman Filter (EKF) estimators. In this chapter, the design of ANFIS architecture for the estimation and control purposes will be explained. Also, the case studies used for the applications will be presented.

4.1 Design of ANFIS

The basic idea behind the neuro-adaptive learning techniques is very simple. These techniques provide a method for the fuzzy modeling procedure to learn information about data set, in order to compute the membership function parameters that best allow the associated fuzzy inference system to track the given input-output data. ANFIS constructs an input-output mapping based on both human knowledge (in the form of fuzzy if-then rules) and simulated input-output data pairs. It serves as a basis for building the set of fuzzy if-then rules with appropriate membership functions to generate the input output pairs.

The parameters associated with the membership functions are open to change through the learning process. The computation of these parameters (or their adjustment) is facilitated by a gradient vector, which provides a measure of how well the ANFIS is modeling the input output data for a given parameter set. Once the gradient vector is obtained, backpropagation or hybrid learning algorithm, described in the previous chapter, can be applied in order to adjust the parameters.

As stated previously, ANFIS can be used in modeling, estimating and controlling studies in chemical engineering processes similar to other artificial intelligence methods such as NNs and Fuzzy Logic (FL). In this work, the designed ANFIS is utilized as an estimator and a controller. Estimation is done for compositions from the temperature measurements in continuous and batch distillation columns whereas adaptive control strategy that needs no separate process network model with ANFIS controller is utilized in a pH system for set point tracking problem.

4.1.1 ANFIS as an Estimator

ANFIS can be used for the estimation of some dependent variables in chemical process. The designed ANFIS estimator is used to infer the compositions from measurable tray temperatures in batch and continuous distillation columns. Estimation scheme is shown in Figure 4.1. In estimator design process, different ANFISs are constructed and trained to find the architecture that gives the best performance as an estimator.

In order to design an estimator, first, training data sets should be generated to train the estimator networks. These data sets consist of estimator inputs and desired output values. They are produced from the process input-

output data. Since, ANFIS is a data processing method, it is important that the input-output data must be within the sufficient operational range including the maximum and minimum values for both input and output variables of the system. If this is not provided, estimator performance cannot be guaranteed and thus the designed estimator will not be accurate. Having generated the training data, estimators that have different architectures are trained with the obtained data sets.

Performances of the trained estimators are evaluated with model simulations and best estimator architecture is obtained. These simulations are made to verify and to generalize the ANFIS structures. Verification is done to show how good the estimator structure learned the given training data. This is carried out by simulating the column models with specific initial process inputs used in obtaining training data sets. Generalization capabilities of the estimators are found with other simulations in which input process variables are in operational range but not used in training data formation.

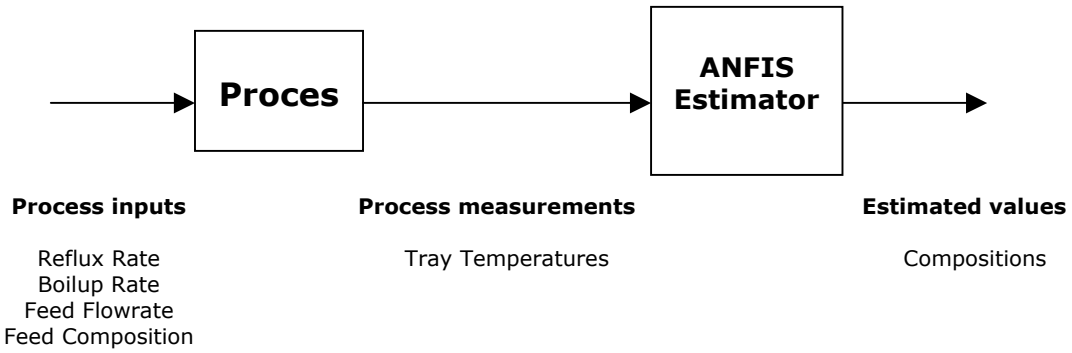

Figure 4.1 Estimation using ANFIS estimator

ANFIS estimator design consists of two parts: constructing and training. In constructing part, structure parameters are determined. These are type and number of input Membership Functions (MFs), and type of output MF. Any of several MFs such as Triangular, Trapezoidal and Gaussian can be used as an

input MF. Frequently used MFs in literature are the Triangular and Gaussian. For this reason, they are chosen as input MF type in this study. Number of MFs on each input can be chosen as 3, 5, and 7 to define the linguistic labels significantly. Effective partition of the input space is important and it can decrease the rule number and thus increase the speed in both learning and application phase. Output MFs can be either a constant or in linear form. Both of these two forms are used for the output MF in this study. Having described the number and type of input MFs, the estimator rule base is constituted. Since, there is no standard method to utilize the expert knowledge; automatic rule generation (grid partition) method is usually preferred (Castillo and Melin 2000). According to this method, for instance, an ANFIS model with two inputs and three MFs on each input would result in 3^2=9 Takagi-Sugeno fuzzy if-then rules automatically. Although this method can require much computational knowledge especially in systems that have to be defined with many inputs, it is used in this study due to advantage of MATLAB software. Therefore, rule bases of the estimators are formed automatically with the number of inputs and number of MFs. After the ANFIS structure is constructed, learning algorithm and training parameters are chosen. As mentioned in the previous chapter, backpropagation or hybrid learning can be used as a learning algorithm. The hybrid learning algorithm is used in this study. Parameters in the algorithm are epoch size (presentation of the entire data set), error tolerance, initial step size, step size decrease rate, and step size increase rate. Since there is no exact method in literature to find the optimum of these parameters a trial and error procedure is used. In all trainings, they are taken as 10, $1x10^{-5}$, 0.01, 0.9, and 1.1, respectively as default constant value as proposed in MATLAB.

MATLAB fuzzy logic toolbox is used to design ANFIS estimators' structures. Using the given training data set, the toolbox constructs an ANFIS structure using either a backpropagation algorithm alone, or in combination with least

squares type of method (hybrid algorithm). ANFIS model can be generated either from the command line, or through the ANFIS editor GUI. In this study, ANFIS Editor GUI is used to generate the ANFIS models with the chosen design parameters in construction phase. Written MATLAB code is used to train the ANFIS structure in the training step. This code is given Appendix B.1. The use of the ANFIS editor GUI can be found in program help files.

The steps in ANFIS estimator design in this study utilizing the MATLAB fuzzy logic toolbox are as follows:

1. Generated training data is loaded to the Editor GUI.

2. Design parameters, number of input MF, type of input and output MF, are chosen. Thus, initial ANFIS structure is formed.

3. The code for the training is run with the initial structure.

4. ANFIS structure constituted after training is saved to use as an estimator.

4.1.2 ANFIS as a Controller

In the literature, approaches of using NNs for the control of nonlinear processes literature are usually model based, i.e., they have one NN for modeling the dynamics of the process and a second NN that acts as the controller. However, NN structures are not generally mean. A network consists of several connection weights. Therefore, any adaptive model-based control scheme will have to deal with updating a very large number of weights. The iterative inversion of the forward model in a Model Predictive Control (MPC) scheme and the online adaptation of controller network parameters in an adaptive control scheme require considerable computational power and the

convergence of such schemes within a sampling interval is not guaranteed. Hence, the scope of applications using such control schemes is severely limited in practice (Jutan and Krishnapura 2000).

The original purpose of fuzzy logic control, proposed by Mamdani in 1975, is to mimic the behavior of a human operator able to a control complex plant satisfactorily. When a fuzzy controller is constructed, "knowledge acquisition", which takes a human operator's knowledge and generates fuzzy if-then rules is needed to perform as the backbone for a fuzzy controller that behaves like the original human operator. Usually, "linguistic and numerical" information are the types of information from a human operator. An experienced human operator usually summarizes his or her reasoning processes to find the final control decisions as a set of fuzzy if-then rules with imprecise but roughly correct MFs. These MFs are obtained with a certain amount of trial and error plus a lengthy interview process. This corresponds to linguistic information. But, it is possible to obtain data observed by human and human's corresponding actions as a set of desired input-output data pairs. Next, this data can be used as training data in constructing a fuzzy controller. Before the Neuro-Fuzzy approaches, most controller design methods used only the linguistic information to build a fuzzy controller. Manual trial and error processes were involved to fine tune the MFs. Therefore, linguistic information can be used to identify the structure of the fuzzy controller, and then numerical information can be used to identify the parameters such that the fuzzy controller can reproduce the desired action more accurately.

To deal with the problems mentioned above in control applications, ANFIS can be used. An adaptive Neuro-Fuzzy controller with a small number of weights can be designed by using the ANFIS architecture. The ANFIS structure, with very few weights, can overcome the problem of excessive tuning parameters and the

need for modeling of the process by a separate network model such as NN, fuzzy or ANFIS.

Inverse learning is one of the methods of designing Neuro-Fuzzy controllers. It involves two phases: learning and application phases. In the learning phase training set is obtained by generating inputs randomly, and observing the corresponding outputs produced by the plant. In the application phase, the ANFIS identifier is copied to the ANFIS controller for generating the desired output. Learning phase and application phase of inverse learning are shown in Figures 4.2 and 4.3, respectively. This method seems straightforward and only one learning task is needed to find the inverse of the plant. It assumes the existence of the inverse plant, which is not valid in general. Minimization of the network error does not guarantee minimization of the overall system error. However, inverse learning is an indirect approach that tries to minimize the network output error instead of overall system error (defined as the difference between desired and actual trajectories). Instead of this method, "specialized learning" (illustrated in Figure 4.4) can be used as an alternative that tries to minimize the system error directly by backpropagating error signals through the plant block. In order to backpropagate the error signals through the plant, a model representing the behavior of the plant is needed. In other words, Jacobian of the plant, $\frac{\partial y}{\partial u}$, is required. It can be estimated online from the changes of plant's inputs and outputs. The desired behavior of the overall system can also be implicitly specified by a (usually linear) model that is able to achieve the control goal satisfactorily. In literature, Yamamura et. al. (1988) used an iterative approach to evaluate the plant Jacobian information. Ydtsie (1990) also used a similar approach, called one-step ahead indirect adaptive control, with a single linear neuron in the output layer and many nonlinear neurons in the hidden layer of the network. Saerens and Soquet (1991) used the sign of the

process gain to approximate the process Jacobian. Juan and Krishnapura (2000) proposed the use of sigmoidal function that has the property of having continuous derivatives to approximate the plant Jacobian. Their approach of using the sigmoidal function to represent the unmodifaible process layer in controller process network gives an approximation to both the process gain as well as its sign.

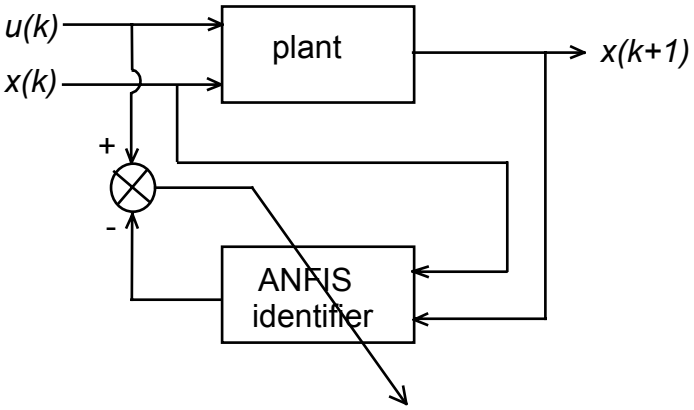

Figure4.2 Learning phase of the Inverse Learning

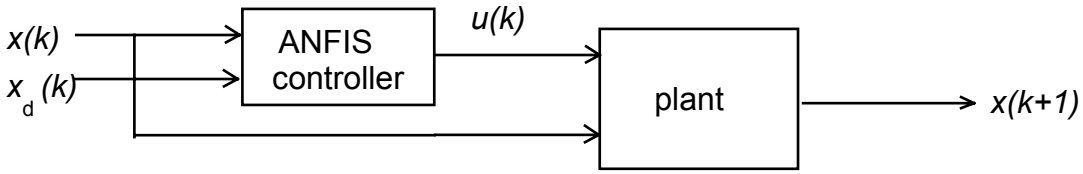

Figure 4.3 Application phase of the Inverse Learning

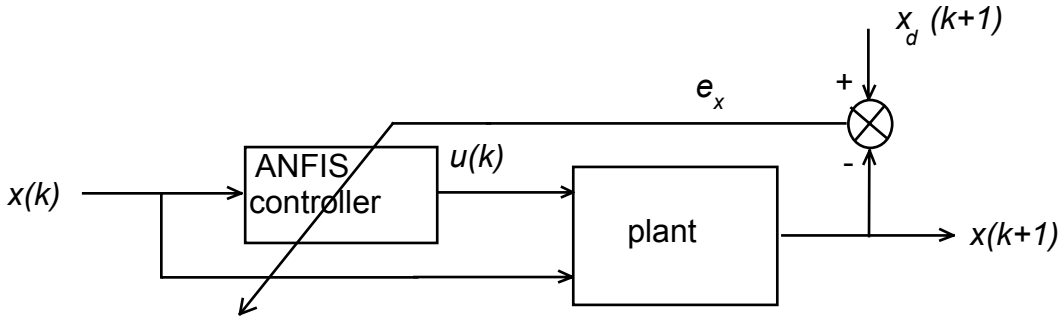

Figure 4.4 Specialized Learning

In the control part of this study, specialized learning algorithm shown in Figure 4.4 is used in pH system to see the performance of the ANFIS as a controller. Therefore, a simple ANFIS controller designed. The sigmoidal function approximation proposed by Jutan and Krishnapura (2000) is used to approximate the plant Jacobian.

When designing the ANFIS controller, first, inputs to the controllers are decided. Two inputs are chosen. They are the controller and process outputs at the previous sampling instant. These are base flow rate and pH, respectively. Since Gaussian MF has two parameters, it is chosen as an input MF. Three MFs labeled with the Small (S), Medium (M) and Big (B) as linguistic variables are used for input MFs. Output membership function type is chosen as constant form explained in detail in Chapter 3.3. Thus, ANFIS controller, with two inputs and three MFs on each input, has nine control rules in the controller rule base due to the grid partition method mentioned in previous section. Generated ANFIS controller structure is given in Figure 3.11. In the controller, total number of adjusted parameters is 21. Since each input is graded with three MFs and one of them consists of two parameters, twelve of adjusted parameters come from the premise part of the controller network. Consequent part of the controller consists of nine parameters due to the nine rules with constant output. Controller parameters are adapted at each sampling instant. A code for simple backpropagation algorithm is written to update the parameters of the ANFIS controller in closed-loop system simulation.

The performance error for the pH system is $e = \left(pH_d - pH\right)$. The cost (error) function to be optimized is defined as follows:

$$E = \frac{1}{2}e^2 = \frac{1}{2}\left(pH_d - pH\right)^2 = E(\alpha) \tag{4.1}$$

where α is one of the controller parameters. The calculation of the cost function gradient $\frac{\partial E}{\partial \alpha}$ is done by applying the chain rule to optimize the parameter α as

$$\frac{\partial E}{\partial \alpha} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial pH}\frac{\partial pH}{\partial u}\frac{\partial u}{\partial \alpha} \tag{4.2}$$

$$\frac{\partial E}{\partial \alpha} = -\left(pH_{d} - pH\right)\frac{\partial pH}{\partial u}\frac{\partial u}{\partial \alpha} \tag{4.3}$$

where $\frac{\partial pH}{\partial u}$ is the plant Jacobian. To minimize the error, the parameter change should be in the negative gradient direction. Therefore,

$$\Delta\alpha = -\eta\frac{\partial E}{\partial \alpha} \tag{4.4}$$

Adaptation formula for the parameter α is calculated as

$$\alpha(k+1) = \alpha(k) + \Delta\alpha \tag{4.5}$$

$$\alpha(k+1) = \alpha(k) - \eta\frac{\partial E}{\partial \alpha} \tag{4.6}$$

All parameters in the controller are adapted according to Equation 4.6.

4.2 Case Studies

In this section, the cases in which ANFIS methodologies are implemented will be given. In the first part, the multicomponent industrial continuous distillation column will be described. The case of batch distillation column will be given in the second part. In the third part of this section, the pH system will be presented.

4.2.1 Industrial Continuous Distillation Column

The industrial continuous distillation column used in this study is C_3-C_4 Splitter column. This column exists in Kırıkkale refinery, in Turkey. Figure 4.5 presents the sketch of the column. A mixture of propane, i-butane, n-butane, and i-pentane enters the column from the 22^{nd} tray and propane and n-butane are separated as a top and bottom product, respectively. Top product composition purity is controlled by manipulating the reflux flow rate. The bottom product purity is controlled by measuring the temperature of the bottoms and manipulating the steam flow rate to the reboiler. Liquid heights in the column bottom and receiver drum are controlled by adjusting the bottoms and liquid distillate flow rates, respectively. The pressure in the column is controlled by manipulating the overhead vapor flow rate to the receiver drum. Design parameters and the operating data of the column are given in Table 4.1.

The unsteady state simulation program for the column mentioned above was first written in FORTRAN by Alkaya (1990). Then, it was improved by Kaya (2000) and modified by Dokucu (2002). Bahar (2003) adapted the simulation code to MATLAB software without changing the main algorithm. Details of the algorithm of the unsteady state model can be found in the study of Dokucu (2002).

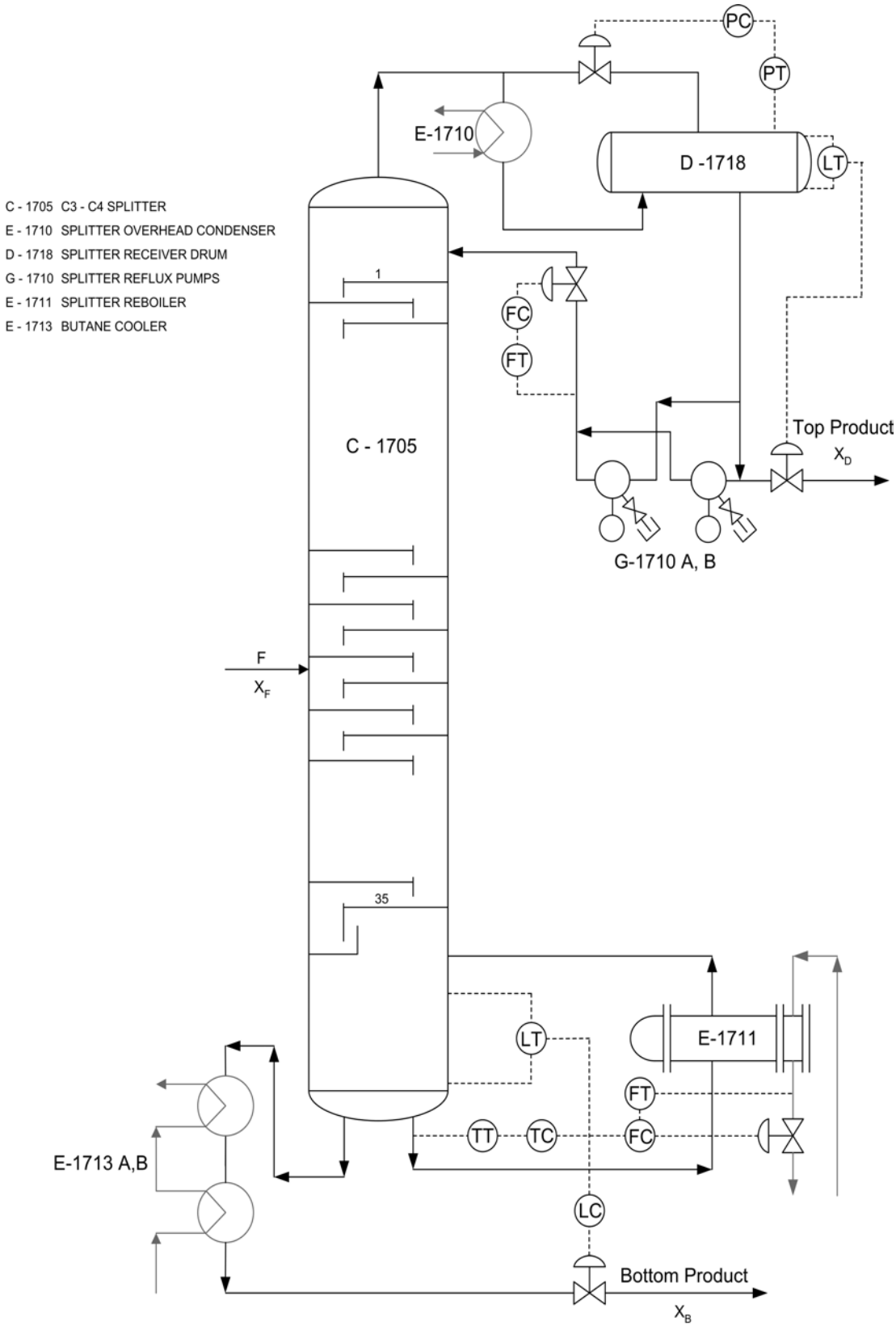

Figure 4.5 C3-C_4 Splitter column (Dokucu 2002)

Table 4.1 Plant Data (Dokucu 2002)

Column Specifications	
Number of Trays	35
Column Inside Diameter	1000 mm
Tray Spacing	600 mm
Weir Length	880 mm
Weir Height	50 mm
Maximum Capacity (% of nominal design)	110 %
Minimum Capacity (% of nominal design)	50 %
Feed Condition	
Feed Rate	118.53 kmol/hr
Feed Pressure	18.04 bar
Feed Temperature	84 ^{0}C
Feed Composition (mole fraction)	
Propane	0.3933
i-Butane	0.2384
n-Butane	0.3678
i-Pentane	0.0005
Operational Values (Design)	
Maximum Pressure Drop for One Tray	5 mm Hg
Top Tray Pressure	16.18 bar
Bottom Tray Pressure	16.67 bar
Top Tray Temperature	48 ^{0}C
Bottom Tray Temperature	98 ^{0}C
Reflux Rate	161.51 kmol/hr
Distillate Rate	32.76 kmol/hr
Bottoms Rate	85.77 kmol/hr
Reboiler Duty	1930 MW

4.2.2 Batch Distillation Column

The batch distillation column simulated by Mujtaba et. al. (1993) is used in the second study. Figure 4.6 illustrates the sketch of the batch column. This column separates a mixture of cyclo-hexane, n-heptane and toluene. Design parameters for the case column is given in Table 4.2. The column is under the perfect control of reflux drum level and reflux ratio (R) is used as the manipulated variable in order to realize the optimal operation policy recommended by Mujtaba et. al. (1993). In this policy, a switching time between R and shortcuts was optimized according to the minimization of the capacity factor. Table 4.3 shows the parameters of the optimal reflux policy profile given by Mujtaba et. al. (1993).

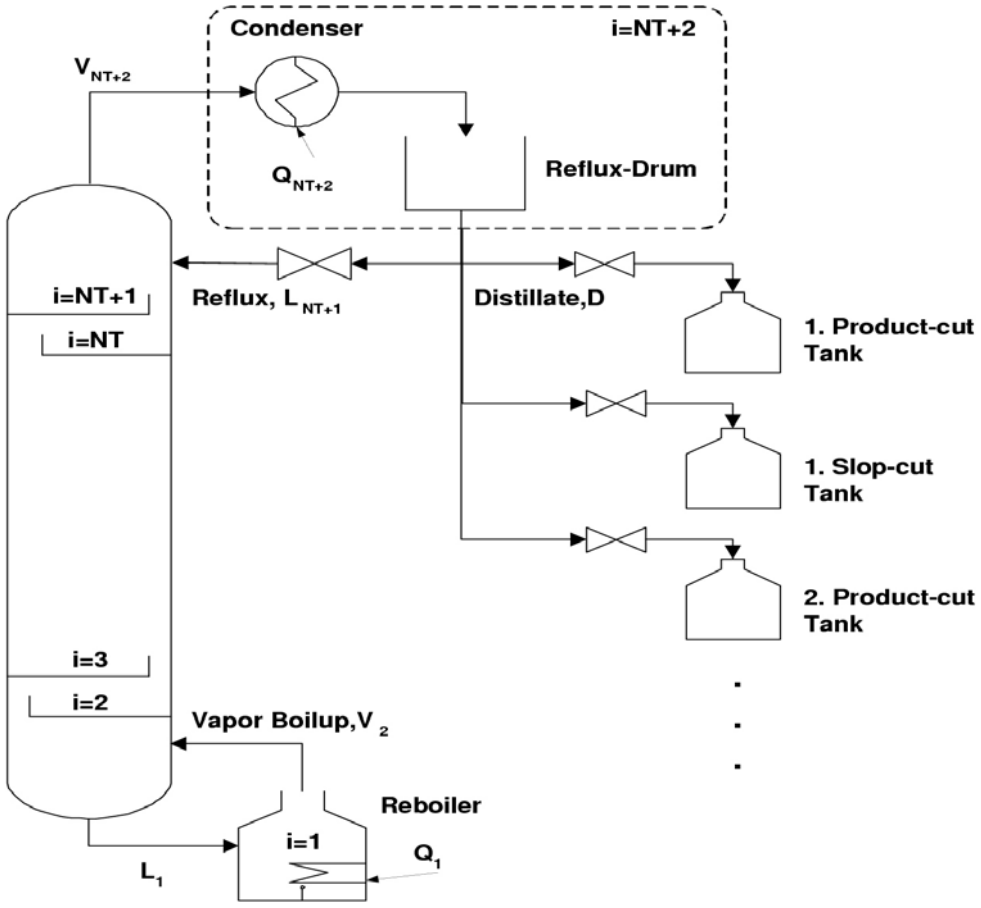

Figure 4.6 Batch Distillation Column (Yıldız 2003)

Table 4.2 Design parameters for the batch distillation column (Yıldız 2003)

Number of trays	8
Condenser-Reflux-Drum Holdup	0.02 kmol
Trays Holdup	0.01 kmol
Maximum Boil-up rate	2.75 kmol/h

Table 4.3 Optimal reflux ratio policy parameters (Yıldız 2003)

Amount of fresh feed		2.93 kmol
Feed composition:		
cyclo-hexane		0.407
n-heptane		0.394
toluene		0.199
Desired purity of comp. 1 in Product-cut 1		0.9
Desired purity of comp. 2 in Product-cut 2		0.8
Optimum Reflux Profile		
Time Interval (hour)	*Reflux Ratio*	
0-2.04	0.875	
2.04-3.4	0.911	
3.4-6.17	0.933	
6.17-6.51	0.831	
6.51-8.35	0.876	

The simulation code for the column was written in MATLAB by Yıldız (2003). He developed a rigorous batch distillation column model and tested the validity of the model by using operating conditions of the column given in literature. Assumptions made in model development are negligible vapor holdup, constant volume of tray liquid holdups, negligible fluid dynamics lags, adiabatic operation and ideal trays. Details of the model development and simulation code of the system can be found in the study of Yıldız (2003).

4.2.3 pH System

The pH system was first studied by Sain (1989). Sain investigated the effect of large dead time in the measuring line on the control performance of different controller tuning methods in her study. Then, Kaffashi (1998) and Nasrellah (1998) applied the intelligent process control techniques. Akbay (2002)

also studied the system experimentally using sliding mode and fuzzy sliding mode control techniques.

The pH system under study can be shown schematically in Figure 4.7. The acetic acid (CH_3COOH) stream (process stream), with concentration c_1, enters the CSTR at a flow rate of F_1 while the sodium hydroxide (NaOH) base stream (titrating stream) of concentration c_2, flows into reactor at a rate of F_2. The reactor has a constant volume V and the mixture in the reactor is assumed to be perfectly. The process output being measured is the pH of the exiting stream. The objective of the system is to control of the effluent pH by manipulating the base stream flow, F_2, into the system. x_1 and x_2 are the total ion concentrations of the acetate and sodium respectively.

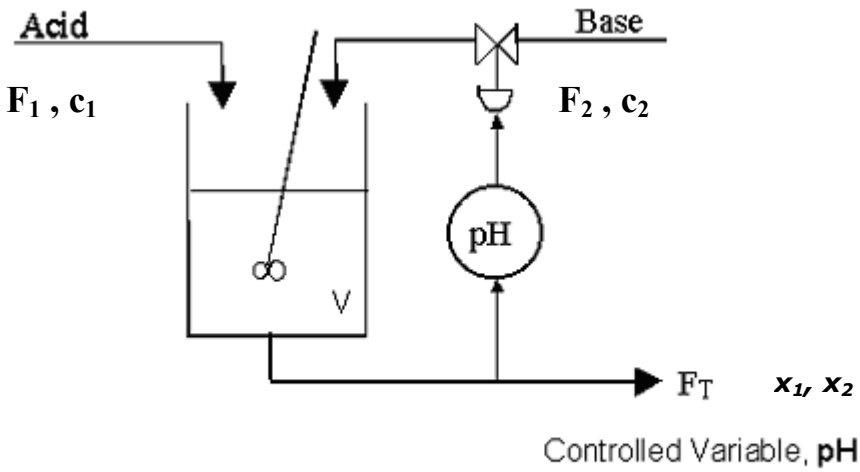

Figure 4.7 pH reactor

In order to model this simple dynamic pH system, a method proposed by Wright and Kravaris (1991) is used in this study. According to the Wright and Kravaris, the material balances around the dynamic CSTR system with the assumptions of constant volume and perfect mixing are of the form:

$$V\frac{dx_i}{dt} = F(c_i - x_i) + (\alpha_i - x_i)u \qquad i == 1,....., n \tag{4.7}$$

where x_i = total ion concentration of the *i*th acidic or basic species in the effluent stream (state variables), F = flow rate of the process stream, V = volume of the CSTR, α_i = total ion concentration of *i*th species in process stream, and u = flow rate of titrating stream (manipulated variable). Equation 4.6 forms the state model for the system. The pH is calculated from the following pH Equation:

$$\sum_{i=1}^{n} a_i(pH)x_i + A(pH) = 0 \tag{4.8}$$

where pH =–log[H^+], $a_i(pH)$ are weighting factors, and $A(pH)$ is given by the following equation:

$$A(pH) = 10^{-pH} - K_w 10^{pH} \tag{4.9}$$

where K_w is the water equilibrium constant. Equations 4.7 and 4.8 constitute a general mathematical model for any pH process. The discrete forms of these equations are:

$$(x_i)_{k+1} = (x_i)_k \exp\left[-\left(\frac{F_k + u_k}{V}\right)\Delta t\right] - \frac{F_k(c_i)_k + u_k\alpha_i}{F_k + u_k}\left\{1 - \exp\left[-\left(\frac{F_k + u_k}{V}\right)\Delta t\right]\right\} \tag{4.10}$$

$$\sum_{i=1}^{n} a_i(pH_{k+1})(x_i)_{k+1} + A(pH_{k+1}) = 0 \tag{4.11}$$

In the pH system under study, consisting of acetic acid being titrated by sodium hydroxide, the total acetate (x_1) and total sodium (x_2) concentrations are defined as follows:

$$x_1 = [CH_3COOH^-] + [CH_3COOH] \tag{4.12}$$

$$x_2 = [Na^+] \tag{4.13}$$

The acetate balance and sodium balance can be given by the following equations:

$$V\frac{dx_1}{dt} = F_1 c_1 - (F_1 + F_2)x_1 \tag{4.14}$$

$$V\frac{dx_2}{dt} = F_2 c_2 - (F_1 + F_2)x_2 \tag{4.15}$$

Figure 4.8 represents steady state gain variation in the pH to changes in base stream flow rate of the system, with the acid stream flow rate held constant. CSTR parameters used are taken the study of McAvoy (1990).

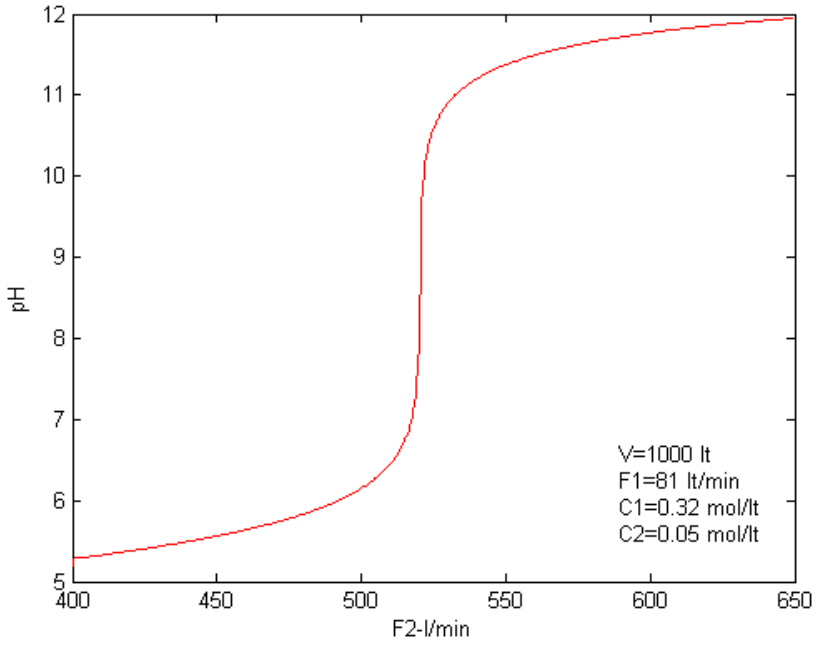

Figure4.8 Steady state pH gain curve

CHAPTER 5

RESULTS AND DISCUSSION

In this study, the aim is to design ANFIS architectures for chemical processes in estimation problems and to see their performances. For this purpose, ANFIS architecture is applied to batch and continuous distillation columns to estimate the compositions from measured tray temperatures. It is also tried as a controller in pH reactor control system.

In this chapter, simulation results will be given and be discussed in detail. Design and implementations of ANFIS estimators will be demonstrated. Controller performance of ANFIS for set point tracking in pH reactor will also be presented.

5.1 Estimation in Continuous Distillation Column

In the continuous distillation column for estimation purposes, two parallel ANFIS estimators are designed to estimate the top (propane) and bottom (butane) product compositions from tray temperatures. Static mapping of the compositions from the temperatures are achieved using the ANFIS estimators. This is performed in four phases. In the first phase, estimators' inputs are selected. Then, in the second phase, training data sets are generated. Estimator

structures are trained in the third phase. And, in the last phase, simulations are done to obtain the results for performance evaluations.

5.1.1 Selection of Estimator Inputs

In estimators, estimation accuracy can be easily affected from the inputs behavior. Also estimator performance depends strongly on the number of inputs. For these reasons, selection of the inputs is a critical issue in estimator design process.

Bahar (2003) designed NN estimator for the continuous distillation column under study to estimate the bottom and top compositions from temperatures and past composition values. She applied a Singular Value Decomposition (SVD) technique to select the sensor locations. According to the SVD, for a NC component system, NC-1 temperature measurements are needed for the composition estimation. Hence, three trays from top and three from bottom were found to estimate the top (propane) and bottom (butane) compositions. These are 31^{st}, 32^{nd}, and 33^{rd} trays for the top, and 10^{th}, 11^{th}, and 12^{th} trays for the bottom.

In this study, it is aimed to estimate the compositions only from temperature measurements. However, since ANFIS has a single output, only one of the product compositions can be estimated using the temperature values. There is no need to use past composition values as estimator inputs as NN needs. Also, as the number of measurements is increased as system inputs, structure complexity is increased which affects the convergence of the problem. Therefore, it is decided to use three-tray measurement for estimation process as suggested by SVD. Thus, by referring to the Bahar's study, 31st, 32^{nd}, and 33^{rd} trays are selected to estimate the top product (propane) composition and 10^{th}, 11^{th}, and 12^{th} trays are selected to estimate the bottom product (butane)

composition in the column. The estimation scheme for continuous distillation column is shown in Figure 5.1.

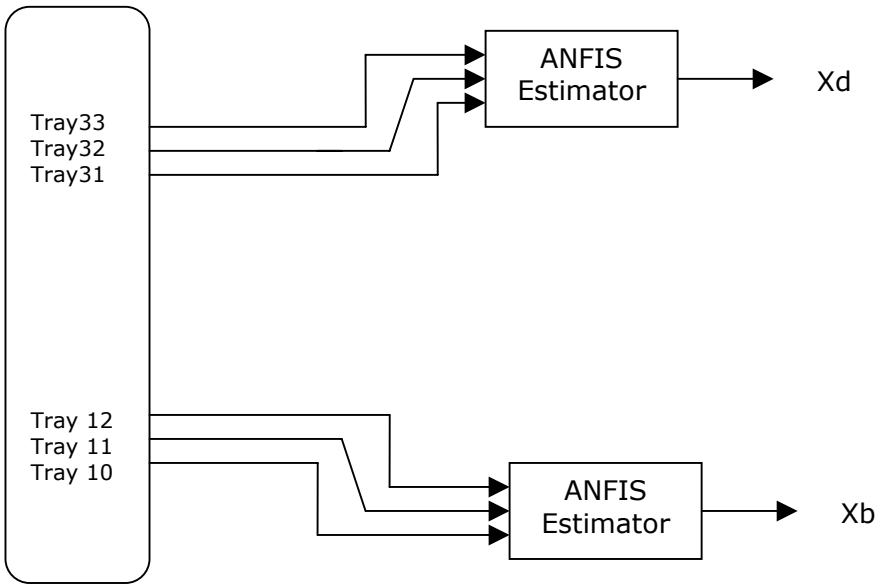

Figure5.1 Estimation scheme for continuous distillation column

5.1.2 Generation of Training Data

If the operating input-output data are outside their training data range, estimator will not operate accurately. As a result, the training data set should possess sufficient operational range including the maximum and minimum values for input-output variables. The data set should include data for each process variable, evenly distributed throughout the range for which estimation is desired.

The maximum and minimum values of reflux and boilup rates in the column were determined by Bahar (2003) by looking at the closed-loop responses of the system without the estimator obtained by Dokucu (2002) to 10% increase in feed flow rate. This corresponds to approximately maximum +7% change in the reflux rate, and maximum +5% change in the boilup rate. These are considered to be their maximum changes in operation. The percent changes for each process variable used in training data generation are shown in

Table 5.1. Thus, model simulations are done to obtain the input-output data by using these values. Then, tray temperature values and corresponding top and bottom product compositions are collected. In training data sets (matrix of 4 columns), first three columns correspond to tray temperature values and the last column corresponds to composition values.

Table 5.1 Range of Process Variables

Process Variable	% changes
Reflux Rate, R	+1, +4, +7
Boilup Rate, Q	+1, +3, +5
Feed Flowrate, F	+1, +5, +10
Feed Composition, z_F	-1, -5, -10

5.1.3 Training of ANFIS estimators

Estimator structure design and training are realized as explained in the previous chapter using MATLAB software. First, generated training data is loaded using the GUI Editor. Then, with chosen design parameters, initial estimator structure is constructed. For example, if three triangular MFs are used for each input and constant output MF is chosen, GUI Editor determines the initial parameters of triangular MFs automatically using loaded data and constructs the initial Tri3con (three triangular MFs for each input and constant output MF) ANFIS structure. Trainings of the structures are done by running the written code in MATLAB. This code is given in Appendix B.1. All structures are trained in the same way only by changing the training data. In the design of top product estimator, training data set corresponded to 31^{st}, 32^{nd}, and 33^{rd} trays temperatures and actual top product composition. In the bottom product estimator design, data set that includes 10^{th}, 11^{th}, and 12^{th} trays' temperatures and actual bottom product composition.

5.1.4 Simulations results

After the training of the ANFIS structures, performances of estimators are investigated through the model for both the verification and generalization tests. These tests are made by utilizing the different estimator structures. The responses of the compositions to a 4% increase in reflux rate and a 5% increase in feed rate are obtained by simulations to verify the estimator's learning performances. Also, reflux rate and feed rate are increased by 5% and 7% respectively to see the generalization capabilities of the estimators. In all simulations, the Integral of the Absolute Error (IAE) scores for the error between the actual and estimated compositions are calculated as the performance criteria. Simulation results are given in Tables 5.2-5.5. Verification capabilities of the estimators can be followed considering the IAE scores giving how well these different estimator structures can generalize what they have learned.

Table 5.2 Verification: 5% increase in Feed Flowrate

Input MF	**Number of input MF**	**Output MF**	**IAE score Top product xd**	**IAE score Bottom product xb**
Triangular	3	constant	0	0
Triangular	3	linear	0	0
Triangular	5	constant	0	0
Triangular	5	linear	0	0
Triangular	7	constant	0	0
Triangular	7	linear	0	0
Gaussian	3	constant	0	0
Gaussian	3	linear	0	0
Gaussian	5	constant	0	0
Gaussian	5	linear	0	0
Gaussian	7	constant	0	0
Gaussian	7	linear	0	0

Table 5.3 Verification: 4% increase in Reflux Rate

Input MF	Number of input MF	Output MF	IAE score Top product xd	IAE score Bottom product xb
Triangular	3	constant	0.00049	0.0038
Triangular	3	linear	0.00017	0.0011
Triangular	5	constant	0.00026	0.0035
Triangular	5	linear	0.00015	0.0007
Triangular	7	constant	0.00025	0.0032
Triangular	7	linear	0.00019	0.0008
Gaussian	3	constant	0.00172	0.0038
Gaussian	3	linear	0.00014	0.0014
Gaussian	5	constant	0.00037	0.0026
Gaussian	5	linear	0.00012	0.0006
Gaussian	7	constant	0.00027	0.0021
Gaussian	7	linear	0.00048	0.0023

Table 5.4 Generalization: 7% increase in Feed Flowrate

Input MF	Number of input MF	Output MF	IAE score Top product xd	IAE score Bottom product xb
Triangular	3	constant	0	0
Triangular	3	linear	0	0
Triangular	5	constant	0	0
Triangular	5	linear	0	0
Triangular	7	constant	0	0
Triangular	7	linear	0	0
Gaussian	3	constant	0	0
Gaussian	3	linear	0	0
Gaussian	5	constant	0	0
Gaussian	5	linear	0	0
Gaussian	7	constant	0	0
Gaussian	7	linear	0	0

Table 5.5 Generalization: 5% increase in Reflux rate

Input MF	Number of input MF	Output MF	IAE score Top product xd	IAE score Bottom product xb
Triangular	3	constant	0.0003	0.0124
Triangular	3	linear	0.0001	0.0075
Triangular	5	constant	0.0030	0.0290
Triangular	5	linear	0.0004	0.0083
Triangular	7	constant	0.0003	0.0054
Triangular	7	linear	0.0004	0.0219
Gaussian	3	constant	0.0018	0.0312
Gaussian	3	linear	0.0013	0.0420
Gaussian	5	constant	0.0036	0.0376
Gaussian	5	linear	0.0014	0.0364
Gaussian	7	constant	0.0086	0.0543
Gaussian	7	linear	0.0032	0.0528

As can be seen from the IAE scores in Table 5.2 and 5.3 that all structures have learned the training data exactly. Learning performances of the estimators are close the each other's. Figures related with verification tests can be seen in Appendix A.1.

When investigating the generalization capabilities, it is seen from Table 5.4 that all structures give actual composition values when the system is disturbed by feed flow rate changes. Their performances are also very good in estimating the top product composition as reflux ratio changes. However, it is observed that predictions of Triangular structures are better considering the IAE scores than that of Gaussian structures for bottom product composition in reflux ratio changes. Table 5.5 shows that minimum IAE score for top product composition is achieved from the Tri3lin structure. For bottom product, although Tri7con, Tri5lin, and Tri3lin structures show almost similar performance, Tri7con structure has the minimum IAE score. Thus, Tri3lin structure can be selected as the estimator architecture for top product composition. Figure 5.2 illustrates the Tri3lin structure performance in terms of top and bottom product compositions responses to 5% increase in reflux ratio. Performances of Tri7con and Tri5lin can be seen in Figure 5.3 and 5.4. Figures that show other structure's generalization performances are also given in Appendix A.2.

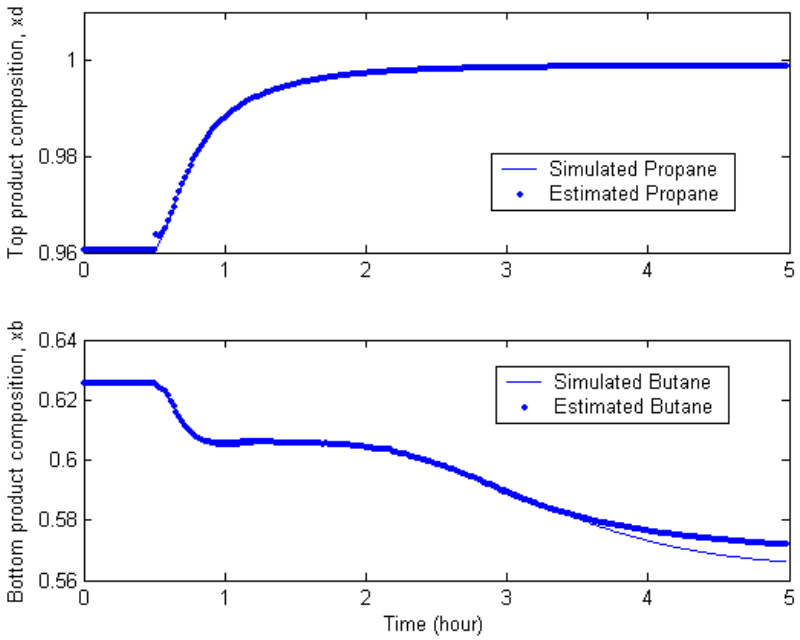

Figure 5.2 Tri3lin structure performance to 5% increase in reflux rate with the IAE scores 0.00014 and 0.0074 for top and bottom product

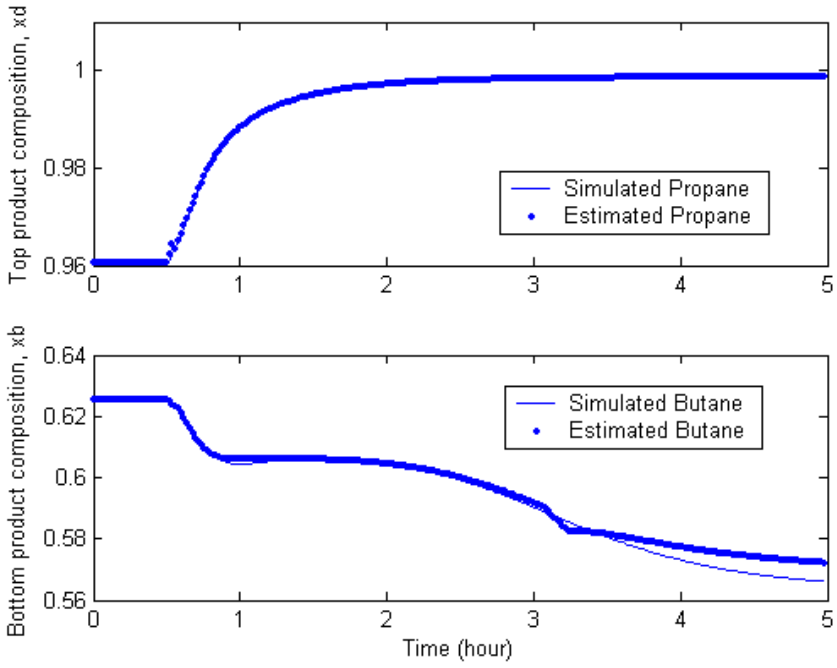

Figure 5.3 Tri5lin structure performance to 5% increase in reflux rate with the IAE scores 0.00044 and 0.0088 for top and bottom product

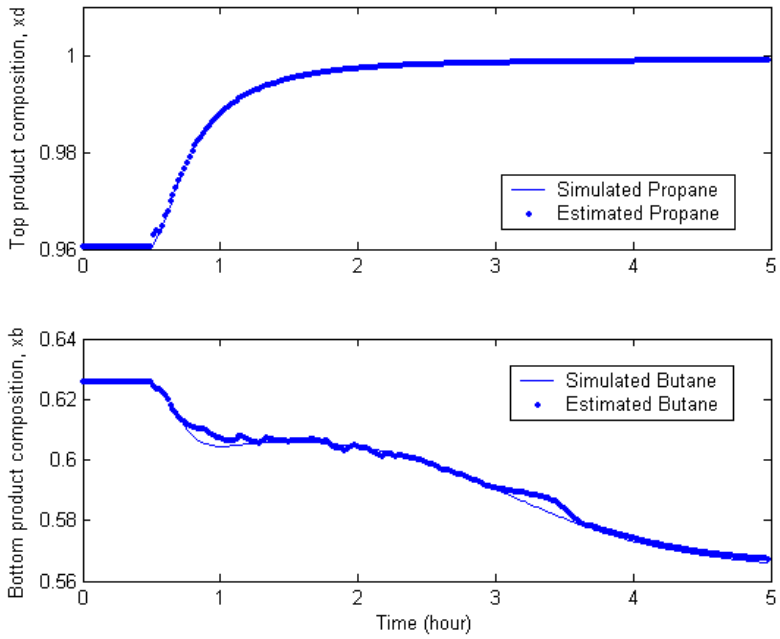

Figure 5.4 Tri7con structure performance to 5% increase in reflux rate with the IAE scores 0.00034 and 0.0054 for top and bottom product

It can be seen from the Figure 5.2 and 5.3 that structures with linear output show excellent performance up to 3.5 hours of response time. Tri3lin structure is somewhat better than Tri5lin structure. It is also found that defining of the inputs with many MFs do not improve performances of the structures that have linear output. However, in Figure 5.4, it is seen that that structure with seven MFs in input and with constant output results in good estimates for the steady state value with slight deviations from the actual values in short response time.

Therefore, in order to determine the estimator structures, another generalization simulation is made. The reflux ratio is increased by 3% and 6%, and then Tri3lin and Tri7con estimator performances are investigated. Figures 5.5-5.9 illustrate the performances of Tri3lin and Tri7con structures, respectively.

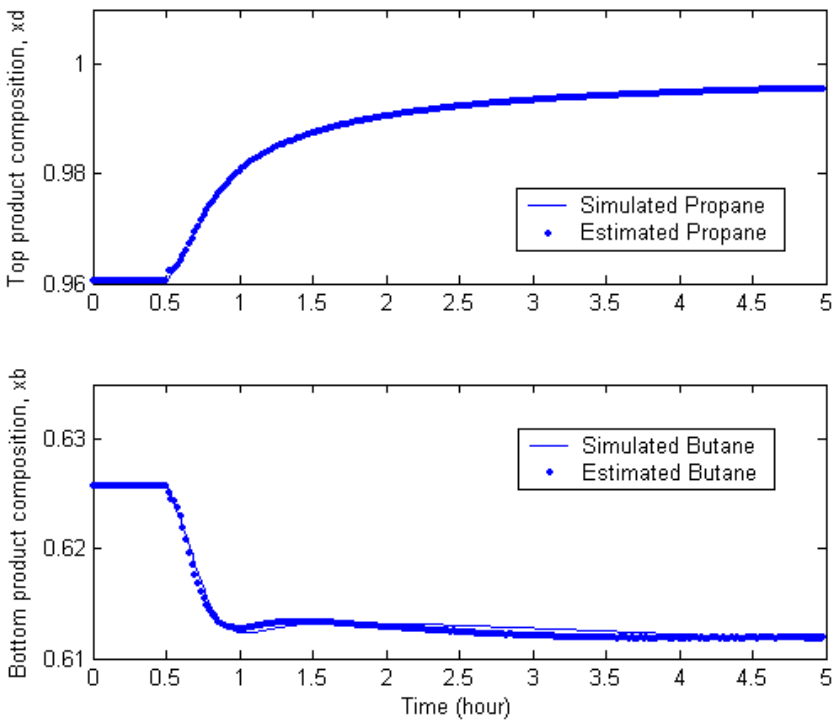

Figure 5.5 Tri3lin structure performance to 3% increase in reflux rate with IAE scores 0.0002 and 0.0018, for top and bottom product

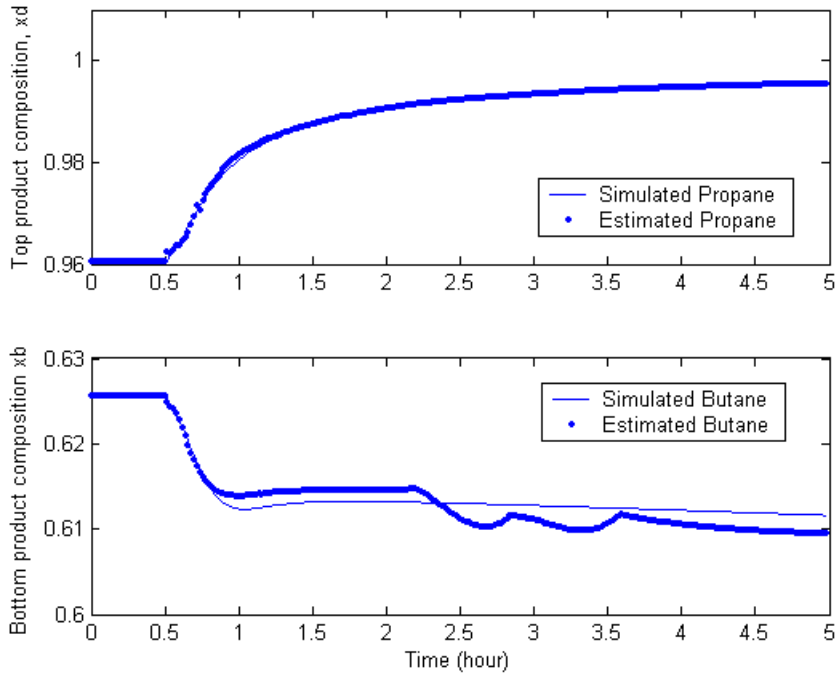

Figure 5.6 Tri7con structure performance to 3% increase in reflux rate with IAE scores 0.0008 and 0.0151, for top and bottom product

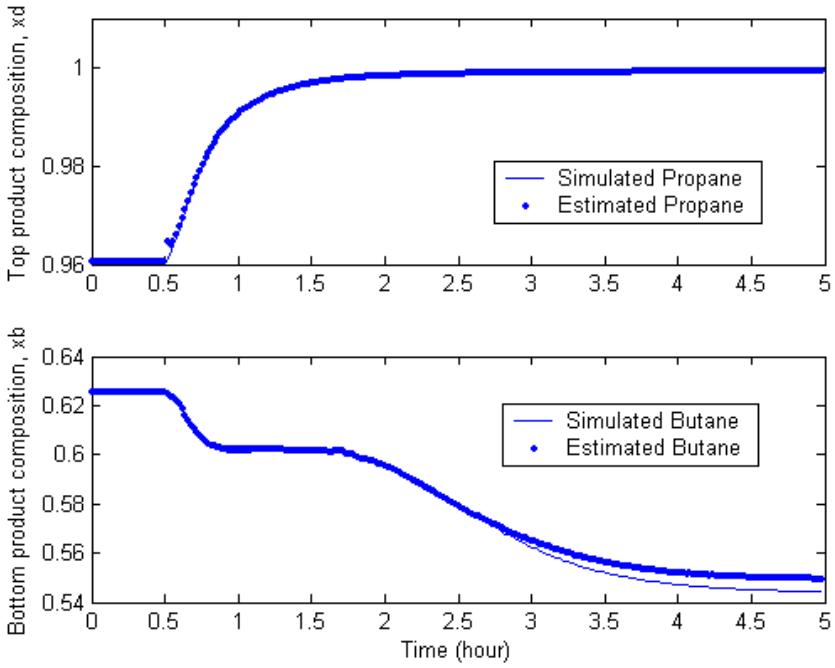

Figure 5.7 Tri3lin structure performance to 6% increase in reflux rate with IAE scores 0.00014 and 0.0108, for top and bottom product

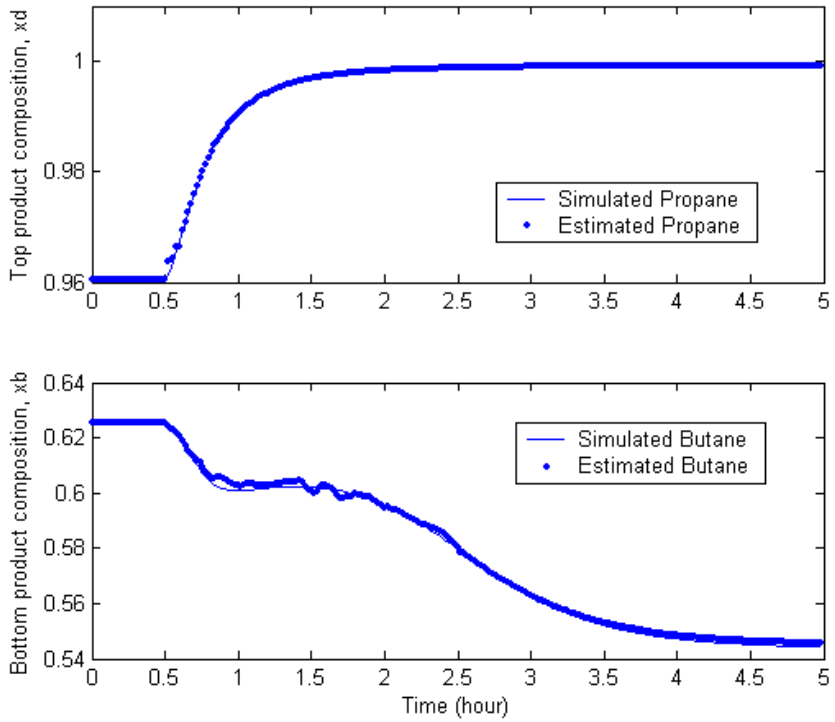

Figure 5.8 Tri7con structure performance to 6% increase in reflux rate with IAE scores 0.00032 and 0.00487, for top and bottom product

As can be seen from the Figures 5.5-5.9 and IAE scores, the Tri3lin structure performance is better to estimate the top product composition and is selected as the top product estimator. However, when the results are evaluated for bottom product composition Tri3lin and Tri7con structure's performances are similar. Although IAE scores of Tri7con structure are smaller in total, Tri3lin structure performance is very good up to 3 hours of response time. The Tri7con structure performance becomes also weak especially in small increase in reflux ratio. This can be seen in Figure 5.6 as the reflux ratio is increased by 3%.

In this study, Tri3lin and Tri7con structure performance are also evaluated for the other compositions in the column. It is seen from the Figures 5.9 and 5.10 that Tri3lin structure performance is very good for other compositions in the top of the column. Figures 5.11 to 5.14 show the structure's performances for the compositions in bottom. As can be seen from these figures that Tri7con structure performance changes after 3 hours of response time similar to the Tri3lin structure. It can also be seen from the Figures 5.12 and 5.14 that Tri3lin structure performance is better than Tri7con structure for the i-butane composition in bottom. If the many training data are collected from the part of simulations after 3 hours, better learning of the Tri3lin structure can be achieved and thus estimator performance can be developed.

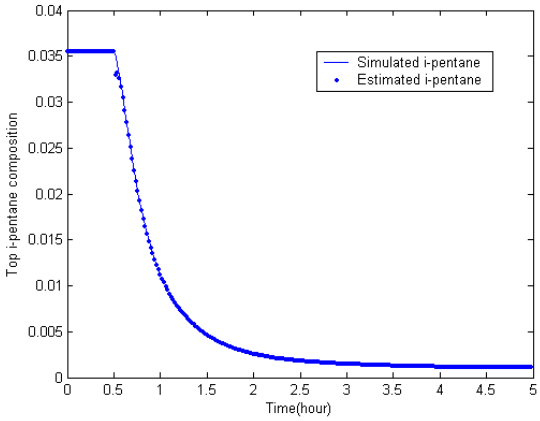

Figure 5.9 Tri3lin structure performance to 5% increase in reflux rate for the i-pentane composition in top with IAE score, 0.00012

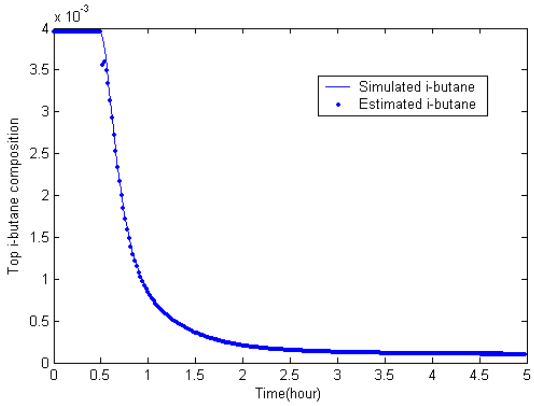

Figure 5.10 Tri3lin structure performance to 5% increase in reflux rate for the i-butane composition in top with IAE score, 0.00047

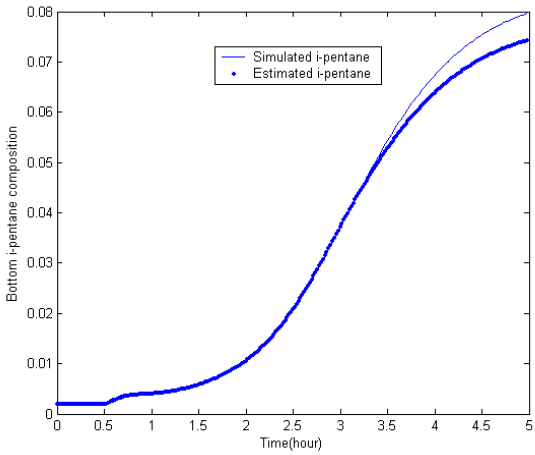

Figure 5.11 Tri3lin structure performance to 5% increase in reflux rate for the i-pentane composition in bottom with IAE score, 0.0065

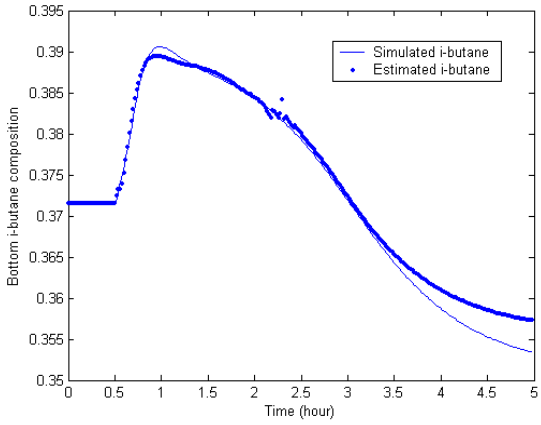

Figure 5.12 Tri3lin structure performance to 5% increase in reflux rate for the i-butane composition in bottom with IAE score, 0.0056

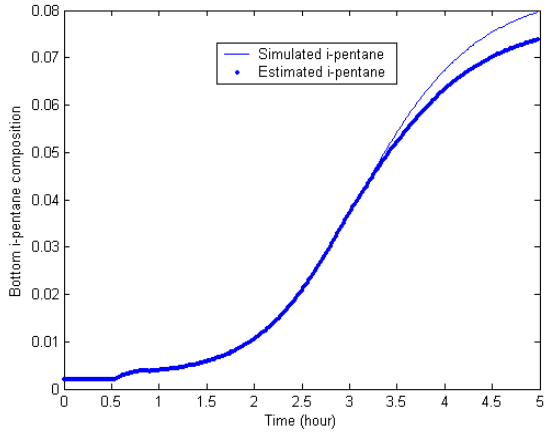

Figure 5.13 Tri7con structure performance to 5% increase in reflux rate for the i-pentane composition in bottom with IAE score 0.0075

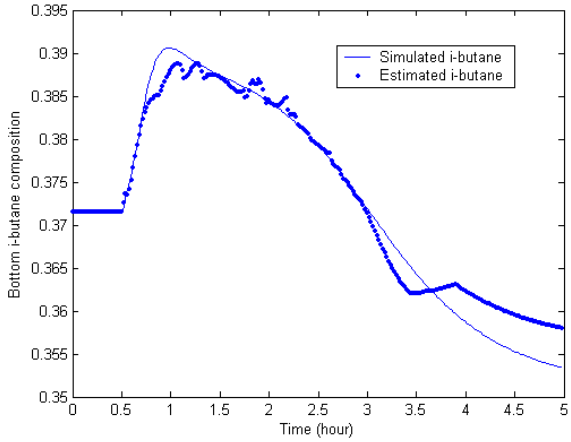

Figure 5.14 Tri7con structure performance to 5% increase in reflux rate for the i-butane composition in bottom with IAE score, 0.0096

Therefore, evaluating all the obtained simulation results; Tri3lin structure can be used as ANFIS estimator to predict the top and bottom product compositions, and the other compositions, in the continuous distillation column under study. It can be implemented to the real plant and online estimation of the compositions from temperatures can be achieved. Tri3lin fuzzy inference system files in MATLAB and rule base are given in Appendix B.2 to B.5. Its parameters can be seen in these files. Input MFs are also shown in Figure A.30 and A.31. The column simulation code can be found in the study of Bahar (2003).

Designed ANFIS estimators are also compared with NN estimator developed by Bahar (2003). This NN estimator was designed to be used in MPC framework for the control of product compositions of the column. Estimator's performances for the 5% increase in reflux rate for top and bottom product compositions are shown in Figures 5.7 and 5.8, respectively. It is seen that ANFIS estimator is very good compared to the NN estimator.

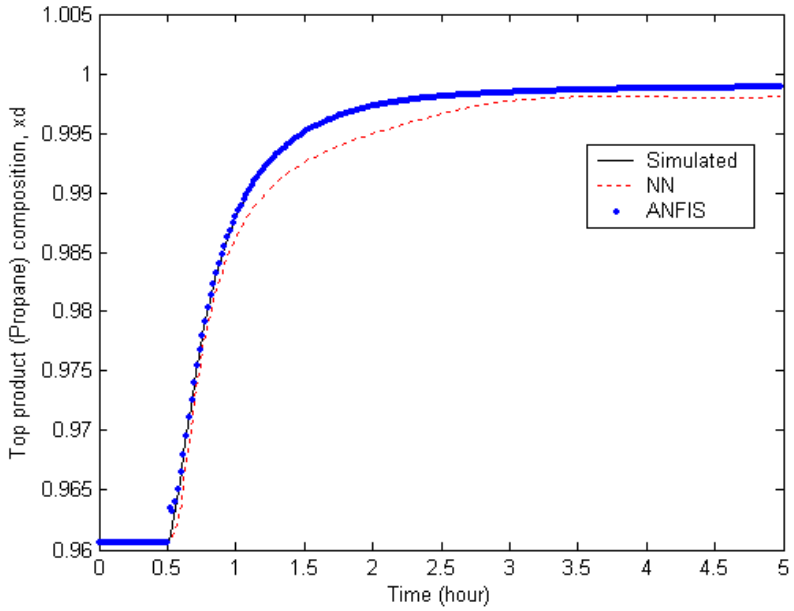

Figure 5.15 Comparison of the ANFIS and NN estimators for top product compositions for a 5% increase in reflux rate with the IAE scores, 0.00014 0.0059, respectively

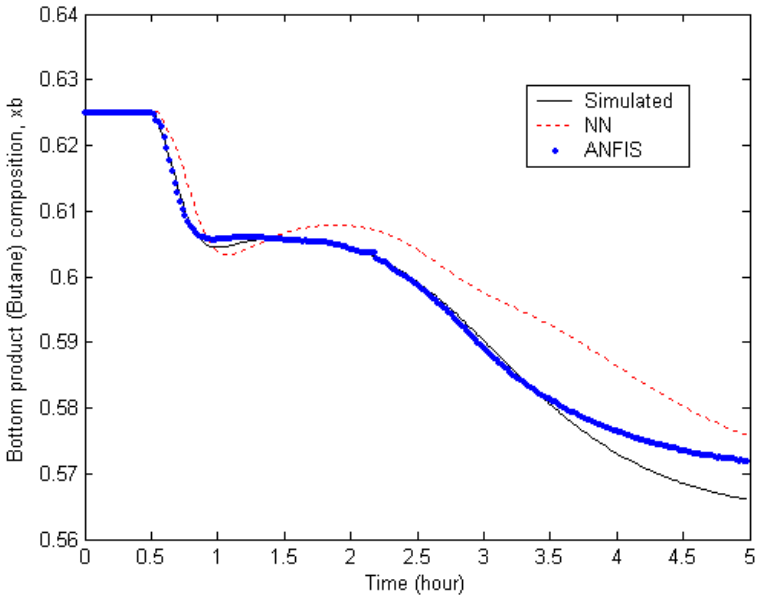

Figure 5.16 Comparison of the ANFIS and NN estimators for bottom product compositions for a 5% increase in reflux rate with the IAE scores, 0.0074 0.0308, respectively

In the designed NN estimator in Bahar's study which predicts the top and bottom compositions, input vector also considers the past composition values. The past composition values are also the outputs of the estimator. Therefore, the estimator, which can produce inaccurate composition values in the past, is used for the estimator at next time steps. This brings errors to the system and it reduces the system performance. In NN estimator, many input output data pairs are needed for training and normalization process. If the input and output variables are not of the same order of magnitude, some variables may appear to have more significance then they actually do. Thus, training data and network inputs and outputs are needed to be normalized to values between 0 and 1. Besides, system with many inputs and parameters can require additional computational power, especially in online applications. As a result, although different estimators should be designed for each composition, it can be said that, ANFIS can be utilized for a better estimation in distillation columns instead of NNs.

5.2 Estimation in Batch Distillation Column

In this part of the study, it is aimed to investigate the ANFIS structure performances in batch distillation column. For this purpose, three parallel ANFIS estimators are developed to predict the reflux drum compositions; cyclo-hexane (C_1), n-heptane (C_2) and toluene (C_3) from tray temperatures by applying the same procedure as implemented in the case of continuous distillation column.

5.2.1 Selection of Estimator Inputs

In continuous distillation column, since four components were separated, three tray temperatures were used as the estimator's input according to the SVD method. It was also seen from the results that three tray temperature measurements were sufficient to estimate the compositions. If the SVD method (NC-1) is applied in batch distillation column, since three components are separated, two tray temperature measurements should be selected and used for estimation in batch distillation column.

In the case study of continuous distillation column, number of trays on which the temperature can be measured is 37. However, batch distillation column under study has 8 trays. Besides, batch distillation process has much complex characteristics than continuous distillation columns. Therefore in order to reflect the column dynamic well, instead of two trays, three trays are used in batch distillation column without SVD analysis. Thus, 2^{nd}, 5^{th} and 9^{th} trays, one from bottom, one from middle and one from top of the column, are selected and used for the composition estimation.

5.2.2 Generation of training data

As stated in the previous chapter, the column is worked under optimal reflux ratio policy of Mujtaba (1993). Optimal reflux ratio policy and column

design parameters can be seen in Table 4.2 and 4.3. Rigorous model for column developed by Yıldız (2003) is used with these column parameters. Different simulations are done with this model by changing the initial composition of the feed charge to the column to generate the training data for estimators. Initial fractions for feed used in simulations are given in Table 5.6. C_1, C_2 and C_3 are the compositions of cyclo-hexane, n-heptane and toluene, respectively. Having collected the input output data, three different training data sets are formed and used in training.

Table 5.6 Initial feed compositions

Run No	**C_{1init}**	**C_{2init}**	**C_{3init}**
1	0.20	0.20	0.60
2	0.25	0.50	0.25
3	0.30	0.15	0.55
4	0.35	0.40	0.25
5	0.40	0.30	0.30
6	0.50	0.25	0.25
7	0.60	0.35	0.05
8	0.22	0.60	0.18

5.2.3 Training of ANFIS structures

Estimations based on simulations for continuous column indicated that, Triangular structures are better than Gaussian structures both in verification and generalization. Therefore, only Triangular structure's performances are used in batch distillation studies. These structures are trained with generated training data sets. Thus, number of trainings and test simulations are decreased.

5.2.4 Simulations results

After trained the structures, estimators' verification and generalization capabilities are tested using the rigorous model of the column as a real plant.

Verification test is done using Run No 6 from the Table 5.6 as the initial feed compositions. Figures in Appendix A.3 illustrate this verification performance of structures. IAE scores are tabulated in Table 5.7. It can be seen from Figures in Appendix A.3 that learning performances of the constant output structures are better than linear output structures. This can also be seen from the IAE scores given in Table 5.7. Therefore, it is decided to use only constant output structures in generalization tests.

First generalization test is done with the initial fractions of C_{init} [0.407; 0.394; 0.199]. Results are shown in Figures 5.17, 5.18 and 5.19. Tri3con and Tri5con performance are nearly same but Tri7con structure performance is not good as the others. Therefore, another generalization test is done to make better decision.

Table 5.7 Verification test results with the initial fractions of [0.5; 0.25; 0.25]

Input MF	**Number of input MF**	**Output MF**	**IAE score of C1**	**IAE score of C2**	**IAE score of C3**	**Total IAE score**
Triangular	3	constant	0.04211	0.0972	0.0644	0.20374
Triangular	3	linear	0.0783	0.1845	0.1036	0.3664
Triangular	5	constant	0.0521	0.1073	0.0627	0.2200
Triangular	5	linear	0.1173	0.3009	0.17918	0.5975
Triangular	7	constant	0.079	0.1594	0.0873	0.3258
Triangular	7	linear	0.123	0.2761	0.1652	0.5644

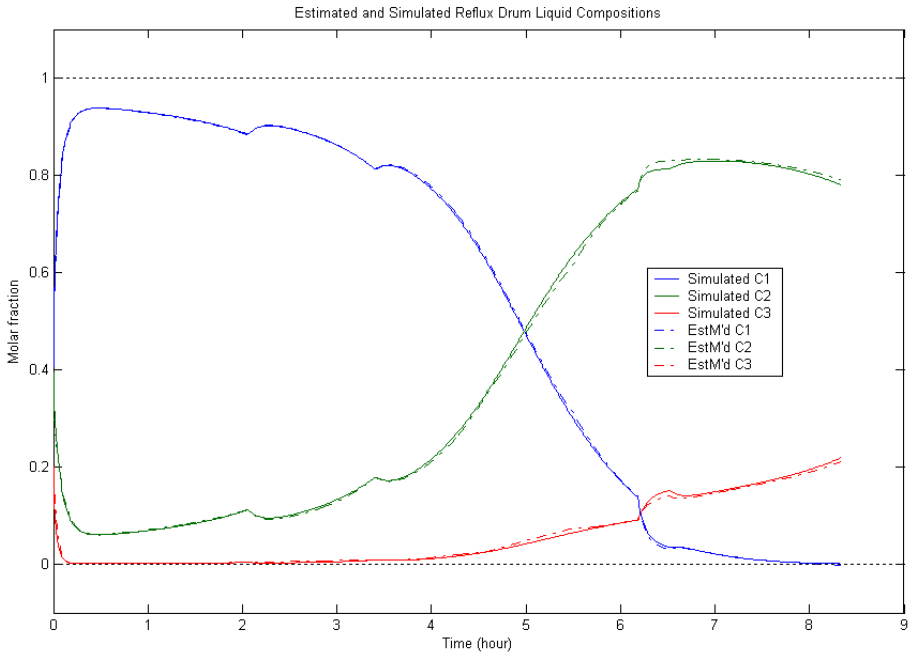

Figure 5.17 Generalization of Tri3con structure; C_{init} [0.407; 0.394; 0.199] with the IAE scores 0.0213, 0.0428 and 0.0251, respectively

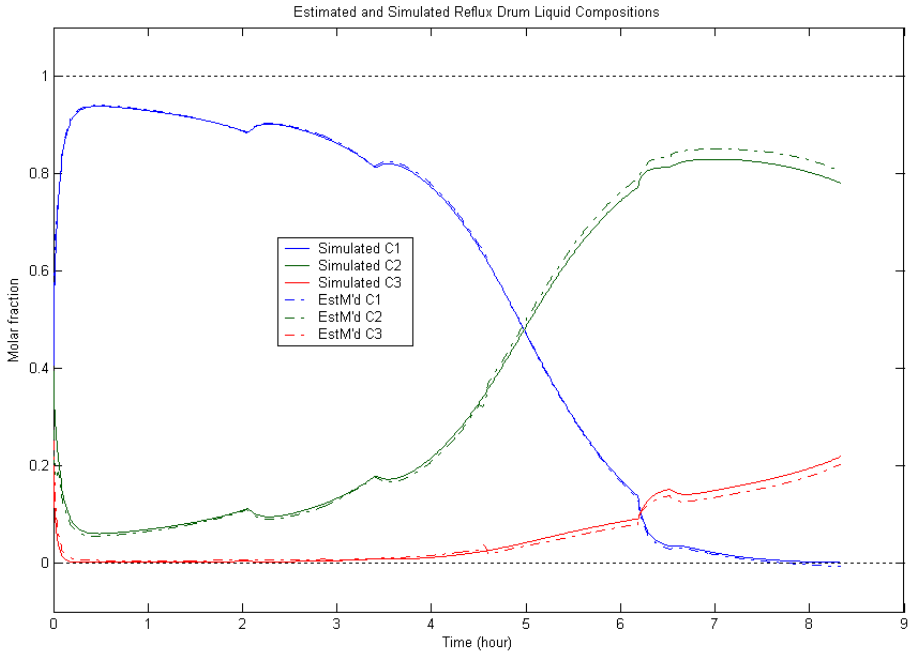

Figure 5.18 Generalization of Tri5con structure; C_{init} [0.407; 0.394; 0.199] with the IAE scores 0.0285, 0.1002 and 0.0645, respectively

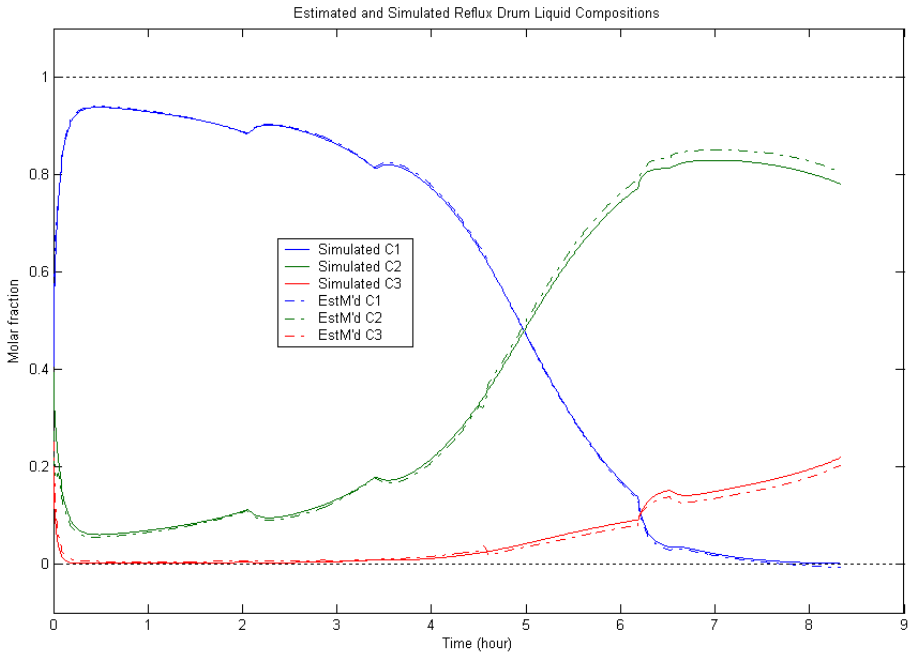

Figure 5.19 Generalization of Tri7con structure; C_{init} [0.407; 0.394; 0.199] with the IAE scores 0.0609, 0.2114 and 0.1335, respectively

Second generalization test is done with the initial fractions of C_{init} [0.25; 0.35; 0.40]. Figure 5.20, 5.21 and 5.22 shows the estimator performances. It can be seen that Tri3con shows the best performance among the constant output structures.

Moreover, additional generalization simulations are done and Tri3con structure performance is investigated in detail with the different initial feed fractions. It can be seen from Figures 5.23 to 5.25 and from IAE scores that Tri3con structure performance is very good. This structure can be used as ANFIS estimator to predict the reflux drum compostions from the 2^{nd}, 5^{th} and 9^{th} trays temperatures in batch distillation column under study. Tri3con structure parameters, estimator rules and simulation code for the batch distilllation column are given in Appendix from B.6 to B.9. In addition, Figures of input MFs are presented in Appendix A.32, A.33 and A.34.

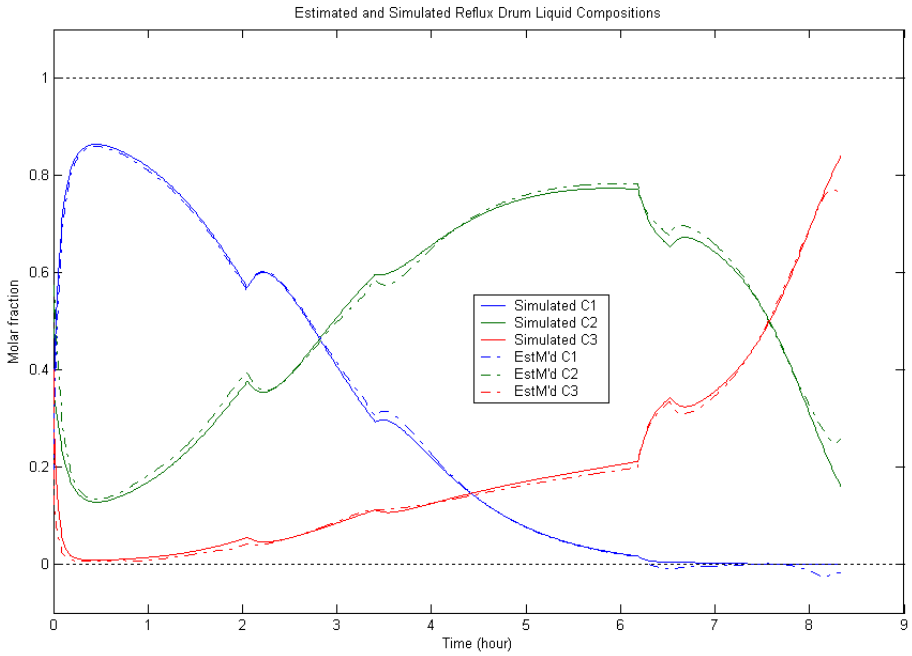

Figure 5.20 Generalization of Tri3con structure; C_{init} [0.25; 0.35; 0.40] with the IAE scores 0.0552, 0.1126 and 0.0626, respectively

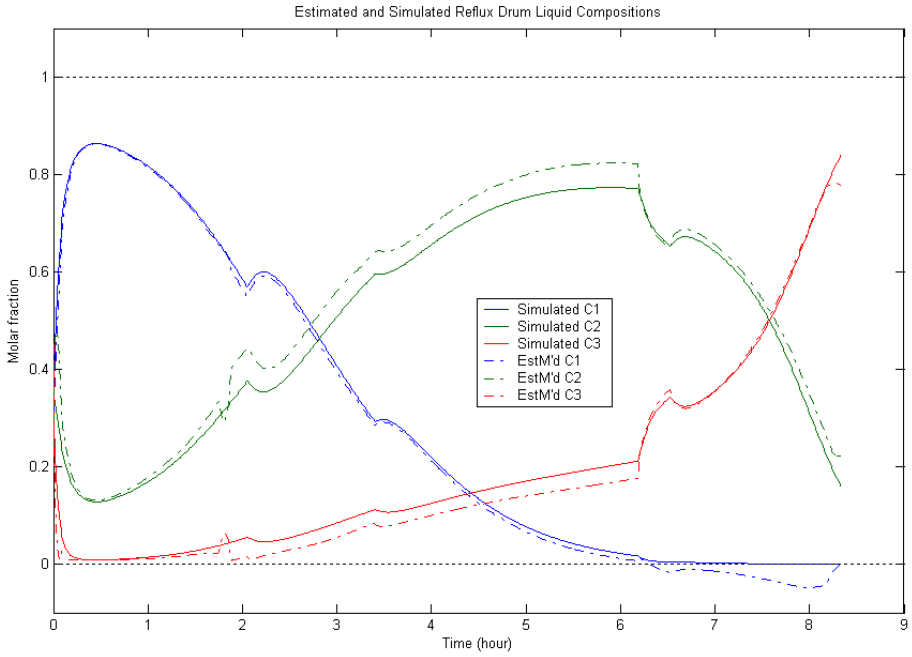

Figure5.21 Generalization of Tri5con structure; C_{init} [0.25; 0.35; 0.40] with the IAE scores 0.1088, 0.2849 and 0.1605, respectively

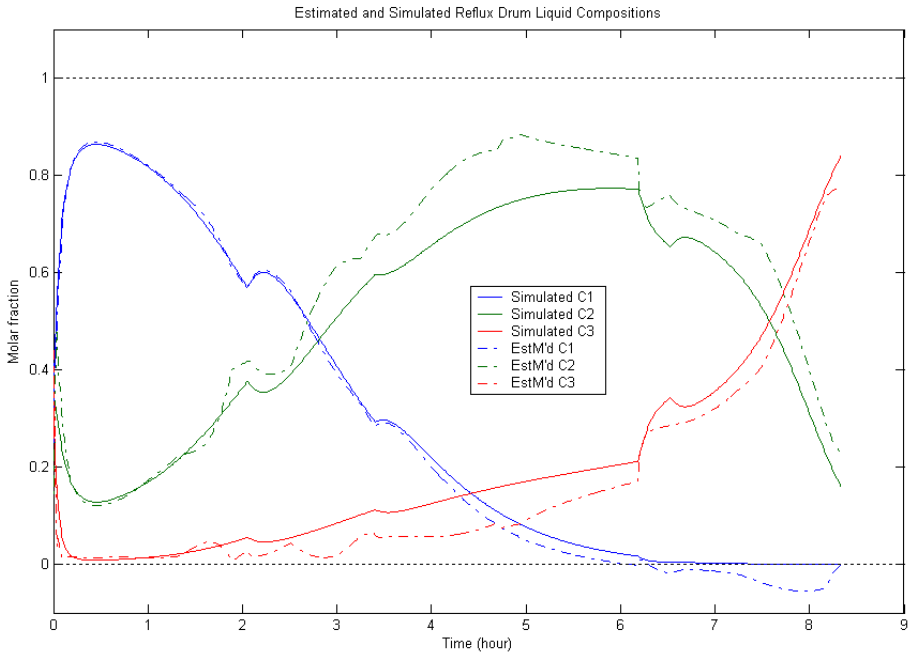

Figure 5.22 Generalization of Tri7con structure; C_{init} [0.25; 0.35; 0.40] with the IAE scores 0.1459, 0.5813 and 0.3378, respectively

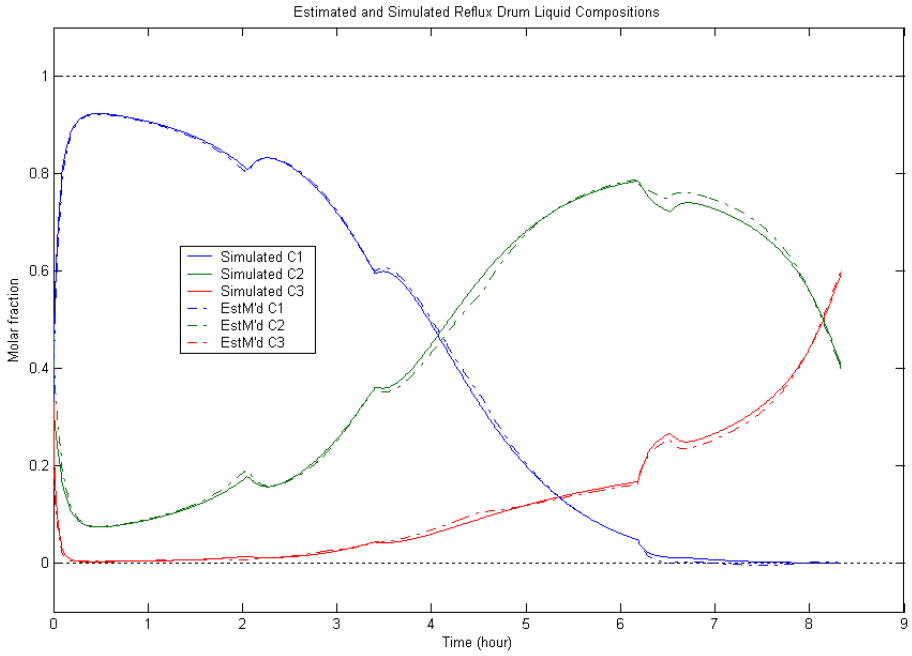

Figure 5.23 Generalization of Tri3con structure; C_{init} [0.34; 0.33; 0.33] with the IAE scores 0.0456, 0.0826 and 0.0451, respectively

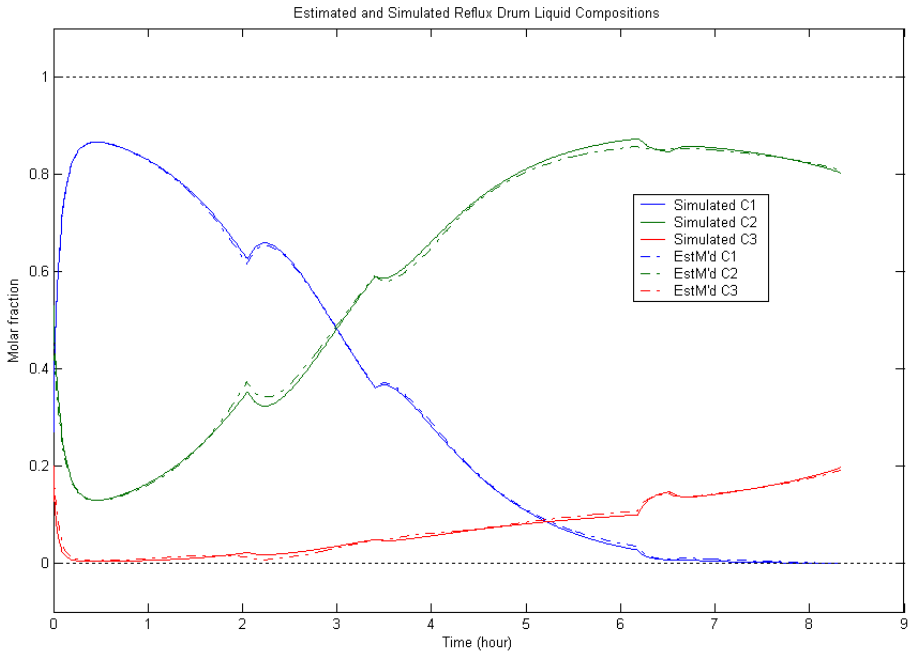

Figure 5.24 Generalization of Tri3con structure; C_{init} [0.27; 0.53; 0.20] with the IAE scores 0.0292, 0.0583 and 0.0349, respectively

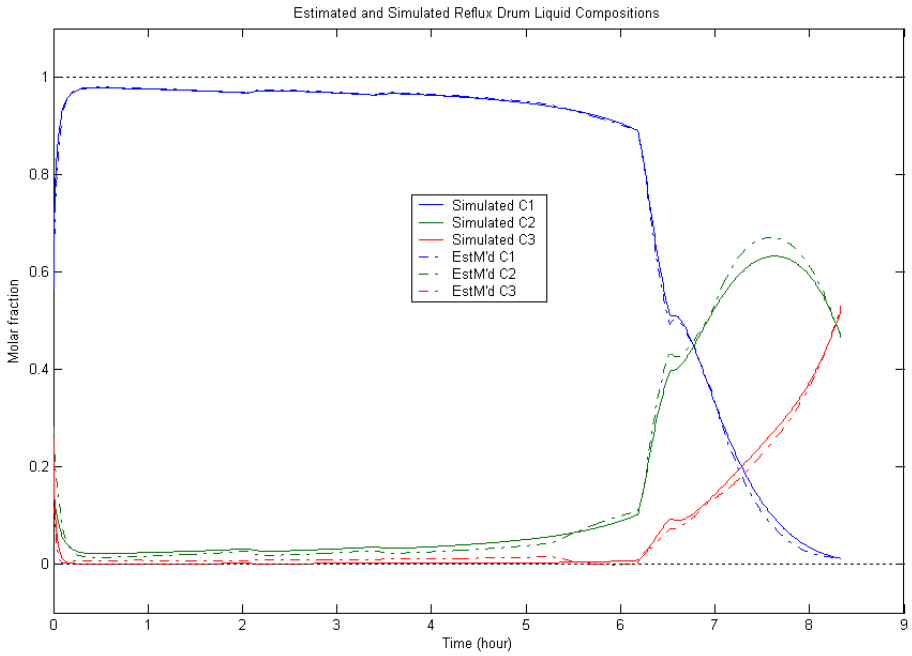

Figure 5.25 Generalization of Tri3con structure; C_{init} [0.58; 0.16; 0.26] with the IAE scores 0.0389, 0.1073 and 0.0741, respectively

Yıldız (2003) designed an Extended Kalman Filter (EKF) to estimate the compositions in the reflux drum for the batch distillation column under study. Figure 5.26 illustrates the performance of the EKF estimator for the reflux drum compositions with the initial fractions of feed C_{init} [0.407; 0.394; 0.199]. For the same initial fractions, Tri3con ANFIS estimator performance was already given in Figure 5.19. Estimators's performances are shown in Figure 5.27. As can be seen from the Figure 5.27 that ANFIS performance is quite well according to the EKF.

EKF is the optimal state estimator while ANFIS can be used in estimation studies of distillation columns. If the sensors provide perfect and complete data about a system without any measurement corruption by noise and without device inaccuracies, trained ANFIS structures can be implemented to the system for online estimation of the state variables. Also, there is no need for a simplified mathematical model of the system; that is needed for the EKF design.

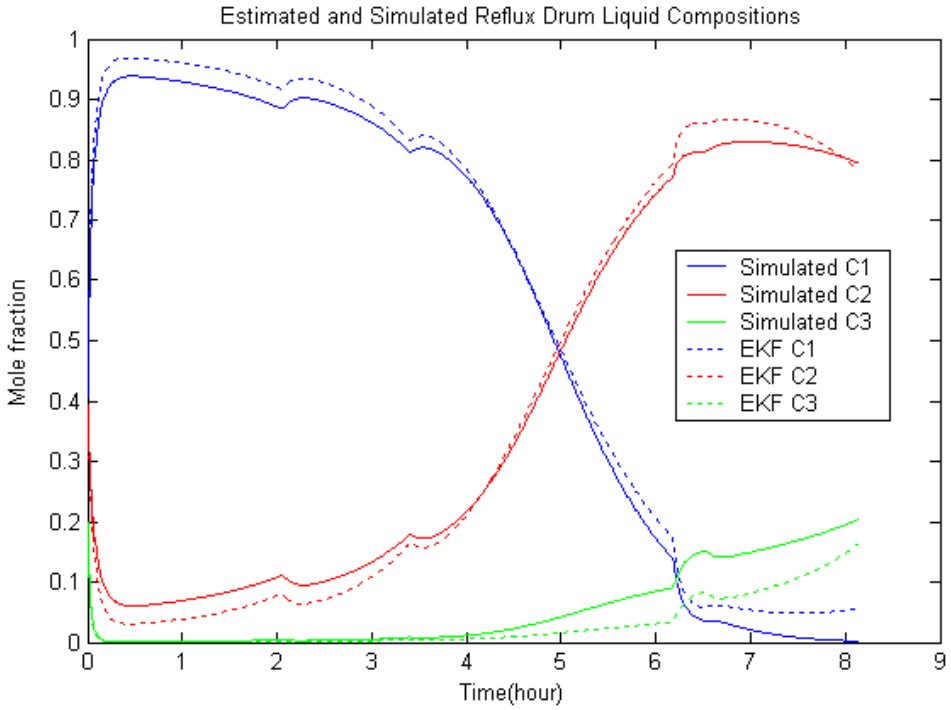

Figure 5.26 EKF estimator performance; C_{init} [0.407; 0.394; 0.199] with the IAE scores 0.2211, 0.1933 and 0.1966, respectively

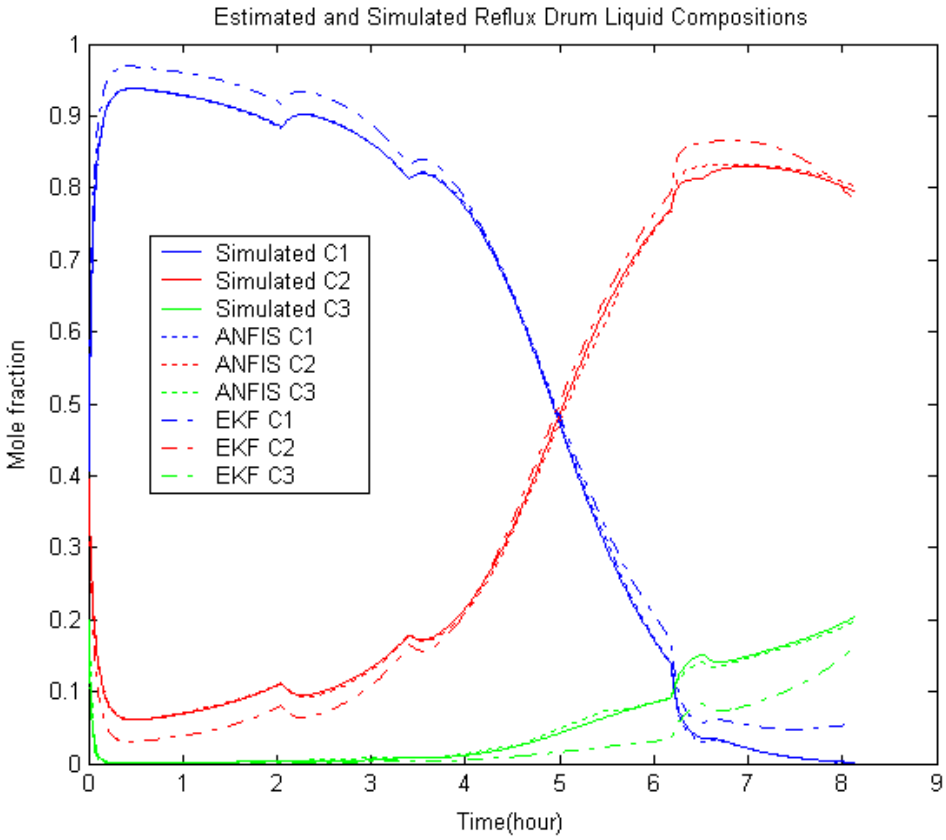

Figure 5.27 Comparison of ANFIS and EKF estimators

In this study, design parameters of the system are selected as the ANFIS structure parameters. Temperature sensors locations were not investigated in detail. Actually, according to the SVD method NC-1 number of tray temperature is sufficient to use as the estimator inputs. Therefore, in the system under study, only two tray temperatures can be used. However, as stated previously in the batch distillation column study, three temperature locations were selected as 2nd, 5th and 9th trays (one from bottom, one from middle and one from top of the column) to represent the column dynamic well. The designed ANFIS estimator performances with two tray temperatures are shown in Figure 5.28 and 5.29. These two trays are selected from the top of the column as 8th and 9th trays without applying SVD analysis. Figure 5.28 illustrates the performance of the estimator with the initial feed fractions of 0.58, 0.16 and 0.26. The estimator performance with three inputs for the same initial feed fractions was shown in

Figure 5.25. When these two Figures 5.25 and 5.28 and IAE scores are compared, it is observed that two input estimator performance is as good as than three input estimator. However, initial fraction estimation of C_2 and C_3 is not very close (0.1677/0.64, 0.2822/-0.101). Figure 5.29 also illustrates the performance of the estimator with the initial fractions of 0.407, 0.399, and 0.199. The estimator performance with three inputs for the same initial fractions was shown in Figure 5.17. It can be seen from the Figure 5.17 and 5.29 that the estimator perfromance with two inputs is as good as three input estimator with the IAE comparison but initial fraction estimation of C_3 (0.20/-0.10) is not achieved. When the total IAE scores are compared in these two runs tested with different initial feed composition, it can be concluded that the estimation can be done using only two tray temperatures but initial composition fractions can not be estimate accurately.

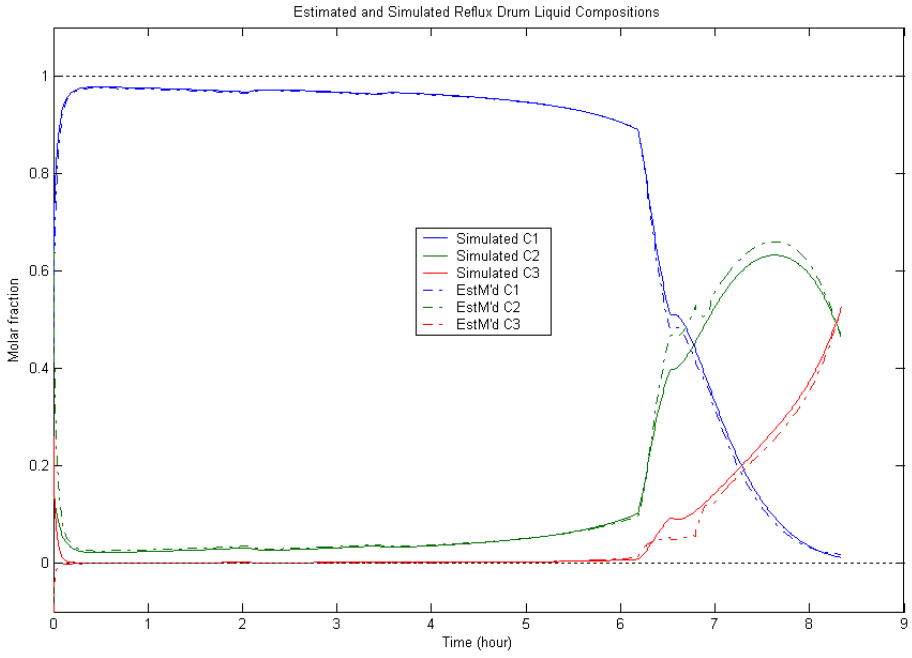

Figure 5.28 ANFIS estimator performance with two estimator inputs with the initial fractions; C_{init} [0.58; 0.16; 0.26] and with the IAE scores 0.0441, 0.0938 and 0.0569, respectively

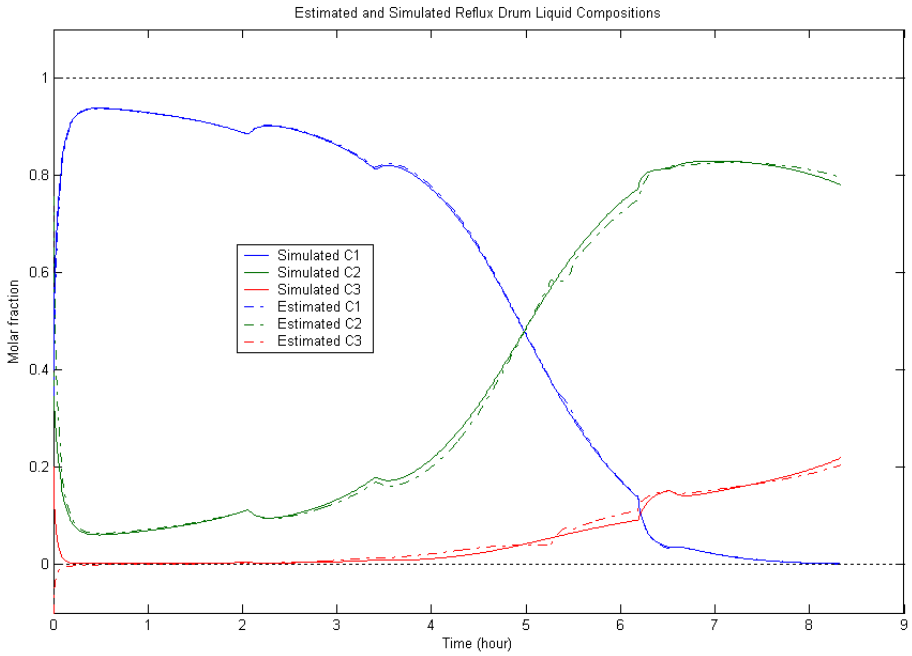

Figure5.29 ANFIS estimator performance with two estimator inputs with the initial fractions; C_{init} [0.407; 0.394; 0.199] and with the IAE scores 0.029, 0.076 and 0.045, respectively

5.3 Control of the pH reactor

The control of a pH system is often considered to be the benchmark for nonlinear process control because of the highly nonlinear behavior exhibited by pH dynamics. In this study, it is aimed to use ANFIS as a controller in a pH control system. For this purpose, ANFIS controller is designed and used in an adaptive way in the pH control scheme. Figure 5.30 illustrates the adaptive control scheme for the pH system under study. Developed pH model is used as a real plant in this scheme. The objective of the system is to control the effluent pH by manipulating base stream flow. Inputs to the controller at each sampling instant are plant and controller output, pH(k-1) and F_2(k-1), respectively at previous sampling instant. Controller output is the new plant input, F_2(k).

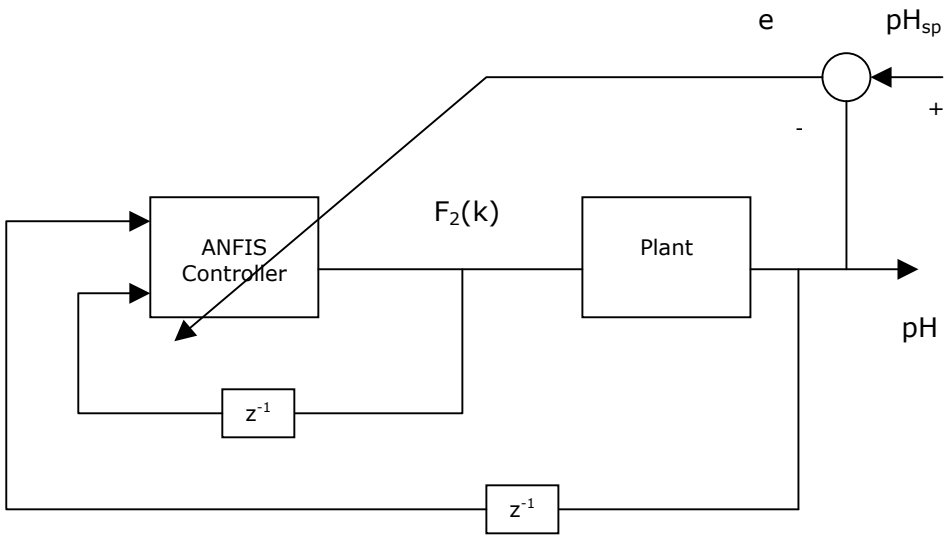

Figure5.30 Adaptive pH control scheme

In this study, Gaussian MFs are only used as the input MFs. The controller inputs are labeled with three linguistic variables Small (S), Medium (M) and Big (B), and output MF type is constant. Hence, a controller structure that has few parameters is constructed. Since the controller has two inputs and three MFs on each input, its rule base consists of nine rules. These rules are given in Table 5.8. Unlike the estimation part, backpropagation algorithm is used for controller parameter adaptation instead of hybrid learning algorithm due to the computational simplicity. A code for parameter update is written according to the backpropagation algorithm explained in Chapter 4.1.

Table 5.8 Rule base of the controller

Rule	Controller rule base
1	If pH(k-1) is S and F_2(k-1) is S then F_2(k) is f_1
2	If pH(k-1) is S and F_2(k-1) is M then F_2(k) is f_2
3	If pH(k-1) is S and F_2(k-1) is B then F_2(k) is f_3
4	If pH(k-1) is M and F_2(k-1) is S then F_2(k) is f_4
5	If pH(k-1) is M and F_2(k-1) is M then F_2(k) is f_5
6	If pH(k-1) is M and F_2(k-1) is B then F_2(k) is f_6
7	If pH(k-1) is B and F_2(k-1) is S then F_2(k) is f_7
8	If pH(k-1) is B and F_2(k-1) is M then F_2(k) is f_8
9	If pH(k-1) is B and F_2(k-1) is B then F_2(k) is f_9

In the plant Jacobian for calculating the gradient in backpropagation algorithm (Equation 4.2), approximation of sigmoidal function is used. In the scheme, process is assumed to form an unmodifiable layer of the lumped controller-process network. Hence, using sigmoidal function to represent the process layer gives an approximation process gain as well as its sign. Actually, this approach can be more appropriate for pH system compared to other nonlinear process due to *S* shape the steady state gain curve in Figure 4.8 Having obtained the parameters by trial and error, the function is found as follows:

$$pH\left(F_2\right)=\frac{14}{1+e^{\left[-0.012\left(F_2-515\right)\right]}} \tag{5.1}$$

Approximation is shown in Figure 5.31. Hence, the plant Jacobian, $\frac{\partial pH}{\partial u}$, is

$$\frac{\partial pH}{\partial F_2}=\frac{0.168e^{\left[-0.012\left(F_2-515\right)\right]}}{\left(1+e^{\left[-0.012\left(F_2-515\right)\right]}\right)^2} \tag{5.2}$$

The controller parameters can be adapted at each sampling instant according to the Equation 4.6. However, initial parameters of the controller must be determined suitably for the convergence of the system to start the simulation. Thus, training data is obtained from open loop simulation by changing the base stream flow rate of the system while the acid stream flow rate is held constant. Training data set is generated from pH and base flow rate values and then controller is trained. Initial CSTR parameters used in the present study are the same as in Bhat and McAvoy (1990) and given in Table 5.8. Closed loop simulation code including controller structure, model and parameter adaptation are given Appendix B.10.

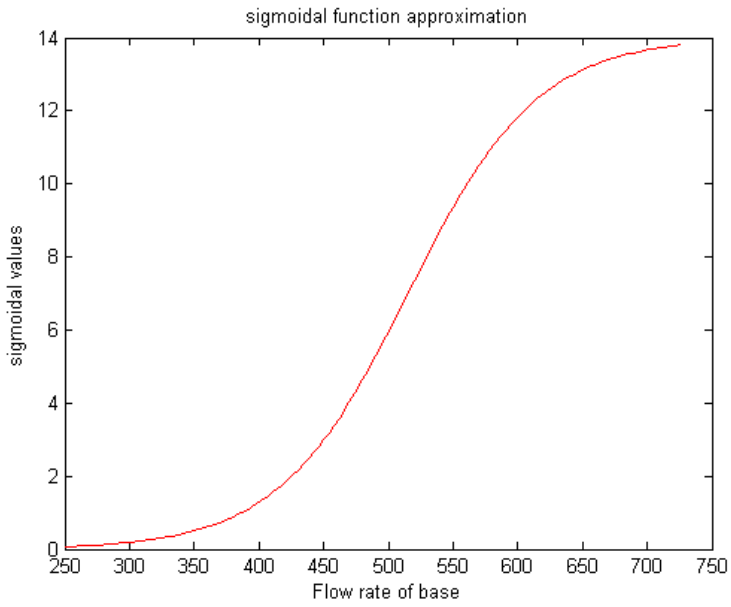

Figure5.31 Sigmoidal function approximation

Figure 5.32 represents the closed-loop system responding to set point changes with the proposed ANFIS controller in the loop. The pH set point is varied from an initial value of 6.94 to a value of 5 and then to a new value of 9. As can be seen from the Figure 4.8, this system exhibits highly nonlinear behavior and has very high gains around the neutral zone (pH=7). A small change in the base stream input flow results in changing the output pH significantly in this steep region. Hence, it poses quite a challenging problem to control the pH of the process around the electroneutrality point, where the gain surface is very steep. Although the controller does not have to work hard for the first set point change due to the flat region of the gain surface, it has to balance the closed-loop system on the steep surface for the second set point change.

As can be seen from Figure 5.32 that ANFIS controller is able to drive the closed-loop system to the desired values. While there is no oscillatory response for the first set point change from 6.94 to 5, there are some oscillations in the process response for the second change in set point from 5 to 9. This is to be expected because even a very small corrective action by the controller to

increase and decrease base flow rate in this region will cause a large variation in the process output, pH.

Table 5.9 Initial operating conditions of the pH CSTR

Parameter	**Value**	Unit
V	1000	l
F_1	81	l/min
F_2	515	l/min
C_1	0.32	mol/l
C_2	0.05	mol/l
x_1	0.0432	mol/l
X_2	0.0435	mol/l

The performance of ANFIS controller is also compared with that of the NN controller studied by Jutan and Krishnapura (2000) for the same system under study. The process response with NN controller is presented in Figure 5.33. It is seen from the figures that although system responses follow the same trajectory, the process controlled with ANFIS controller is slower and reaches the steady state values later with the oscillations. Actually, both controllers, ANFIS and NN, use the backpropagation algorithm with constant learning rate, but performance of NN is better than that of ANFIS. This points out that classical backpropagation algorithm is not effective for ANFIS controller. Therefore, learning rate, η, during the simulation can be decreased or increased to improve the convergence performance of the controller. Although computational power is needed, controller performance can also be increased by using the hybrid-learning algorithm. Besides, inputs to the controller can be increased and type of MFs for inputs and outputs can be changed or to improve the performance of ANFIS. As previously stated, the sigmoidal function is used to approximate the plant Jacobian. It can be seen from the Figure 5.21 that actual steady state gain

of the system curve cannot be captured with this approximation method. Thus, other alternative plant Jacobian methods can be used to improve the controller performance. For instance, a crude estimate can be obtained by approximating plant Jacobian directly from the changes in the plant's input and output during the consecutive sampling instants.

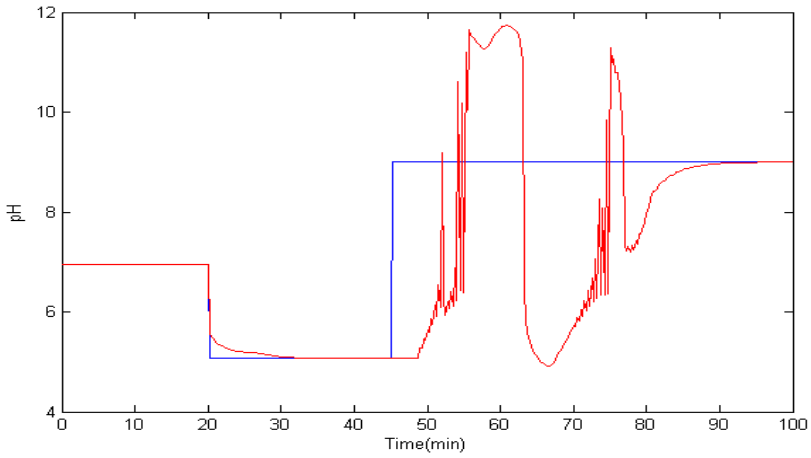

Figure5.32 pH control-set point tracking with ANFIS controller

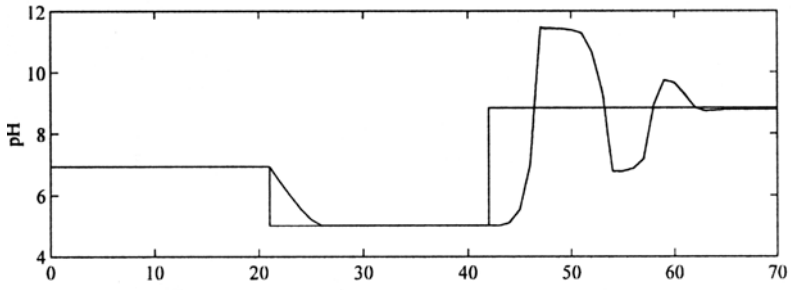

Figure5.33 pH control-set point tracking with NN controller

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

In this study, estimation of the compositions from the trays temperature measurements in continuous and batch distillation columns, and the pH control in CSTR are achieved by using the ANFIS architecture.

1. In continuous distillation column, ANFIS structures are trained and tested to infer one of the components of the top and bottom product compositions. It is concluded that all estimators' structures verification and generalization capabilities are very good especially in feed flow rate changes. Triangular structures are better than Gaussian structures that are used in membership functions (MF).

2. Best performance is obtained by Tri3lin (three triangular MFs for each input and linear output MF) ANFIS structure for both top and bottom product estimation. Tri3lin ANFIS estimator performance is compared with NN estimator and it is seen that performance of the ANFIS is better than that of NN.

3. In batch distillation column, three parallel Triangular ANFIS structures are trained and tested for the three components in reflux drum. It is seen that, structures that have constant output MF

performances are better. The best performance is obtained by Tri3con ANFIS structure for all compositions in reflux drum. Tri3con ANFIS estimator is compared with EKF estimator and it is seen that ANFIS is better than EKF.

4. In batch distillation column, designed estimator using two tray temperatures is also evaluated. It is seen that, the estimator performance with two tray temperatures is as good three tray temperatures. However, initial composition values are not estimated accurately.

5. The performance of ANFIS is also investigated in pH control problem. ANFIS structure is used as a controller in adaptive Neuro-Fuzzy control scheme for set point tracking. It is concluded that the ANFIS controller drives the closed loop system to the desired set point values.

6. ANFIS controller performance is compared with NN controller. It is concluded that convergence of ANFIS with backpropagation algorithm is slower than that of NN.

7. As a recommendation for further studies, using hybrid-learning algorithm that combines the backpropagation learning algorithm and least squares estimates can increase the performance of the ANFIS especially in controller applications.

8. If sufficient process input-output data is collected form the system, ANFIS methodology can be used for estimation and modeling purposes in chemical processes.

9. ANFIS can be also used for control purposes in chemical processes especially in Neuro-Fuzzy control structures.

REFERENCES

1. Akbay, Ü., (2002), " Fuzzy and sliding mode control of a pH neutralization system", MSc. Thesis, Chem. Eng. Dep., M.E.T.U, Ankara
2. Aliev, R.A., (2001) Soft computing and its applications", Singapore; New Jersey: World Scientific
3. Alkaya, D., (1990), "Determination of a Suitable Measurement Structure for Better Control of Distillation Columns", MSc. Thesis, Middle East Technical University, Ankara.
4. Aoyama, A., Doyle, F.J., Venkatasubramanian, V., (1995), Control-affine fuzzy neural network approach for nonlinear process control", Journal of process control, Vol. 5, No. 6, pp. 375-386
5. Bafas, G., Tsekouras, G., Sarimveis, H., Rapto, C., " A fuzzy logic approach for the classification of product qualitative characteristics", Computers chem. Engng., Vol. 26, pp. 429-438
6. Bahar, A., (2003), "An Artificial Neural network Estimator Design for the Inferential Model Predictive Control of Industrial Multicomponent Distillation Column", ", MSc. Thesis, Chem. Eng. Dep., M.E.T.U, Ankara
7. Belarbi, K., Mezaache, A., Bettou, K., (2000), "Fuzzy neural network for estimation and fuzzy controller design: simulation study for a pulp batch digester", Journal of process control, Vol. 10, pp. 35-41
8. Benz, R., Menzl, S., Stühler, M., (1996), "A self adaptive computer based pH measurement and fuzzy control system", Wat. Res., Vol. 30, No. 4, pp. 981-991
9. Bhat, N., McAvoy, T.J., (1990), "Use of Neural Nets for Dynamic Modeling and Control of Chemical Process Systems", Computers chem. Engng, Vol. 14, No. 4/5, pp. 573-583
10. Budman, H., Kavchak, M., (1999), "Adaptive neural network structures for nonlinear process estimation and control", Computers and chem. Engng, Vol. 23, pp. 1209-1228

11. Bulsari, A.B., (1995), "Neural networks for chemical engineers", Amsterdam; New York: Elsevier

12. Burden, A.C., Tantelea, R.Z., Deshpade, P.B., (2003), "Control and optimize nonlinear systemm", Chemical Engineering Progress Magazine, February, pp. 63-73

13. Castillo, O., Melin, P., (2001), "Soft Computing for Control Nonlinear Dynamical System", Physica-Verlag Heidelberg, New York

14. Cutler, C.R., and Ramaker, B.L., (1979), "Dynamic Matrix Control – A Computer Control Algorithm", AIChE National Meeting, Houston, Texas.

15. Czogala, E., Leski, J., (2000), "Fuzzy and Neuro-Fuzzy Intelligent Systems", Physica-Verlag Heidelberg, New York

16. Dagli, C.H., Sun, G., Thammano, A., (1997), "Dynamic neuro fuzzy control of the nonlinear process", Comp. and chem. Engng, Vol. 33, Nos. 1-2, pp. 413-416

17. Dokucu, M., (2002), "Multivariable Model Predictive Controller Design for an Industrial Distillation Column", MSc. Thesis, Middle East Technical University, Ankara.

18. Fisher, D.G., Wong, C.H., Shah, S.L., Bourker, M.M., (2000), "Adaptive fuzzy relational predictive control", fuzzy Sets and systems, Vol. 115, pp. 247-260

19. Fong, K.F., Loh, P.A., Looi, K.O., (1995), "Neural network modelling and control strategies for a pH process", Journal of Process Control, Vol. 5, No. 6, pp. 355-362

20. Gustafson, D.E., Kessel, W.C., (1978), "Fuzzy clusters with a fuzzy covariance matrix", IEEE conference and decision and control, pp. 761-766

21. Hancheng, Q., Bocai, X., Shangzheng, L., Fogen, W., (2002), "Fuzzy neural network modeling of material properties", Journal of Material Processing Technology, Vol. 122, pp. 196-200

22. Himmelblau, D.M., MacMurray, J., C., (1995), "Modeling and Control of a Packed Distillation Column Using Artificial Neural Networks", Computers chem. Engng, Vol. 19, No. 10., pp. 1077-1088

23. Himmelblau, D.M., Kanjala, T.W., Cheng, Y., (1996), "Resolving problems in closed loop nonlinear process identification using IRN", Computers chem. Engng, Vol. 20, No. 10, pp. 1159-1176

24. Himmelblau, D.M., (2000), "Applications of Artificial Neural Networks in Chemical Engineering", Korean J. Chem. Eng., Vol. 17, No. 4, pp. 373-392

25. Hussain, M.A., (1999), "Review of the applications of neural networks in chemical process control: simulation and online implementation", Artificial Intelligence in Engineering, Vol. 13, pp. 55-68

26. Jain, C.H., (1997), " Soft computing techniques in knowledge based intelligent engineering systems: approaches and applications", Physica-Verlag Heidelberg, New York

27. Jang, R.J., (1992), "Self Learning Fuzzy Controllers Based on Temporal Backpropagation", IEEE Transections on Neural Networks, Vol. 3, No. 5, pp. 714-725

28. Jang, R.J., (1993), "ANFIS: Adaptive-Network-Based Fuzzy Inference System", IEEE Transections on Man, and Cybernetics, Vol. 23, No. 3, pp. 665-683

29. Jutan, A., Krisnapura, G.Y., (2000), "A neural adaptive controller", Chemical Engineering Science, Vol 50, pp. 3803-3812

30. Kaffashi, F., (1998), " Some sliding mode control applications in brushless dc motor and pH systems", MSc. Thesis, Chem. Eng. Dep., M.E.T.U, Ankara

31. Kartalopoulos, V.S., (1996), "Understanding Neural Networks and Fuzzy Logic: Basic Concepts and its Applications", IEEE Press, New York

32. Kaya, M., (2000), "Control of an Industrial Multi Component High Purity Distillation Column with a Model Predictive Controller", MSc. Thesis, Middle East Technical University, Ankara.

33. Kecman, V., (2001), "Learning and Soft Computing", A Bradford Book The MIT Press Cambridge, Massachusetts London, England

34. Kim, Y.H., Kim, M.H., (1995), "Experimental application of combined fuzzy and predictive control to a binary distillation column", Journal of Chemical Engineering Japan, Vol. 28, No. 5, pp. 617-620

35. Leiviska, K., (2001), "Industrial Applications of Soft Computing", Physica-Verlag Heidelberg, New York

36. Lisboa, P.G.J., (1992), " Neural Networks: current applications", London ; New York : Chapman & Hall

37. Mamdani, E.H., Assilian, S., (1975), "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller", International Journal of Man-Machine Studies, Vol. 7, pp. 1-13

38. Matko, D., Biosizzo, K.K., Skrjanc, I., (1997), "Fuzzy predictive control of highly nonlinear pH process", Computers chem. Engng, Vol. 21, pp. 613-618

39. Martinez, E.C., Wilson, J.A., (1997), "Neuro-fuzzy modeling and control of a batch process involving simultaneous reaction and distillation", Computers chem. Engng, Vol. 21, Suppl., pp. 1233-1238

40. McAvoy T.J., Bhat, N., Minderman, P.A., (1990) "Modeling Chemical Process Systems via Neural Computation", April, pp. 24-30, IEEE Control System Magazine

41. McCulloch, W.S., Pitts, W., (1943), " A logical calculus of the ideas immament in nervous activity", Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-133

42. Mujtaba, I.M., Macchietto, S., (1993), "Optimal Operation of Multicomponent Batch Distillation-Multiperiod Formulation and Solution", Computers chem. Engng, Vol. 17, pp. 1191-1207

43. Nasrellah, K., (1998) "Simulation study of an adaptive fuzzy knowledge based controller (FKBC) and neural network (NNC) on a pH system", MSc. Thesis, Chem. Eng. Dep., M.E.T.U, Ankara

44. Peng, S.T., Chen, T.C., (1999), "Intelligent process control using neuro fuzzy techniques", Journal of Process Control, Vol. 9, pp. 493-503

45. Prode, H., Dubios, D., (1992), "Fuzzy sets in approximate reasoning, Part 1: Inference with possibility distrubitions", Fuzzy sets and Systems, Vol. 40, pp. 203-244

46. Rao, D.H., Gupta, M.M., (1994), "On the principles of fuzzy neural networks", Fuzzy Sets and Systems, Vol. 61, pp. 1-18

47. Reznik, L., Dimitrov, V., Kacprzyk, J., (1998), " Fuzzy systems design: social and engineering applications", Physica-Verlag Heidelberg, New York

48. Roffel, B., Betler, H.L., Pascal, F.C., (2003), "Combining prior knowledge with data driven modeling of a batch distillation column including start-up", Computers chem. Engng, Vol. 27, pp. 1021-1030

49. Rosenblatt, F., (1958), "The perceptron: A probabilistic model for information storage and organization in the brain", Psychological Review, Vol. 65, pp. 386-408

50. Rummelhart, D., McClelland, J., (1986) " Parallel Distrubeted Processing", MIT Press, Cambridge

51. Rutkowska, D., (2002), "Neuro-Fuzzy Architectures and Hybrid Learning", Physica-Verlag Heidelberg, New York

52. Saerens, M., Soquet, A., (1991), "Neural controller based on backpropagation algorithm", IEEE procedings Part F, 138, pp. 155-162

53. Sain, S., (1989), "Internal model control studies in a pH control loop", MSc. Thesis, Chem. Eng. Dep., M.E.T.U, Ankara

54. Sarimveis, H., Alexandridis, A., Tsekouras, G., Bafas, G., (2002), "A Fast and Efficient Algorithm for Training Radial Basis Function Neural Networks Based on Fuzzy Partition of the Input Space", Ind. Eng. Chem. Res., Vol. 41, pp. 751-759

55. Sinha, K.N., Gupta, M.M., (2000), "Soft computing and intelligent systems: theory and applications", Academic Press, San Diego

56. Takagi, T., Sugeno, M., (1985), "Fuzzy Identification of Systems and its Applications to Modeling and Control", IEEE Transactions on Systems, Man and Cybernetics, Vol. 15, pp. 116-132

57. Tsoukalas, L.H. and Uhrig, R.E., (1997), "Fuzzy and Neural Approaches in Engineering", John Wiley & Sons Inc.

58. Uchikawa, Y., Furushashi, T., Horikawa, S., (1992), "On Fuzzy Modeling Using Fuzzy Neurral Networks with the Backpropagation Algorithm", IEEE Transections on Neural Networks, Vol. 3, No. 5, pp. 801-806

59. Uchikawa, Y., Furushashi, T., Horikawa, S., Hasegewa, T., (1995), "On design of adaptive fuzzy controller using fuzzy neural networks and a description of its dynamical behaviour", Fuzzy Sets and Systems, Vol. 71, pp. 3-23

Venkateswarlu, C., Avantika, S., (2001), "Optimal state estimation of multicomponent batch distillation", Chemical Engineering Science, Vol. 56, pp. 5771-5786

60. Wang, H., Youngseok, O., Yoon, E.S., (1998), "Strategies for modeling and control of nonlinear chemical processes using neural networks", Computers chem. Engng, Vol. 22, Supp., pp. 823-826

61. Wright, R.A., Kravaris, C., (1991), " Nonlinear Control of pH Processes Using the Strong Acid Eqiuvalent", Ind. Eng. Chem. Res., Vol. 30, No. 7

62. Yamamura, A.A., Sideris, A., Psaltis, D., (1988), "A Multilayered Neural Network Controller", IEEE Control Systems Magazine, April, pp. 17-21

63. Ydstie, B.E., (1990), "Forecasting and control using adaptive connectionist networks", Computers chem. Engng, Vol. 14, pp. 583-599

64. Yıldız, U., (2003) "Multicomponent Batch Distillation Column Simulation and State Observer", MSc. Thesis, Middle East Technical University, Ankara.

65. Zadeh, L.A., (1965), "Fuzzy Sets", Journal of Information Sciences, Vol. 8, pp. 338-353

66. Zadeh, L.A., (1971), "Similarity Relations and Fuzzy Ordering", Journal of Information Sciences, Vol. 3, pp. 177-206

APPENDIX A

FIGURES

A.1 Verification results in continuous distillation column

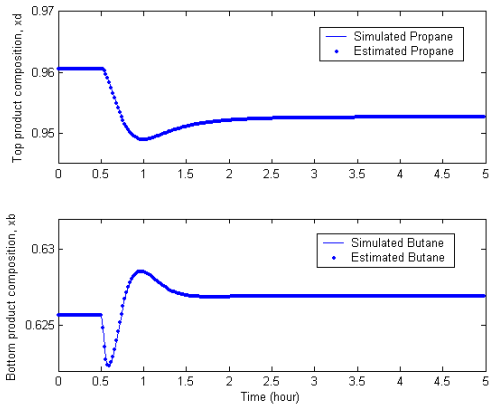

Figure A.1 All structure responses to 5% increase in feed flowrate

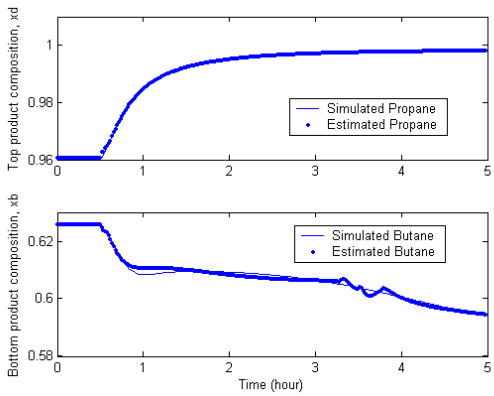

Figure A.2 Tri3con structure response to 4% increase in Reflux rate

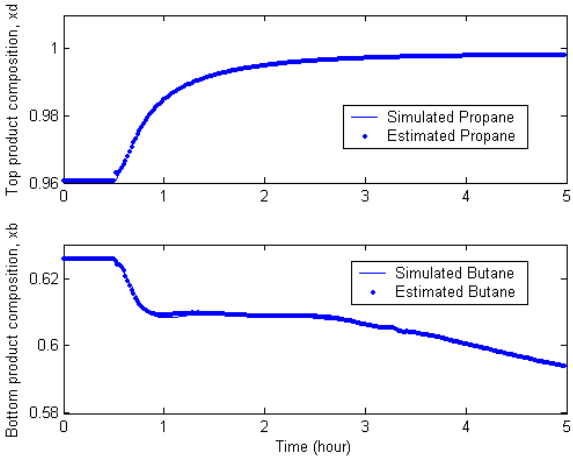

Figure A.3 Tri3lin structure response to 4% increase in Reflux rate

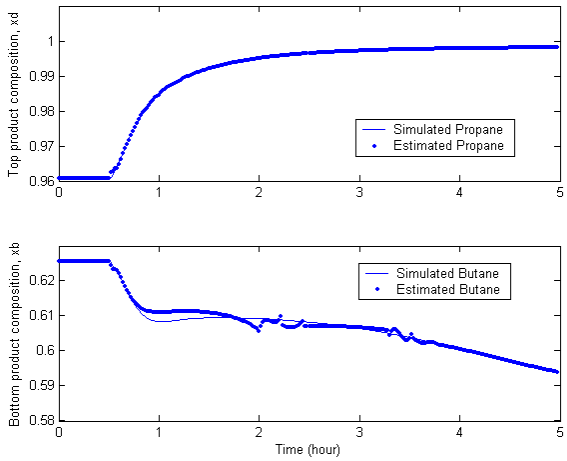

Figure A.4 Tri5con structure response to 4% increase in Reflux rate

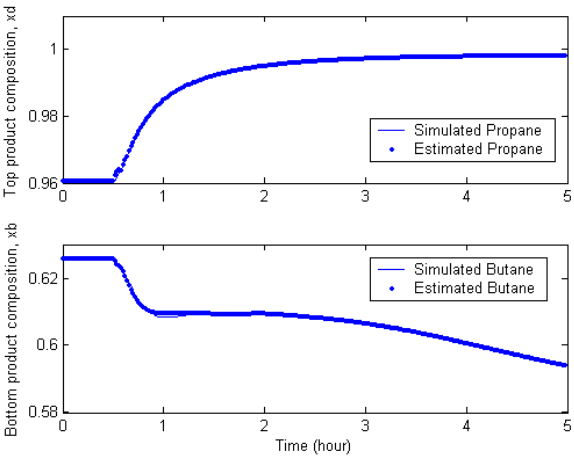

Figure A.5 Tri5lin structure response to 4% increase in Reflux rate

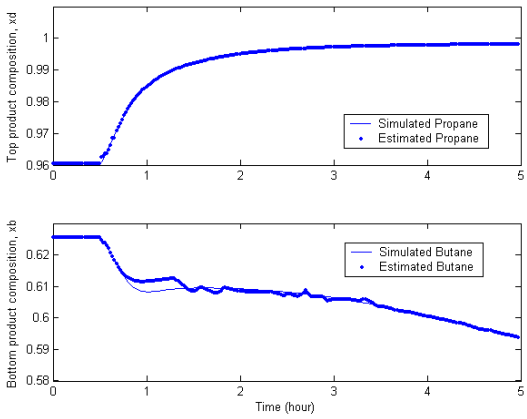

Figure A.6 Tri7con structure response to 4% increase in Reflux rate

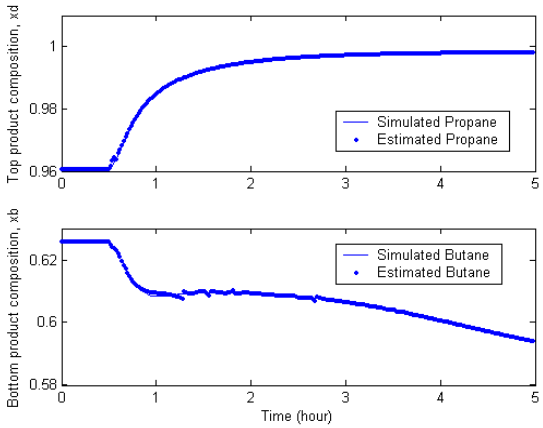

Figure A.7 Tri7lin structure response to 4% increase in Reflux rate

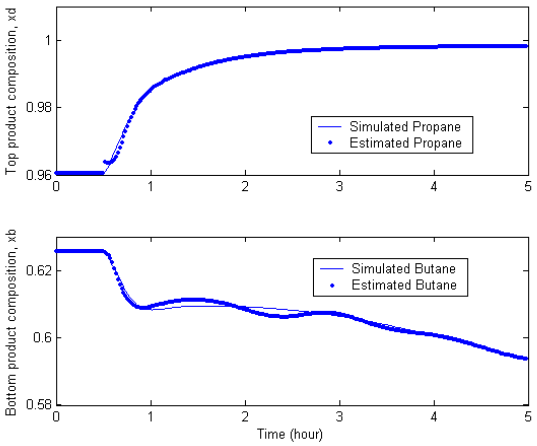

Figure A.8 Gauss3con structure response to 4% increase in Reflux rate

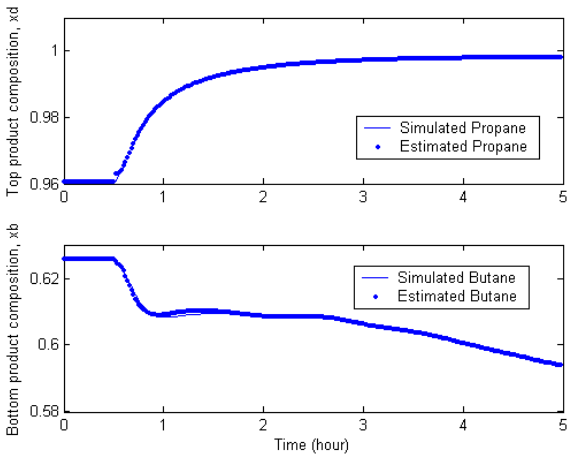

Figure A.9 Gauss3lin structure response to 4% increase in Reflux rate

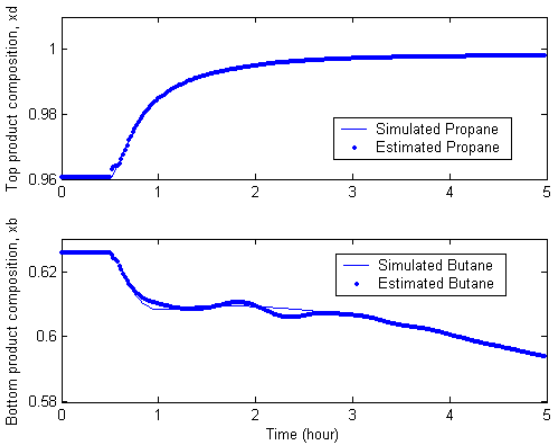

Figure A.10 Gauss5con structure response to 4% increase in Reflux rate

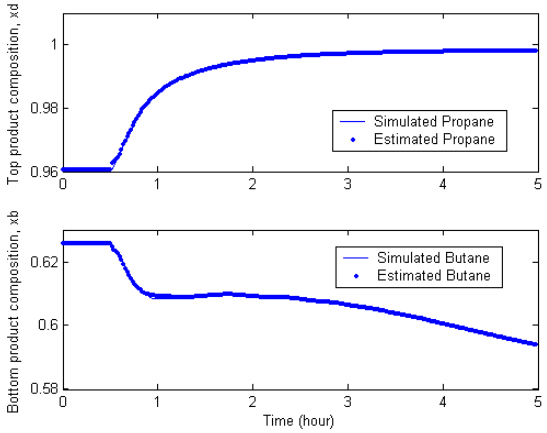

Figure A.11 Gauss5lin structure response to 4% increase in Reflux rate

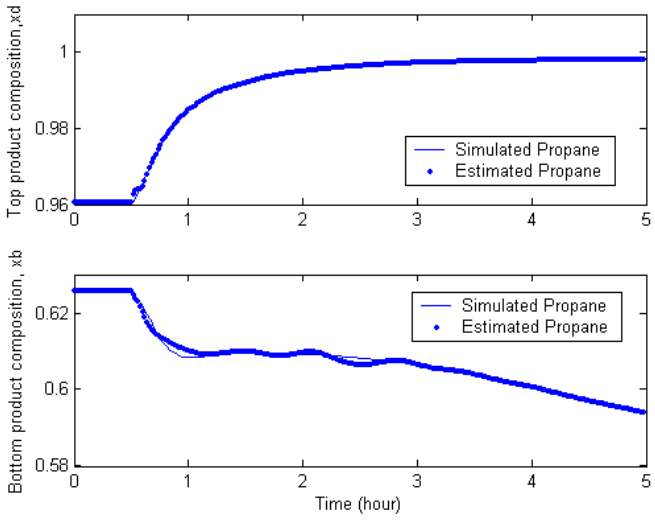

Figure A.12 Gauss7con structure response to 4% increase in Reflux rate

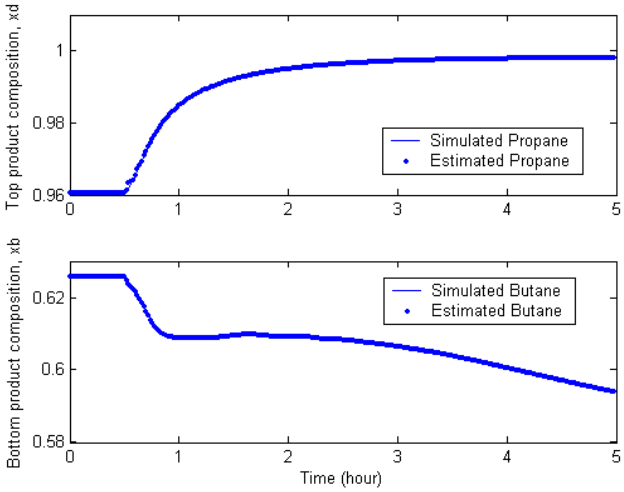

Figure A.13 Gauss7lin structure response to 4% increase in Reflux rate

A.2 Generalization results in continuous distillation column

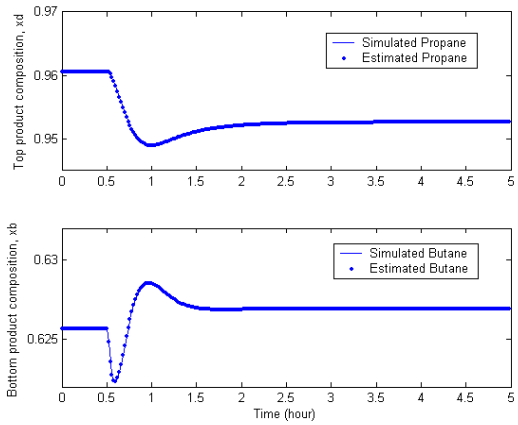

Figure A.14 All structures responses to 7% increase in feed flow rate

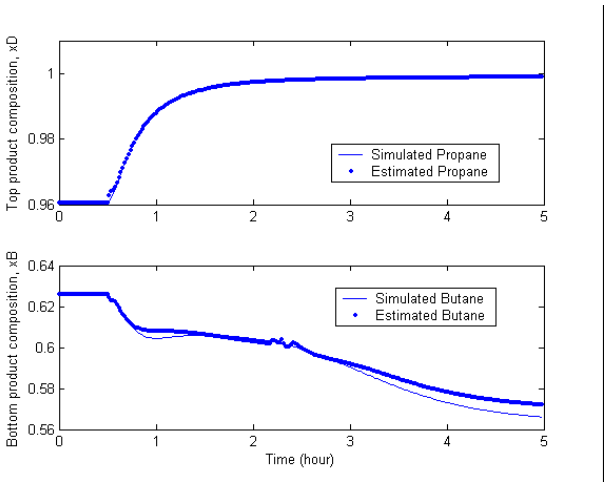

Figure A.15 Tri3con structure response to 5% increase in reflux rate

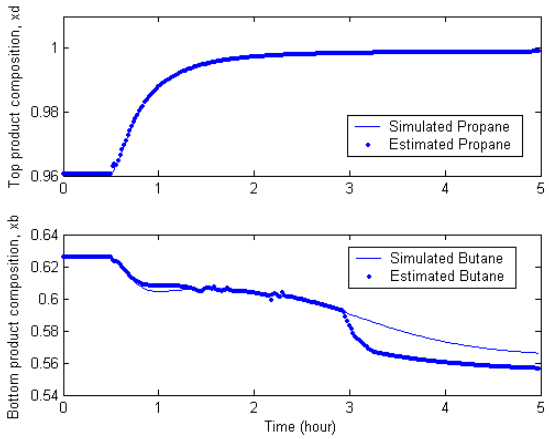

Figure A.16 Tri5con structure response to 5% increase in reflux rate

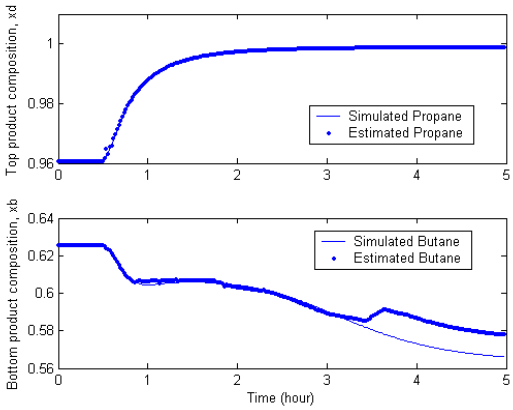

Figure A.17 Tri7lin structure response to 5% increase in reflux rate

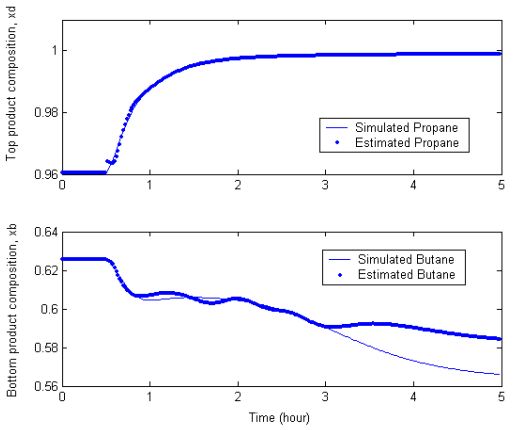

Figure A.18 Gauss3con structure response to 5% increase in reflux rate

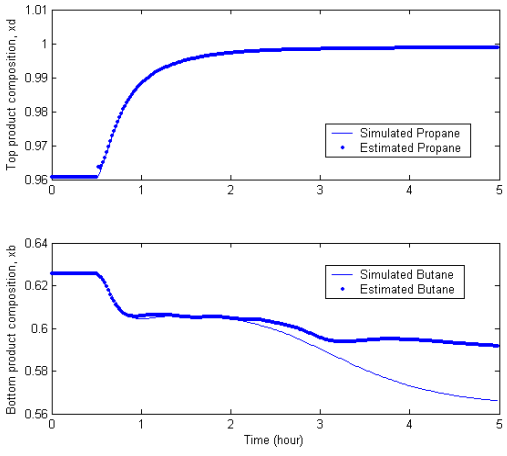

Figure A.19 Gauss3lin structure response to 5% increase in reflux rate

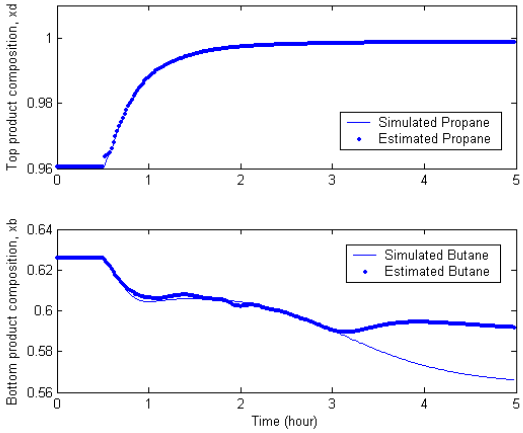

Figure A.20 Gauss5con structure response to 5% increase in reflux rate

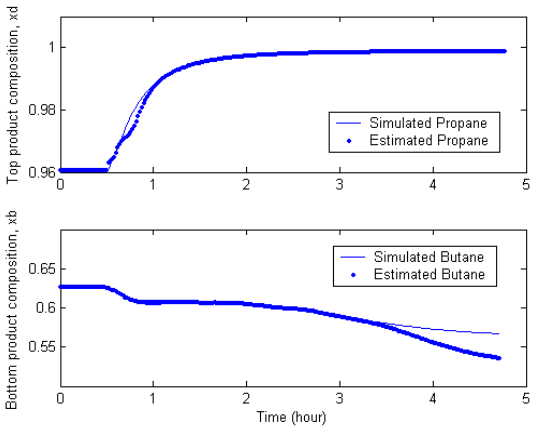

Figure A.21 Gauss5lin structure response to 5% increase in reflux rate

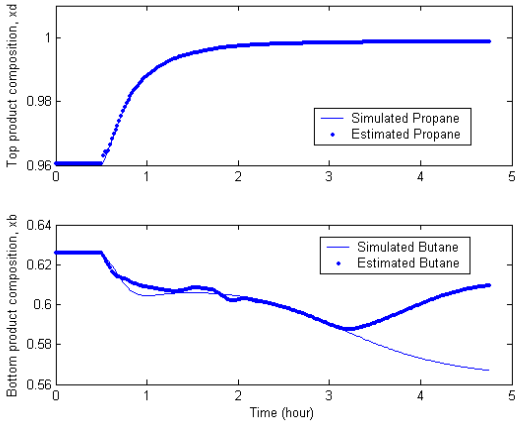

Figure A.22 Gauss7con structure response to 5% increase in reflux rate

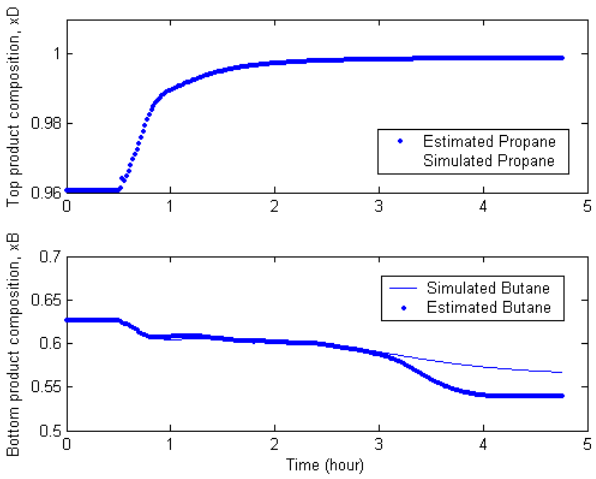

Figure A.23 Gauss7lin structure response to 5% increase in reflux rate

A.3 Verification results in batch distillation column

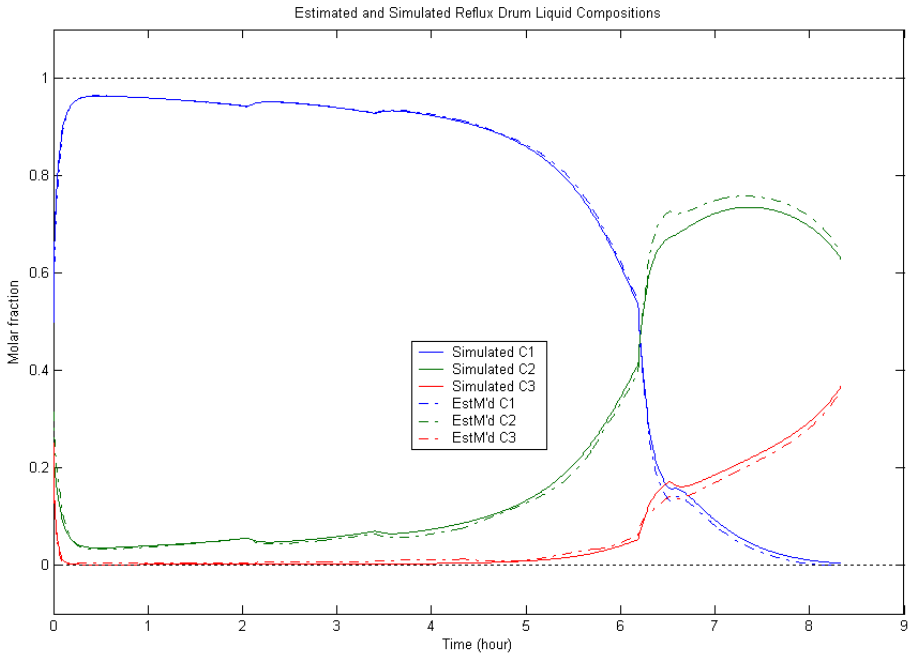

Figure A.24 Verification performance of Tri3con structure

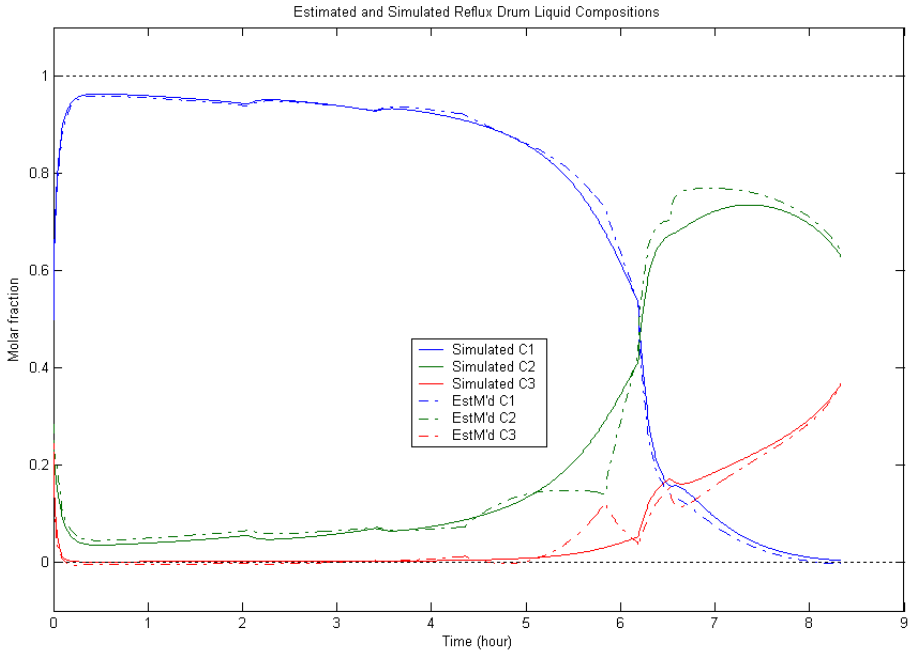

Figure A.25 Verification performance of Tri3lin structure

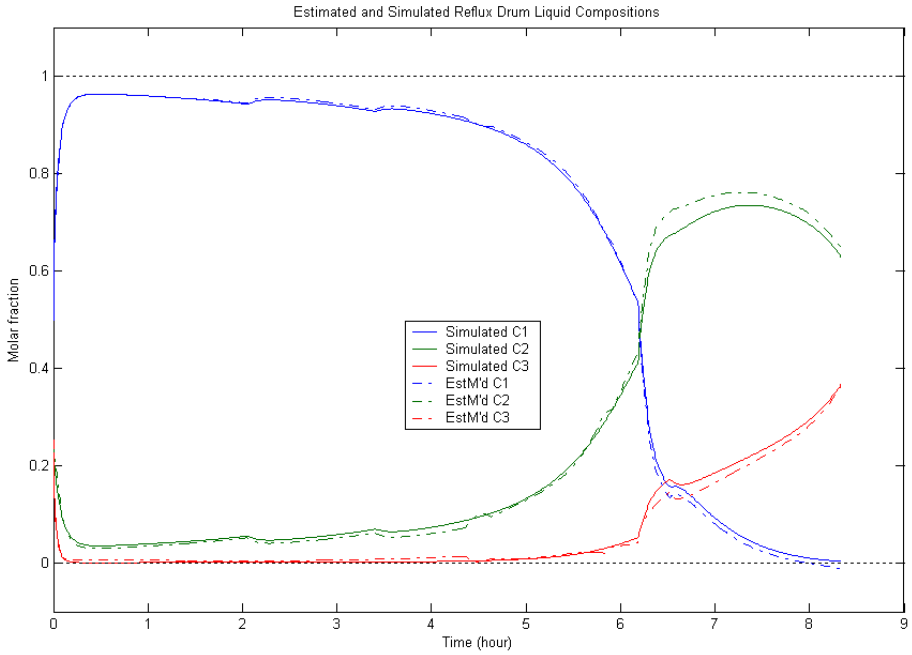

Figure A.26 Verification performance of Tri5con structure

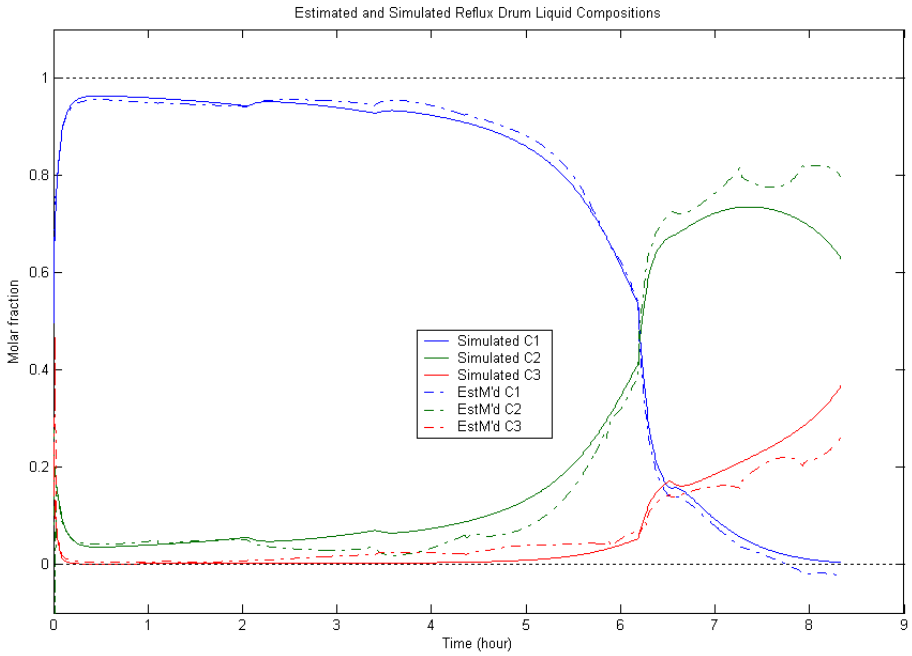

Figure A.27 Verification performance of Tri5lin structure

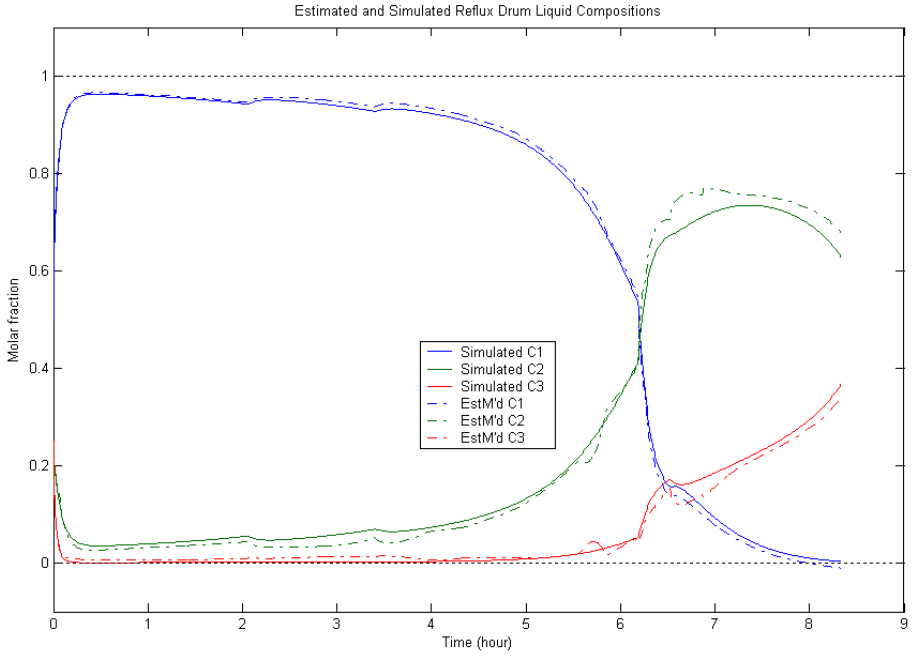

Figure A.28 Verification performance of Tri7con structure

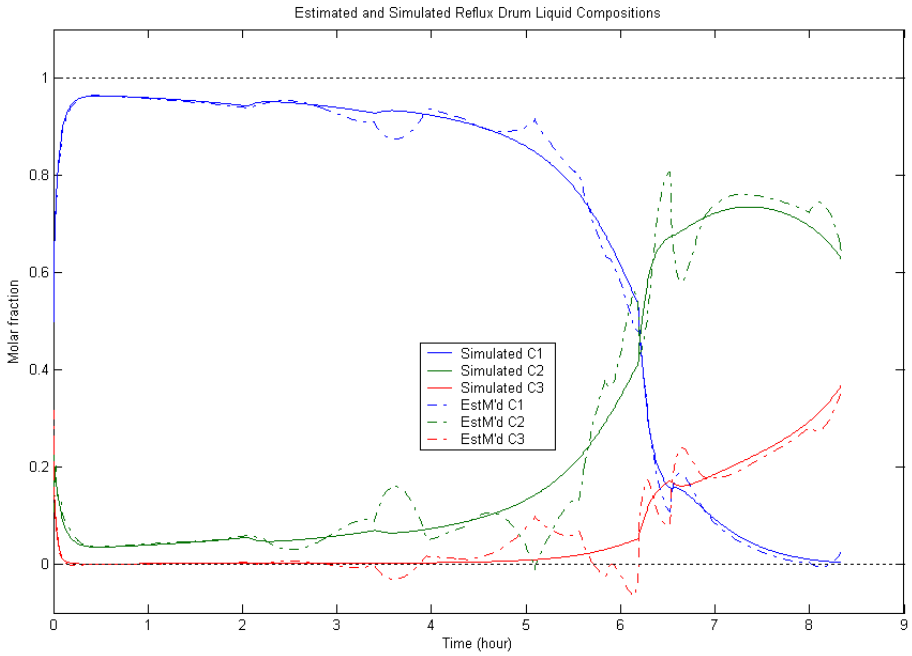

Figure A.29 Verification performance of Tri7lin structure

A.4 Input MFs for estimators

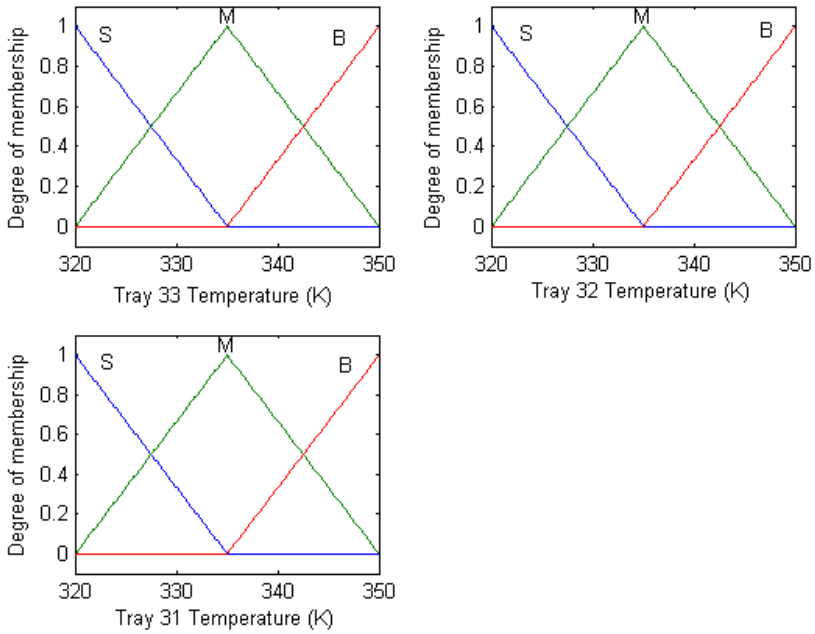

Figure A.30 Input MFs for top product composition estimator

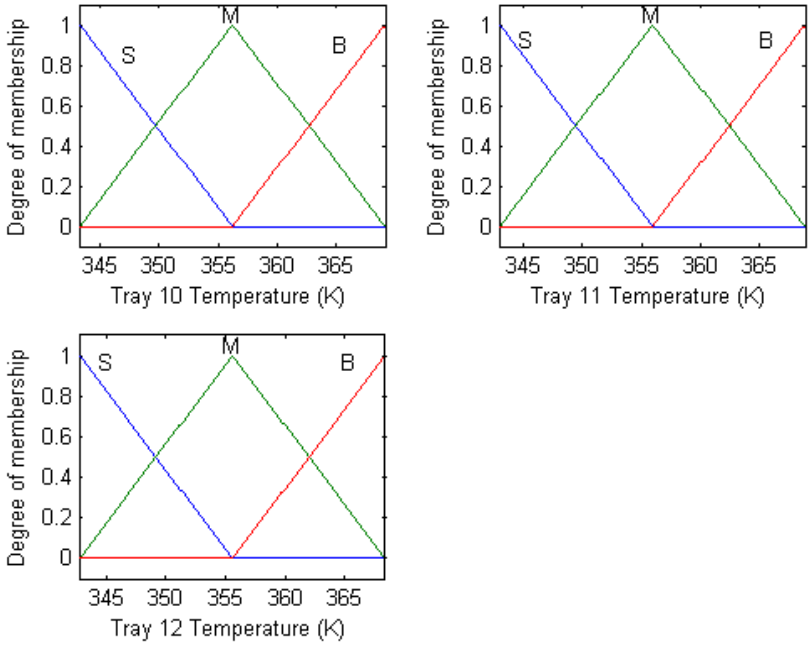

Figure A.31 Input MFs for bottom product composition estimator

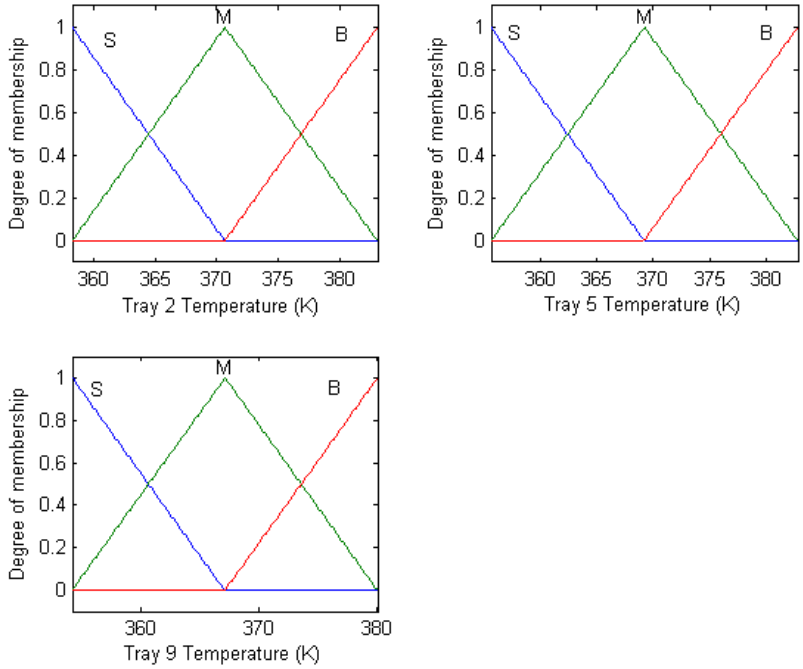

Figure A.32 Input MFs for reflux drum composition C_1 estimator

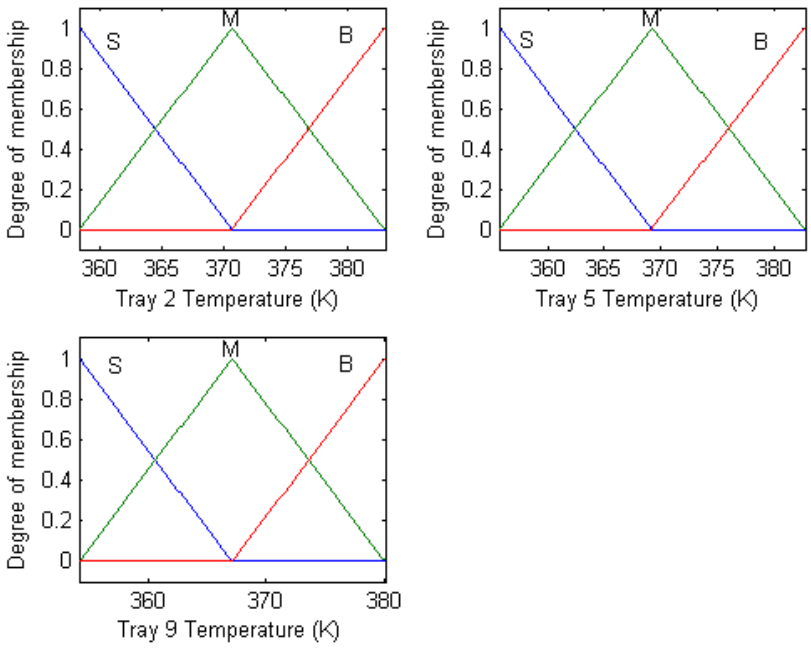

Figure A.33 Input MFs for reflux drum composition C_2 estimator

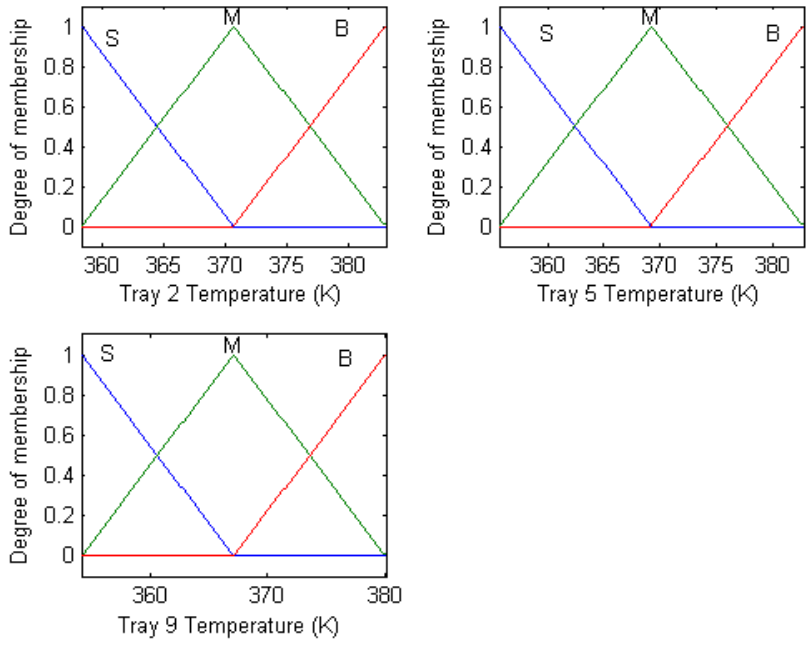

Figure A.34 Input MFs for reflux drum composition C_3 estimator

APPENDIX B

SOURCE CODES, FUNCTIONS AND SYSTEM FILES

B.1 Training.m

```
% ANFIS
% data set
% Training data sets ( Traindatxd, Traindatxb )

 load('Traindatxd.mat');

% Training parameters
%trnOpt(1): number of training epochs
Number_of_epoch=10  ;
%trnOpt(2): error tolerance
error_tolerance=1e-5 ;
%trnOpt(3): initial step-size
int_step_size=0.01 ;
%trnOpt(4): step-size decrease rate
ss_dec_rate=0.9  ;
%trnOpt(5): step-size increase rate
ss_inc_rate=1.1;

 % ANFIS  Esimator training

 anfis=readfis('initanf1');
[fismat1,error1,ss] =anfis(Traindatxd,anfis,...
[Number_of_epoch error_tolerance int_step_size ss_dec_rate ss_inc_rate],[1 1 1 1],1);
writefis(fismat1,'anfisxd');
save('trainxdinall');
```

B.2 ANFIS Tri3lin estimator structure for top product comp.

```
 [system]
Name='anfis estimator for top porduct composition'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=27
AndMethod='prod'
OrMethod='probor'
ImpMethod='prod'
AggMethod='max'
DefuzzMethod='wtaver'
```

[Input1]
Name='Tray-33-Temperature'
Range=[320 350]
NumMFs=3
MF1='S':'trimf',[305 320 335]
MF2='M':'trimf',[320 335 350]
MF3='B':'trimf',[335 350 365]

[Input2]
Name='Tray-32-Temperature'
Range=[320 350]
NumMFs=3
MF1='S':'trimf',[305 320 335]
MF2='M':'trimf',[319.920634920635 334.920634920635 349.920634920635]
MF3='B':'trimf',[335 350 365]

[Input3]
Name='Tray-31-Temperature'
Range=[320 350]
NumMFs=3
MF1='S':'trimf',[305 320 335]
MF2='M':'trimf',[320 335 350]
MF3='B':'trimf',[335 350 365]

[Output1]
Name='Top-product-comp.'
Range=[0.898131644 0.99937822]
NumMFs=27
MF1='out1mf1':'linear',[0.22313658093071 -0.101577676329315 -0.118383541149573 -0.00187249919229055]
MF2='out1mf2':'linear',[-0.0351757456478434 -0.028975746592769 0.067072516405571
0.00163548870682386]
MF3='out1mf3':'linear',[0 0 0 0]
MF4='out1mf4':'linear',[0.211498804326545 -0.102040268548731 -0.0961474825789182
0.00118084118080798]
MF5='out1mf5':'linear',[-0.0532897398973157 0.0700247770940657 -0.00829069403658541 -
0.00143972345287384]
MF6='out1mf6':'linear',[0 0 0 0]
MF7='out1mf7':'linear',[0 0 0 0]
MF8='out1mf8':'linear',[0 0 0 0]
MF9='out1mf9':'linear',[0 0 0 0]
MF10='out1mf10':'linear',[0.0422249759832046 0.174316019595751 -0.225029353687692 -
0.00303042066941992]
MF11='out1mf11':'linear',[-0.269896536737564 0.217150843689906 0.0386028391136344
0.00233823314152329]
MF12='out1mf12':'linear',[0 0 0 0]
MF13='out1mf13':'linear',[0.133559345483427 -0.170226876161504 0.0518939698721778
0.0177952298743884]
MF14='out1mf14':'linear',[-0.0172076320955233 -0.177358824942259 0.197349605997108 -
0.0170627963722117]
MF15='out1mf15':'linear',[-0.0181907550557125 -0.0138072088995837 0.0254045586067765
0.000846318157304845]
MF16='out1mf16':'linear',[-0.352585102378571 0.359302995086343 0.0362991960113568 -
0.0132236435571341]
MF17='out1mf17':'linear',[0.0717587597464706 -0.143330024569982 0.0902429198071236
0.0138978977081317]
MF18='out1mf18':'linear',[-0.0103333059978692 0.0229049196456512 0.0232688078446274 -
0.000784053920055757]
MF19='out1mf19':'linear',[-0.162261058907324 -0.0970886227997957 0.248190428690175
0.00490711761782341]
MF20='out1mf20':'linear',[0.117414214791879 -0.040747606431513 -0.108192946316506 -
0.00426949300240609]
MF21='out1mf21':'linear',[0 0 0 0]
MF22='out1mf22':'linear',[0.0105241342648875 0.317754311881693 -0.321345105526963 -
0.0170962540169099]

MF23='out1mf23':'linear',[-0.137341000997744 -0.0910454817863009 0.227000108646743 0.0170540282304605]
MF24='out1mf24':'linear',[-0.0406378624141448 0.00197467814367782 -0.00570202424669482 -0.000978129141067654]
MF25='out1mf25':'linear',[0.219153117784947 -0.0811445303977888 -0.105645061269921 0.0122756817340606]
MF26='out1mf26':'linear',[0.0213397476861223 -0.0840424415286495 0.0803940527241428 -0.0130362777873095]
MF27='out1mf27':'linear',[0.0802006376359789 0.00287745417725242 -0.0811585007774142 0.000950072554021044]

[Rules]
1 1 1, 1 (1) : 1
1 1 2, 2 (1) : 1
1 1 3, 3 (1) : 1
1 2 1, 4 (1) : 1
1 2 2, 5 (1) : 1
1 2 3, 6 (1) : 1
1 3 1, 7 (1) : 1
1 3 2, 8 (1) : 1
1 3 3, 9 (1) : 1
2 1 1, 10 (1) : 1
2 1 2, 11 (1) : 1
2 1 3, 12 (1) : 1
2 2 1, 13 (1) : 1
2 2 2, 14 (1) : 1
2 2 3, 15 (1) : 1
2 3 1, 16 (1) : 1
2 3 2, 17 (1) : 1
2 3 3, 18 (1) : 1
3 1 1, 19 (1) : 1
3 1 2, 20 (1) : 1
3 1 3, 21 (1) : 1
3 2 1, 22 (1) : 1
3 2 2, 23 (1) : 1
3 2 3, 24 (1) : 1
3 3 1, 25 (1) : 1
3 3 2, 26 (1) : 1
3 3 3, 27 (1) : 1

B.3 ANFIS Tri3lin estimator structure for bottom product comp.

[system]
Name='anfisbottom'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=27
AndMethod='prod'
OrMethod='probor'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='wtaver'

[Input1]
Name='Tray-10-Temperature'
Range=[343.1713556 369.2226122]
NumMFs=3
MF1='S':'trimf',[330.1457273 343.1713556 356.1969839]
MF2='M':'trimf',[343.102572531217 356.128200831217 369.153829131217]
MF3='B':'trimf',[356.1969839 369.2226122 382.2482405]

[Input2]

Name='Tray-11-Temperature'
Range=[342.9109127 368.8577226]
NumMFs=3
MF1='S':'trimf',[329.93750775 342.9109127 355.88431765]
MF2='M':'trimf',[342.842129631217 355.815534581217 368.788939531217]
MF3='B':'trimf',[355.88431765 368.8577226 381.83112755]

[Input3]
Name='Tray-12-Temperature'
Range=[342.7218287 368.4181872]
NumMFs=3
MF1='S':'trimf',[329.87364945 342.7218287 355.57000795]
MF2='M':'trimf',[342.7218287 355.57000795 368.4181872]
MF3='B':'trimf',[355.57000795 368.4181872 381.26636645]

[Output1]
Name='Bottom-product-comp.'
Range=[0.525944598 0.648049865]
NumMFs=27
MF1='out1mf1':'linear',[-0.349006709314208 0.789861798976041 -0.438841308414577 -0.156602408449348]
MF2='out1mf2':'linear',[-0.67158482894933 0.102610826707204 0.678257844206966 0.151617063059198]
MF3='out1mf3':'linear',[0 0 0 0]
MF4='out1mf4':'linear',[-0.0357790665226616 -0.176496225844082 0.0201025449450525 0.152617122139784]
MF5='out1mf5':'linear',[-0.0121620677174623 -0.237844065072512 0.204774866626714 -0.146470007480143]
MF6='out1mf6':'linear',[0 0 0 0]
MF7='out1mf7':'linear',[0 0 0 0]
MF8='out1mf8':'linear',[0 0 0 0]
MF9='out1mf9':'linear',[0 0 0 0]
MF10='out1mf10':'linear',[0.611946578276047 -0.267276453176755 -0.258492680271999 0.151840486062888]
MF11='out1mf11':'linear',[0.0072712135116738 -0.065861592075415 0.176425659316952 -
0.146584680729087]
MF12='out1mf12':'linear',[0 0 0 0]
MF13='out1mf13':'linear',[0.28470983444281 0.0265516132188946 -0.377895538163907 -0.146312777108544]
MF14='out1mf14':'linear',[-0.666352781526469 1.41395662740005 -0.745954947506552 0.196840685060586]
MF15='out1mf15':'linear',[-0.185469225025512 -0.720802132691032 0.975814053013067 -
0.054270116616474]
MF16='out1mf16':'linear',[-0.282484424828341 -0.0835110836211585 0.160057566493536 -
0.000871806849476438]
MF17='out1mf17':'linear',[0.351404643466166 -0.512263928259577 0.0457526583637452 -
0.0513780630297902]
MF18='out1mf18':'linear',[1.02112944858637 -0.443932167824431 -0.645573377038038 0.0492036576477876]
MF19='out1mf19':'linear',[0.10285016134605 0.0632682468118817 0.0152074531447263 5.99305282609536e-
005]
MF20='out1mf20':'linear',[0.273364438202161 0.0624189200358236 -0.195517227884446 -
0.000564865625630779]
MF21='out1mf21':'linear',[0 0 0 0]
MF22='out1mf22':'linear',[0.0682866512560048 0.00416992355938628 -0.0208292080190554
0.000161319022466568]
MF23='out1mf23':'linear',[0.720757034862477 -0.178655381411407 -0.494744086589861 -
0.0536465120139853]
MF24='out1mf24':'linear',[0.0315700630529699 -0.567226860643969 0.630942120956261
0.0519387277108186]
MF25='out1mf25':'linear',[-0.0694729644377239 -0.0308579107465855 0.0178180752333713 -
0.000112188197231817]
MF26='out1mf26':'linear',[-0.0711607880729906 -0.606822295538997 0.65322974638599
0.0503245313074935]
MF27='out1mf27':'linear',[-1.09296199619905 1.47076997460625 -0.375289370909959 -0.0476623763197282]

[Rules]
1 1 1, 1 (1) : 1
1 1 2, 2 (1) : 1
1 1 3, 3 (1) : 1
1 2 1, 4 (1) : 1
1 2 2, 5 (1) : 1
1 2 3, 6 (1) : 1
1 3 1, 7 (1) : 1

1 3 2, 8 (1) : 1
1 3 3, 9 (1) : 1
2 1 1, 10 (1) : 1
2 1 2, 11 (1) : 1
2 1 3, 12 (1) : 1
2 2 1, 13 (1) : 1
2 2 2, 14 (1) : 1
2 2 3, 15 (1) : 1
2 3 1, 16 (1) : 1
2 3 2, 17 (1) : 1
2 3 3, 18 (1) : 1
3 1 1, 19 (1) : 1
3 1 2, 20 (1) : 1
3 1 3, 21 (1) : 1
3 2 1, 22 (1) : 1
3 2 2, 23 (1) : 1
3 2 3, 24 (1) : 1
3 3 1, 25 (1) : 1
3 3 2, 26 (1) : 1
3 3 3, 27 (1) : 1

B.4 Rule base of the top product estimator

If (Tray 31 Temp. is S) and (Tray 32 Temp. is S) and (Tray 33 Temp. is S) then Top prod. Comp. is outmf1
If (Tray 31 Temp. is S) and (Tray 32 Temp. is S) and (Tray 33 Temp. is M) then Top prod. Comp. is outmf2
If (Tray 31 Temp. is S) and (Tray 32 Temp. is S) and (Tray 33 Temp. is B) then Top prod. Comp. is outmf3
If (Tray 31 Temp. is S) and (Tray 32 Temp. is M) and (Tray 33 Temp. is S) then Top prod. Comp. is outmf4
If (Tray 31 Temp. is S) and (Tray 32 Temp. is M) and (Tray 33 Temp. is M) then Top prod. Comp. is outmf5
If (Tray 31 Temp. is S) and (Tray 32 Temp. is M) and (Tray 33 Temp. is B) then Top prod. Comp. is outmf6
If (Tray 31 Temp. is S) and (Tray 32 Temp. is B) and (Tray 33 Temp. is S) then Top prod. Comp. is outmf7
If (Tray 31 Temp. is S) and (Tray 32 Temp. is B) and (Tray 33 Temp. is M) then Top prod. Comp. is outmf8
If (Tray 31 Temp. is S) and (Tray 32 Temp. is B) and (Tray 33 Temp. is B) then Top prod. Comp. is outmf9
If (Tray 31 Temp. is M) and (Tray 32 Temp. is S) and (Tray 33 Temp. is S) then Top prod. Comp. is outmf10
If (Tray 31 Temp. is M) and (Tray 32 Temp. is S) and (Tray 33 Temp. is M) then Top prod. Comp. is outmf11
If (Tray 31 Temp. is M) and (Tray 32 Temp. is S) and (Tray 33 Temp. is B) then Top prod. Comp. is outmf12
If (Tray 31 Temp. is M) and (Tray 32 Temp. is M) and (Tray 33 Temp. is S) then Top prod. Comp. is outmf13
If (Tray 31 Temp. is M) and (Tray 32 Temp. is M) and (Tray 33 Temp. is M) then Top prod. Comp. is outmf14
If (Tray 31 Temp. is M) and (Tray 32 Temp. is M) and (Tray 33 Temp. is B) then Top prod. Comp. is outmf15
If (Tray 31 Temp. is M) and (Tray 32 Temp. is B) and (Tray 33 Temp. is S) then Top prod. Comp. is outmf16
If (Tray 31 Temp. is M) and (Tray 32 Temp. is B) and (Tray 33 Temp. is M) then Top prod. Comp. is outmf17
If (Tray 31 Temp. is M) and (Tray 32 Temp. is B) and (Tray 33 Temp. is B) then Top prod. Comp. is outmf18
If (Tray 31 Temp. is B) and (Tray 32 Temp. is S) and (Tray 33 Temp. is S) then Top prod. Comp. is outmf19
If (Tray 31 Temp. is B) and (Tray 32 Temp. is S) and (Tray 33 Temp. is M) then Top prod. Comp. is outmf20
If (Tray 31 Temp. is B) and (Tray 32 Temp. is S) and (Tray 33 Temp. is B) then Top prod. Comp. is outmf21
If (Tray 31 Temp. is B) and (Tray 32 Temp. is M) and (Tray 33 Temp. is S) then Top prod. Comp. is outmf22
If (Tray 31 Temp. is B) and (Tray 32 Temp. is M) and (Tray 33 Temp. is M) then Top prod. Comp. is outmf23
If (Tray 31 Temp. is B) and (Tray 32 Temp. is M) and (Tray 33 Temp. is B) then Top prod. Comp. is outmf24
If (Tray 31 Temp. is B) and (Tray 32 Temp. is B) and (Tray 33 Temp. is S) then Top prod. Comp. is outmf25
If (Tray 31 Temp. is B) and (Tray 32 Temp. is B) and (Tray 33 Temp. is M) then Top prod. Comp. is outmf26
If (Tray 31 Temp. is B) and (Tray 32 Temp. is B) and (Tray 33 Temp. is B) then Top prod. Comp. is outmf27

B.5 Rule base of the bottom product estimator

If (Tray 10 Temp. is S) and (Tray 11 Temp. is S) and (Tray 12 Temp. is S) then Bottom prod. Comp. is outmf1
If (Tray 10 Temp. is S) and (Tray 11 Temp. is S) and (Tray 12 Temp. is M) then Bottom prod. Comp. is outmf2
If (Tray 10 Temp. is S) and (Tray 11 Temp. is S) and (Tray 12 Temp. is B) then Bottom prod. Comp. is outmf3
If (Tray 10 Temp. is S) and (Tray 11 Temp. is M) and (Tray 12 Temp. is S) then Bottom prod. Comp. is outmf4
If (Tray 10 Temp. is S) and (Tray 11 Temp. is M) and (Tray 12 Temp. is M) then Bottom prod. Comp. is outmf5
If (Tray 10 Temp. is S) and (Tray 11 Temp. is M) and (Tray 12 Temp. is B) then Bottom prod. Comp. is outmf6
If (Tray 10 Temp. is S) and (Tray 11 Temp. is B) and (Tray 12 Temp. is S) then Bottom prod. Comp. is outmf7
If (Tray 10 Temp. is S) and (Tray 11 Temp. is B) and (Tray 12 Temp. is M) then Bottom prod. Comp. is outmf8
If (Tray 10 Temp. is S) and (Tray 11 Temp. is B) and (Tray 12 Temp. is B) then Bottom prod. Comp. is outmf9

If (Tray 10 Temp. is M) and (Tray 11 Temp. is S) and (Tray 12 Temp. is S) then Bottom prod. Comp. is outmf10
If (Tray 10 Temp. is M) and (Tray 11 Temp. is S) and (Tray 12 Temp. is M) then Bottom prod. Comp. is outmf11
If (Tray 10 Temp. is M) and (Tray 11 Temp. is S) and (Tray 12 Temp. is B) then Bottom prod. Comp. is outmf12
If (Tray 10 Temp. is M) and (Tray 11 Temp. is M) and (Tray 12 Temp. is S) then Bottom prod. Comp. is outmf13
If (Tray 10 Temp. is M) and (Tray 11 Temp. is M) and (Tray 12 Temp. is M) then Bottom prod.Comp is outmf14
If (Tray 10 Temp. is M) and (Tray 11 Temp. is M) and (Tray 12 Temp. is B) then Bottom prod. Comp is outmf15
If (Tray 10 Temp. is M) and (Tray 11 Temp. is B) and (Tray 12 Temp. is S) then Bottom prod. Comp. is outmf16
If (Tray 10 Temp. is M) and (Tray 11 Temp. is B) and (Tray 12 Temp. is M) then Bottom prod.Comp. is outmf17
If (Tray 10 Temp. is M) and (Tray 11 Temp. is B) and (Tray 12 Temp. is B) then Bottom prod. Comp. is outmf18
If (Tray 10 Temp. is B) and (Tray 11Temp. is S) and (Tray 12 Temp. is S) then Bottom prod. Comp. is outmf19
If (Tray 10 Temp. is B) and (Tray 11Temp. is S) and (Tray 12 Temp. is M) then Bottom prod. Comp. is outmf20
If (Tray 10 Temp. is B) and (Tray 11Temp. is S) and (Tray 12 Temp. is B) then Bottom prod. Comp. is outmf21
If (Tray 10 Temp. is B) and (Tray 11Temp. is M) and (Tray 12 Temp. is S) then Bottom prod. Comp. is outmf22
If (Tray 10 Temp. is B) and (Tray 11Temp. is M) and (Tray 12 Temp. is M) then Bottom prod. Comp. is outmf23
If (Tray 10 Temp. is B) and (Tray 11 Temp. is M) and (Tray 12 Temp. is B) then Bottom prod. Comp. is outmf24
If (Tray 10 Temp. is B) and (Tray 11 Temp. is B) and (Tray 12 Temp. is S) then Bottom prod. Comp. is outmf25
If (Tray 10 Temp. is B) and (Tray 11 Temp. is B) and (Tray 12 Temp. is M) then Bottom prod. Comp. is outmf26
If (Tray 10 Temp. is B) and (Tray 11 Temp. is B) and (Tray 12 Temp. is B) then Bottom prod. Comp. is outmf27

B.6 ANFIS Tri3con estimator structure for reflux drum comp C_1

[System]
Name='anfis C_1'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=27
AndMethod='prod'
OrMethod='probor'
ImpMethod='prod'
AggMethod='max'
DefuzzMethod='wtaver'

[Input1]
Name='Tray-2-Temp.'
Range=[358.256868 383.071248]
NumMFs=3
MF1='in1mf1':'trimf',[345.849678 358.257787053743 370.667990280714]
MF2='in1mf2':'trimf',[358.25664412853 370.66723709827 383.068055214585]
MF3='in1mf3':'trimf',[370.659775244075 383.073507020429 395.478438]

[Input2]
Name=' Tray-5-Temp.'
Range=[355.674577 382.802466]
NumMFs=3
MF1='in2mf1':'trimf',[342.1106325 355.675084276322 369.231529977593]
MF2='in2mf2':'trimf',[355.674152827397 369.241651973919 382.801902728499]
MF3='in2mf3':'trimf',[369.1598469572 382.805114907896 396.3664105]

[Input3]
Name=' Tray-9-Temp.'
Range=[354.170049 380.046657]
NumMFs=3
MF1='in3mf1':'trimf',[341.231745002973 354.180074059252 367.073787268364]
MF2='in3mf2':'trimf',[354.160061545438 367.1302307862 380.033419123547]
MF3='in3mf3':'trimf',[367.122538896839 380.058542849881 392.984961]

[Output]
Name='Comp. C_1'
Range=[1.5e-005 0.981329]
NumMFs=27
MF1='out1mf1':'constant',[0.982006680242058]
MF2='out1mf2':'constant',[0.190386332362015]

MF3='out1mf3':'constant',[-22.392942830294]
MF4='out1mf4':'constant',[0.947654056698508]
MF5='out1mf5':'constant',[0.500176205791321]
MF6='out1mf6':'constant',[1.98172335598331]
MF7='out1mf7':'constant',[1.63216430887809]
MF8='out1mf8':'constant',[64.2152937876542]
MF9='out1mf9':'constant',[-2.16171634197359]
MF10='out1mf10':'constant',[0.985853911189988]
MF11='out1mf11':'constant',[0.0126149008570596]
MF12='out1mf12':'constant',[-7.27752424098632]
MF13='out1mf13':'constant',[1.02073154201006]
MF14='out1mf14':'constant',[0.272507560874275]
MF15='out1mf15':'constant',[-0.637030883156609]
MF16='out1mf16':'constant',[1.23097569336719]
MF17='out1mf17':'constant',[0.274343646675519]
MF18='out1mf18':'constant',[0.310940392774676]
MF19='out1mf19':'constant',[0.739079672434684]
MF20='out1mf20':'constant',[1.86028137773717]
MF21='out1mf21':'constant',[0]
MF22='out1mf22':'constant',[0.930816885753836]
MF23='out1mf23':'constant',[0.487873652203564]
MF24='out1mf24':'constant',[-1.00721820556956]
MF25='out1mf25':'constant',[0.877354177674599]
MF26='out1mf26':'constant',[0.406048965409524]
MF27='out1mf27':'constant',[-0.0189171616149951]

[Rules]
1 1 1, 1 (1) : 1
1 1 2, 2 (1) : 1
1 1 3, 3 (1) : 1
1 2 1, 4 (1) : 1
1 2 2, 5 (1) : 1
1 2 3, 6 (1) : 1
1 3 1, 7 (1) : 1
1 3 2, 8 (1) : 1
1 3 3, 9 (1) : 1
2 1 1, 10 (1) : 1
2 1 2, 11 (1) : 1
2 1 3, 12 (1) : 1
2 2 1, 13 (1) : 1
2 2 2, 14 (1) : 1
2 2 3, 15 (1) : 1
2 3 1, 16 (1) : 1
2 3 2, 17 (1) : 1
2 3 3, 18 (1) : 1
3 1 1, 19 (1) : 1
3 1 2, 20 (1) : 1
3 1 3, 21 (1) : 1
3 2 1, 22 (1) : 1
3 2 2, 23 (1) : 1
3 2 3, 24 (1) : 1
3 3 1, 25 (1) : 1
3 3 2, 26 (1) : 1
3 3 3, 27 (1) : 1

B.7 ANFIS Tri3con estimator structure for reflux drum comp C_2

[System]
Name='anfis C_2'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=27

AndMethod='prod'
OrMethod='probor'
ImpMethod='prod'
AggMethod='max'
DefuzzMethod='wtaver'

[Input1]
Name=' Tray-2-Temp '
Range=[358.256868 383.071248]
NumMFs=3
MF1='in1mf1':'trimf',[345.849678 358.258020067984 370.650508349578]
MF2='in1mf2':'trimf',[358.256652678929 370.666544446069 383.068503718612]
MF3='in1mf3':'trimf',[370.652494858713 383.072582023573 395.478438]

[Input2]
Name=' Tray-5-Temp '
Range=[355.674577 382.802466]
NumMFs=3
MF1='in2mf1':'trimf',[342.1106325 355.675935544443 369.235575338006]
MF2='in2mf2':'trimf',[355.673253114779 369.241094568775 382.802422379147]
MF3='in2mf3':'trimf',[369.152004345474 382.803766900634 396.3664105]

[Input3]
Name=' Tray-9-Temp '
Range=[354.170049 380.046657]
NumMFs=3
MF1='in3mf1':'trimf',[341.231745000671 354.175272362572 367.091043927468]
MF2='in3mf2':'trimf',[354.164866448959 367.120903034683 380.039391874954]
MF3='in3mf3':'trimf',[367.121646703249 380.053994324431 392.984961]

[Output1]
Name='Comp. C_2'
Range=[0.017719 0.911318]
NumMFs=27
MF1='out1mf1':'constant',[0.0175128892155713]
MF2='out1mf2':'constant',[0.66194083678541]
MF3='out1mf3':'constant',[15.7519201989238]
MF4='out1mf4':'constant',[0.208583718989921]
MF5='out1mf5':'constant',[-0.324609779818212]
MF6='out1mf6':'constant',[-17.6021302022835]
MF7='out1mf7':'constant',[14.5235804704464]
MF8='out1mf8':'constant',[-164.49232448122]
MF9='out1mf9':'constant',[-1.06793546474844]
MF10='out1mf10':'constant',[-0.00379217733848338]
MF11='out1mf11':'constant',[1.22123499217628]
MF12='out1mf12':'constant',[24.4876979210084]
MF13='out1mf13':'constant',[-0.0626900990626596]
MF14='out1mf14':'constant',[0.737662134600771]
MF15='out1mf15':'constant',[1.92797448029802]
MF16='out1mf16':'constant',[-0.559988566690381]
MF17='out1mf17':'constant',[0.435961374860723]
MF18='out1mf18':'constant',[-1.08004305482104]
MF19='out1mf19':'constant',[0.63774404745321]
MF20='out1mf20':'constant',[-2.32177213105731]
MF21='out1mf21':'constant',[0]
MF22='out1mf22':'constant',[0.0782133147793849]
MF23='out1mf23':'constant',[0.336526917871792]
MF24='out1mf24':'constant',[2.44509930463474]
MF25='out1mf25':'constant',[0.661946624774005]
MF26='out1mf26':'constant',[0.368125882283921]
MF27='out1mf27':'constant',[0.259922743359131]

[Rules]
1 1 1, 1 (1) : 1
1 1 2, 2 (1) : 1
1 1 3, 3 (1) : 1

1 2 1, 4 (1) : 1
1 2 2, 5 (1) : 1
1 2 3, 6 (1) : 1
1 3 1, 7 (1) : 1
1 3 2, 8 (1) : 1
1 3 3, 9 (1) : 1
2 1 1, 10 (1) : 1
2 1 2, 11 (1) : 1
2 1 3, 12 (1) : 1
2 2 1, 13 (1) : 1
2 2 2, 14 (1) : 1
2 2 3, 15 (1) : 1
2 3 1, 16 (1) : 1
2 3 2, 17 (1) : 1
2 3 3, 18 (1) : 1
3 1 1, 19 (1) : 1
3 1 2, 20 (1) : 1
3 1 3, 21 (1) : 1
3 2 1, 22 (1) : 1
3 2 2, 23 (1) : 1
3 2 3, 24 (1) : 1
3 3 1, 25 (1) : 1
3 3 2, 26 (1) : 1
3 3 3, 27 (1) : 1

B.8 ANFIS Tri3con estimator structure for reflux drum comp C_3

[System]
Name='anfis C_3'
Type='sugeno'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=27
AndMethod='prod'
OrMethod='probor'
ImpMethod='prod'
AggMethod='max'
DefuzzMethod='wtaver'

[Input1]
Name='Tray-2-Temp.'
Range=[358.256868 383.071248]
NumMFs=3
MF1='in1mf1':'trimf',[345.849678 358.258089007431 370.64180876486]
MF2='in1mf2':'trimf',[358.256641854165 370.665215170102 383.071310865505]
MF3='in1mf3':'trimf',[370.649061840508 383.071184937993 395.478438]

[Input2]
Name=' Tray-5-Temp.'
Range=[355.674577 382.802466]
NumMFs=3
MF1='in2mf1':'trimf',[342.110632500002 355.676450718881 369.238310038246]
MF2='in2mf2':'trimf',[355.672723333751 369.239788922644 382.802571841195]
MF3='in2mf3':'trimf',[369.151728320173 382.801945678647 396.3664105]

[Input3]
Name=' Tray-9-Temp.'
Range=[354.170049 380.046657]
NumMFs=3
MF1='in3mf1':'trimf',[341.231744999983 354.174402030284 367.112961306089]

MF2='in3mf2':'trimf',[354.165731354943 367.11447951232 380.045416894431]
MF3='in3mf3':'trimf',[367.092865214924 380.04843018439 392.984961]

[Output1]
Name='Comp. C_3'
Range=[0.000138 0.773586]
NumMFs=27
MF1='out1mf1':'constant',[-0.00109007290656566]
MF2='out1mf2':'constant',[0.272462794301091]
MF3='out1mf3':'constant',[-1.94272417978808]
MF4='out1mf4':'constant',[-0.188110472462333]
MF5='out1mf5':'constant',[0.743936155026231]
MF6='out1mf6':'constant',[16.8997986036545]
MF7='out1mf7':'constant',[-9.23722199686824]
MF8='out1mf8':'constant',[102.873314242532]
MF9='out1mf9':'constant',[1.73665894986522]
MF10='out1mf10':'constant',[0.0212873232679715]
MF11='out1mf11':'constant',[-0.186980425524613]
MF12='out1mf12':'constant',[-14.1374715484766]
MF13='out1mf13':'constant',[0.0320467547490414]
MF14='out1mf14':'constant',[-9.270867910532e-005]
MF15='out1mf15':'constant',[-0.310140042864206]
MF16='out1mf16':'constant',[0.258765665447207]
MF17='out1mf17':'constant',[0.218936660423444]
MF18='out1mf18':'constant',[1.93976844948546]
MF19='out1mf19':'constant',[-0.420721937336112]
MF20='out1mf20':'constant',[1.8048593600609]
MF21='out1mf21':'constant',[0]
MF22='out1mf22':'constant',[-0.0135449045280753]
MF23='out1mf23':'constant',[0.197791351576457]
MF24='out1mf24':'constant',[-0.529261703124254]
MF25='out1mf25':'constant',[-0.478811223831693]
MF26='out1mf26':'constant',[0.239919755475814]
MF27='out1mf27':'constant',[0.758932284850586]

[Rules]
1 1 1, 1 (1) : 1
1 1 2, 2 (1) : 1
1 1 3, 3 (1) : 1
1 2 1, 4 (1) : 1
1 2 2, 5 (1) : 1
1 2 3, 6 (1) : 1
1 3 1, 7 (1) : 1
1 3 2, 8 (1) : 1
1 3 3, 9 (1) : 1
2 1 1, 10 (1) : 1
2 1 2, 11 (1) : 1
2 1 3, 12 (1) : 1
2 2 1, 13 (1) : 1
2 2 2, 14 (1) : 1
2 2 3, 15 (1) : 1
2 3 1, 16 (1) : 1
2 3 2, 17 (1) : 1
2 3 3, 18 (1) : 1
3 1 1, 19 (1) : 1
3 1 2, 20 (1) : 1
3 1 3, 21 (1) : 1
3 2 1, 22 (1) : 1
3 2 2, 23 (1) : 1
3 2 3, 24 (1) : 1

3 3 1, 25 (1) : 1
3 3 2, 26 (1) : 1
3 3 3, 27 (1) : 1

B.9 Rule base of the reflux drum compositions estimators

If (Tray 2 Temp. is S) and (Tray 5 Temp. is S) and (Tray 9 Temp. is S) then Comp. C ...s outmf1
If (Tray 2 Temp. is S) and (Tray 5 Temp. is S) and (Tray 9 Temp. is M) then Comp. C ... is outmf2
If (Tray 2 Temp. is S) and (Tray 5Temp. is S) and (Tray 9 Temp. is B) then Comp. C ... is outmf3
If (Tray 2 Temp. is S) and (Tray 5 Temp. is M) and (Tray 9 Temp. is S) then Comp. C ... is outmf4
If (Tray 2 Temp. is S) and (Tray 5 Temp. is M) and (Tray 9 Temp. is M) then Comp. C ... is outmf5
If (Tray 2 Temp. is S) and (Tray 5 Temp. is M) and (Tray 9 Temp. is B) then Comp. C ... is outmf6
If (Tray 2 Temp. is S) and (Tray 5 Temp. is B) and (Tray 9 Temp. is S) then Comp. C ... is outmf7
If (Tray 2 Temp. is S) and (Tray 5 Temp. is B) and (Tray 9 Temp. is M) then Comp. C ... is outmf8
If (Tray 2 Temp. is S) and (Tray 5 Temp. is B) and (Tray 9 Temp. is B) then Comp. C ... is outmf9
If (Tray 2 Temp. is M) and (Tray 5 Temp. is S) and (Tray 9 Temp. is S) then Comp. C ... is outmf10
If (Tray 2 Temp. is M) and (Tray 5 Temp. is S) and (Tray 9 Temp. is M) then Comp. C ... is outmf11
If (Tray 2 Temp. is M) and (Tray 5 Temp. is S) and (Tray 9 Temp. is B) then Comp. C ... is outmf12
If (Tray 2 Temp. is M) and (Tray 5 Temp. is M) and (Tray 9 Temp. is S) then Comp. C ... is outmf13
If (Tray 2 Temp. is M) and (Tray 5 Temp. is M) and (Tray 9 Temp. is M) then Comp. C ... is outmf14
If (Tray 2 Temp. is M) and (Tray 5 Temp. is M) and (Tray 9 Temp. is B) then Comp. C ... is outmf15
If (Tray 2 Temp. is M) and (Tray 5 Temp. is B) and (Tray 9 Temp. is S) then Comp. C ... is outmf16
If (Tray 2 Temp. is M) and (Tray 5 Temp. is B) and (Tray 9 Temp. is M) then Comp. C ... is outmf17
If (Tray 2 Temp. is M) and (Tray 5 Temp. is B) and (Tray 9 Temp. is B) then Comp. C ... is outmf18
If (Tray 2 Temp. is B) and (Tray 5 Temp. is S) and (Tray 9 Temp. is S) then Comp. C ... is outmf19
If (Tray 2 Temp. is B) and (Tray 5 Temp. is S) and (Tray 9 Temp. is M) then Comp. C ... is outmf20
If (Tray 2 Temp. is B) and (Tray 5 Temp. is S) and (Tray 9 Temp. is B) then Comp. C ... is outmf21
If (Tray 2 Temp. is B) and (Tray 5 Temp. is M) and (Tray 9 Temp. is S) then Comp. C ... is outmf22
If (Tray 2 Temp. is B) and (Tray 5 Temp. is M) and (Tray 9 Temp. is M) then Comp. C ... is outmf23
If (Tray 2 Temp. is B) and (Tray 5 Temp. is M) and (Tray 9 Temp. is B) then Comp. C ... is outmf24
If (Tray 2 Temp. is B) and (Tray 5 Temp. is B) and (Tray 9 Temp. is S) then Comp. C ...is outmf25
If (Tray 2 Temp. is B) and (Tray 5 Temp. is B) and (Tray 9 Temp. is M) then Comp. C ... is outmf26
If (Tray 2 Temp. is B) and (Tray 5 Temp. is B) and (Tray 9 Temp. is B) then Comp. C ... is outmf27

B.10 Simulation code of pH control system

Parameter1.m

```
% pH Modelling ( Wright and Kravaris, 1991)
% Implementing this strategy to controlling problem of pH reactor om MC AVoy (1972)
% Acetic acid and NaOH input streams
% Content of the Acid feed :  HAc
% Initial concentrations  HAc :(C2)
% Initial concentration NaOH (alfa) :(alfa3)
con2=0.00d0;
con1=0.32d0;
alfa3=0.05d0;
% Constant Flow rate of Acid stream (L/min)
F=81;
% Volume of the tank (L)
V=1000 ;
% Dissociation constant of Acetic Acid at (25 C).
pKa=4.76d0 ;
%  NaOH or Na in the tank initially, total Na concentration
x3(1)=0.04320469795531 ;
% Assuming there will be no HCl left drom the dissociation of HCl x2(0)=C2(0) (M)
x2(1)=con2;
% Finding eguation obtained using Ka for x1(1)
x1(1)=0.04348993295291;
%Initial time is zero
time(1)=0.0d0;
```

```
time(2)=1;
% Initial pH in the tank from initial total hydrogen concentration
pH(1)=6.94;
sp=6.94;
% Initial Flow rate (lt/min)
u(1)=515;
v(1)=515;
%Sampling interval (min)
delt=1;% Set point tracking
delsp=0;
%Learning rate
n=0.01;
% jacobian parameters
sa=0.0115;
sc=515;
% Anfis initial parameters
  b1=0.200574360279041 ;
  c1=5.28061128448098;
    b2=0.200052784216443 ;
  c2=5.69939525951916;
  b3=0.186851285106911 ;
  c3=6.13804978912724;
  B1=25.2143373079753  ;
  C1=400.00001723419;
  B2=25.214317364033 ;
  C2=459.374935088131;
  B3=25.2141941522302 ;
  C3=518.750006915436;
   r1=372.386;
  r2=391.352;
  r3=-39.80488;
  r4=658.9136;
  r5=526.310;
  r6=509.505;
  r7=0.20478;
  r8=408.027;
  r9=527.604;
```

Parameter2.m

```
% Content of the Acid feed :  HAc
% Initial concentrations  HAc :(C2), HCl (C1)
% Initial concentration NaOH (alfa) :(alfa3)
con2=0.00d0;
con1=0.32d0;
alfa3=0.05d0;
% Constant Flow rate of Acid stream (L/min)
F=81;
% Volume of the tank (L)
V=1000 ;
% Dissociation constant of Acetic Acid at (25 C).
pKa=4.76d0 ;
%  NaOH or Na in the tank initially, total Na concentration
x3(51)=0.0373 ;
% Assuming there will be no HCl left drom the dissociation of HCl x2(0)=C2(0) (M)
x2(51)=con2;
% Finding eguation obtained using Ka for x1(1)
x1(51)=0.0814;
time(50)=43;
time(51)=43.25;
%Sampling interval (min)
delt=0.25;
% Set point tracking
delsp=0;
%Learning rate
n=0.01;
```

```
% jacobian parameters
sa=0.0115;
sc=515;
% Anfis initial parameters
 b1=0.85640;
 c1=5.266;
 b2=0.439;
 c2=32.0436;
   b3=0.5358;
 c3=11.9802;
  B1=62.63957;
 C1=300;
   B2=62.5913 ;
 C2=447.49;
   B3=62.5331;
 C3=595;
 r1=236.6003;
 r2=382.0986;
 r3=591.499;
 r4=1323;
 r5=812.699;
 r6=249.5989;
 r7=4030;
 r8=433.39;
 r9=626.299;
```

pHcontsim.m

```
% Closed loop simulation of pH CSTR reactor system.
% Written by Evren Güner
% Simulation part1
% Controller and CSTR Parameter initalization
 parga;
 for k=1:48;
% Solution of states for each time step in pH system
% GOVERNING EQUATIONS
% xi(k+1)=xi(k).exp[-((Fk+uk)/V).delt]+[(Fk.(ci)k+uk.&i)/Fk+uk].[1-exp(-((Fk+uk)/V).delt)]
% ai(ph(k+1)).xi(k+1)+A(ph(k+1))=0
    x1(k+1)=[x1(k)*exp(-((F+u(k))/V)*delt)]+[(F*(con1))/(F+u(k))]*[1-exp(-((F+u(k))/V)*delt)] ;
    x2(k+1)=0;
    x3(k+1)=[x3(k)*exp(-((F+u(k))/V)*delt)]+[(u(k)*(alfa3))/(F+u(k))]*[1-exp(-((F+u(k))/V)*delt)] ;
    A=x1(k+1);
    B=x2(k+1);
    C=x3(k+1);
%Solving pH values using states at time step
 D=FZERO(@necati, 7, optimset, A,B,C);
 time(k+1)=time(k)+delt ;
 pH(k+1)=D;
 H=pH(k);
if k<20;
    sp=6.94;
    delt=1;
else if k>=21 & k<=41
     sp=5.0;
     delt=1;
 else
     sp=9.0;
     delt=0.25;
 end
end
 pHsp(k+1)=sp;
 g=pH(k+1)-pH(k);
% ANFIS Controller ;  Five Layer connectionist network
% Layer1 , Calculating the MF's values for inputs ( F2(k) and pH(k)
% Layer2 , Calculating the rule firing strengths
% Layer3 , Calculating the normalized firing strengths ( ratio of ht ith rule's
```

```
%firing strength to the sum of all rule's FS)
% Layer4 , Calculating the rules outputs
% Layer5 ; Calculating the over all output ; F2(k+1)
% Inputs to the ANFIS Controller
% v(k) ; manipulated variable ( Flow Rate of Base F2(k))
% pH(k) ; pH value of the system
% Output ; F2(k+1) ;manipulated variable
% Membership functions for the input values choosen as Gaussian type
% LAYER 1 Calculating the MF's values for inputs
% For v(k); Ma's
% Ma1 ; small for pH(k)
Ma1=exp(-0.5*(((pH(k)-c1)/b1)^2));
delc1=(1/(b1^3))*(pH(k)-c1)*exp(-0.5*(((pH(k)-c1)/b1)^2));
delb1=(1/(b1^3))*((pH(k)-c1)^2)*exp(-0.5*(((pH(k)-c1)/b1)^2));
% Ma2 ; medium for pH(k)
Ma2=exp(-0.5*(((pH(k)-c2)/b2)^2));
delc2=(1/(b2^3))*(pH(k)-c2)*exp(-0.5*(((pH(k)-c2)/b1)^2));
delb2=(1/(b2^3))*((pH(k)-c2)^2)*exp(-0.5*(((pH(k)-c2)/b2)^2));
% Ma3 ; big for v(k)
Ma3=exp(-0.5*(((pH(k)-c3)/b3)^2));
delc3=(1/(b3^3))*(pH(k)-c3)*exp(-0.5*(((pH(k)-c3)/b3)^2));
delb3=(1/(b3^3))*((pH(k)-c3)^2)*exp(-0.5*(((pH(k)-c3)/b3)^2));
% For F2(k) ; Mb's
% Mb1 ;  small for v(k)
Mb1=exp(-0.5*(((v(k)-C1)/B1)^2));
delC1=(1/(B1^3))*(v(k)-C1)*exp(-0.5*(((v(k)-C1)/B1)^2));
delB1=(1/(B1^3))*((v(k)-C1)^2)*exp(-0.5*(((v(k)-C1)/B1)^2));
% Mb2 ;  medium for v(k)
Mb2=exp(-0.5*(((v(k)-C2)/B2)^2));
delC2=(1/(B2^3))*(v(k)-C2)*exp(-0.5*(((v(k)-C2)/B2)^2));
delB2=(1/(B2^3))*((v(k)-C2)^2)*exp(-0.5*(((v(k)-C2)/B2)^2));
% Mb3 ; big for v(k)
Mb3=exp(-0.5*(((v(k)-C3)/B3)^2));
delC3=(1/(B3^3))*(v(k)-C3)*exp(-0.5*(((v(k)-C3)/B2)^2));
delB3=(1/(B3^3))*((v(k)-C3)^2)*exp(-0.5*(((v(k)-C3)/B3)^2));
% LAYER2 Calculating the rule firing strengths
% w1; If pH(k) is S and v(k) is S then f1 = r1
 w1=Ma1*Mb1 ;
% w2;  If pH(k) is S and v(k) is M then f2 = r2
 w2=Ma1*Mb2 ;
% w3 ; If pH(k) is S and v(k) is B then f3 = r3
 w3=Ma1*Mb3 ;
% w4 ; If pH(k) is M and v(k) is S then f4 = r4
 w4=Ma2*Mb1 ;
% w5 ; If pH(k) is M and v(k) is M then f5 = r5
 w5=Ma2*Mb2 ;
 % w6 ; If pH(k) is   M and v(k) is B then f6 = r6
 w6=Ma2*Mb3 ;
 % w7 ; If pH(k) is   B and v(k) is    S   then f7 = r7
 w7=Ma3*Mb1 ;
 % w8 ; If pH(k) is   B and v(k) is    M   then f8 = r8
 w8=Ma3*Mb2 ;
 % w9 ; If pH(k) is   B and v(k) is    B   then f9 = r9
 w9=Ma3*Mb3;
 % LAYER 3 , Calculating the normalized firing strengths Wi ( ratio of ht ith rule's
 %firing strength to the sum of all rule's FS)
 % Sum of the firing strengths
 wT=w1+w2+w3+w4+w5+w6+w7+w8+w9 ;
 % Normalized firing strengths
 W1=w1/wT; W2=w2/wT ; W3=w3/wT ; W4=w4/wT; W5=w5/wT ; W6=w6/wT; W7=w7/wT; W8=w8/wT ;
W9=w9/wT;
 % LAYER 4 , Calculating the rules outputs
 f1=(r1) ;f2=(r2);f3=(r3);f4=(r4);f5=(r5);
 f6=(r6) ;f7=(r7);f8=(r8) ;f9=(r9);
 o1=W1*f1 ;  o2=W2*f2 ;  o3=W3*f3 ;  o4=W4*f4 ;  o5=W5*f5;
 o6=W6*f6;  o7=W7*f7 ;  o8=W8*f8;  o9=W9*f9 ;
```

```
 % LAYER 5 ; Calculating the over all output ; Fbase(k) at time t
if k<20;
   u(k+1)=515;
else
   u(k+1)=(o1+o2+o3+o4+o5+o6+o7+o8+o9);
end
v(k+1)=u(k+1);
% For plant Jacobian derivative of the pH with respect to Fbase is
% calculated from the sigmoid model of the plant
% pH= 14/(1+exp(-sa*(u(k)-sc)));
% Plant Jacobian:  delpH/delFbase
jacobian=(14*sa*exp(-sa*(u(k)-sc)))/((1+exp(-sa*(u(k)-sc)))^2);
% Adaptation of parameters according to the equations derived from the EBP algorithm
% n; learning rate
% MFs  parameters b1,c1,b2,c2,b3,c3,B1,C1,B2,C2,B3,C3 the parameters of the
% gaussian MFs
b1=b1-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delb1)/wT)*(Mb1*r1+Mb2*r2+Mb3*r3));
c1=c1-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delc1)/wT)*(Mb1*r1+Mb2*r2+Mb3*r3));
b2=b2-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delb2)/wT)*(Mb1*r4+Mb2*r5+Mb3*r6));
c2=c2-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delc2)/wT)*(Mb1*r4+Mb2*r5+Mb3*r6));
b3=b3-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delb3)/wT)*(Mb1*r7+Mb2*r8+Mb3*r9));
c3=c3-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delc3)/wT)*(Mb1*r7+Mb2*r8+Mb3*r9));
B1=B1-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delB1)/wT)*(Ma1*r1+Ma2*r2+Ma3*r3));
C1=C1-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delC1)/wT)*(Ma1*r1+Ma2*r2+Ma3*r3));
B2=B2-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delB2)/wT)*(Ma1*r4+Ma2*r5+Ma3*r6));
C2=C2-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delC2)/wT)*(Ma1*r4+Ma2*r5+Ma3*r6));
B3=B3-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delB3)/wT)*(Ma1*r7+Ma2*r8+Ma3*r9));
C3=C3-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delC3)/wT)*(Ma1*r7+Ma2*r8+Ma3*r9));
% Consequent parameters ; ri
 r1=r1-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W1];
 r2=r2-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W2];
 r3=r3-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W3];
 r4=r4-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W4];
 r5=r5-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W5];
 r6=r6-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W6];
 r7=r7-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W7];
 r8=r8-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W8];
 r9=r9-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W9];
end
% Saving data
simpart1(:,1)=time';
simpart1(:,2)=u';
simpart1(:,3)=pH';
save simpart1 simpart1 ;
clear;
% Simulation part2
% Controller and CSTR Parameter initalization
 parga2;
 for k=50:400;
   x1(k+1)=[x1(k)*exp(-((F+u(k))/V)*delt)]+[(F*(con1))/(F+u(k))]*[1-exp(-((F+u(k))/V)*delt)] ;
   x2(k+1)=0;
   x3(k+1)=[x3(k)*exp(-((F+u(k))/V)*delt)]+[(u(k)*(alfa3))/(F+u(k))]*[1-exp(-((F+u(k))/V)*delt)] ;
   A=x1(k+1);
   B=x2(k+1);
   C=x3(k+1);
 D=FZERO(@necati, 7, optimset, A,B,C);
 time(k+1)=time(k)+delt ;
 pH(k+1)=D;
 H=pH(k);
if k<20;
   sp=5.0;
else
sp=9.0;
end
 pHsp(k+1)=sp;
 g=pH(k+1)-pH(k);
```

```
% LAYER 1 Calculating the MF's values for inputs
% For v(k); Ma's
% Ma1 ; small for pH(k)
Ma1=exp(-0.5*(((pH(k)-c1)/b1)^2));
delc1=(1/(b1^3))*(pH(k)-c1)*exp(-0.5*(((pH(k)-c1)/b1)^2));
delb1=(1/(b1^3))*((pH(k)-c1)^2)*exp(-0.5*(((pH(k)-c1)/b1)^2));
% Ma2 ; medium for pH(k)
Ma2=exp(-0.5*(((pH(k)-c2)/b2)^2));
delc2=(1/(b2^3))*(pH(k)-c2)*exp(-0.5*(((pH(k)-c2)/b1)^2));
delb2=(1/(b2^3))*((pH(k)-c2)^2)*exp(-0.5*(((pH(k)-c2)/b2)^2));
% Ma3 ; big for v(k)
Ma3=exp(-0.5*(((pH(k)-c3)/b3)^2));
delc3=(1/(b3^3))*(pH(k)-c3)*exp(-0.5*(((pH(k)-c3)/b3)^2));
delb3=(1/(b3^3))*((pH(k)-c3)^2)*exp(-0.5*(((pH(k)-c3)/b3)^2));
% For pH(k) ; Mb's
% Mb1 ;  small for v(k)
Mb1=exp(-0.5*(((v(k)-C1)/B1)^2));
delC1=(1/(B1^3))*(v(k)-C1)*exp(-0.5*(((v(k)-C1)/B1)^2));
delB1=(1/(B1^3))*((v(k)-C1)^2)*exp(-0.5*(((v(k)-C1)/B1)^2));
% Mb2 ;  medium for v(k)
Mb2=exp(-0.5*(((v(k)-C2)/B2)^2));
delC2=(1/(B2^3))*(v(k)-C2)*exp(-0.5*(((v(k)-C2)/B2)^2));
delB2=(1/(B2^3))*((v(k)-C2)^2)*exp(-0.5*(((v(k)-C2)/B2)^2));
% Mb3 ; big for v(k)
Mb3=exp(-0.5*(((v(k)-C3)/B3)^2));
delC3=(1/(B3^3))*(v(k)-C3)*exp(-0.5*(((v(k)-C3)/B2)^2));
delB3=(1/(B3^3))*((v(k)-C3)^2)*exp(-0.5*(((v(k)-C3)/B3)^2));
% LAYER2 Calculating the rule firing strengths
% w1; If pH(k) is S and v(k) is S then f1 = r1
 w1=Ma1*Mb1 ;
% w2;  If pH(k) is S and v(k) is M then f2 = r2
 w2=Ma1*Mb2 ;
% w3 ; If pH(k) is S and v(k) is B then f3 = r3
 w3=Ma1*Mb3 ;
% w4 ; If pH(k) is M and v(k) is S then f4 = r4
 w4=Ma2*Mb1 ;
% w5 ; If pH(k) is M and v(k) is M then f5 = r5
 w5=Ma2*Mb2 ;
 % w6 ; If pH(k) is   M and v(k) is B then f6 = r6
 w6=Ma2*Mb3 ;
 % w7 ; If pH(k) is   B and v(k) is    S   then f7 = r7
 w7=Ma3*Mb1 ;
 % w8 ; If pH(k) is   B and v(k) is    M   then f8 = r8
 w8=Ma3*Mb2 ;
 % w9 ; If pH(k) is   B and v(k) is    B   then f9 = r9
 w9=Ma3*Mb3;
 % LAYER 3 , Calculating the normalized firing strengths Wi ( ratio of ht
 % ith rule's firing strength to the sum of all rule's FS)
 % Sum of the firing strengths
 wT=w1+w2+w3+w4+w5+w6+w7+w8+w9 ;
 % Normalized firing strengths
 W1=w1/wT; W2=w2/wT ; W3=w3/wT ; W4=w4/wT; W5=w5/wT ; W6=w6/wT; W7=w7/wT; W8=w8/wT ;
W9=w9/wT;
 % LAYER 4 , Calculating the rules outputs
 f1=(r1) ;f2=(r2);f3=(r3);f4=(r4);f5=(r5);
 f6=(r6) ;f7=(r7);f8=(r8) ;f9=(r9);
 o1=W1*f1 ;  o2=W2*f2 ;  o3=W3*f3 ;  o4=W4*f4 ;  o5=W5*f5;
 o6=W6*f6;  o7=W7*f7 ;  o8=W8*f8;  o9=W9*f9 ;
 % LAYER 5 ; Calculating the over all output ; Fbase(k) at time t
 u(k+1)=(o1+o2+o3+o4+o5+o6+o7+o8+o9);
 v(k+1)=u(k+1);
% For plant Jacobian derivative of the pH with respect to Fbase is
jacobian=(14*sa*exp(-sa*(u(k)-sc)))/((1+exp(-sa*(u(k)-sc)))^2);
% Adaptation of parameters according to the equations derived from the EBP algorithm
b1=b1-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delb1)/wT)*(Mb1*r1+Mb2*r2+Mb3*r3));
c1=c1-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delc1)/wT)*(Mb1*r1+Mb2*r2+Mb3*r3));
```

```
b2=b2-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delb2)/wT)*(Mb1*r4+Mb2*r5+Mb3*r6));
c2=c2-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delc2)/wT)*(Mb1*r4+Mb2*r5+Mb3*r6));
b3=b3-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delb3)/wT)*(Mb1*r7+Mb2*r8+Mb3*r9));
c3=c3-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delc3)/wT)*(Mb1*r7+Mb2*r8+Mb3*r9));
B1=B1-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delB1)/wT)*(Ma1*r1+Ma2*r2+Ma3*r3));
C1=C1-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delC1)/wT)*(Ma1*r1+Ma2*r2+Ma3*r3));
B2=B2-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delB2)/wT)*(Ma1*r4+Ma2*r5+Ma3*r6));
C2=C2-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delC2)/wT)*(Ma1*r4+Ma2*r5+Ma3*r6));
B3=B3-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delB3)/wT)*(Ma1*r7+Ma2*r8+Ma3*r9));
C3=C3-n*((pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*((delC3)/wT)*(Ma1*r7+Ma2*r8+Ma3*r9));
% Consequent parameters ; ri
 r1=r1-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W1];
 r2=r2-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W2];
 r3=r3-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W3];
 r4=r4-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W4];
 r5=r5-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W5];
 r6=r6-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W6];
 r7=r7-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W7];
 r8=r8-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W8];
 r9=r9-n*[(pHsp(k+1)-pH(k+1))*(-1)*(jacobian)*W9];
end
% Saving data
tim=time'; U=u'; PH=pH';
simpart2(:,1)=tim(50:400,:);
simpart2(:,2)=U(50:400,:);
simpart2(:,3)=PH(50:400,:);
save simpart2 simpart2 ;
clear;
load simpart1;
load simpart2;
simdata=[simpart1;simpart2];
save simdata simdata;
clear;
```